Package ‘GeneGroupAnalysis’

September 24, 2012

Type Package

Title Gene Functional Class Analysis

Version 1.2.0

Date 2011-10-05

Author Alejandro Quiroz-Zarate and John Quackenbush

Maintainer Alejandro Quiroz-Zarate <`aquiroz@hsph.harvard.edu`>

Description R package providing functions to peform gene-set significance analysis over simple cross-sectional or time series data designs.

biocViews
GeneExpression, DifferentialExpression, MultipleComparisons,CrossSectional, TimeCourse

Depends R (>= 2.10), MCMCpack, GO.db, breastCancerVDX,rheumaticConditionWOLLBOLD, hgu133a.db, hgu133plus2.db

Imports AnnotationDbi, annotate, tcltk

License Artistic-2.0

Collate ArrayInfoFun.R GeneMaxVarFun.R GrpMeanDataFun.R MakeAffyOrderGrp.Fun.R MeanMatGrps.Aux.ts.R GeneGrps2AffyGrpsFun.R SizeGOAffyGrps.R MCMCData.cs.R MCMCData.ts.R Alfa.Aux.cs.R Alfa0NPCFun.ts.R AlfaAllCFun.ts.R AlfaFun.cs.R AlfaNPCFun.ts.R Beta.Aux.cs.R BetaCFun.ts.R BetaFun.cs.R PiFun.R RhoFun.R SgmAlfaFun.cs.R SgmAlfaFun.ts.R SgmAprGrps.Aux.ts.R SgmY.Aux.cs.R SgmYCFun.ts.R SgmYFun.cs.R GibbsFun.cs.R GibbsAllFun.ts.R GibbsNPFun.ts.R

LazyLoad yes

URL `http://compbio.dfci.harvard.edu/`

R topics documented:

`GeneGroupAnalysis-package` *The Gene Group Analysis package for Gene Functional Class Analysis.*

Description

Functions to perform gene-set significance analysis method for data sets with cross-sectional or time series designs.

Details

Package:	GeneGroupAnalysis
Type:	Package
Version:	1.0
Date:	2011-10-05
License:	Artistics-2.0
LazyLoad:	yes

For detailed examples on the use and manipulation of the functions please see the package Vignette.

Author(s)

A. Quiroz-Zarate and John Quackenbush `<aquiroz@hsph.harvard.edu>`

References

Quiroz-Zarate A and Quackenbush J. "Manuscript in preparation. 2012."

`ArrayInfoFun` *Function that extracts the required information from the data set.*

Description

This function captures the Affymetrix identifiers and the gene symbols associated to the data set of interest in order to prepare the data set for the Gibb's sampler execution.

Usage

```
ArrayInfoFun(data, array.symbols)
```

Arguments

`data`	Data set of interest in the form of a matrix: columns for patients, rows for Affymetrix identifiers.
`array.symbols`	Mapping between Affymetrix identifiers and gene symbols for the associated Affymetrix platform of the data set of interest.

Details

This function captures the necessary information from the data set of interest.

Value

This function returns a list containing the following results:

`genes.name`	A vector of gene symbols off all the Affymetrix identifiers (rows of the data set) that have a gene symbol.
`probes.name`	A vector of Affymetrix identifiers (rows of the data set).
`probes.ids`	A vector containing the row number of their respective Affymetrix identifier (rows of the data set).
`genes.name.unique`	A vector of unique gene symbols off all the Affymetrix identifiers (rows of the data set) in the data set.

Author(s)

A. Quiroz-Zarate and John Quackenbush.

See Also

See the `GeneGroupAnalysis` Vignette for examples on how to use this function and the help of the function `GibbsAllFun.ts` for a detailed example of its use.

Examples

```
#- For an example on the use of this function go to:
#- GibbsAllFun.ts
```

`GeneGrps2AffyGrpsFun` *Function that translates GO hierarchical tree of gene symbols to a groups of Affymetrix identifiers of interest.*

Description

This function converts GO hierarchical trees of gene symbols to groups of their associated Affymetrix identifiers.

Usage

```
GeneGrps2AffyGrpsFun(GO.group, level, gene.u.name, affy.max.indexes)
```

Arguments

`GO.group`	The GO ontology that is of interest to consider in the analysis. ("BP","MF","CC")
`level`	Depth of the GO hierarchical tree of the ontology chosen. Level can only be an integer between 3 and 10.
`gene.u.name`	An `ArrayInfoFun` item containing in a vector the unique gene symbols considered on the data set of interest.
`affy.max.indexes`	A `GeneMaxVarFun` item consisting of the vector with the Affymetrix identifiers associated to the genes that had the maximum variability across phenotypes.

Value

This function returns a list containing the following results:

`index.GO.grps`	List of row number groups for their associated gene symbols.
`gene.GO.grps`	List of gene symbol groups for their associated GO functional class.
`GO.grps`	List of GO functional classes and their associated children functional classes.

Author(s)

A. Quiroz-Zarate and John Quackenbush.

See Also

See the `GeneGroupAnalysis` Vignette for examples on how to use this function and the help of the function `GibbsAllFun.ts` for a detailed example of its use.

Examples

```
#- For an example on the use of this function go to:
#- GibbsAllFun.ts
```

`GeneMaxVarFun` *Function to compute the Affymetrix identifiers with the maximum variability between phenotypes.*

Description

Some Affymetrix identifiers map to the same gene symbol. This implies that there exist more than one row number from the data set associated to a gene symbol. This function computes the respective row number associated to the Affymetrix identifier the presents the maximum variability across phenotypes for every group of Affymetrix identifiers mapping to gene symbols.

Usage

```
GeneMaxVarFun(ArrayInfoFun.ans, data, phenotype.a, phenotype.b)
```

Arguments

`ArrayInfoFun.ans`	A `ArrayInfoFun` item of the data set of interest.
`data`	Data set of interest in the form of a matrix: columns for patients, rows for Affymetrix identifiers.
`phenotype.a`	Vector of indices with the column identifiers of phenotype a.
`phenotype.b`	Vector of indices with the column identifiers of phenotype b.

Details

This function captures the row numbers associated to the Affymetrix identifiers with the maximum variability between phenotypes for every group of Affymetrix identifiers mapping to gene symbols.

Value

This function returns a list containing the following results:

`IndexMaxVar`	A vector with the row number ids of the respective Affymetrix identifiers with the maximum variability across phenotypes.
`AffyIdsMaxVar`	A vector with the Affymetrix identifiers associated to the "indexes.good"

Author(s)

A. Quiroz-Zarate and John Quackenbush.

See Also

See the `GeneGroupAnalysis` Vignette for examples on how to use this function and the help of the function `GibbsAllFun.ts` for a detailed example of its use.

Examples

```
#- For an example on the use of this function go to:
#- GibbsAllFun.ts
```

`GGAdata` *List containing a collection of gene symbols with their associated GO term and GO hierarchical trees for each ontology*

Description

This dataset contains list containing a matrix with the gene symbols associated with their respective GO term, based on the from the MSigDB at the Broad Institute. It also contains lists of GO hierarchical trees based on the ontologies and the MSigDB from the Broad Institute.

Usage

```
data(GGAdata)
```

Format

A matrix containing the following information by rows:

- `GGAdata$MSigDBAnnotation[1:20,]`: The first 20 rows of the data base from MSigDD. It displays the gene symbols, GO ids, Ontology for who the GO id belongs (C-CC, F-MF, P-BP) and a description of the gene function.
- `GGAdata$CClevels[[3]]`: The CC hierarchical tree of depth 3.
- `GGAdata$MFlevels[[3]]`: The MF hierarchical tree of depth 3.
- `GGAdata$BPlevels[[3]]`: The BP hierarchical tree of depth 3.

Details

This dataset contains list containing a matrix with the gene symbols associated with their respective GO term, based on the from the MSigDB at the Broad Institute. As well it contains lists of GO hierarchical trees based on the ontologies and the MSigDB from the Broad Institute. This data set enables the construction of the gene groups of interest for the analysis of the methodology proposed.

Source

http://www.broadinstitute.org/gsea/msigdb/collections.jsp#C5

References

Quiroz-Zarate A and Quackenbush J (2012). "Manuscript in preparation".

Examples

```
#- load the dataset
data(GGAdata,package="GeneGroupAnalysis")
#- show the first 20 rows and columns.
GGAdata$MSigDBAnnotation[1:20,]
```

`GibbsAllFun.ts` *Function to compute the Gibb's sampler for the series data design.*

Description

This function obtains the posterior samples for the parameters of the linear model on the mean of the gene functional classes under a time series data design. These posterior samples are for the parameters for the model that identifies gene functional classes that present a parallel and non-parallel time course trajectories between phenotypes.

Usage

```
GibbsAllFun.ts(y.mu.a, y.mu.b, grp.sz, beta.mat, alfa.mat, sgm.alfa, rho, pi.i, mm, aa.pi, lmbd,
```

Arguments

`y.mu.a`	A `MCMCData.ts` item with the preprocessed data matrix for phenotype a.
`y.mu.b`	A `MCMCData.ts` item with the preprocessed data matrix for phenotype b.
`grp.sz`	A `MCMCData.ts` item with the preprocessed data matrix for phenotype b.
`beta.mat`	Matrix for the posterior samples for the parameter Beta.
`alfa.mat`	Matrix for the posterior samples for the parameter Alpha.
`sgm.alfa`	Matrix for the posterior samples of the variance of the prior distribution for the parameter Alpha.
`rho`	Vector for the posterior samples for the parameter Rho.
`pi.i`	Vector for the posteriors samples for the parameter Pi.
`mm`	A real number for the mean of the Beta distribution for the non-zero part for the parameter Pi.
`aa.pi`	A real number for the precision parameter of the Beta distribution for the non-zero part for the parameter Pi.
`lmbd`	A prior estimate for the distribution for the variance assumed on the data.
`df.lmbd`	An integer defining the degrees of freedom for the matrix apriori estimate for the distribution for the variance assumed on the data.
`lmbd.alf`	A prior estimate for the prior distribution for the variance for the parameter Alpha.
`indexes`	Vector of integers defining the indexes in the data that delineate the different time points for the sample identifiers (columns of the data set).
`num.time.pnts`	An integer defining the number of time points considered.
`num.pcnts`	An integer defining the number of subjects considered.
`apriori.diff.exp`	An integer for the expected number of apriori differentially expressed gene groups.
`nsim`	An integer defining the total number of iterations for the Gibb's sampler.
`burn.in`	An integer defining the total number of iterations for the Gibb's sampler.
`often`	An integer defining the frequency of the iterations to print the number of differentially expressed gene functional classes up to those iterations.

`prob.cut.off`	A real number defining the cut off on the probability of functional gene classes differentially expressed between phenotypes across the time course. The number of functional gene classes that pass this threshold will be printed put during the iteration procedure of the Gibb's sampler.

Details

This function calculates the posterior samples for the parameters of interest for the time series data design, where the objective is to calculate the functional gene classes that are differentially expressed between phenotypes across the time course. During the iteration process, this function prints out the number of differentially expressed gene functional classes with the desired probability, defined by prob.cut.off.

Value

This function returns a list containing the following results:

`Beta`	Matrix with the posterior samples for the parameter beta.
`Alfa`	Matrix with the posterior samples for the parameter alpha.
`Pi`	Vector with the posterior samples for the parameter pi.
`Rho`	Vector with the posterior samples for the parameter rho.

Author(s)

A. Quiroz-Zarate and John Quackenbush.

See Also

See the `GeneGroupAnalysis` Vignette for examples on how to use this function.

Examples

```
library(annotate)
library(rheumaticConditionWOLLBOLD)
library(hgu133plus2.db)

data(wollbold,package="rheumaticConditionWOLLBOLD")
#----Normalized expression data set
woll.data.exp=exprs(wollbold)
#---- Phenotypic information from the data set
#pData(wollbold)
#---ER status of patients
TGF.exp=1:30
TNF.exp=31:60

#--- Checking that the columns correspond to their respective phenotype data id
all(colnames(woll.data.exp)==rownames(pData(wollbold)))
#---Checking the annotation of the data
annotation(wollbold)
array.info=ArrayInfoFun(woll.data.exp,hgu133plus2SYMBOL)
genes.max.var=GeneMaxVarFun(array.info,woll.data.exp,TGF.exp,TNF.exp)
GO2Gene.grps=GeneGrps2AffyGrpsFun("CC",3,array.info$genes.name.unique,genes.max.var$IndexMaxVar)
GO.grps=GO2Gene.grps$GO.grps

Wollbold09WrkngGrps=SizeGOAffyGrps(GO2Gene.grps$index.GO.grps,100)
```

```
indexes.1=c(1,2,3,4,5,6)
Wollbold09MCMCData=MCMCData.ts(Wollbold09WrkngGrps$groups,GO2Gene.grps$index.GO.grps,
GO2Gene.grps$GO.grps,woll.data.exp,TGF.exp,TNF.exp,indexes.1,5)

nsim = 40
burn.in = 10
Grps.apriori.diff.exp = 23
shape = 3
scale = 0.1
mm.pi = 0.75
aa.pi = 10
often = 20
cut.off = 0.7
df.lambda = 10
no.time.pnts = 5
no.pcnts = 6
v.SS.i = array(0, c(length(Wollbold09MCMCData$proc.GO), no.time.pnts, no.time.pnts,8,2))
v.SS.Al = array(0, c(no.time.pnts , no.time.pnts , 2))
v.beta.i = array(0, c(length(Wollbold09MCMCData$proc.GO), no.time.pnts, nsim))
v.alfa.i = array(0, c(length(Wollbold09MCMCData$proc.GO), no.time.pnts, nsim))
v.pi.a.i = matrix(0, length(Wollbold09MCMCData$proc.GO), nsim)
v.rho.a = rep(0, nsim)
v.beta.i[, , 1] = rnorm(length(Wollbold09MCMCData$proc.GO), 0, 1)
v.alfa.i[, , 1] = rnorm(length(Wollbold09MCMCData$proc.GO), 0, 1)
v.SS.Al[, , 1] = diag(0.1, no.time.pnts)
v.pi.a.i[, 1] = runif(length(Wollbold09MCMCData$proc.GO))
v.rho.a[1] = 0.1
L0.alfa = diag(0.005,no.time.pnts)
indexes.1 = c(0, 6, 12, 18, 24)

results.MCMC = GibbsAllFun.ts(Wollbold09MCMCData$y.mu.a, Wollbold09MCMCData$y.mu.b, Wollbold09WrkngGrps$gr
```

GibbsFun.cs

Function to compute the Gibb's sampler for the cross-sectional data design.

Description

This function obtains the posterior samples for the parameters of the linear model on the mean of the gene functional classes under a cross-sectional data design.

Usage

```
GibbsFun.cs(y.mu.a, y.mu.b, grp.sz, beta.mat, alfa.mat, sgm.y.a, sgm.y.b, sgm.alfa, rho, pi.i, m
```

Arguments

`y.mu.a`	A `MCMCData.cs` item with the preprocessed data matrix for phenotype a.
`y.mu.b`	A `MCMCData.cs` item with the preprocessed data matrix for phenotype b.
`grp.sz`	A `SizeGOAffyGrps` item containing a vector with the set sizes of the gene groups with the size desired.
`beta.mat`	Matrix for the posterior samples for the parameter Beta.
`alfa.mat`	Matrix for the posterior samples for the parameter Alpha.

`sgm.y.a`	Matrix for the posterior samples of the variance parameter of the distribution assumed on the data for phenotype a.
`sgm.y.b`	Matrix for the posterior samples of the variance parameter of the distribution assumed on the data for phenotype b.
`sgm.alfa`	Vector for the posterior samples of the variance of the prior distribution for the parameter Alpha.
`rho`	Vector for the posterior samples for the parameter Rho.
`pi.i`	Matrix for the posterior samples for the parameter Pi.
`mm`	A real number for the mean of the Beta distribution for the non-zero part for the parameter Pi.
`aa`	A real number for the shape parameter of the inverse gamma distribution of the variance parameter of the distribution assumed on the data.
`bb`	A real number for the scale parameter of the inverse gamma distribution of the variance parameter of the distribution assumed on the data.
`aa.pi`	A real number for the precision parameter of the Beta distribution for the non-zero part for the parameter Pi.
`apriori.diff.exp`	An integer for the expected number of apriori differentially expressed gene groups.
`nsim`	An integer defining the total number of iterations for the Gibb's sampler.
`burn.in`	An integer defining the number of iterations that define the burn-in period for the posterior samples generated.
`often`	An integer defining the frequency of the iterations to print the number of differentially expressed gene functional classes up to those iterations.
`prob.cut.off`	A real number defining the cut off on the probability of functional gene classes differentially expressed across phenotypes. The number of functional gene classes that pass this threshold will be printed put during the iteration procedure of the Gibb's sampler.

Details

This function calculates the posterior samples for the parameters of interest for the cross-sectional data design, where the objective is to calculate the functional gene classes that are differentially expressed across phenotypes. During the iteration process, this function prints out the number of differentially expressed gene functional classes with the desired probability, defined by prob.cut.off.

Value

This function returns a list containing the following results:

`Beta`	Matrix with the posterior samples for the parameter beta.
`Sgma.Y.A`	Matrix with the posterior samples for the variance parameter of the distribution assumed on the data for phenotype a.
`Alfa`	Matrix with the posterior samples for the parameter alpha.
`Sgma.Y.B`	Matrix with the posterior samples for the variance parameter of the distribution assumed on the data for phenotype b.
`Pi`	Vector with the posterior samples for the parameter pi.
`Sgm.alfa`	Vector with the posterior samples for the variance of the prior distribution for the parameter alpha.
`Rho`	Vector with the posterior samples for the parameter rho.

Author(s)

A. Quiroz-Zarate and John Quackenbush.

See Also

A detailed example on the use of this function is provided in the `GeneGroupAnalysis` Vignette.

Examples

```
#- vignette("GeneGroupAnalysis")
```

`GibbsNPFun.ts` *Function to compute the Gibb's sampler for the series data design.*

Description

This function obtains the posterior samples for the parameters of the linear model on the mean of the gene functional classes under a time series data design. These posterior samples are for the parameters for the model that identifies only gene functional classes that present non-parallel time course trajectories between phenotypes.

Usage

```
GibbsNPFun.ts(y.mu.a, y.mu.b, grp.sz, beta.mat, alfa.0.vec, alfa.mat, sgm.alfa, rho, pi.i, mm, a
```

Arguments

`y.mu.a`	A `MCMCData.ts` item with the preprocessed data matrix for phenotype a.
`y.mu.b`	A `MCMCData.ts` item with the preprocessed data matrix for phenotype b.
`grp.sz`	A `SizeGOAffyGrps` item containing a vector with the set sizes of the gene groups with the size desired.
`beta.mat`	Matrix for the posterior samples for the parameter Beta.
`alfa.0.vec`	Vector for the posterior samples for the baseline parameter for Alpha.
`alfa.mat`	Matrix for the posterior samples for the parameter Alpha.
`sgm.alfa`	Matrix for the posterior samples of the variance of the prior distribution for the parameter Alpha.
`rho`	Vector for the posterior samples for the parameter Rho.
`pi.i`	Vector for the posterior samples for the parameter Pi.
`mm`	A real number for the mean of the Beta distribution for the non-zero part for the parameter Pi.
`aa.pi`	A real number for the precision parameter of the Beta distribution for the non-zero part for the parameter Pi.
`lmbd`	A prior estimate for the distribution for the variance assumed on the data.
`df.lmbd`	An integer defining the degrees of freedom for the matrix apriori estimate for the distribution for the variance assumed on the data.

`lmbd.alf`	A prior estimate for the prior distribution for the variance for the parameter Alpha.
`indexes.1`	Vector of integers defining the indexes in the data that delineate the different time points for the sample identifiers (columns of the data set).
`indexes.2`	Vector of integers defining the indexes in the data that delineate the different time points for the sample identifiers (columns of the data set) of the baseline parameter of Alpha.
`num.time.pnts`	An integer defining the number of time points considered.
`num.pcnts`	An integer defining the number of subjects considered.
`apriori.diff.exp`	An integer for the expected number of apriori differentially expressed gene groups.
`nsim`	An integer defining the total number of iterations for the Gibb's sampler.
`burn.in`	An integer defining the number of iterations that define the burn-in period for the posterior samples generated.
`often`	An integer defining the frequency of the iterations to print the number of differentially expressed gene functional classes up to those iterations.
`prob.cut.off`	A real number defining the cut off on the probability of functional gene classes differentially expressed between phenotypes across the time course. The number of functional gene classes that pass this threshold will be printed put during the iteration procedure of the Gibb's sampler.

Details

This function calculates the posterior samples for the parameters of interest for the time series data design, where the objective is to calculate the functional gene classes that are differentially expressed between phenotypes across the time course. During the iteration process, this function prints out the number of differentially expressed gene functional classes with the desired probability, defined by prob.cut.off.

Value

This function returns a list containing the following results:

`Beta`	Matrix with the posterior samples for the parameter beta.
`Alfa.0`	Vector with the posterior samples for the baseline for the parameter alpha.
`Alfa`	Matrix with the posterior samples for the parameter alpha.
`Pi`	Vector with the posterior samples for the parameter pi.
`Rho`	Vector with the posterior samples for the parameter rho.

Author(s)

A. Quiroz-Zarate and John Quackenbush.

See Also

A detailed example on the use of this function is provided in the `GeneGroupAnalysis` Vignette.

Examples

```
#- vignette("GeneGroupAnalysis")
```

`MCMCData.cs`	*Function that computes the required data set objects for the Gibb's sampler under a cross-sectional data design.*

Description

This function provides several data objects that are required in the Gibb's sampler iteration procedure. The first data objects are a transformation of the original data set into a data matrix with mean gene expression measurements where the rows of the matrix correspond to functional gene classes and the columns to the original samples provided. Finally this function provides a list of GO (Gene Ontology) processes identifiers that meet with the size requirements.

Usage

```
MCMCData.cs(wrk.grps, data.grps, GO.grps, data, phenotype.a, phenotype.b)
```

Arguments

`wrk.grps`	A `SizeGOAffyGrps` item containing the list of row numbers associated to each gene functional class.
`data.grps`	A `GeneGrps2AffyGrpsFun` item containing the list of row numbers of the respecitve gene symbols for each gene functional class.
`GO.grps`	A `GeneGrps2AffyGrpsFun` item containing the list of GO gene functional classes.
`data`	Data set of interest in the form of a matrix: columns for patients, rows for Affymetrix identifiers.
`phenotype.a`	Vector of indices with the column identifiers of phenotype a.
`phenotype.b`	Vector of indices with the column identifiers of phenotype b.

Value

This function returns a list containing the following results:

`y.mu.a`	Data matrix with observations for the functional gene classes for phenotype a.
`y.mu.b`	Data matrix with observations for the functional gene classes for phenotype b.
`proc.GO`	A vector with the GO identifiers of the functional gene classes with the minimum number of elements desired.

Author(s)

A. Quiroz-Zarate and John Quackenbush.

See Also

A detailed example on the use of this function is provided in the `GeneGroupAnalysis` Vignette.

Examples

```
#- vignette("GeneGroupAnalysis")
```

`MCMCData.ts` *Function that computes the required data set objects for the Gibb's sampler under a time series data design.*

Description

This function provides several data objects that are required in the Gibb's sampler iteration procedure. The first data objects are a transformation of the original data set into a data matrix with mean gene expression measurements where the rows of the matrix correspond to functional gene classes and the columns to the original samples provided. It also provides the covariance matrix required for the posterior sample calculation of the covariance of the data. Finally this function provides a list of GO (Gene Ontology) processes identifiers that meet with the size requirements.

Usage

```
MCMCData.ts(wrk.grps, data.grps, GO.proc, data, phenotype.a, phenotype.b, indexes, num.time.pnts
```

Arguments

`wrk.grps`	A `SizeGOAffyGrps` item containing the list of row numbers associated to each gene functional class.
`data.grps`	A `GeneGrps2AffyGrpsFun` item containing the list of row numbers of the respective gene symbols for each gene functional class.
`GO.proc`	A `GeneGrps2AffyGrpsFun` item containing the list of GO gene functional classes.
`data`	Data set of interest in the form of a matrix: columns for patients, rows for Affymetrix identifiers.
`phenotype.a`	Vector of indices with the column identifiers of phenotype a.
`phenotype.b`	Vector of indices with the column identifiers of phenotype b.
`indexes`	Vector with the order in which the subjects are recorded across the time course in the data set.
`num.time.pnts`	Number of time points considered.

Value

This function returns a list containing the following results:

`y.mu.a`	Data matrix with observations for the functional gene classes for phenotype a.
`y.mu.b`	Data matrix with observations for the functional gene classes for phenotype b.
`lambda`	A priori matrix estimate for the covariance matrix to be used for the posterior sample estimates for the variance of the distribution assumed on the data.
`proc.GO`	A vector with the GO identifiers of the functional gene classes with the minimum number of elements desired.

Author(s)

A. Quiroz-Zarate and John Quackenbush.

See Also

See the `GeneGroupAnalysis` Vignette for examples on how to use this function and the help of the function `GibbsAllFun.ts` for a detailed example of its use.

Examples

```
#- For an example on the use of this function go to:
#- GibbsAllFun.ts
```

`SizeGOAffyGrps` *Function to compute gene functional classes with a desired size.*

Description

This obtains the gene functional classes with the minimum desired size as well as providing the size of such functional classes.

Usage

```
SizeGOAffyGrps(affy.GO.grps, size)
```

Arguments

`affy.GO.grps`	A `GeneGrps2AffyGrpsFun` item with a list of groups with the row number of their respective Affymetrix identifiers of the gene functional classes.
`size`	An integer defining the minimum gene functional size desired.

Value

This function returns a list containing the following results:

`groups`	A list of groups with the row numbers of the respective Affymetrix identifiers of the the functional gene classes, with the group size desired.
`group.size`	Vector with the set sizes of the groups with the size desired.

Author(s)

A. Quiroz-Zarate and John Quackenbush.

See Also

See the `GeneGroupAnalysis` Vignette for examples on how to use this function and the help of the function `GibbsAllFun.ts` for a detailed example of its use.

Examples

```
#- For an example on the use of this function go to:
#- GibbsAllFun.ts
```

Index