Package 'qusage'

October 24, 2025

Version 2.43.0

Date 2013-01-20

Title qusage: Quantitative Set Analysis for Gene Expression

Author Christopher Bolen and Gur Yaari, with contributions from Juilee Thakar, Hailong Meng, Jacob Turner, Derek Blankenship, and Steven Kleinstein

Maintainer Christopher Bolen <cbolen1@gmail.com>

Depends R (>= 2.10), limma (>= 3.14), methods

Imports utils, Biobase, nlme, emmeans, fftw

Description This package is an implementation the Quantitative Set Analysis for Gene Expression (QuSAGE) method described in (Yaari G. et al, Nucl Acids Res, 2013). This is a novel Gene Set Enrichment-type test, which is designed to provide a faster, more accurate, and easier to understand test for gene expression studies. qusage accounts for inter-gene correlations using the Variance Inflation Factor technique proposed by Wu et al. (Nucleic Acids Res, 2012). In addition, rather than simply evaluating the deviation from a null hypothesis with a single number (a P value), qusage quantifies gene set activity with a complete probability density function (PDF). From this PDF, P values and confidence intervals can be easily extracted. Preserving the PDF also allows for post-hoc analysis (e.g., pair-wise comparisons of gene set activity) while maintaining statistical traceability. Finally, while qusage is compatible with individual gene statistics from existing methods (e.g., LIMMA), a Welch-based method is implemented that is shown to improve specificity. The QuSAGE package also includes a mixed effects model implementation, as described in (Turner JA et al, BMC Bioinformatics, 2015), and a meta-analysis framework as described in (Meng H, et al. PLoS Comput Biol. 2019). For questions, contact Chris Bolen (cbolen1@gmail.com) or Steven Kleinstein (steven.kleinstein@yale.edu)

License GPL (>= 2)

URL http://clip.med.yale.edu/qusage

biocViews GeneSetEnrichment, Microarray, RNASeq, Software, ImmunoOncology

git_url https://git.bioconductor.org/packages/qusage

git_branch devel

git_last_commit 2c42099

git_last_commit_date 2025-04-15

Repository Bioconductor 3.23

Date/Publication 2025-10-24

Contents

`aggregateGeneSet` *Calculate Pathway Activation*

Description

Combine individual gene differential expresseion for each pathway

Usage

```
aggregateGeneSet(geneResults, geneSets, n.points=2^12, silent=TRUE)
```

Arguments

geneResults	A QSarray object, as generated by makeComparison
geneSets	A list of pathways to be compared. See description for more details.
n.points	The number of points at which to sample the convoluted t-distribution. See Details for more information on appropriate values for this parameter.
silent	If false, print a "." after every fifth pathway, as a way to track progress.

Details

This function convolutes individual gene t-distributions into a single PDF for each gene set.

The *geneSets* parameter can either be provided as a vector describing a single gene set, or a list of vectors representing a group of gene sets (such as the ones available from Broad's Molecular Signatures Database). Each pathway must be a character vector with entries matching the row names of *eset*. If a pathway does not contain any values matching the rownames of *eset*, a warning will be printed, and the function will return NAs for the values of that pathway.

By default the parameter *n.points* is set to 2^12, or 4096 points, which will give very accurate p-values in most cases. Sampling at more points will increase the accuracy of the resulting p-values, but will also linearly increase the amount of time needed to calculate the result. With larger sample sizes, as few as 1/4 this number of points can be used without seriously affecting the accuracy of the resulting p-values, however when there are a small number of samples (i.e. less than 8 samples total), the t-distribution must be sampled over a much wider range, and the number of points needed for sampling should be increased accordingly. It is recommended that when running `aggregateGeneSet` with less than 8 samples the number of points be increased to at least 2^15 or 2^16. It may also be useful to test higher values of this parameter, as it can often result in a much more significant p-value with small sample sizes.

The PDF for each individual gene set is generated by using numerical convolution applied to the individual gene PDFs. Briefly, a Fast Fourier Transform (FFT) is calculated for each individual gene PDF to arrive at a k-component vector. The product of each component across all of the genes is then taken to arrive at a new k-component vector for the gene set. The real part of the resulting product is then transformed back to a PDF using a reverse FFT, and assured to be normalized and centered around zero. The mean of the combined PDF is simply the mean fold change of the individual genes. The range for sampling is determined by the lowest degrees of freedom of the individual genes, such that at most 10^-8 of the cumulative distribution at the tails are excluded (i.e., assumed to be 0). For example, when nu = 3, the range is (-480,480), and when nu = 120, the range is (-6,6).

Technically, the output of this step is the PDF of the *sum* of differences in expressions over all genes in the gene set under the assumption that the genes are independent. In order to estimate the *mean* differential expression PDF, this distribution is scaled by a factor of 1/N, where N is the number of genes in the gene set. The resulting PDFs of the input gene sets are stored as a matrix in *path.PDF* slot of the returned QSarray object. However, the x-coordinates for these PDFs are not stored in the QSarray object, and must be calculated using the getXcoords function.

Value

A QSarray object.

Examples

```
##create example data
eset = matrix(rnorm(500*20),500,20, dimnames=list(1:500,1:20))
labels = c(rep("A",10),rep("B",10))

##first 30 genes are differentially expressed
eset[1:30, labels=="B"] = eset[1:30, labels=="B"] + 1

##compare the two groups
geneResults = makeComparison(eset, labels, "B-A")

##aggregate data for gene sets
geneSets = list(set1=1:30, set2=31:60)
set.results = aggregateGeneSet(geneResults, geneSets)
```

---

calcBayesCI *Calculate pathway Confidence Intervals*

Description

A function to calculate the confidence intervals for each of the gene sets in the *geneResults* object

Usage

```
calcBayesCI(QSarray,low=0.025,up=1-low,addVIF=!is.null(QSarray$vif))
```

Arguments

QSarray	A QSarray object, as generated by either makeComparison or aggregateGeneSet
low, up	the lower and upper bounds of the confidence interval.
addVIF	a logical indicating whether the VIF should be used to calculate the variance of the pathway.

Details

This function can be used to calculate a confidence interval (CI) for the gene sets in QSarray. By default, a 95% CI is calculated, with the lower and upper bounds at 0.025% and 0.975%, respectively. This function is used in plotCIs to plot the confidence intervals of each pathway.

Value

Matrix of size (2 x numPathways(QSarray)) containing the lower and upper bounds of the confidence intervals for each pathway in QSarray.

---

calcVIF *Calculate Variance Inflation Factor*

Description

A function to calculate the Variance Inflation Factor (VIF) for each of the gene sets in the *geneResults* object

Usage

```
calcVIF(eset, geneResults, useCAMERA = geneResults$var.method=="Pooled",
        useAllData = TRUE)
```

Arguments

eset	An objet of class ExpressionSet containing log normalized expression data (as created by the affy and lumi packages), OR a matrix of log2(expression values). This must be the same dataset that was used to create geneResults
geneResults	A QSarray object, as generated by either makeComparison or aggregateGeneSet
useCAMERA	The method used to calculate variance. See the description for more details.
useAllData	Boolean parameter determining whether to use all data in eset to calculate the VIF, or to only use data from the groups being contrasted. Only used if useCAMERA is set to FALSE

Details

This method calculates the Variance Inflation Factor (VIF) for each gene set in *geneSets*, which is used to correct for the correlation of genes in the gene set. This method builds off of a technique proposed by Wu et al. (Nucleic Acids Res, 2012), which calculates the VIF for each gene set based on the correlation of the genes in that set. The Wu et al. method, referred to as CAMERA, uses the linear model framework created by LIMMA to calculate gene-gene correlations, but consequently it must assume equal variance not only between all groups in the dataset, but also across each gene in the gene set. While this assumption leads to a slightly more computationally efficient VIF calculation, it is not valid for most gene sets, and its violation can greatly impact specificity.

This function provides two options for calculating the VIF: the CAMERA method established by Wu et al. (if *useCAMERA* is TRUE), or an alternative implementation of the VIF calculation (if *useCAMERA* is FALSE) which does not assume equal variance of individual groups or genes. By default, calcVIF will choose *useCAMERA* based on the options specified in makeComparison. If var.equal was set to TRUE, then by default the variance will be calculated using CAMERA.

If the internal VIF calculation is used (i.e. useCAMERA=FALSE), the parameter *useAllData* can be specified to determine which samples in *eset* should be used to calculate the VIF. By default

(useAllData=TRUE), all of the samples in *eset* will be used to calculate the VIF. If useAllData=FALSE, only the samples in *eset* which were used to generate *geneResults* will be included in the calculation. Generally, using all data will provide a more accurate esimate of the gene-gene correlations, but if the samples in *eset* are from very different conditions (e.g. different tissues or platforms), it may make more sense to limit the VIF calculation to a subset of samples.

Value

A version of *geneResults* with VIF added into the object.

Examples

```
##create example data
eset = matrix(rnorm(500*20),500,20, dimnames=list(1:500,1:20))
labels = c(rep("A",10),rep("B",10))

##a few of the genes are made to be strongly correlated
corGenes = t(apply(eset[1:30,],1,sort))
eset[1:30,] = corGenes[,sample(1:ncol(eset))]

##genes 1:60 are differentially expressed
eset[1:60, labels=="B"] = eset[1:60, labels=="B"] + 1
geneSets = list(cor.set=1:30, random.set=31:60)

##Run qusage
geneResults = makeComparison(eset, labels, "B-A")
set.results = aggregateGeneSet(geneResults, geneSets)

##calc VIF for gene sets
set.results = calcVIF(eset, set.results)

##Look at results with and without VIF
par(mfrow=c(1,2))
plotDensityCurves(set.results, addVIF=FALSE, col=1:2, main="No VIF")
plotDensityCurves(set.results, addVIF=TRUE, col=1:2, main="With VIF")
legend("topleft",legend=names(geneSets),col=1:2, lty=1)
```

---

combinePDFs *Combine PDFs from multiple QuSAGE comparisons*

Description

This function combines the results of multiple qusage runs into a single, joint PDF. The resulting PDFs will be the average of the PDFs from each individual QSarray object, weighted by the number of samples in each dataset.

Usage

```
combinePDFs(QSarrayList, n.points=2^14)
```

Arguments

QSarrayList	A list of QSarray objects, each generated from the same list of geneSets
n.points	integer; the number of points at which to sample the convoluted t-distribution. For best results, this should be about 2-4 times higher than the n.points used in the individual QSarray objects.

Details

Like aggregateGeneSet, combinePDFs uses numerical convolution to calculate the combined PDFs for individual pathways, with each individual PDF weighted by the total number of samples used in the comparison. This method is useful for meta-analysis of multiple datasets, or for a meta comparison where the difference between two QuSAGE pdfs is of interest.

The results of combinePDFs can be plotted (on a pathway-by-pathway basis) using the plotCombinedPDF function, or by simply calling "plot" on a QSarray object which contains the QSlist field.

Value

A QSarray object containing a convolution of the original QSarrays. The new QSarray object will contain an additional field, QSlist, containing the input QSarrayList.

Examples

```
##create example data - a set of 500 genes normally distributed across 40 patients
  eset = matrix(rnorm(500*40),500,40, dimnames=list(1:500,1:40))
  labels = rep(c("A","B","C","D"),each=10)

##create a number of gene sets with varying levels of differential expression.
  geneSets = list()
  for(i in 0:10){
    genes = ((30*i)+1):(30*(i+1))
    eset[genes,labels=="B"] = eset[genes,labels=="B"] + 2 + rnorm(1)
    eset[genes,labels=="D"] = eset[genes,labels=="D"] + 1 + rnorm(1)

    geneSets[[paste("Set",i)]] = genes
  }


##calculate qusage results
  qsList = lapply(c("B-A","D-C"), function(comparison){
    qusage(eset,labels, comparison, geneSets)
  })

##combine the two QSarrays
  qsComb = combinePDFs(qsList)

  plot(qsComb, path.index=1)
```

`fluExample` *Example gene expression set*

Description

This is a matrix containing microarray gene expression values taken from a publicly available dataset (GEO ID: GSE30550; Huang Y et al. PLoS Genet 2011). This dataset contains samples from 17 patients who were exposed to Influenza and had blood drawn approximately every 8 hours for a week. Patients were classified as either symptomatic (sx) or asymptomatic (asx) based on the severity of their symptoms.

The portion of the dataset included here contains a total of 15 time points, including a pre-exposure point (time 0) and points approximately every 8 hours up to 108 hours. `eset.full` is arranged by donor, and the information for each sample is contained in a table, `flu.meta`. The metadata table contains 7 columns:

`SampleID` The GEO sample IDs, matching the column names of eset.full

`Subject` the subject ID

`Hours` The hour-post-infection that the sample was collected (stored as a factor)

`Hours.Numeric` Same as above, but stored as a numeric vector

`Condition` The condition of the donor, either symptomatic (sx) or asymptomatic (asx)

`Gender` The donor gender

`Age` Age of the donor at recruitment

Usage

```
eset.full
flu.meta
```

Format

`eset.full` is a matrix of gene expression measurements, with rows of genes and columns representing samples. `flu.meta` is a data frame, with rows of samples and columns of metadata information.

Source

http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE30550

References

Huang Y et al. (PLoS Genet 2011)

fluVaccine *Gene expression sets from Flu Vaccine trials*

Description

`fluVaccine` is a list containing data from two separate Flu Vaccine treatment studies, `GSE59635` and `GSE59654`. The top-level list contains two entries, `esets` and `phenoData`.

`fluVaccine$esets` is a list containing two expression matrices. Each matrix, which is labeled with the name of the GEO dataset it originates from, contains rows of gene expression levels and columns of samples.

`fluVaccine$phenoData` is a list containing two data frames. The rows of each data frame correspond to the columns in the expression matrices, and each data frame contains the following columns:

subjectID The subject ID

responder Factor describing the subject's response to flu vaccination, either "high" or "low".

bloodDrawDay The day post-vaccination that the sample was taken.

Usage

```
fluVaccine
```

Format

`fluVaccine` is a list.

Source

http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE59635 http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE59654

References

Thakar J et al., Aging, 2015

GeneSets *Example Gene Sets*

Description

ISG.geneSet A vector contains a list of probable ISGs based on Schoggins et al. (Nature, 2011).

MSIG.geneSet A list containing a set of vectors representing the MSigDB's Canonical Pathways gene set database.

BTM.geneSet A list containing a set of vectors. Each entry describes one of the blood transcription modules (BTMs) from Li et al (Nat Immunol., 2014)

Usage

```
ISG.geneSet
MSIG.geneSets
BTM.geneSet
```

Source

http://www.broadinstitute.org/gsea/msigdb/

References

ISG: Schoggins et al. (Nature, 2011).

MSIG: Liberzon et al. (Bioinformatics, 2011)

BTM: Li et al. (Nat Immunol., 2014).

---

getXcoords *Get the X coordinates for the points of the PDF*

Description

Calculates the x-coordinates for the PDF of a given pathway.

Usage

```
getXcoords(QSarray, path.index=1, addVIF=!is.null(QSarray$vif))
```

Arguments

QSarray	A QSarray object as output by qusage (or aggregateGeneSet)
path.index	either an integer between 1 and numPathways(QSarray), or the name of the pathway to retrieve.
addVIF	a logical indicating whether to use the VIF when calculating the variance

Details

The calculation of the x-coordinates for a PDF is not straightforward, and as such they are not included in the QSarray object initially. During the numerical convolution step, the gene set PDF is calculated at a number of points (equal to `QSarray$n.points`) over a range defined by:

`c(path.mean - range, path.mean + range)`

However, the resulting PDF is actually the *sum* of the individual gene PDFs, rather than the desired *average* PDF. Therefore the range which is stored in the resulting QSarray is divided by the number of genes in the pathway, `QSarray$path.size`.

In addition, the width of the PDF can be expanded by the Variance Inflation Factor (VIF), which is equivalent to multiplying the range of the x-coordinates by the `sqrt(VIF)`. If the parameter

addVIF=TRUE, the VIF calculatd using the calcVIF method will be included in the calculation of the x-coordinates.

In general, the x-coordinates for a pathway are calculated for each point n using the following formula:

$$x_n = (-1 + \frac{2(n-1)}{N_{pts}-1}) \times r \times \sqrt{VIF} + \hat{\mu}_{path}$$

Value

A numeric vector of length QSarray$n.points.

Examples

```
##create example data
eset = matrix(rnorm(500*20),500,20, dimnames=list(1:500,1:20))
labels = c(rep("A",10),rep("B",10))

##first 30 genes are differentially expressed
eset[1:30, labels=="B"] = eset[1:30, labels=="B"] + 1
geneSets = list(diff.set=1:30, base.set=31:60)

##Run qusage
set.results = qusage(eset, labels, "B-A", geneSets)

##Plot the PDF (see also: plotDensityCurves() )
x = getXcoords(set.results, 1)
y = set.results$path.PDF[,1]
plot(x,y, type="l")
```

---

makeComparison *Compare Genes Between Two Groups*

Description

A function to calculate comparisons between groups in a dataset.

Usage

```
makeComparison(eset, labels, contrast, pairVector=NULL,
               var.equal = FALSE, bayesEstimation = TRUE,
               min.variance.factor=10^-8)
```

Arguments

`eset`	An objet of class ExpressionSet containing log normalized expression data (as created by the affy and lumi packages), OR a matrix of log2(expression values), with rows of features and columns of samples
`labels`	Vector of labels representing each column of eset
`contrast`	A string describing which of the groups in *labels* we want to compare. This is usually of the form 'trt-ctrl', where 'trt' and 'ctrl' are groups represented in *labels*.
`pairVector`	A vector of factors (usually just 1,2,3,etc.) indicating which samples are paired. This is often just a vector of patient IDs or something similar. If not provided, all samples are assumed to be independent.
`var.equal`	A logical variable indicating whether to treat the two variances as being equal. If TRUE then the pooled variance is used to estimate the variance otherwise the Welch approximation is used.
`bayesEstimation`	A logical variable. If true, use a bayesian framework to estimate the standard deviation (via limma's eBayes function).
`min.variance.factor`	A factor to add to the SDs to ensure that none are equal to 0. Only used if var.equal==FALSE or bayesEstimation==FALSE.

Details

This function is the first step in the qusage algorithm. It defines the t-distributions for each gene in the input dataset by calculating the fold change and standard deviation between two groups of samples.

There are two primary methods to compare two groups of data, based on whether variances of the genes in the two groups should be considered equal (as specified by the the parameter var.equal). If var.equal=F, the t-distributions are estimated using a Welch's formalism, which is implemented internally. Else, the LIMMA package is used to calculate the t-distribution of each gene using a pooled formalism.

A note on var.equal: LIMMA's linear model function can only be run when assuming equal variances. If var.equal==TRUE, then a linear model will be created on the entire dataset at once. One benefit of using LIMMA's pooled variance calculation is that the linear models allow for more complicated comparisons (e.g. "(A+B)-C" or similar). This may be of interest to some users, but in order to do this, you must assume equal variances between all groups.

One caveat regarding paired samples: LIMMA can not fit a linear model when the paired samples are convoluted with the groups (e.g. one set of paired (trt vs mock) samples in patients with disease, combined with a set of paired samples from healthy controls). If var.equal==TRUE, these groups must be run separately to correctly fit the model (e.g. run disease first, then healthy controls).

Value

A QSarray object.

Examples

```
##create example data
eset = matrix(rnorm(500*20),500,20, dimnames=list(1:500,1:20))
labels = c(rep("A",10),rep("B",10))

##first 30 genes are differentially expressed
eset[1:30, labels=="B"] = eset[1:30, labels=="B"] + 1

##compare the two groups
results = makeComparison(eset, labels, "B-A")

## Paired Samples
##Group A and group B are two samples from the same set of 10 patients
pairVector = c(1:10,1:10)
results.paired = makeComparison(eset, labels,"B-A",pairVector=pairVector)
```

`newQSarray` *The qusage Array Object*

Description

The constructor for the QSarray object. Should primarily be used internally by `qusage` or `makeComparison`. See `QSarray-class` for a full description of the fields in the QSarray object.

Usage

```
newQSarray(obj=NULL,
        ...
       )
```

Arguments

`obj`	The features of QSarray can be supplied as a list of objects. The objects in the list must be named appropriately. See `QSarray-class` for a description of the parameters which can be stored in the QSarray object.
`...`	The fields of the QSarray object can also be specified individually. If *obj* is specified and additional fields are provided, the parameters will be combined together into a single QSarray object, with the parameters specified by ... replacing those in *obj* (this will also produce a warning).

Details

This is the constructor for use in creating QSarray objects. This is primarily intended for internal use, but advanced users may find it useful to construct their own QSarray objects without going through the process of running qusage.

In order to create a QSarray object from scratch, the constructor requires the following three fields: `mean`, `sd`, and `dof`. All other fields can be either left blank or added after. Note that in some cases, various methods will not be able to run without more information. For a complete list of the fields that the QSarray object can contain, refer to `QSarray-class`.

`plotCIs` *Plot Pathway Mean and Confidence Intervals*

Description

Functions for plotting the mean and confidence intervals of a set of pathways.

Usage

```
plotCIs(QSarray,
        path.index=1:numPathways(QSarray),
        sort.by=c("mean","p","none"),
        lowerBound=0.025,
        upperBound=1-lowerBound,

        col=NULL,
        use.p.colors=TRUE,
        p.breaks=NULL,
        p.adjust.method = "fdr",
        addLegend=use.p.colors,
        lowerColorBar="none",
        lowerColorBar.cols=NULL,

        addGrid=TRUE,
        x.labels=NULL,
        cex.xaxis=1,
        shift=0.0,

        add=FALSE,
        ylim=NULL, xlim=NULL,
        ylab=NULL, xlab=NULL,
        main=NULL,
        sub=NULL,
        type="p",
        ...
        )
```

Arguments

`QSarray`	QSarray object

`path.index` vector describing which pathways to plot. Can either be numeric or a character vector containing the names of the pathways to plot.

`sort.by` One of c("mean","p","none") indicating the order that the pathways should be plotted in. If "none", the pathways will not be reordered, and the order specified in path.index will be maintained

`lowerBound, upperBound`
numeric indicating the lower and upper bounds of the confidence intervals. Default is for a 95% confidence interval.

`col` an optional vector indicating the color for the points. If `use.p.colors=FALSE` is specified, these colors will also be used for the error bars.

`use.p.colors` logical indicating whether error bars should be colored based on the significance of the p-value.

`p.breaks` a vector indicating where the breaks in the p-value color scheme should be. By default, breaks will be at 0.001, 0.005, 0.01, 0.05, & 0.1

`p.adjust.method`
The method to use to adjust the p-values. Must be one of the methods in `p.adjust.methods`.

`addLegend` a logical specifying if a legend for the p-value color scheme be plotted

`lowerColorBar` Options for plotting a color bar below each point. Automatically generated color bars have not yet been implemented, but custom bars can be created using the "lowerColorBar.cols" parameter.

`lowerColorBar.cols`
a vector of colors to plot as a bar below each point.

`addGrid` Should guiding dashed lines be plotted?

`x.labels` character vector of the same length as `path.index` giving the names of the pathways. By default, will use the names stored in QSarray.

`cex.xaxis` set cex parameter seperately for x axis label

`shift` a number between 0 and 1 decribing the amount to shift points with respects to the guiding lines and axis labels. Useful when `add=TRUE`

`add` logical indicating whether a new plot should be created. If `FALSE`, a new plot will be generated.

`xlim, ylim, xlab, ylab, main, sub, type, ...`
parameters to be passed on to `plot`

Details

This function uses the data produced by `aggregateGeneSet` to plot the means and confidence intervals of the gene sets in `QSarray`. By default, the gene sets will be ordered by decreasing mean, and the 95% confidence intervals of each point (as calculated by calcBayesCI) will be added. To specify a different order, `sort.by` must be set to `"none"`, and the order specified by `path.index` will be used.

The points in the plot can be optionally color-coded by the significance of the (corrected) p-values. The p-values are adjusted using R's built in `p.adjust` method, which uses the `p.adjust.method` parameter to determine the algorithm being used. The colors of the points are based on the breaks

specified in p.breaks. By default, more significant p-values will be plotted in bright red/green. If use.p.colors is specified and addLegend=TRUE, a legend describing the p-values will be added to the top left corner of the plot. Alternatively, if you want to specify the colors of the points individually, you can provide a vector of colors to the col parameter.

The plotCIs function can also add a color bar along the bottom of the plot to provide additional information about the pathways. We are currently working on implementing various metrics which can be added automatically using the lowerColorBar parameter, but in the mean time, the bar can be added manually by providing a vector of colors the same length as path.index to the lowerColorBar.cols parameter.

Examples

```
##create example data
  eset = matrix(rnorm(500*20),500,20, dimnames=list(1:500,1:20))
  labels = c(rep("A",10),rep("B",10))

  geneSets = list()

##create a number of gene sets with varying levels of differential expression.
  for(i in 0:10){
    genes = ((30*i)+1):(30*(i+1))
    eset[genes,labels=="B"] = eset[genes,labels=="B"] + rnorm(1)

    geneSets[[paste("Set",i)]] = genes
  }


##calculate qusage results
  results = qusage(eset,labels, "B-A", geneSets)

##Plot gene set CIs
  plotCIs(results)
```

---

plotCIsGenes *Plot Gene Mean and Confidence Intervals*

Description

Functions for plotting the mean and confidence intervals of the genes in a pathway.

Usage

```
plotCIsGenes(QSarray,
             path.index=1,
             gene.list=NULL,
             sort.by=NULL,
             lowerBound=0.025,
```

```
upperBound=1-lowerBound,
asBand=FALSE,

col=NULL,
addGrid=TRUE,
x.labels=NULL,
cex.xaxis=1,
shift=0.0,
pathwayCI=c("band","bar","none"),
meanCol=4,

add=FALSE,
ylim=NULL, xlim=NULL,
ylab=NULL, xlab=NULL,
main=NULL,
sub=NULL,
...
)
```

Arguments

QSarray	QSarray object
path.index	which pathway to plot. Can either be numeric or a character vector containing the names of the pathways to plot. Must be of length 1
gene.list	Character vector specifying the genes in the gene set to be plotted. If sort.by='none', the order of these genes will be used. NAs are accepted.
sort.by	one of c(mean,p,none), specifying how to order the genes. If NULL and gene.list is provided, default is "none", else, default is "mean".
lowerBound, upperBound	numeric indicating the lower and upper bounds of the confidence intervals. Default is for a 95% confidence interval.
asBand	logical indicating if CIs should be plotted as a grey band or as arrows
col	an optional vector indicating the color for the points.
addGrid	Should guiding dashed lines be plotted?
x.labels	character vector indicating the names of the genes to be plotted along the x-axis. By default, will use the names stored in QSarray, or gene.list, if specified.
cex.xaxis	set cex parameter seperately for x axis label
shift	a number between 0 and 1 decribing the amount to shift points with respects to the guiding lines and axis labels. Useful when add=TRUE
pathwayCI	A string, one of "band", "bar", or "none", determining whether to add the confidence interval for the gene set PDF to the plot. By default ("band"), a band will be plotted behind the bars for the individual genes. If "bar" is specificied, another error bar will be added before the genes' error bars. To suppress the plotting of the pathway band, specify pathwayCI="none".

meanCol	color for the line indicating the mean of the pathway. Only used if pathwayCI is either 'band' or 'bar'
add	logical indicating whether a new plot should be created. If FALSE, a new plot will be generated.
xlim, ylim, xlab, ylab, main, sub, ...	parameters to be passed on to plot. If NULL, defaults will be used.

Details

This function uses the data produced by makeComparison to plot the means and confidence intervals of the genes in an individual gene set. To only plot the means and CIs of a subset of the genes in a pathway, a list of the genes to be plotted can be specified using the gene.list parameter. By default, the genes will be ordered by decreasing mean, and the 95% confidence intervals of each point will be added. To specify a different order, sort.by must be set to "none", and the order specified by gene.list will be used.

The mean of the overall pathway will automatically be added as a dashed line (with color specified by meanCol), but information on the confidence interval of the aggregated pathway can optionally be plotted as well. If pathwayCI is set to either "band" or "bar", the mean and CI of the gene set will be added to the plot. Specifying "band" will add the CI as a band behind the individual points, whereas "bar" will add an additional point at the left side of the plot with the mean and CI of the pathway itself.

Examples

```
##create example data
   eset = matrix(rnorm(500*20),500,20, dimnames=list(1:500,1:20))
   labels = c(rep("A",10),rep("B",10))

##first 30 genes are differentially expressed for the 2 vs. 1 comparison
   diffSet = 1:30
   eset[diffSet, labels=="B"] = eset[diffSet, labels=="B"] + 1

#a second gene set of non-D.E. genes
   normSet = 31:60

   geneSets = list(diffSet=diffSet, normSet=normSet)

##calculate qusage results
   results = qusage(eset,labels, "B-A", geneSets)

##Plot gene data from first gene set
   plotCIsGenes(results, path.index=1)

##Add a bar to represent the differential expression of the gene set
   plotCIsGenes(results, path.index=1, pathwayCI="bar")
```

plotCombinedPDF *Plot combined PDF for an individual pathway*

Description

A function for plotting out the pdfs for an indivdiual pathway in the QScomb object.

Usage

```
plotCombinedPDF(QScomb,
                path.index=1,
                zeroLine=TRUE,
                comb.lwd=3,path.lwd=1,
                comb.col=par("col"),
                path.col=NULL,
                legend=FALSE,legend.labs=NULL,
                add=FALSE,
                xlim=NULL,ylim=NULL,
                xlab=NULL,ylab=NULL,
                main=NULL,
                type="l",
                ...)
```

Arguments

QScomb	QScomb object
path.index	index describing which pathway to plot. Can either be numeric or a string containing the names of the pathway to plot. Must be of length 1
zeroLine	a logical indicating whether to include a vertical line at 0.
comb.lwd	the lwd for the combined PDF curve.
path.lwd	the lwd for the individual dataset curves. Can be a vector of the same length as QScomb$QSlist specifying the lwd of each individual curve.
comb.col	the color of the combined PDF curve.
path.col	the color of the individual dataset curves. Can be a vector of the same length as QScomb$QSlist specifying the color of each individual curve.
legend	boolean; should a legend be added to the plot? Can also be a character vector specifying the location of the legend (i.e "topleft" vs "topright", etc. See legend for details)
legend.labs	character vector; the names of each dataset in QScomb$QSlist. If not provided, the function will attempt to pull the labels from QScomb$QSlist, or will use default names if those are undefined.
add, xlim, ylim, xlab, ylab, main, type, ...	parameters to be passed on to plot

Details

This function uses the data produced by combinePDFs to plot both the individual dataset PDFs and the combined PDF for a single pathway. By default, `plotCombinedPDF` will plot the PDFs for the first pathway in QScomb, but this behavior can be controlled by the `path.index` parameter.

Examples

```
##create 5 example datasets of different sizes
esets = lapply(1:5, function(i){
  n = 20 + i*5
  eset = matrix(rnorm(500*n),500,n, dimnames=list(1:500,1:n))

  labels = c(rep("A",10+5*floor(i/2)),
             rep("B",10+5*ceiling(i/2))
            )

  ##genes 1:30 are differentially expressed
  eset[1:30, labels=="B"] = eset[1:30, labels=="B"] + rnorm(30,rnorm(1,0.5,0.5),1)

  return(list(eset=eset, labels=labels))
})

##gene sets
geneSets = list(diff.set=1:30, baseline.set=31:60)

##Run qusage on each dataset
set.results = lapply(esets, function(dat){
  qusage(dat$eset, dat$labels, "B-A", geneSets)
})

##run the combinePDFs function
combined = combinePDFs(set.results)

##plot the combined PDF result for "diff.set"
plotCombinedPDF(combined, path.index="diff.set")
```

---

plotDensityCurves *Plot gene set PDFs*

Description

A function for plotting out the pdfs of a set of pathways.

Usage

```
plotDensityCurves(QSarray,
                  path.index=1:numPathways(QSarray),
```

```
                    zeroLine=TRUE,
                    addVIF=!is.null(QSarray$vif),
                    col=NULL,
                    plot=TRUE,
                    add=FALSE,
                    xlim=NULL,ylim=NULL,
                    xlab=NULL,ylab=NULL,
                    type="l",
                    ...)
```

Arguments

QSarray	QSarray object
path.index	vector describing which pathways to plot. Can either be numeric or a character vector containing the names of the pathways to plot.
zeroLine	a logical indicating whether to include a vertical line at 0.
addVIF	a logical indicating whether the VIF should be used to calculate the variance of the pathway.
col	the color of the curves. Can be a vector of the same length as path.index specifying the color of each individual curve.
plot	Logical indicating whether to create the plot. If FALSE, only the coordinates for the plot will be returned, and no new plot will be created.
add, xlim, ylim, xlab, ylab, type, ...	parameters to be passed on to plot

Details

This function uses the data produced by aggregateGeneSet to plot the PDFs of the pathways in QSarray. By default, plotDensityCurves will plot a curve for each pathway in the QSarray pathway, but this behavior can be controlled by the path.index parameter. For the best plots, it is suggested that you limit the number of curves plotted to below ten.

Value

Invisibly returns a list of the same length as path.index, where each entry is a matrix of x- and y-coordinates for that pathway.

Examples

```
 ##create example data
eset = matrix(rnorm(500*20),500,20, dimnames=list(1:500,1:20))
labels = c(rep("A",10),rep("B",10))

##genes 1:30 are differentially expressed
eset[1:30, labels=="B"] = eset[1:30, labels=="B"] + 1
geneSets = list(diff.set=1:30, base.set=31:60)
```

```
##Run qusage
set.results = qusage(eset, labels, "B-A", geneSets)

##Plot results
plotDensityCurves(set.results)

##plot just the first curve with a different color
plotDensityCurves(set.results, path.index=1, col=2, lwd=2)

##plot the CDFs of the curves
coords = plotDensityCurves(set.results, plot=FALSE)
plot(0, type="n", xlim=c(-1,2),ylim=c(0,1),xlab="x",ylab="CDF")
for(i in 1:length(coords)){
  points = coords[[i]]
  x = points$x
  y = cumsum(points$y)/sum(points$y)
  lines(x,y,col=i)
}
```

`plotGeneSetDistributions` *Plot gene and gene set PDFs*

Description

A function for plotting out the pdfs of all the genes in a gene set

Usage

```
plotGeneSetDistributions(QSarray1, QSarray2=NULL,
                         path.index=1,
                         colorScheme="sdHeat",
                         alpha=1,
                         normalizePeaks=FALSE,

                         addBarcode=TRUE,
                         barcode.col=NULL,
                         barcode.hei=0.2,

                         groupLabel=NULL,
                         labelLoc="left",
                         xlab="Activity",
                         ylab=NA,
                         main=NULL,
                         lwds=c(1,3),
                         cex=1,
                         ...)
```

Arguments

QSarray1, QSarray2	QSarray objects containing PDFs of a gene set
path.index	either an integer between 1 and numPathways(QSarray), or the name of the pathway to retrieve. This can be of length 1 or 2 to specify different gene sets for the top and bottom plot (see details)
colorScheme	This parameter specifies the color scheme to be used when plotting the individual gene PDFs. This can either be one of c("rainbow", "sdHeat") for a customized color scheme, or a vector of colors of the same length as the gene set. See the details section for more information.
alpha	numeric value between 0 and 1 specifying the alpha channel for the individual gene curves. Only used if colorScheme is set to one of "rainbow" or "sdHeat"
normalizePeaks	logical indicating whether curve heights will be normalized to the same value.
addBarcode	logical indicating whether a barcode-style plot should be added below the PDFs representing the means activity of each individual gene.
barcode.col	The color used for the bars of the barcode plot. Can be a vector of colors, or a single color which is repeated for each bar in the plot.
barcode.hei	a numeric value specifying the height of the barcode plot relative to the size of the PDF plot.
groupLabel	Vector of labels for the individual plots. If left blank, labels will be generated automatically.
labelLoc	vector of length 1 or 2 determining the location on the plot of where to put the label. One of "left","center", or "right"
lwds	a numeric vector of length 2 specifying the lwd parameters for the gene and gene set curves, respectively.
xlab, ylab, main, cex, ...	parameters to be passed on to plot

Details

The plotGeneSetDistribution function is designed to provide a quick and intuitive look at how individual genes contribute to the overall expression of a gene set. This function plots the PDFs of each individual gene in a gene set alongside the convoluted PDF of those genes. In addition, a barcode plot representing the location of the mean fold change of each individual gene is added by default below the plot. The appearance of the curves can be controlled by the colorScheme and alpha parameters, and the barcode plot by addBarcode, barcode.col, and barcode.hei.

The default colorScheme, sdHeat, will automatically color-code the gene PDFs by their standard deviations, with hotter colors being used for smaller standard deviations. This, along with colorScheme="rainbow", are the only automatic color schemes, but colorScheme also accepts custom colors. This can be a vector of colors in any format accepted by par(col). If the vector provided is shorter than the number of genes in the gene set, the vector will be repeated. NOTE: The order that the colors are used in is not the same as the order of genes in the original gene set. All gene sets are reordered when they are stored in the QSarray$pathways slot, and the vector provided to colorScheme will be used in this order. This also applies to any colors provided to barcode.col

By default, the first pathway in the QSarray object will be plotted. If you wish to change this parameter, you can provide an alternatve pathway using the `path.index` parameter. This can either be an integer between 1 and `numPathways(QSarray1)`, or it can be a string representing the name of the pathway.

The `plotGeneSetDistribution` function can also be used to compare the results from two different pathways or datasets. In order to analyze two different pathways from the same QSarray object, you can provide a `path.index` parameter of length 2 representing the two pathways to be compared. Alternatively, a separate QSarray object can be provided as the parameter `QSarray2`, and the second plot will be drawn from this object. If `QSarray2` is provided and `path.index` is of length 2, the second `path.index` will be drawn from `QSarray2`.

Examples

```
 ##create example data
   eset = matrix(rnorm(500*20),500,20, dimnames=list(1:500,1:20))
   labels = c(rep("A1",5),rep("A2",5),rep("B1",5),rep("B2",5))

##first 30 genes are differentially expressed much more strongly in group "B" than in group "A"
   geneSet = 1:30
   eset[geneSet, labels=="A2"] = eset[geneSet, labels=="A2"] + 1
   eset[geneSet, labels=="B2"] = eset[geneSet, labels=="B2"] + 2

##calculate qusage results
   A.results = qusage(eset,labels, "A2-A1", geneSet)
   B.results = qusage(eset,labels, "B2-B1", geneSet)

##plot the gene set distribution for group A and group B side-by-side
  plotGeneSetDistributions(A.results,B.results)

##add labels to the right side of the plots
 plotGeneSetDistributions(A.results,B.results,groupLabel = c("A2-A1", "B2-B1"), labelLoc="right")

##change the colors of the curves
  plotGeneSetDistributions(A.results,B.results, colorScheme="rainbow")
```

`pVal` *Calculate p-values for gene set activity*

Description

Methods for calculating the significance of gene set activity, compared either to a null hypothesis (`pdf.pVal`), or to a separate PDF (`twoCurve.pVal`).

Usage

```
pdf.pVal(QSarray, alternative=c("two.sided","less","greater"),
         direction=FALSE, addVIF=!is.null(QSarray$vif), selfContained=TRUE)
```

```
twoCurve.pVal(grp1, grp2, path.index1 = 1:numPathways(grp1), path.index2 = 1:numPathways(grp2),
              alternative=c("two.sided","less","greater"), direction=FALSE,
              addVIF=!(is.null(grp1$vif) | is.null(grp2$vif)))
```

Arguments

QSarray, grp1, grp2	A QSarray object as output by qusage (or aggregateGeneSet)
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less". You can specify just the initial letter.
direction	a logical indicating whether the p-values should be signed (i.e. negative fold changes return negative p-values). Ignored if alternative!="two.sided".
addVIF	a logical indicating whether to use the VIF when calculating the variance
selfContained	a logical indicating whether the test should be self-contained or competitive. See details for more information.
path.index1, path.index2	numeric vectors indicating which gene sets in grp1 to compare to grp2. The length of path.index1 and path.index2 must match.

Details

The pVal functions are designed to estimate the level of significance for the gene set activity cacluated using qusage. Because the QSarray object contains gene set information stored as a Probability Density Function (PDF), we can determine significance of an individual gene set using the pdf.pVal function by comparing the PDF to our null hypothesis (zero by default. See below). If alternative="greater", pdf.pVal tests whether the fold change of the gene set is greater than the null mean, and the p-value is calcuated based on the proportion of the lower tail of the PDF which is below the null hypothesis.

There are two options for the null hypothesis in this method, controlled by the logical parameter "selfContained". By default, pdf.pVal performs a self-contained test, where the null hypothesis is that the mean fold change is 0. If selfContained=FALSE is specified, pdf.pVal instead performs a competitive test, where the null hypothesis is the mean fold change of all genes which are not in the pathway.

An individual gene set's PDF can also be compared with a second PDF, created from either comparing a different set of samples or using a different gene set, using the twoCurve.pVal function. This function takes two QSarray objects, grp1 and grp2, and by default compares the PDFs for each gene set in the two QSarray objects in order. However, this behavior can be controlled by the path.index1 and path.index2 parameters, which are numeric vectors specifying which gene sets should be compared. The two vectors must be the same length, and the first index in path.index1 will be compared with the first index in path.index2 and so on.

Value

A vector of p-values for each gene set in QSarray, or for each gene set specified with path.index when using twoCurve.pVal.

Examples

```
##create example data
   eset = matrix(rnorm(500*20),500,20, dimnames=list(1:500,1:20))
   labels = c(rep("A1",5),rep("A2",5),rep("B1",5),rep("B2",5))

   geneSets = list()

##first 30 genes are differentially expressed for the 2 vs. 1 comparison
   geneSets[["simple.diffSet"]] = 1:30
   eset[geneSets[[1]], labels=="A2"] = eset[geneSets[[1]], labels=="A2"] + 1
   eset[geneSets[[1]], labels=="B2"] = eset[geneSets[[1]], labels=="B2"] + 1

##second set of 30 genes different in only group B
   geneSets[["complex.diffSet"]] = 31:60
   eset[geneSets[[2]], labels=="B2"] = eset[geneSets[[2]], labels=="B2"] + 1

#a third gene set of non-D.E. genes
   geneSets[["normSet"]] = 61:90

##calculate qusage results
   A.results = qusage(eset,labels, "A2-A1", geneSets)
   B.results = qusage(eset,labels, "B2-B1", geneSets)

##calculate p-values for initial comparison
   pdf.pVal(A.results)
   pdf.pVal(B.results)

##compare the pdfs of the two groups
  twoCurve.pVal(A.results,B.results)
```

---

qgen *Run qusage while incoprating generalized least squares and linear mixed model analysis at the gene level to account for repeated measures, continous covariates, and confounder adjusting.*

Description

A wrapper function for the three primary steps in the qusage algorithm. The first step replaces the conventional t-test framework, with additional flexibility to incorporate general linear models from the nlme package, specifically the lme and gls function calls.

Usage

```
qgen(eset,design,fixed,geneSets,
     contrast.factor,contrast,
     random=NULL,correlation=NULL,design.sampleid=NULL)
```

Arguments

`eset`	A matrix of log2(expression values), with rows of features and columns of samples.
`design`	A data frame consisting of sample annotation information such as the various fixed effects that wished to be included in modeling and subject identifiers for the inclusion of random effects.It is recommended that the design file have a column which consists of the column names of *eset*. See *design.sampleid*
`fixed`	A one-sided linear formula object describing the fixed-effects part of the model. The formula should always begin with a ~ operator followed by the fixed effect terms, seperated by the + operator.
`geneSets`	Either a list of pathways to be compared, or a vector of gene names representing a single gene set. See Description for more details.
`contrast.factor`	A one sided formula indicating the factor in which the user whishes to create a contrast. ~X1 indicates that a differences between the levels of X1 are of interest, where ~X1*X2 indicates that a difference between the level combinations of the the two factors are of interest. Factors included in the formula must be included in *fixed*.
`contrast`	A character string indicating which specific levels of *contrast.factor* the user wishes to compare. This is usually of the form "TrtA - TrtB". For contrast involving level combinations of the form ~X1*X2, the levels are concatonated ie "TrtADay1 - TrtADay0". The order of concatenation must conform to the order in *contrast.factor*.
`random`	An optional formula to specify random effects and is passed directly to the lme function. For simple repeated measures study designs, the form is usually ~ 1|g where g specifies the donor or subject id's. Defaults to NULL.
`correlation`	An optional corStruct object to specify within-group correlation structure through the residual covariance matrix and is passed directly to the lme or gls functions. Defaults to NULL.
`design.sampleid`	A character string indicating a variable name in *design* indicating the column names of *eset*. If set to NULL, it is assumed that the column ordering in *eset* corresponds to the row ordering in *design*. Defaults to NULL.

Details

This function runs the entire qusage method on the input data, returning a single QSarray object containing the results. Rather than conducting gene level Welch's or paired t-tests, the user can specify more general linear models inherent to lme and gls functions within the nlme package. This requires the specification of a design file to link the necessary covariates and repeated measures information. One consideration when running more complex linear models is that complexity combined with poorly behaved data can sometimes yield convergence issues during the optimization step. We encourage users to run their models on individual genes through gls and lme directly to ensure their models are paramaterized correctly and as expected. Additional filtering of probes may need to be conducted for genes that have little variation as this can cause convergence issues.

Gene sets are commonly obtained from online databases such as Broad's Molecular Signatures Database. Gene set lists can be obtained from these sites in the form of .gmt files, which can be read into R using the read.gmt function. Once the data has been read into R, the information can be passed into the qusage function as either a vector describing a single gene set, or a list of vectors representing a group of gene sets. Each pathway must be a character vector with entries matching the row names of *eset*. If a pathway does not contain any values matching the rownames of *eset*, a warning will be printed, and the function will return NAs for the values of that pathway.

Value

A QSarray object.

Examples

```
##Creating a design file of 20 patients (10 in conditionA/10 in conditionB
# with 5 timepoints)
# In addition, we also create a dummy continous covariate that is partially confounded
# with condition. Condition B will have a higher mean for the covariate than condition A.

des<-data.frame(SampleID=paste("Sample",1:100,sep=""),
                Condition=rep(c("A","B"),each=50),
                Donor=rep(letters[1:20],each=5),
                Time=rep(paste("D",0:4,sep=""),20),
                stringsAsFactors=TRUE)


##Create example data - a set of 500 genes normally dstributed across 20 patients
##with 5 timepoints
eset = matrix(rnorm(500*100),500,100, dimnames=list(1:500,1:100))
colnames(eset)= paste("Sample",1:100,sep="")

##create a number of gene sets with varying levels of differential expression between
##conditions and between timepoints
geneSets = list()
for(i in 0:10){
  genes = ((30*i)+1):(30*(i+1))
  eset[genes,des$Condition=="B"] = eset[genes,des$Condition=="B"] + rnorm(1)
  eset[genes,des$Time=="D1"]=eset[genes,des$Time=="D1"]+rnorm(1)

  geneSets[[paste("Set",i)]] = genes
}

##Adding additional subject specific variability to generate repeated measures correlations
for(i in 1:500){
  eset[i,]<-eset[i,]+rep(rnorm(20,0,4),each=5)
}



##Running a linear mixed model to test for D1 vs D0 for Condition B, with a subject
## specific random effect
```

```
qs.result<-qgen(eset,des,geneSets=geneSets,
                fixed= ~Condition+Time+Time*Condition,random=~1|Donor,
                contrast.factor=~Condition*Time,contrast="BD1-BD0",
                design.sampleid="SampleID")

plot(qs.result)
```

QSarray-class *Class* "QSarray"

Description

A list-based class which contains the results of running qusage. Generally created by qusage or makeComparison

Objects from the Class

QSarray objects should not be created directly, but rather through the makeComparison function. They can also be created manually via a call to the newQSarray function, although this should be done by advanced users only.

Components

QSarray objects do not contain any slots (apart from .Data) but they should contain the following list components:

mean	numeric vector containing mean fold changes for individual genes
SD	numeric vector of standard deviations for individual genes
dof	numeric vector. Degrees of Freedom for each gene
var.method	one of ("Welch's","Pooled"), indicating the method used to calculate the variance
sd.alpha	The factor each sd is multiplied by (either due to the min.variance.factor parameter in makeComparison or bec
labels	The labels as input in makeComparisons, describing the group structure of the data.
pairVector	A vector indicating which samples should be treated as pairs.
contrast	A string describing which of the two groups in labels was compared.

The following additional components are appended to the object by running aggregateGeneSet and calcVIF

pathways	the list of genes in each gene set. Represented as a list of indices.
path.mean	vector describing the mean fold change for each of the pathways provided to AggregateGeneSet
path.PDF	Matrix describing the probability distributions for each of the pathways provided to AggregateGeneSet, where e
path.size	numeric vector containing the number of features in each pathway that mapped to the input data.
ranges	the (uncorrected) range that all PDFs were calculated over. If the VIF is not used to correct the range, the x-coor
n.points	The number of points that the PDF was calculated at. This is equal to the number of rows in path.PDF
vif	the Variance Inflation Factor for each pathway, as calculated by calcVIF

Methods

newQSarray The constructor for the QSarray object. Should primarily be used internally by `qusage` or `makeComparison`. See `newQSarray` for additional details.

numFeatures Returns the number of features (i.e. genes or probesets) in the dataset

numPathways Returns the number of pathways provided to `aggregateGeneSet`

dim dimensions of the QSarray object, as c(numFeatures, numPathways)

print, head Prints a summarized version of all fields in the QSarray object.

summary Prints a brief summary of the QSarray object.

plot Plots the information stored in QSarray by either calling `plotDensityCurves` (if `numPathways` < 10) or `plotCIs` (if `numPathways` >= 10)

qsTable Print a table with a summary of the information on the most significant gene sets in QSarray. See `qsTable` for more details.

Author(s)

Christopher Bolen

---

qsTable *Summary of QSarray Results*

Description

Print a table with a summary of the information on the most significant gene sets in QSarray.

Usage

```
qsTable(QSarray, number=20, sort.by=c("fdr","p","logFC"))
```

Arguments

`QSarray`	A QSarray object
`number`	The number of gene sets to include in the table
`sort.by`	character vector; a list of metrics to be used to sort the gene sets in QSarray. Can be any combination and order of `c("fdr","p","logFC")`, or `NULL` to specify no re-ordering of gene sets.

Details

This method will return a table with a summary of the results of qusage.

Value

A data frame containing the following columns:

- pathway.name - The name of the pathway
- log.fold.change - Average log2 fold change value of the genes in the pathway
- p.Value - The p-value for the gene set, as calculated using pdf.pVal
- FDR - The Benjamini-Hochberg False Discovery rate. Calculated using R's built-in p.adjust method.

Examples

```
 ##create example data
  eset = matrix(rnorm(500*20),500,20, dimnames=list(1:500,1:20))
  labels = c(rep("A",10),rep("B",10))

  geneSets = list()

##create a number of gene sets with varying levels of differential expression.
  for(i in 0:10){
    genes = ((30*i)+1):(30*(i+1))
    eset[genes,labels=="B"] = eset[genes,labels=="B"] + rnorm(1)

    geneSets[[paste("Set",i)]] = genes
  }


##calculate qusage results
  results = qusage(eset,labels, "B-A", geneSets)

  qsTable(results)

##show the first 5 sets, sorted by log fold change
  qsTable(results, number=5, sort.by="logFC")
```

---

qusage *Run qusage on an expression dataset*

Description

A wrapper function for the three primary steps in the qusage algorithm

Usage

```
qusage(eset, labels, contrast, geneSets, pairVector=NULL,
       var.equal=FALSE, filter.genes=FALSE, n.points=2^12)
```

Arguments

eset	An objet of class ExpressionSet containing log normalized expression data (as created by the affy and lumi packages), OR a matrix of log2(expression values), with rows of features and columns of samples
labels	Vector of labels representing each column of eset
contrast	A string describing which of the groups in 'labels' we want to compare. This is usually of the form 'trt-ctrl', where 'trt' and 'ctrl' are groups represented in 'labels'
geneSets	Either a list of pathways to be compared, or a vector of gene names representing a single gene set. See Description for more details.
pairVector	A vector of factors (usually just 1,2,3,etc.) describing the sample pairings. This is often just a vector of patient IDs or something similar. If not provided, all samples are assumed to be independent.
var.equal	A logical variable indicating whether to treat the two variances as being equal. If TRUE then the pooled variance is used to estimate the variance otherwises the Welch approximation is used.
filter.genes	A boolean indicating whether the genes in eset should be filtered to remove genes with low mean and sd.
n.points	The number of points at which to sample the convoluted t-distribution. This should be increased when running qusage with a small number of samples (i.e. 6 or less in total). See `aggregateGeneSet` for more information.

Details

This function runs the entire qusage method on the input data, returning a single QSarray object containing the results of the three primary steps in the qusage algorithm: makeComparison, calcVIF, and aggregateGeneSet. Many of the parameters are left out of this function for simplicity, so for greater control each of the functions must be called separately.

Gene sets are commonly obtained from online databases such as Broad's Molecular Signatures Database. Gene set lists can be obtained from these sites in the form of .gmt files, which can be read into R using the `read.gmt` function. Once the data has been read into R, the information can be passed into the qusage function as either a vector describing a single gene set, or a list of vectors representing a group of gene sets. Each pathway must be a character vector with entries matching the row names of *eset*. If a pathway does not contain any values matching the rownames of *eset*, a warning will be printed, and the function will return NAs for the values of that pathway.

Value

A QSarray object.

Examples

```
##create example data - a set of 500 genes normally distributed across 20 patients
  eset = matrix(rnorm(500*20),500,20, dimnames=list(1:500,1:20))
  labels = c(rep("A",10),rep("B",10))

##create a number of gene sets with varying levels of differential expression.
```

```
  geneSets = list()
  for(i in 0:10){
    genes = ((30*i)+1):(30*(i+1))
    eset[genes,labels=="B"] = eset[genes,labels=="B"] + rnorm(1)

    geneSets[[paste("Set",i)]] = genes
  }


##calculate qusage results
  results = qusage(eset,labels, "B-A", geneSets)
```

`read.gmt` *Read in gene set information from .gmt files*

Description

This function reads in and parses information from the MSigDB's .gmt files. Pathway information will be returned as a list of gene sets.

Usage

```
read.gmt(file)
```

Arguments

`file`	The .gmt file to be read

Details

The .gmt format is a tab-delimited list of gene sets, where each line is a separate gene set. The first column must specify the name of the gene set, and the second column is used for a short description (which this function discards). For complete details on the .gmt format, refer to the Broad Institute's Data Format's page (url: http://www.broadinstitute.org/cancer/software/gsea/wiki/index.php/Data_formats).

Value

A list, where each index represents a separate gene set.

Warning

The function does not check that the file is correctly formatted, and may return incorrect or partial gene sets, e.g. if the first two columns are omitted. Please make sure that files are correctly formatted before reading them in using this function.

Index