Package 'piano'

November 1, 2025

Type Package

Title Platform for integrative analysis of omics data

Version 2.27.0

Date 2022-09-13

Author@R c(person(``Leif'', ``Varemo Wigge'', email = ``piano.rpkg@gmail.com'', role = ``aut''), person(``Intawat'', ``Nookaew'', email = ``piano.rpkg@gmail.com'', role = ``aut''))

Author Leif Varemo Wigge <piano.rpkg@gmail.com> and Intawat Nookaew <piano.rpkg@gmail.com>

Maintainer Leif Varemo Wigge <piano.rpkg@gmail.com>

Depends R (>= 3.5)

Imports BiocGenerics, Biobase, gplots, igraph, relations, marray, fgsea, shiny, DT, htmlwidgets, shinyjs, shinydashboard, visNetwork, scales, grDevices, graphics, stats, utils, methods

Suggests yeast2.db, rsbml, plotrix, limma, affy, plier, affyPLM, gtools, biomaRt, snowfall, AnnotationDbi, knitr, rmarkdown, BiocStyle

Description Piano performs gene set analysis using various statistical methods, from different gene level statistics and a wide range of gene-set collections. Furthermore, the Piano package contains functions for combining the results of multiple runs of gene set analyses.

License GPL (>=2)

biocViews Microarray, Preprocessing, QualityControl, DifferentialExpression, Visualization, GeneExpression, GeneSetEnrichment, Pathways

URL http://www.sysbio.se/piano

BugReports https://github.com/varemo/piano/issues

VignetteBuilder knitr

RoxygenNote 7.1.2

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

git_branch devel

git_last_commit f61c2c0

git_last_commit_date 2025-10-29

Repository Bioconductor 3.23

Date/Publication 2025-10-31

Contents

piano-package . . . 2
consensusHeatmap . . . 3
consensusScores . . . 5
diffExp . . . 7
exploreGSAres . . . 10
extractFactors . . . 11
geneSetSummary . . . 12
GSAheatmap . . . 13
GSAsummaryTable . . . 14
gsa_input . . . 15
gsa_results . . . 16
loadGSC . . . 16
loadMAdata . . . 18
networkPlot . . . 20
networkPlot2 . . . 22
polarPlot . . . 26
runGSA . . . 27
runGSAhyper . . . 32
runQC . . . 34
writeFilesForKiwi . . . 36

Index **38**

`piano-package` *Piano - Platform for Integrative ANalysis of Omics data*

Description

Run gene set analysis with various statistical methods, from different gene level statistics and a wide range of gene-set collections. Furthermore, the Piano package contains functions for combining the results of multiple runs of gene set analyses.

Details

The Piano package consists of two parts. The major part revolves around gene set analysis (GSA), and the central function for this is `runGSA`. There are some downstream functions (e.g. `GSAsummaryTable` and `geneSetSummary`) that handle the results from the GSA. By running `runGSA` multiple times with different settings it is possible to compute consensus gene set scores. Another set of functions (e.g. `consensusScores` and `consensusHeatmap`) take a list of result objects given by `runGSA` for this

step. The second part of the Piano package contains a set of functions devoted for an easy-to-use approach on microarray analysis (wrapped around the **affy** and **limma** packages), which are constructed to integrate nicely with the downstream GSA part. The starting function in this case is `loadMAdata`.

Author(s)

Leif Varemo <piano.rpkg@gmail.com> and Intawat Nookaew <piano.rpkg@gmail.com>

See Also

`runGSA` and `loadMAdata`

`consensusHeatmap` *Heatmap of top consensus gene sets*

Description

Based on multiple result objects from the `runGSA` function, this function computes the consensus scores, based on rank aggregation, for each directionality class and produces a heatmap plot of the results.

Usage

```
consensusHeatmap(
  resList,
  method = "median",
  cutoff = 5,
  adjusted = FALSE,
  plot = TRUE,
  ncharLabel = 25,
  cellnote = "consensusScore",
  columnnames = "full",
  colorkey = TRUE,
  colorgrad = NULL,
  cex = NULL
)
```

Arguments

`resList`	a list where each element is an object of class `GSAres`, as returned by the `runGSA` function.
`method`	a character string selecting the method, either `"mean"`, `"median"`, `"Borda"` or `"Copeland"`.
`cutoff`	the maximum consensus score of a gene set, in any of the directionality classes, to be included in the heatmap.

adjusted	a logical, whether to use adjusted p-values or not. Note that if runGSA was run with the argument adjMethod="none", the adjusted p-values will be equal to the original p-values.
plot	whether or not to draw the heatmap. Setting plot=FALSE allows you to save the heatmap as a matrix without plotting it.
ncharLabel	the number of characters to include in the row labels.
cellnote	a character string selecting the information to be printed inside each cell of the heatmap. Either "consensusScore", "medianPvalue", "nGenes" or "none". Note that the actual heatmap will always be based on the consensus scores.
columnnames	either "full" (default) or "abbr" to use full or abbreviated column labels. Will save some space for the heatmap if set to "abbr"
colorkey	a logical (default TRUE), whether or not to display the colorkey. Will save some space for the heatmap if turned off.
colorgrad	a character vector giving the color names to use in the heatmap.
cex	a numeric, to control the text size.

Details

This function computes the consensus gene set scores for each directionality class based on the results (gene set p-values) listed in resList, using the consensusScores function. For each class, only the GSAres objects in resList that contain p-values for that class are used as a basis for the rank aggregation. Hence, if not all classes are covered by at least 2 GSAres objects in the list, the consensusHeatmap function will not work. The results are displayed in a heatmap showing the consensus scores.

Value

A list, returned invisibly, containing the matrix of consensus scores as represented in the heatmap as well as the matrix of corresponding median p-values and the matrix of number of genes in each gene set (inlcuding the subset of up and down regulated genes for the mixed directional classes).

Author(s)

Leif Varemo <piano.rpkg@gmail.com> and Intawat Nookaew <piano.rpkg@gmail.com>

See Also

piano, runGSA

Examples

```
# Load some example GSA results:
data(gsa_results)

# Consensus heatmap:
dev.new(width=10,height=10)
consensusHeatmap(resList=gsa_results)
```

```
# Store the output:
dev.new(width=10,height=10)
ch <- consensusHeatmap(resList=gsa_results)

# Access the median p-values for gene set s1:
ch$pMat["s1",]
```

---

consensusScores *Top consensus gene sets and boxplot*

Description

Calculates the consensus scores for the gene sets using multiple gene set analysis methods (with runGSA()). Optionally also produces a boxplot to visualize the results.

Usage

```
consensusScores(
  resList,
  class,
  direction,
  n = 50,
  adjusted = FALSE,
  method = "median",
  plot = TRUE,
  cexLabel = 0.8,
  cexLegend = 1,
  showLegend = TRUE,
  rowNames = "names",
  logScale = FALSE,
  main
)
```

Arguments

resList	a list where each element is an object of class GSAres, as returned by the runGSA function.
class	a character string determining the p-values of which directionality class that should be used as significance information for the plot. Can be one of "distinct", "mixed", "non".
direction	a character string giving the direction of regulation, can be either "up" or "down".
n	consensus rank cutoff. All gene sets with consensus rank (see details below) <=n will be included in the plot. Defaults to 50.

adjusted	a logical, whether to use adjusted p-values or not. Note that if runGSA was run with the argument adjMethod="none", the adjusted p-values will be equal to the original p-values.
method	a character string selecting the method, either "mean", "median", "max", "Borda" or "Copeland".
plot	a logical, whether or not to draw the boxplot.
cexLabel	the x- and y-axis label sizes.
cexLegend	the legend text size.
showLegend	a logical, whether or not to show the legend and the indivual method ranks as points in the plot.
rowNames	a character string determining which rownames to use, set to either "ranks" for the consensus rank, "names" for the gene set names, or "none" to omit rownames.
logScale	a logical, whether or not to use log-scale for the x-axis.
main	a character vector giving an alternative title of the plot.

Details

Based on the results given by the elements of resList, preferably representing similar runs with runGSA but with different methods, this function ranks the gene sets for each GSAres object, based on the selected directionality class. Next, the median rank for each gene set is taken as a score for top-ranking gene sets. The highest scoring gene-sets (with consensus rank, i.e. rank(rankScore,ties.method="min"), smaller or equal to n) are selected and depicted in a boxplot, showing the distribution of individual ranks (shown as colored points), as well as the median rank (shown as a red line). As an alternative of using the median rank as consensus score, it is possible to choose the mean or using the Borda or Copeland method, through the method argument. A more conservative approach can also be taken using the maximum rank as a consensus score, prioritizing gene-sets that are consistently ranked high across all GSA runs.

All elements of resList have to be objects containing results for the same number of gene-sets. The ranking procedure handles ties by giving them their minimum rank.

Value

A list containing a matrix of the ranks for the top n gene sets, given by each run, as well as the corresponding matrix of p-values, given by each run.

Author(s)

Leif Varemo <piano.rpkg@gmail.com> and Intawat Nookaew <piano.rpkg@gmail.com>

See Also

piano, runGSA

Examples

```
# Load some example GSA results:
data(gsa_results)

# Consensus scores for the top 50 gene sets (in the non-directional class):
cs <- consensusScores(resList=gsa_results,class="non")

# Access the ranks given to gene set s7 by each individual method:
cs$rankMat["s7",]
```

`diffExp` *Perform differential expression analysis*

Description

Identifies differentially expressed genes by using the linear model approach of **limma**. Optionally produces a Venn diagram, heatmap, Polar plot and volcano plot.

Usage

```
diffExp(
  arrayData,
  contrasts,
  chromosomeMapping,
  fitMethod = "ls",
  adjustMethod = "fdr",
  significance = 0.001,
  plot = TRUE,
  heatmapCutoff = 1e-10,
  volcanoFC = 2,
  colors = c("red", "green", "blue", "yellow", "orange", "purple", "tan", "cyan",
    "gray60", "black"),
  save = FALSE,
  verbose = TRUE
)
```

Arguments

`arrayData`	an object of class `ArrayData`.
`contrasts`	a character vector giving the contrasts to be tested for differential expression. Use `extractFactors` to get allowed contrasts.
`chromosomeMapping`	character string giving the name of the chromosome mapping file, or an object of class `data.frame` or similar containing the chromosome mapping. Required for the Polar plot if the `ArrayData` object lacks annotation information. See details below.

fitMethod	character string giving the fitting method used by lmFit. Can be either "ls" for least squares (default) or "robust" for robust regression.
adjustMethod	character string giving the method to use for adjustment of multiple testing. Can be "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr" (default) or "none". See p.adjust for details.
significance	number giving the significance cutoff level for the Venn diagram and the horizontal line drawn in the volcano plot. Defaults to 0.001.
plot	should plots be produced? Set either to TRUE (default) or FALSE to control all plots, or to a character vector with any combination of "venn", "heatmap", "polarplot" and "volcano", to control the single plots (e.g. plot=c("venn","polarplot") or plot="heatmap").
heatmapCutoff	number giving the significance cutoff level for the heatmap. Defaults to 1e-10.
volcanoFC	number giving the x-coordinates of the vertical lines drawn in the volcano plot. Defaults to 2.
colors	character vector of colors to be used by the Venn diagram and Polar plot.
save	should the figures and p-values be saved? Defaults to FALSE.
verbose	verbose? Defaults to TRUE.

Details

This function uses **limma** to calculate p-values measuring differential expression in the given contrasts. The uniqueFactors given by extractFactors can be used to define a contrast vector, where each element should be a character string on the form "uniqueFactorA - uniqueFactorB", note the space surrounding the -. (See the example below and for extractFactors.)

If appropriate annotation is missing for the ArrayData object the user can suppply this as chromosomeMapping. This should be either a data.frame or a tab delimited text file and include the columns *chromosome* with the chromosome name and *chromosome location* containing the starting position of each gene. A - sign can be used to denote the antisense strand but this will be disregarded while plotting. The rownames should be *probe IDs* or, if using a text file, the first column with a column header should contain the *probe IDs*.

Note that the fitMethod="robust" may need longer time to run.

A Venn diagram can be drawn for up to five contrasts (diffExp() will use vennDiagram).

The heatmap shows normalized expression values of the genes that pass the heatmapCutoff in at least one contrast.

A volcano plot is produced for each contrast showing magnitude of change versus significance.

The Polar plot sorts the genes according to chromosomal location, for each chromosome starting with unknown positions followed by increasing number in the *chromosome location* column. Genes which do not map to any chromosome are listed as U for unknown. The radial lines in the Polar plot are -log10 scaled p-values, so that a longer line means a smaller p-value. This gives an overview of the magnitude of differential expression for each contrast.

Typical usages are:

```
 # Identify significantly changed genes in
'm1' and 'm2' compared to 'wt': diffExp(arrayData, contrasts=c("m1 - wt",
"m2 - wt"))
```

Value

A list with elements:

pValues	data.frame containing adjusted p-values (according to argument adjustMethod) for each contrast
foldChanges	data.frame containing log2 fold changes for each contrast
resTable	a list with an element for each contrast, each being a data.frame with full result information
vennMembers	list containing the gene members of each area of the Venn diagram (only returned when a Venn diagram is drawn)

Author(s)

Leif Varemo <piano.rpkg@gmail.com> and Intawat Nookaew <piano.rpkg@gmail.com>

References

Smyth, G. K. (2005). Limma: linear models for microarray data. In: 'Bioinformatics and Computational Biology Solutions using R and Bioconductor'. R. Gentleman, V. Carey, S. Dudoit, R. Irizarry, W. Huber (eds), Springer, New York, pages 397–420.

See Also

piano, loadMAdata, extractFactors, polarPlot, runGSA, **limma**, venn, heatmap.2

Examples

```
# Get path to example data and setup files:
dataPath <- system.file("extdata", package="piano")

# Load normalized data:
myArrayData <- loadMAdata(datadir=dataPath, dataNorm="norm_data.txt.gz", platform="yeast2")

# Perform differential expression analysis:
pfc <- diffExp(myArrayData, contrasts=c("aerobic_Clim - anaerobic_Clim",
                                        "aerobic_Nlim - anaerobic_Nlim"))

# Order the genes according to p-values, for aerobic_Clim vs anaerobic_Clim:
o <- order(pfc$resTable$'aerobic_Clim - anaerobic_Clim'$P.Value)

# Display statistics for the top 10 significant genes:
pfc$resTable$'aerobic_Clim - anaerobic_Clim'[o[1:10],]
```

exploreGSAres *Explore GSA results*

Description

Explore GSA results interactively in a web browser using shiny.

Usage

```
exploreGSAres(gsaRes, browser = TRUE, geneAnnot = NULL, genesets)
```

Arguments

gsaRes	an object of class GSAres, as returned from runGSA() or an object returned from runGSAhyper().
browser	a logical, whether or not to open the Shiny app in a browser window. Set to FALSE to open an interactive window directly in RStudio.
geneAnnot	a data.frame, containing gene annotation. The first column should be gene IDs matching those in gsares.
genesets	a character vector or list (named or un-named) of character vectors containing subsets of gene-set names that can be selected and displayed in the network plot.

Details

Additional gene-level information, e.g. alternative names or description, can be supplied via the geneAnnot argument. This information will show up in the gene table and the gene summary tabs.

Value

Does not return any object.

Author(s)

Leif Varemo <piano.rpkg@gmail.com>

See Also

piano, runGSA, GSAheatmap, networkPlot2

Examples

```
# Load example input data to GSA:
data("gsa_input")

# Load gene set collection:
gsc <- loadGSC(gsa_input$gsc)

# Run gene set analysis:
```

```
gsares <- runGSA(geneLevelStats=gsa_input$pvals , directions=gsa_input$directions,
                 gsc=gsc, nPerm=500)

# Explore results:
## Not run: exploreGSAres(gsares)
```

`extractFactors` *Extracts* `ArrayData` *factors*

Description

Extracts the factors, given by an `ArrayData` object, that can be used by `diffExp`

Usage

```
extractFactors(arrayData)
```

Arguments

`arrayData`	an `ArrayData` object.

Value

A `list` with elements:

`factors`	Assigns one factor to each array
`uniqueFactors`	The unique factors that can be used to form contrasts

Author(s)

Leif Varemo <piano.rpkg@gmail.com> and Intawat Nookaew <piano.rpkg@gmail.com>

See Also

piano, `diffExp`

Examples

```
# Get path to example data and setup files:
dataPath <- system.file("extdata", package="piano")

# Load normalized data:
myArrayData <- loadMAdata(datadir=dataPath, dataNorm="norm_data.txt.gz", platform="yeast2")

#Extract the factors that can be used in the call to diffExp:
extractFactors(myArrayData)
```

geneSetSummary *Gene set summary*

Description

Returns a summary of the statistics and gene members of a given gene set in a GSAres object.

Usage

```
geneSetSummary(gsaRes, geneSet)
```

Arguments

gsaRes	an object of class GSAres, as returned from runGSA().
geneSet	a character string giving the name of a gene-set.

Details

This function can be used to access information on specific gene sets of interest. The same results are available for all gene sets using GSAsummaryTable.

Value

A list with the elements name, containing the gene-set name, geneLevelStats, containing the gene-level statistics of the member genes, directions, containing the directions of the member genes, and stats, a table of the gene set statistics and p-values.

Author(s)

Leif Varemo <varemo@chalmers.se> and Intawat Nookaew <intawat@chalmers.se>

See Also

piano, runGSA, GSAsummaryTable

Examples

```
# Load example input data to GSA:
data("gsa_input")

# Load gene set collection:
gsc <- loadGSC(gsa_input$gsc)

# Run gene set analysis:
gsares <- runGSA(geneLevelStats=gsa_input$pvals , directions=gsa_input$directions,
                 gsc=gsc, nPerm=500)

# Get info on a specific gene set:
geneSetSummary(gsares,"s1")
```

GSAheatmap *Heatmap of top significant gene sets*

Description

This function selects the top scoring (most significant) gene sets for each directionality class and produces a heatmap plot of the results.

Usage

```
GSAheatmap(
  gsaRes,
  cutoff = 5,
  adjusted = FALSE,
  ncharLabel = 25,
  cellnote = "pvalue",
  columnnames = "full",
  colorkey = TRUE,
  colorgrad = NULL,
  cex = NULL
)
```

Arguments

gsaRes	an object of class GSAres, as returned from runGSA().
cutoff	an integer n, so that the top n gene sets (plus possible ties) in each directionality class will be included in the heatmap.
adjusted	a logical, whether to use adjusted p-values or not. Note that if runGSA was run with the argument adjMethod="none", the adjusted p-values will be equal to the original p-values.
ncharLabel	the number of characters to include in the row labels.
cellnote	a character string selecting the information to be printed inside each cell of the heatmap. Either "pvalue", "rank", "nGenes" or "none". Note that the actual heatmap will always be based on the gene set ranks.
columnnames	either "full" (default) or "abbr" to use full or abbreviated column labels. Will save some space for the heatmap if set to "abbr"
colorkey	a logical (default TRUE), whether or not to display the colorkey. Will save some space for the heatmap if turned off.
colorgrad	a character vector giving the color names to use in the heatmap.
cex	a numeric, to control the text size.

Details

This function selects the top significant gene sets in each directionality class and draws a heatmap of the results. It provides a quick summary alternative to the GSAsummaryTable function or the networkPlot.

Value

A list, returned invisibly, containing the matrix of p-values (adjusted or non-adjusted depending on the settings) as represented in the heatmap as well as the matrix of corresponding ranks and the matrix of number of genes in each gene set (inlcuding the subset of up and down regulated genes for the mixed directional classes).

Author(s)

Leif Varemo <piano.rpkg@gmail.com> and Intawat Nookaew <piano.rpkg@gmail.com>

See Also

piano, runGSA, GSAsummaryTable, networkPlot2, exploreGSAres

Examples

```
# Load example input data to GSA:
data("gsa_input")

# Load gene set collection:
gsc <- loadGSC(gsa_input$gsc)

# Run gene set analysis:
gsares <- runGSA(geneLevelStats=gsa_input$pvals , directions=gsa_input$directions,
                 gsc=gsc, nPerm=500)

# Make heatmap:
dev.new(width=10,height=10)
GSAheatmap(gsares)
```

GSAsummaryTable *Gene set analysis summary table*

Description

Displays or saves a summary table of the results from runGSA.

Usage

```
GSAsummaryTable(gsaRes, save = FALSE, file = NULL)
```

Arguments

gsaRes	an object of class GSAres, as returned from runGSA().
save	a logical, whether or not to save the table.
file	a character string giving the file name to save to.

Details

The table is by default saved as an .xls file, if `file` is unused.

Value

The summary table as a data.frame (returned invisibly if `save=TRUE`).

Author(s)

Leif Varemo <piano.rpkg@gmail.com> and Intawat Nookaew <piano.rpkg@gmail.com>

See Also

piano, runGSA, networkPlot, GSAheatmap

Examples

```
# Load example input data to GSA:
data("gsa_input")

# Load gene set collection:
gsc <- loadGSC(gsa_input$gsc)

# Run gene set analysis:
gsares <- runGSA(geneLevelStats=gsa_input$pvals , directions=gsa_input$directions,
                 gsc=gsc, nPerm=500)

# Summary table:
GSAsummaryTable(gsares)
```

---

gsa_input *Random input data for gene set analysis*

Description

This data set is completely randomly generated and contains p-values for 2000 genes, fold-changes for those genes and a gene set collection giving the connection between genes and 50 gene sets. Only intended to be used as example data for runGSA.

Format

A list containing 3 elements: gsa_input$pvals and gsa_input$directions are numeric vectors, gsa_input$gsc is a two-column matrix with gene names in the first column and gene set names in the second.

gsa_results *Gene set analysis result data*

Description

This data set contains gene set analysis results, as returned by the runGSA function, that is used as example data for downstream functions. The input data to runGSA was randomly generated and is accessible through data(gsa_input).

Format

A list where each element is an object of class GSAres, as returned by runGSA.

loadGSC *Load a gene set collection*

Description

Load a gene set collection, to be used in runGSA, in GMT, SBML or SIF format, or optionally from a data.frame.

Usage

```
loadGSC(file, type = "auto", addInfo)
```

Arguments

file	a character string, giving the name of the file containing the gene set collection. Optionally an object that can be coerced into a two-column data.frame, the first column containing genes and the second gene sets, representing all "gene"-to-"gene set" connections.
type	a character string giving the file type. Can be either of "gmt", "sbml", "sif". If set to "auto" the type will be taken from the file extension. If the gene-set collection is loaded into R from another source and stored in a data.frame, it can be loaded with the setting "data.frame".
addInfo	an optional data.frame with two columns, the first containging the gene set names and the second containing additional information for each gene set. Some additional info may load automatically from the different file types.

Details

This function is used to create a gene-set collection object to be used with runGSA.

The "gmt" files available from the Molecular Signatures Database (http://www.broadinstitute.org/gsea/msigdb/) can be loaded using loadGSC. This website is a valuable resource and contains several different collections of gene sets.

By using the functionality of e.g. the biomaRt package, a gene-set collection with custom gene names (matching the statistics used in runGSA) can easily be compiled into a two-column data.frame (column order: genes, gene sets) and loaded with type="data.frame".

If a sif-file is used it is assumed that the first column contains gene sets and the third column contains genes.

A genome-scale metabolic model in SBML format can be used to define gene sets. In this case, metabolites will be the gene sets, containing all the genes that code for enzymes catalyzing reactions in which the metabolite takes part in. In order to load an SBML-file it is required that libSBML and rsbml is installed. Note that the SBML loading is an experimental feature and is highly dependent on the version and format of the SBML file and requires it to contain gene associations for the reactions. By examining the returned GSC object it is easy to see if the correct gene sets were loaded.

Value

A list like object of class GSC containing two elements. The first is gsc, a list of the gene sets, each element a character vector of genes. The second element is addInfo, a data.frame containing the optional additional information.

Author(s)

Leif Varemo <piano.rpkg@gmail.com> and Intawat Nookaew <piano.rpkg@gmail.com>

See Also

piano, runGSA

Examples

```
# Randomly generated gene sets:
g <- sort(paste("g",floor(runif(100)*500+1),sep=""))
g <- c(g,sort(paste("g",floor(runif(900)*1000+1),sep="")))
g <- c(g,sort(paste("g",floor(runif(1000)*2000+1),sep="")))
s <- paste("s",floor(rbeta(2000,0.9,1.7)*50+1),sep="")

# Make data.frame:
gsc <- cbind(g,s)

# Load gene set collection from data.frame:
gsc <- loadGSC(gsc)
```

`loadMAdata` *Load and preprocess microarray data*

Description

Loads, preprocesses and annotates microarray data to be further used by downstream functions in the **piano** package.

Usage

```
loadMAdata(
  datadir = getwd(),
  setup = "setup.txt",
  dataNorm,
  platform = "NULL",
  annotation,
  normalization = "plier",
  filter = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

`datadir`	character string giving the directory in which to look for the data. Defaults to `getwd()`.
`setup`	character string giving the name of the file containing the experimental setup, or an object of class `data.frame` or similar containing the experimental setup. Defaults to `"setup.txt"`, see details below for more information.
`dataNorm`	character string giving the name of the normalized data, or an object of class `data.frame` or similar containing the normalized data. Only to be used if the user wishes to start with normalized data rather then CEL files.
`platform`	character string giving the name of the platform, can be either `"yeast2"` or `NULL`. See details below for more information.
`annotation`	character string giving the name of the annotation file, or an object of class `data.frame` or similar containing the annotation information. The annotation should consist of the columns *Gene name*, *Chromosome* and *Chromosome location*. Not required if `platform="yeast2"`.
`normalization`	character string giving the normalization method, can be either `"plier"`, `"rma"` or `"mas5"`. Defaults to `"plier"`.
`filter`	should the data be filtered? If `TRUE` then probes not present in the annotation will be discarded. Defaults to `TRUE`.
`verbose`	verbose? Defaults to `TRUE`.
`...`	additional arguments to be passed to `ReadAffy`.

Details

This function requires at least two inputs: (1) data, either CEL files in the directory specified by `datadir` or normalized data specified by `dataNorm`, and (2) experimental setup specified by `setup`.

The setup shold be either a tab delimited text file with column headers or a `data.frame`. The first column should contain the names of the CEL files or the column names used for the normalized data, please be sure to use names valid as column names, e.g. avoid names starting with numbers. Additional columns should assign attributes in some category to each array. (For an example run the example below and look at the object `myArrayData$setup`.)

The **piano** package is customized for yeast 2.0 arrays and annotation will work automatically, if the cdfName of the arrays equals *Yeast_2*. If using normalized yeast 2.0 data as input, the user needs to set the argument `platform="yeast2"` to tell the function to use yeast annotation. If other platforms than yeast 2.0 is used, set `platform=NULL` (default) and supply appropriate annotation by the argument `annotation`. Note that the cdfName will override `platform`, so it can still be set to `NULL` for yeast 2.0 CEL files. Note also that `annotation` overrides `platform`, so if the user wants to use an alternative annotation for yeast, this can be done simply by specifying this in `annotation`.

The annotation should have the column headers *Gene name*, *Chromosome* and *Chromosome location*. The *Gene name* is used in the heatmap in `diffExp` and the *Chromosome* and *Chromosome location* is used by the `polarPlot`. The rownames (or first column if using a text file) should contain the *probe IDs*. If using a text file the first column should have the header *probeID* or similar. The filtering step discards all probes not listed in the annotation.

Normalization is performed on all CEL file data using one of the Affymetrix methods: PLIER (`"plier"`) as implemented by `justPlier`, RMA (Robust Multi-Array Average) (`"rma"`) expression measure as implemented by `rma` or MAS 5.0 expression measure `"mas5"` as implemented by `mas5`.

It is possible to pass additional arguments to `ReadAffy`, e.g. `cdfname` as this might be required for some types of CEL files.

Value

An `ArrayData` object (which is essentially a `list`) with the following elements:

`dataRaw`	raw data as an AffyBatch object
`dataNorm`	`data.frame` containing normalized expression values
`setup`	`data.frame` containing experimental setup
`annotation`	`data.frame` containing annotation

Depending on input arguments the `ArrayData` object may not include `dataRaw` and/or `annotation`.

Author(s)

Leif Varemo <piano.rpkg@gmail.com> and Intawat Nookaew <piano.rpkg@gmail.com>

References

Gautier, L., Cope, L., Bolstad, B. M., and Irizarry, R. A. affy - analysis of Affymetrix GeneChip data at the probe level. *Bioinformatics.* **20**, 3, 307-315 (2004).

See Also

piano, runQC, diffExp, ReadAffy, expresso, justPlier, yeast2.db

Examples

```
# Get path to example data and setup files:
dataPath <- system.file("extdata", package="piano")

# Load normalized data:
myArrayData <- loadMAdata(datadir=dataPath, dataNorm="norm_data.txt.gz", platform="yeast2")

# Print to look at details:
myArrayData
```

---

networkPlot *Gene set network plot*

Description

Draws a network with gene sets as nodes and the thickness of the edges correlating to the number of shared genes. The gene set significance is visualized as color intensities. Gives an overview of the influence of overlap on significant gene sets.

Usage

```
networkPlot(
  gsaRes,
  class,
  direction,
  adjusted = FALSE,
  significance = 0.001,
  geneSets = NULL,
  overlap = 1,
  lay = 1,
  label = "names",
  cexLabel = 0.9,
  ncharLabel = 25,
  cexLegend = 1,
  nodeSize = c(10, 40),
  edgeWidth = c(1, 15),
  edgeColor = NULL,
  scoreColors = NULL,
  main
)
```

Arguments

gsaRes	an object of class GSAres, as returned from runGSA() or an object returned from runGSAhyper().
class	a character string determining the p-values of which directionality class that should be used as significance information for the plot. Can be one of "distinct", "mixed", "non". Has to be "non" if the result from runGSAhyper() is used.
direction	a character string giving the direction of regulation, can be either "up", "down" or "both" (for class="distinct" only).
adjusted	a logical, if adjusted p-values should be used, or not. Note that if runGSA was run with the argument adjMethod="none", the adjusted p-values will be equal to the original p-values.
significance	the significance cut-off that determines which gene sets are included in the plot. Defaults to 0.001.
geneSets	a character vector of gene set names, to be included in the plot. Defaults to NULL, but if given, the argument significance will not be used.
overlap	a positive numerical. Determines the smallest number of sharing genes between two gene-sets that is needed in order to draw a line/edge between the gene-sets. Defaults to 1.
lay	a numerical between 1-5, or a layout function (see layout in the igraph package). 1-5 sets the layout to one of the five default layout for the network plot.
label	a character string, either "names" ,"numbers", "numbersAndSizes" or "namesAndSizes", determining the labels used for the nodes. The names are the gene set names, numbers is an arbritary numbered list of the gene sets used in the plot connected to the named list returned by the funtion. Sizes are the gene set sizes, e.g. the number of genes.
cexLabel	the text size of the node labels.
ncharLabel	the number of characters to include in the node labels.
cexLegend	the text size of the legend.
nodeSize	a numerical vector of length 2 giving the maximum and minimum node sizes. The node size represents the size of the gene set, and all values will be scaled to the given interval.
edgeWidth	a numerical vector of length 2 giving the maximum and minimum edge widths. The edge width represents the number of shared genes between two gene sets, and all values will be scaled to the given interval.
edgeColor	a character vector giving the colors to use for increasing edge width. Can also be set to a single color. Defaults to a gray-scale.
scoreColors	a character vector giving the colors from which the gradient used for node coloring will be created. In the case of class="distinct" and direction="both" the first half of the vector will be used for the up-regulated gene sets and the second part will be used for the down-regulated gene sets.
main	an optional character vector setting the title of the plot.

Details

In the case of class="distinct" and direction="both", the distinct directional p-values (pDistinctDirUp and pDistinctDirDn, see runGSA) will be used in combination. Using the geneSets and lay arguments, multiple comparative plots (i.e. with the same layout) can be drawn, based for instance on the output gene set list from other network plots with different directionality classes.

Value

Returns a list with two components: geneSets containing the names and numbers of the gene sets in the plot, and layout, containing the saved layout of the plot, which can be passed back to the lay argument in order to draw a subsequent plot with the same layout.

Author(s)

Leif Varemo <piano.rpkg@gmail.com> and Intawat Nookaew <piano.rpkg@gmail.com>

See Also

piano, runGSA, GSAheatmap, networkPlot2, exploreGSAres, layout

Examples

```
# Load example input data to GSA:
data("gsa_input")

# Load gene set collection:
gsc <- loadGSC(gsa_input$gsc)

# Run gene set analysis:
gsares <- runGSA(geneLevelStats=gsa_input$pvals , directions=gsa_input$directions,
                 gsc=gsc, nPerm=500)

# Network plot:
networkPlot(gsares,class="non",significance=0.01)

# Use circular layout and save the layout:
nw <- networkPlot(gsares,class="non",significance=0.01,lay=5)

# Use the saved layout to overlay the distinct-directional p-values for easy comparison.
# Note that the gene sets are now not selected based on a significance cutoff, but from a list:
networkPlot(gsares,class="distinct",direction="both",lay=nw$layout,geneSets=nw$geneSets)
```

---

networkPlot2 *Gene set network plot*

Description

Draws a network with gene sets as nodes and the thickness of the edges correlating to the number of shared genes. The gene set significance is visualized as color intensities. Gives an overview of the influence of overlap on significant gene sets. Uses package `visNetwork` for plotting.

Usage

```
networkPlot2(
  gsaRes,
  class,
  direction,
  adjusted = TRUE,
  significance = 0.001,
  geneSets = NULL,
  lay = "visNetwork",
  physics = TRUE,
  overlap = 0.1,
  label = "names",
  labelSize = 22,
  ncharLabel = 25,
  nodeSize = c(10, 40),
  edgeWidth = c(1, 15),
  edgeColor = NULL,
  scoreColors = NULL,
  naColor = "yellow",
  main,
  submain,
  seed = 1,
  maxAllowedNodes = Inf,
  shiny = FALSE
)
```

Arguments

`gsaRes`	an object of class `GSAres`, as returned from `runGSA()` or an object returned from `runGSAhyper()`.
`class`	a character string determining the p-values of which directionality class that should be used as significance information for the plot. Can be one of `"distinct"`, `"mixed"`, `"non"`. Has to be `"non"` if the result from `runGSAhyper()` is used.
`direction`	a character string giving the direction of regulation, can be either `"up"`, `"down"` or `"both"` (for `class="distinct"` only).
`adjusted`	a logical, if adjusted p-values should be used, or not. Note that if `runGSA` was run with the argument `adjMethod="none"`, the adjusted p-values will be equal to the original p-values.
`significance`	the significance cut-off that determines which gene sets are included in the plot. Defaults to 0.001.

geneSets	a character vector of gene set names, to be included in the plot. Defaults to NULL, but if given, the argument significance will be ignored.
lay	One of "visNetwork" (or "1"), "layout_nicely" (or "2"), "layout_as_star" (or "3"), "layout_with_fr" (or "4"), "layout_with_kk" (or "5"), "layout_with_sugiyama" (or "6"), "layout_in_circle" (or "7"), "layout_on_grid" (or "8"), "layout_as_tree", "layout_on_sphere", "layout_randomly", "layout_with_dh", "layout_with_gem", "layout_with_graphopt", "layout_with_lgl", "layout_with_mds"
physics	logical, whether or not to use physics simulation.
overlap	a positive numerical. Determines the smallest number or fraction of sharing genes between two gene-sets that is needed in order to draw a line/edge between the gene-sets. If >= 1, the argument is interpreted as number of genes. If between 0 and 1, the argument is interprested as the fraction of genes of the smalles gene-set in a given pair. Defaults to 0.1.
label	a character string, either "names" ,"numbers", "numbersAndSizes" or "namesAndSizes", determining the labels used for the nodes. The names are the gene set names, numbers is an arbritary numbered list of the gene sets used in the plot connected to the named list returned by the funtion (see example). Sizes are the gene set sizes, e.g. the number of genes.
labelSize	the text size of the node labels.
ncharLabel	the number of characters to include in the node labels.
nodeSize	a numerical vector of length 2 giving the maximum and minimum node sizes. The node size represents the size of the gene set, and all values will be scaled to the given interval.
edgeWidth	a numerical vector of length 2 giving the maximum and minimum edge widths. The edge width represents the number of shared genes between two gene sets, and all values will be scaled to the given interval.
edgeColor	a character vector giving the colors to use for increasing edge width. Can also be set to a single color. Defaults to a gray-scale.
scoreColors	a character vector giving the colors from which the gradient used for node coloring will be created. In the case of class="distinct" and direction="both" the first half of the vector will be used for the up-regulated gene sets and the second part will be used for the down-regulated gene sets.
naColor	the color for gene-sets when selected p-value is NA
main	an optional character vector setting the title of the plot.
submain	an optional character vector setting the subtitle of the plot.
seed	random seed for reproducible layouts
maxAllowedNodes	if the set parameters results in a network with more than maxAllowedNodes, a error if given instead of drawing the network.
shiny	Only for internal use. Set to FALSE by default.

Details

In the case of class="distinct" and direction="both", the distinct directional p-values (pDistinctDirUp and pDistinctDirDn, see runGSA) will be used in combination.

Value

Returns an object of class `visNetwork` that can be further manipulated, see examples.

Author(s)

Leif Varemo <piano.rpkg@gmail.com> and Intawat Nookaew <piano.rpkg@gmail.com>

See Also

piano, runGSA, GSAheatmap, exploreGSAres

Examples

```
# Load example input data to GSA:
data("gsa_input")

# Load gene set collection:
gsc <- loadGSC(gsa_input$gsc)

# Run gene set analysis:
gsares <- runGSA(geneLevelStats=gsa_input$pvals , directions=gsa_input$directions,
                 gsc=gsc, nPerm=500)

# Network plot:
networkPlot2(gsares, class="non", significance=0.1)

# Display number to gene-set name mapping:
res <- networkPlot2(gsares, class="non", significance=0.1)
res$x$nodes[,c("id","geneSetNames")]

# Examples of reusing res later:

# Draw same again:
require(visNetwork)
visNetwork(res$x$nodes,res$x$edges)
# os simly just:
res

# Draw only essential, rest is default:
visNetwork(res$x$nodes[,c("id","label")],res$x$edges[,c("from","to")])

# Add custom options:
visNetwork(res$x$nodes[,c("id","label")],res$x$edges[,c("from","to")]) %>%
visIgraphLayout("layout_in_circle")

# Other example:
res %>% visNodes(shadow=FALSE)

# See package visNetwork for more examples
```

polarPlot *Polar plot*

Description

Produces a Polar plot, mapping p-values to chromosome location. This function is used by diffExp.

Usage

```
polarPlot(
  pValues,
  chromosomeMapping,
  colors = c("red", "green", "blue", "yellow", "orange", "purple", "tan", "cyan",
    "gray60", "black"),
  save = FALSE,
  verbose = TRUE
)
```

Arguments

pValues	a data.frame containing p-values for different contrasts in different columns. Column names are used as contrast names. Maximum number of columns allowed are ten.
chromosomeMapping	character string giving the name of the chromosome mapping file, or an object of class data.frame or similar containing the chromosome mapping. See details below.
colors	character vector of colors to be used by the Polar plot.
save	should the figures be saved? Defaults to FALSE.
verbose	verbose? Defaults to TRUE.

Details

This function is mainly used by diffExp but can also be used separately by the user.

The argument chromosomeMapping should be either a data.frame or a tab delimited text file and include the columns *chromosome* with the chromosome name and *chromosome location* containing the starting position of each gene. A - sign can be used to denote the antisense strand but this will be disregarded while plotting. The rownames should be *probe IDs* or, if using a text file, the first column with a column header should contain the *probe IDs*. If relying on an ArrayData object (called arrayData) and containing an annotation field, the chromosomeMapping can be set to arrayData$annotation[,c(2,3)] (see the example below).

The Polar plot sorts the genes according to chromosomal location, for each chromosome starting with unknown positions followed by increasing number in the *chromosome location* column. Genes which do not map to any chromosome are listed as U for unknown. The radial lines in the Polar plot are -log10 scaled p-values, so that a longer line means a smaller p-value. This gives an overview of the magnitude of differential expression for each contrast.

Value

Does not return any object.

Author(s)

Leif Varemo <piano.rpkg@gmail.com> and Intawat Nookaew <piano.rpkg@gmail.com>

See Also

piano, diffExp, radial.plot

Examples

```
# Get path to example data and setup files:
dataPath <- system.file("extdata", package="piano")

# Load normalized data:
myArrayData <- loadMAdata(datadir=dataPath, dataNorm="norm_data.txt.gz", platform="yeast2")

# Perform differential expression analysis:
pfc <- diffExp(myArrayData, plot=FALSE,
               contrasts=c("aerobic_Clim - anaerobic_Clim",
                           "aerobic_Nlim - anaerobic_Nlim"))

# Get chromosome mapping from myArrayData:
chrMap <- myArrayData$annotation[,c(2,3)]
# Get p-values from pfc
pval <- pfc$pValues
# Draw the polar plot:
polarPlot(pval, chromosomeMapping=chrMap)
```

---

runGSA *Gene set analysis*

Description

Performs gene set analysis (GSA) based on a given number of gene-level statistics and a gene set collection, using a variety of available methods, returning the gene set statistics and p-values of different directionality classes.

Usage

```
runGSA(
  geneLevelStats,
  directions = NULL,
  geneSetStat = "mean",
  signifMethod = "geneSampling",
```

```
  adjMethod = "fdr",
  gsc,
  gsSizeLim = c(1, Inf),
  permStats = NULL,
  permDirections = NULL,
  nPerm = 10000,
  gseaParam = 1,
  ncpus = 1,
  verbose = TRUE
)
```

Arguments

geneLevelStats	a vector or a one-column data.frame or matrix, containing the gene level statistics. Gene level statistics can be e.g. p-values, t-values or F-values.
directions	a vector or a one-column data.frame or matrix, containing fold-change like values for the related gene-level statistics. This is mainly used if statistics are p-values or F-values, but not required. The values should be positive or negative, but only the sign information will be used, so the actual value will not matter.
geneSetStat	the statistical GSA method to use. Can be one of "fisher", "stouffer", "reporter", "tailStrength", "wilcoxon", "mean", "median", "sum", "maxmean", "gsea", "fgsea" or "page". See below for details.
signifMethod	the method for significance assessment of gene sets, i.e. p-value calculation. Can be one of "geneSampling", "samplePermutation" or "nullDist"
adjMethod	the method for adjusting for multiple testing. Can be any of the methods supported by p.adjust, i.e. "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr" or "none". The exception is for geneSetStat="gsea", where only the options "fdr" and "none" can be used.
gsc	a gene set collection given as an object of class GSC as returned by the loadGSC function.
gsSizeLim	a vector of length two, giving the minimum and maximum gene set size (number of member genes) to be kept for the analysis. Defaults to c(1,Inf).
permStats	a matrix with permutated gene-level statistics (columns) for each gene (rows). This should be calculated by the user by randomizing the sample labels in the original data, and recalculating the gene level statistics for each comparison a large number of times, thus generating a vector (rows in the matrix) of background statistics for each gene. This argument is required and only used if signifMethod="samplePermutation".
permDirections	similar to permStats, but should instead contain fold-change like values for the related permutated statistics. This is mainly used if the statistics are p-values or F-values, but not required. The values should be positive or negative, but only the sign information will be used, so the actual value will not matter. This argument is only used if signifMethod="samplePermutation", but not required. Note however, that if directions is give then also permDirections is required, and vice versa.

nPerm	the number of permutations to use for gene sampling, i.e. if signifMethod="geneSampling". The original Reporter features algorithm (geneSetStat="reporter" and signifMethod="nullDist") also uses a permutation step which is controlled by nPerm.
gseaParam	the exponent parameter of the GSEA and FGSEA approach. This defaults to 1, as recommended by the GSEA authors.
ncpus	the number of cpus to use. If larger than 1, the gene permutation part will be run in parallel and thus decrease runtime. Requires R package **snowfall** to be installed. Should be set so that nPerm/ncpus is a positive integer. (Not used by FGSEA.)
verbose	a logical. Whether or not to display progress messages during the analysis.

Details

The rownames of geneLevelStats and directions should be identical and match the names of the members of the gene sets in gsc. If geneSetStat is set to "fisher", "stouffer", "reporter" or "tailStrength" only p-values are allowed as geneLevelStats. If geneSetStat is set to "maxmean", "gsea", "fgsea" or "page" only t-like geneLevelStats are allowed (e.g. t-values, fold-changes).

For geneSetStat set to "fisher", "stouffer", "reporter", "wilcoxon" or "page", the gene set p-values can be calculated from a theoretical null-distribution, in this case, set signifMethod="nullDist". For all methods signifMethod="geneSampling" or signifMethod="samplePermutation" can be used, except for "fgsea" where only signifMethod="geneSampling" is allowed. If signifMethod="geneSampling" gene sampling is used, meaning that the gene labels are randomized nPerm times and the gene set statistics are recalculated so that a background distribution for each original gene set is acquired. The gene set p-values are calculated based on this background distribution. Similarly if signifMethod="samplePermutation" sample permutation is used. In this case the argument permStats (and optionally permDirections) has to be supplied.

The runGSA function returns p-values for each gene set. Depending on the choice of methods and gene statistics up to three classes of p-values can be calculated, describing different aspects of regulation directionality. The three directionality classes are Distinct-directional, Mixed-directional and Non-directional. The non-directional p-values (pNonDirectional) are calculated based on absolute values of the gene statistics (or p-values without sign information), meaning that gene sets containing a high portion of significant genes, independent of direction, will turn up significant. That is, gene-sets with a low pNonDirectional should be interpreted to be significantly affected by gene regulation, but there can be a mix of both up and down regulation involved. The mixed-directional p-values (pMixedDirUp and pMixedDirDn) are calculated using the subset of the gene statistics that are up-regulated and down-regulated, respectively. This means that a gene set with a low pMixedDirUp will have a component of significantly up-regulated genes, disregardful of the extent of down-regulated genes, and the reverse for pMixedDirDn. This also means that one can get gene sets that are both significantly affected by down-regulation and significantly affected by up-regulation at the same time. Note that sample permutation cannot be used to calculate pMixedDirUp and pMixedDirDn since the subset sizes will differ. Finally, the distinct-directional p-values (pDistinctDirup and pDistinctDirDn) are calculated from statistics with sign information (e.g. t-statistics). In this case, if a gene set contains both up- and down-regulated genes, they will cancel out each other. A gene-set with a low pDistinctDirUp will be significantly affected by up-regulation, but not a mix of up- and down-regulation (as in the case of the mixed-directional and non-directional p-values). In order to be able to calculate distinct-directional gene set p-values

while using p-values as gene-level statistics, the gene-level p-values are transformed as follows: The up-regulated portion of the p-values are divided by 2 (scaled to range between 0-0.5) and the down-regulated portion of p-values are set to 1-p/2 (scaled to range between 1-0.5). This means that a significantly down-regulated gene will get a p-value close to 1. These new p-values are used as input to the gene-set analysis procedure to get `pDistinctDirUp`. Similarly, the opposite is done, so that the up-regulated portion is scaled between 1-0.5 and the down-regulated between 0-0.5 to get the `pDistinctDirDn`.

Value

A list-like object of class `GSAres` containing the following elements:

`geneStatType`	The interpretated type of gene-level statistics
`geneSetStat`	The method for gene set statistic calculation
`signifMethod`	The method for significance estimation
`adjMethod`	The method of adjustment for multiple testing
`info`	A list object with detailed info number of genes and gene sets
`gsSizeLim`	The selected gene set size limits
`gsStatName`	The name of the gene set statistic type
`nPerm`	The number of permutations
`gseaParam`	The GSEA parameter
`geneLevelStats`	The input gene-level statistics
`directions`	The input directions
`gsc`	The input gene set collection
`nGenesTot`	The total number of genes in each gene set
`nGenesUp`	The number of up-regulated genes in each gene set
`nGenesDn`	The number of down-regulated genes in each gene set
`statDistinctDir`	Gene set statistics of the distinct-directional class
`statDistinctDirUp`	Gene set statistics of the distinct-directional class
`statDistinctDirDn`	Gene set statistics of the distinct-directional class
`statNonDirectional`	Gene set statistics of the non-directional class
`statMixedDirUp`	Gene set statistics of the mixed-directional class
`statMixedDirDn`	Gene set statistics of the mixed-directional class
`pDistinctDirUp`	Gene set p-values of the distinct-directional class
`pDistinctDirDn`	Gene set p-values of the distinct-directional class
`pNonDirectional`	Gene set p-values of the non-directional class
`pMixedDirUp`	Gene set p-values of the mixed-directional class

pMixedDirDn Gene set p-values of the mixed-directional class

pAdjDistinctDirUp
Adjusted gene set p-values of the distinct-directional class

pAdjDistinctDirDn
Adjusted gene set p-values of the distinct-directional class

pAdjNonDirectional
Adjusted gene set p-values of the non-directional class

pAdjMixedDirUp Adjusted gene set p-values of the mixed-directional class

pAdjMixedDirDn Adjusted gene set p-values of the mixed-directional class

runtime The execution time in seconds

Author(s)

Leif Varemo <piano.rpkg@gmail.com> and Intawat Nookaew <piano.rpkg@gmail.com>

References

Fisher, R. Statistical methods for research workers. Oliver and Boyd, Edinburgh, (1932).

Stouffer, S., Suchman, E., Devinney, L., Star, S., and Williams Jr, R. The American soldier: adjustment during army life. Princeton University Press, Oxford, England, (1949).

Patil, K. and Nielsen, J. Uncovering transcriptional regulation of metabolism by using metabolic network topology. Proceedings of the National Academy of Sciences of the United States of America 102(8), 2685 (2005).

Oliveira, A., Patil, K., and Nielsen, J. Architecture of transcriptional regulatory circuits is knitted over the topology of bio-molecular interaction networks. BMC Systems Biology 2(1), 17 (2008).

Kim, S. and Volsky, D. Page: parametric analysis of gene set enrichment. BMC bioinformatics 6(1), 144 (2005).

Taylor, J. and Tibshirani, R. A tail strength measure for assessing the overall univariate significance in a dataset. Biostatistics 7(2), 167-181 (2006).

Mootha, V., Lindgren, C., Eriksson, K., Subramanian, A., Sihag, S., Lehar, J., Puigserver, P., Carlsson, E., Ridderstrale, M., Laurila, E., et al. Pgc-1-alpha-responsive genes involved in oxidative phosphorylation are coordinately downregulated in human diabetes. Nature genetics 34(3), 267-273 (2003).

Subramanian, A., Tamayo, P., Mootha, V., Mukherjee, S., Ebert, B., Gillette, M., Paulovich, A., Pomeroy, S., Golub, T., Lander, E., et al. Gene set enrichment analysis: a knowledgebased approach for interpreting genom-wide expression profiles. Proceedings of the National Academy of Sciences of the United States of America 102(43), 15545-15550 (2005).

Efron, B. and Tibshirani, R. On testing the significance of sets of genes. The Annals of Applied Statistics 1, 107-129 (2007).

See Also

piano, loadGSC, GSAsummaryTable, geneSetSummary, networkPlot2, exploreGSAres, **samr**, **limma**, **GSA**, **fgsea**

Examples

```
# Load example input data to GSA:
data("gsa_input")

# Load gene set collection:
gsc <- loadGSC(gsa_input$gsc)

# Run gene set analysis:
gsares <- runGSA(geneLevelStats=gsa_input$pvals , directions=gsa_input$directions,
                 gsc=gsc, nPerm=500)
```

`runGSAhyper` *Gene set analysis with Fisher's exact test*

Description

Performs gene set analysis (GSA) based on a list of significant genes and a gene set collection, using Fisher's exact test, returning the gene set p-values.

Usage

```
runGSAhyper(
  genes,
  pvalues,
  pcutoff,
  universe,
  gsc,
  gsSizeLim = c(1, Inf),
  adjMethod = "fdr"
)
```

Arguments

`genes`	a vector of all genes in your experiment, or a small list of significant genes.
`pvalues`	a vector (or object to be coerced into one) of pvalues for genes or a binary vector with 0 for significant genes. Defaults to rep(0,length(genes)), i.e. genes is a vector of genes of interest.
`pcutoff`	p-value cutoff for significant genes. Defaults to 0 if pvalues are binary. If p-values are spread in [0,1] defaults to 0.05.
`universe`	a vector of genes that represent the universe. Defaults to genes if pvalues are not all 0. If pvalues are all 0, defaults to all unique genes in gsc.
`gsc`	a gene set collection given as an object of class `GSC` as returned by the `loadGSC` function.

gsSizeLim	a vector of length two, giving the minimum and maximum gene set size (number of member genes) to be kept for the analysis. Defaults to c(1,Inf).
adjMethod	the method for adjusting for multiple testing. Can be any of the methods supported by p.adjust, i.e. "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr" or "none".

Details

The statistical test performed is a one-tailed Fisher's exact test on the contingency table with columns "In gene set" and "Not in gene set" and rows "Significant" and "Non-significant" (this is equivalent to a hypergeometric test).

Command run for gene set i:

fisher.test(res$contingencyTable[[i]], alternative="greater"),

the res$contingencyTable object is available from the object returned from runGSAhyper.

The main difference between runGSA and runGSAhyper is that runGSA uses the gene-level statistics (numerical values for each gene) to calculate the gene set p-values, whereas runGSAhyper only uses the group membership of each gene (in/not in gene set, significant/non-significant). This means that for runGSAhyper a p-value cut-off for determining significant genes has to be chosen by the user and after this, all significant genes will be seen as equally significant (i.e. the actual p-values are not used). The advantage with runGSAhyper is that you can use it to find enriched gene sets when you only have a list of interesting genes, without any statistics.

Value

A list-like object containing the following elements:

pvalues	a vector of gene set p-values
p.adj	a vector of gene set p-values, adjusted for multiple testing
resTab	a full result table
contingencyTable	a list of the contingency tables used for each gene set
gsc	the input gene set collection

Author(s)

Leif Varemo <piano.rpkg@gmail.com> and Intawat Nookaew <piano.rpkg@gmail.com>

See Also

piano, loadGSC, runGSA, fisher.test, phyper, networkPlot

Examples

```
# Load example input data (dummy p-values and gene set collection):
data("gsa_input")

# Load gene set collection:
gsc <- loadGSC(gsa_input$gsc)
```

```
# Randomly select 100 genes of interest (as an example):
genes <- sample(unique(gsa_input$gsc[,1]),100)

# Run gene set analysis using Fisher's exact test:
res <- runGSAhyper(genes, gsc=gsc)

# If you have p-values for the genes and want to make a cutoff for significance:
genes <- names(gsa_input$pvals) # All gene names
p <- gsa_input$pvals # p-values for all genes
res <- runGSAhyper(genes, p, pcutoff=0.001, gsc=gsc)

# If the 20 first genes are the interesting/significant ones they can be selected
# with a binary vector:
significant <- c(rep(0,20),rep(1,length(genes)-20))
res <- runGSAhyper(genes, significant, gsc=gsc)
```

---

runQC *Run quality control*

Description

Performs a set of quality control methods and produces the results as figures.

Usage

```
runQC(
  arrayData,
  rnaDeg = TRUE,
  nuseRle = TRUE,
  hist = TRUE,
  boxplot = TRUE,
  pca = TRUE,
  colorFactor = 1,
  colors = c("red", "green", "blue", "yellow", "orange", "purple", "tan", "cyan",
    "gray60", "black", "white"),
  save = FALSE,
  verbose = TRUE
)
```

Arguments

arrayData	an object of class ArrayData.
rnaDeg	should RNA degradation be detected? Defaults to TRUE.

nuseRle	should Normalized Unscaled Standard Errors (NUSE) and Relative Log Expressions (RLE) be calculated? Defaults to TRUE.
hist	produce histograms of expression values? Defaults to TRUE.
boxplot	produce boxplots of expression values? Defaults to TRUE.
pca	should PCA be run? Defaults to TRUE.
colorFactor	a number specifying which column of the setup (given by the ArrayData object) should be used for coloring information for the PCA. Defaults to 1.
colors	a character vector of colors to be used in the PCA plot.
save	should the figures be saved? Defaults to FALSE.
verbose	verbose? Defaults to TRUE.

Details

This function is essentially a wrapper for various available quality control functions for AffyBatch objects and normalized microarray data. RNA degradation (rnaDeg=TRUE) and NUSE & RLE (nuseRle=TRUE) require raw data (a dataRaw element in the ArrayData object).

The PCA uses prcomp on centralized normalized data.

Typical usages are:

```
 # Run all quality controls:
runQC(arrayData)
```

Value

Does not return any object.

Author(s)

Leif Varemo <piano.rpkg@gmail.com> and Intawat Nookaew <piano.rpkg@gmail.com>

References

Brettschneider J, Collin F, Bolstad BM, and Speed TP. Quality assessment for short oligonucleotide arrays. *Technometrics.* (2007) In press

See Also

piano, loadMAdata, diffExp, AffyRNAdeg, fitPLM, AffyBatch, prcomp

Examples

```
 # Get path to example data and setup files:
 dataPath <- system.file("extdata", package="piano")

 # Load normalized data:
myArrayData <- loadMAdata(datadir=dataPath, dataNorm="norm_data.txt.gz", platform="yeast2")

 # Run PCA only:
```

```
runQC(myArrayData,rnaDeg=FALSE, nuseRle=FALSE, hist=FALSE, boxplot=FALSE)
```

`writeFilesForKiwi` *Write files for Kiwi*

Description

Given a single object or a list of objects of class GSAres, extract the information needed for visualization in the external python function Kiwi and write it to files that can be used as input.

Usage

```
writeFilesForKiwi(gsaRes, label = "", overwrite = FALSE)
```

Arguments

`gsaRes`	either an object of class `GSAres` or a list where each element is an object of class `GSAres`, as returned by the `runGSA` function.
`label`	a character string that will be appended to the names of the resulting files.
`overwrite`	a logical, whether or not to overwrite existing files with identical names.

Details

This function takes the result from a gene set analysis as returned by the `runGSA` function and writes three files that can be directly used as input to Kiwi. Kiwi is a external function i python that can be used for network-based visualization of the GSA results (http://sysbio.se/kiwi).

Value

Three files are written in the current directory. GSC.txt contains the gene-gene set associations, i.e. the gene set collection. GLS.txt contains the gene-level statistics. GSS.txt contains the gene set statistics.

Author(s)

Leif Varemo <piano.rpkg@gmail.com>

See Also

piano, `runGSA`, `networkPlot`

Examples

```
# Load some example GSA results:
data(gsa_results)

# Write the files:
writeFilesForKiwi(gsa_results,"exp1")
```

Index

∗ **datasets**
    gsa_input, 15
    gsa_results, 16

AffyBatch, *35*
AffyRNAdeg, *35*

consensusHeatmap, *2*, 3
consensusScores, *2*, 5

diffExp, 7, *11*, *20*, *26, 27*, *35*

exploreGSAres, 10, *14*, *22*, *25*, *31*
expresso, *20*
extractFactors, *7–9*, 11

fgsea, *31*
fisher.test, *33*
fitPLM, *35*

geneSetSummary, *2*, 12, *31*
GSA, *31*
gsa_input, 15
gsa_results, 16
GSAheatmap, *10*, 13, *15*, *22*, *25*
GSAsummaryTable, *2*, *12–14*, 14, *31*

heatmap.2, *9*

justPlier, *19, 20*

layout, *21, 22*
limma, *3*, *8, 9*, *31*
loadGSC, 16, *28*, *31–33*
loadMAdata, *3*, *9*, 18, *35*

mas5, *19*

networkPlot, *13*, *15*, 20, *33*, *36*
networkPlot2, *10*, *14*, *22*, 22, *31*

phyper, *33*
piano, *4*, *6*, *9–12*, *14, 15*, *17, 18*, *20*, *22*, *25*, *27*, *31*, *33*, *35, 36*
piano *(*piano-package*)*, 2
piano-package, 2
polarPlot, *9*, 26
prcomp, *35*

radial.plot, *27*
ReadAffy, *19, 20*
rma, *19*
runGSA, *2–4*, *6*, *9, 10*, *12*, *14–17*, *22*, *24, 25*, *27*, *33*, *36*
runGSAhyper, 32
runQC, *20*, 34

samr, *31*

venn, *9*

writeFilesForKiwi, 36

yeast2.db, *20*