PEPPER

February 2, 2018

`convert.probe.to.gene.expression`
Convert probe level expression to gene level

Description

Convert probe level expression to gene level

Usage

```
convert.probe.to.gene.expression(expr, gene.mapping, selection.method = "iqr")
```

Arguments

`expr`	Expression matrix.
`gene.mapping`	Probe-gene mapping.
`selection.method`	How to handle multiple probes corresponding to the same gene. Defaults to iqr (probe with highest IQR). Other options var (highest variance), med (median of the probes)

Value

a data frame containin gene level expression data

Examples

```
#expr.gene = convert.probe.to.gene.expression(expr, gene.mapping)
```

`convert.sample.mapping.to.case.control` *Convert sample mapping to case control from conditions*

Description

Convert sample mapping to case control from conditions

Usage

```
convert.sample.mapping.to.case.control(sample.mapping, states.case,
  states.control, out.file = NULL)
```

Arguments

`sample.mapping`	Sample-condition mapping for the data set.
`states.case`	Conditions to be used as case
`states.control`	Conditions to be used as background
`out.file`	File name to write the converted mapping

Value

a data frame containin sample mapping

Examples

```
#sample.mapping = convert.sample.mapping.to.case.control(sample.mapping,
#      states.control = c("healthy donor"),
#     states.case = c("tuberculosis", "latent tuberculosis infection"))
```

`fetch.expression.data` *Fetch expression data from GEO / or given folder*

Description

Fetch expression data from GEO / or given folder

Usage

```
fetch.expression.data(geo.id, sample.mapping.column = "characteristics_ch1",
  do.log2 = NULL, probe.conversion = NULL, conversion.mapping = NULL,
  conversion.mapping.function = NULL, output.dir = getwd(),
  geo.id.sub = NULL, reprocess = NULL)
```

Arguments

`geo.id`	GEO id
`sample.mapping.column`	The column in the sample annotation that contains phenotype info (default: "characteristics_ch1")
`do.log2`	Apply log2 transformation to the expression values. If NULL (default), deduced from the data (code from GEO2R at NCBI)
`probe.conversion`	Convert probe level expression to gene level using provided annotation label (uses platform specific annotations downloaded from GEO) Defaults to NULL (no conversion). In case of multiple probes, probe with absolute max value is chosen
`conversion.mapping`	Mapping of platform specific ids to user provided ids
`conversion.mapping.function`	Function to process the mapped name such that it matches with the ids provided in conversion.map
`output.dir`	Directory where all files will be written (defaults to current working dir)
`geo.id.sub`	GEO id for the sub-set (e.g., specific to the platform)

Value

A list containing 3 data frames: expression matrix, sample mapping, gene mapping

Examples

```
gds.data = fetch.expression.data("GDS4966", do.log2=F, probe.conversion="Gene ID")
expr = gds.data$expr
sample.mapping = gds.data$sample.mapping
```

`find.de.genes.limma` *Find differentially expressed genes using LIMMA*

Description

Find differentially expressed genes using LIMMA

Usage

```
find.de.genes.limma(expr, sample.mapping, states, out.file = NULL,
  state.background = NULL, adjust.method = "BH", cutoff = 0.05)
```

Arguments

`expr`	expression matrix.
`sample.mapping`	Sample - condition mapping.
`states`	Conditions to be considered as case.
`out.file`	File to write output. If NULL (default) not used.

`state.background`	Condition to be considered as control.
`adjust.method`	Multiple hypothesis testing correction method. Defaults to BH.
`cutoff`	Adjust p-value cutoff. Defaults to 0.05

Value

Data frame with results.

`find.de.genes` *Find differentially expressed genes*

Description

Find differentially expressed genes

Usage

```
find.de.genes(expr, sample.mapping, states, method = "limma",
  out.file = NULL, state.background = NULL, adjust.method = "BH",
  cutoff = 0.05, functional.enrichment = NULL)
```

Arguments

`expr`	expression matrix.
`sample.mapping`	Sample - condition mapping.
`states`	Conditions to be considered as case.
`method`	Differential expression analysis method: limma (Default) | sam | welch.
`out.file`	File to write output. If NULL (default) not used.
`state.background`	Condition to be considered as control.
`adjust.method`	Multiple hypothesis testing correction method. Defaults to BH.
`cutoff`	Adjust p-value cutoff. Defaults to 0.05
`functional.enrichment`	GO or KEGG based functional enrichment analysis

Value

data frame with results

Examples

```
gds.data = fetch.expression.data("GDS4966", do.log2=F, probe.conversion="Gene ID")
expr = gds.data$expr
sample.mapping = gds.data$sample.mapping
sample.mapping = convert.sample.mapping.to.case.control(sample.mapping,
      states.control = c("healthy donor"),
    states.case = c("tuberculosis", "latent tuberculosis infection"))
d = find.de.genes(expr, sample.mapping, c("case", "control"), method="limma")
```

`find.de.genes.sam` *Find differentially expressed genes using SAM*

Description

Find differentially expressed genes using SAM

Usage

```
find.de.genes.sam(expr, sample.mapping, states, out.file = NULL,
  state.background = NULL, adjust.method = "BH", cutoff = 0.05)
```

Arguments

`expr`	expression matrix.
`sample.mapping`	Sample - condition mapping.
`states`	Conditions to be considered as case.
`out.file`	File to write output. If NULL (default) not used.
`state.background`	Condition to be considered as control.
`adjust.method`	Multiple hypothesis testing correction method. Defaults to BH.
`cutoff`	Adjust p-value cutoff. Defaults to 0.05

Value

Data frame with results.

`find.de.genes.welch` *Find differentially expressed genes using Welch (t-test w/ unequal variance) test*

Description

Find differentially expressed genes using Welch (t-test w/ unequal variance) test

Usage

```
find.de.genes.welch(expr, sample.mapping, states, out.file = NULL,
  state.background = NULL, adjust.method = "BH", cutoff = 0.05)
```

Arguments

`expr`	expression matrix.
`sample.mapping`	Sample - condition mapping.
`states`	Conditions to be considered as case.
`out.file`	File to write output. If NULL (default) not used.
`state.background`	Condition to be considered as control.
`adjust.method`	Multiple hypothesis testing correction method. Defaults to BH.
`cutoff`	Adjust p-value cutoff. Defaults to 0.05.

Value

Data frame with results

`get.data.set` *Get expression data set*

Description

Get expression data set

Usage

```
get.data.set(geo.id, output.dir, is.annotation = F)
```

Arguments

`geo.id`	GEO id.
`output.dir`	Output directory to write / look for files.

Value

data set

`get.fdr.matrix` *Get fdr matrix*

Description

Calculates FDRs from z-scores for each gene in each sample.

Usage

```
get.fdr.matrix(z, adjust.method, out.file = NULL)
```

Arguments

`z`	Data frame containing z-scores (probes vs samples).
`adjust.method`	P-value correction method (see p.adjust).
`out.file`	Output file for writing z score matrix.

Value

Data frame containing FDR values.

`get.peeps.from.z.matrix` *Get peeps from z-score matrix*

Description

Returns peeps for each sample in a given z-score matrix.

Usage

```
get.peeps.from.z.matrix(z, cutoff, convert.to.pvalues)
```

Arguments

`z`	Matrix containing z-scores (genes vs samples), tab separated.
`cutoff`	Threshold for deciding peeps, either a z-score or adjusted p-value (if convert.to.pvalues=T).
`convert.to.pvalues`	Flag to convert z-scores to p-values. If TRUE, the z-scores are converted to P-values which are then corrected for multiple hypothesis testing.

Value

Data frame containing sample name and geneid of genes in the peeps

Examples

```
peeps <- get.peeps.from.z.matrix(z, cutoff=0.05, convert.to.pvalues=T)
```

`get.z.matrix` *Get z matrix*

Description

Returns z-score matrix for a given GEO data set. The z-scores are calculated for each gene in each sample using the mean and sd over provided control samples.

Usage

```
get.z.matrix(expr, sample.mapping, states.control = NULL,
  states.case = NULL, method = "mean", out.file = NULL)
```

Arguments

`expr`	Expression matrix (genes vs samples), tab separated.
`sample.mapping`	Sample - condition mapping.
`states.control`	Label of control (background) samples. If NULL sample.mapping is assumed to include the following types: "case" "control".
`states.case`	Label of case (disease) samples.
`method`	Method to calculate the z score, defaults to mean and sd, use median for med and mad.
`out.file`	Output file for writing z score matrix.

Value

Data frame containing z-scores

Examples

```
gds.data = fetch.expression.data("GDS4966", do.log2=F, probe.conversion="Gene ID")
expr = gds.data$expr
sample.mapping = gds.data$sample.mapping
z = get.z.matrix(expr, sample.mapping,
states.control = c("healthy donor"),
states.case = c("tuberculosis", "latent tuberculosis infection"))
```

---

get.z.score *Get z score*

Description

Calculates z-score for each gene in each sample using the mean and sd over provided control samples.

Usage

```
get.z.score(expr, samples.background, method = "mean",
  samples.to.exclude = NULL)
```

Arguments

expr	Expression matrix (probes vs samples).
samples.background	Names of the background (control) samples.
method	Method to calculate the z score, defaults to mean and sd, use median for med and mad.
samples.to.exclude	Names of the samples to be excluded from background (for CV).

Value

Data frame containing z-scores

Index