Package 'tRanslatome'

October 17, 2020

Type Package

Title Comparison between multiple levels of gene expression

Version 1.26.0

Date 2020-03-03

Author Toma Tebaldi, Erik Dassi, Galena Kostoska

Maintainer Toma Tebaldi <tebaldi@science.unitn.it>, Erik Dassi <erik.dassi@unitn.it>

Depends R (>= 2.15.0), methods, limma, sigPathway, anota, DESeq, edgeR, RankProd, topGO, org.Hs.eg.db, GOSemSim, Heatplus, gplots, plotrix, Biobase

Description Detection of differentially expressed genes (DEGs) from the comparison of two biological conditions (treated vs. untreated, diseased vs. normal, mutant vs. wild-type) among different levels of gene expression (transcriptome ,translatome, proteome), using several statistical methods: Rank Product, Translational Efficiency, t-test, Limma, ANOTA, DESeq, edgeR. Possibility to plot the results with scatterplots, histograms, MA plots, standard deviation (SD) plots, coefficient of variation (CV) plots. Detection of significantly enriched post-transcriptional regulatory factors (RBPs, miRNAs, etc) and Gene Ontology terms in the lists of DEGs previously identified for the two expression levels. Comparison of GO terms enriched only in one of the levels or in both. Calculation of the semantic similarity score between the lists of enriched GO terms coming from the two expression levels. Visual examination and comparison of the enriched terms with heatmaps, radar plots and barplots.

License GPL-3

LazyLoad yes

biocViews CellBiology, GeneRegulation, Regulation, GeneExpression, DifferentialExpression, Microarray, HighThroughputSequencing, QualityControl, GO, MultipleComparisons, Bioinformatics

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

git_branch RELEASE_3_11

git_last_commit c3a2f15

git_last_commit_date 2020-04-27

Date/Publication 2020-10-16

R topics documented:

`tRanslatome-package` *tRanslatome*

Description

Description: Detection of differentially expressed genes (DEGs) from the comparison of two biological conditions (treated vs. untreated, diseased vs. normal, mutant vs. wild-type) among different levels of gene expression (transcriptome ,translatome, proteome), using several statistical methods: Translational Efficiency, Rank Product, t-test, Limma, ANOTA, DESeq, edgeR. Possibility to plot the results with scatterplots, histograms, MA plots, standard deviation (SD) plots, coefficient of variation (CV) plots. Detection of significantly enriched Gene Ontology terms in the lists of DEGs previously identified for the two expression levels. Comparison of GO terms enriched only in one of the levels or in both. Calculation of the semantic similarity score between the lists of enriched GO terms coming from the two expression levels. Visual examination and comparison of the GO terms with heatmaps, radar plots and barplots.

`average.similarity.scores` *labelLevelsGOSetsHelpfile*

Description

This function displays an object of class `character` specifying the names of the two levels compared in the experiment. It takes as input an object of class `GOsims`.

Usage

```
average.similarity.scores(object)
```

Arguments

`object`	an object of class `GOsims`.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

`GOsims` `GOComparison`

Examples

```
data(tRanslatomeSampleData)
average.similarity.scores(CCComparison)
```

computeDEGs *computeDEGsHelpfile*

Description

This function takes as input an object of the class TranslatomeDataset which contains a normalized data matrix coming from high throughput experiment. It takes as an input a character label specifying the method that we want to employ in order to detect DEGs(t-test, translational efficiency, ANOTA, DESeq, edgeR, RP, limma) and returns an object of the class DEGs, in which each gene is assigned an expression class: up- or down-regulated at the first level, up- or down-regulated at the second level, up-regulated at both levels, down-regulated at both levels, up-regulated at the first level and down-regulated at the second level and vice versa.

Usage

```
computeDEGs(object, method="limma", significance.threshold= 0.05,
    FC.threshold= 0, log.transformed = FALSE, mult.cor=TRUE)
```

Arguments

object	an object of class TranslatomeDataset
method	a character string that specifies the method that the user wants to employ in the differential expression analysis. It can have one the following values: limma, t-test, RP, TE, ANOTA, DESeq and none.By default, this value is set to limma,
significance.threshold	a numeric value specifying the threshold on the statistical significance below which the genes are considered as differentially expressed, the default is set to 0.05,
FC.threshold	a numeric value specifying the threshold on the absolute log2 fold change, above which the genes are considered as differentially expressed, the default is set to 0,
log.transformed	a boolean variable specifying whether the signals contained in expr.matrix have been previously log2 transformed. By default it is set to FALSE,
mult.cor	a boolean variable specifying whether the significance threshold is applied on the multiple test corrected or on the original p-values obtained from the DEGs detection method. By default it is set to TRUE,

Details

Signals contained in expr.matrix should be previously normalized with standard methods (quantile, percentile shift, ...) when data is coming from microarrays or in the appropriate cases when it is coming from sequencing experiments.

Value

An object of class DEGs

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

References

Smyth GK. (2004) Linear models and empirical Bayes methods for assessing differential expression in microarray experiments. Stat Appl Genet Mol Biol., 3:Article3.

Tian L, Greenberg SA, Kong SW, Altschuler J, Kohane IS, Park PJ.(2005) Discovering statistically significant pathways in expression profiing studies. Proc Natl Acad Sci USA, 102(38):13544-9.

Courtes FC et al. (2013) Translatome analysis of CHO cells identify key growth genes. Journal of Biotechnology, 167, 215-24.

Breitling R, Armengaud P, Amtmann A, Herzyk P.(2004) Rank products: a simple, yet powerful, new method to detect differentially regulated genes in replicated microarray experiments. FEBS Lett., 573(1-3):83-92.

Tusher VG, Tibshirani R, Chu G.(2001) Significance analysis of microarrays applied to the ionizing radiation response. Proc Natl Acad Sci USA., 2001, 98(9):5116-21.

Larsson O, Sonenberg N, Nadon R.(2011) anota: Analysis of differential translation in genome-wide studies. Bioinformatics, 27(10):1440-1.

Anders S, Huber W.(2010) Differential expression analysis for sequence count data. Genome Biology, 11(10):R106.

Robinson MD, McCarthy DJ, Smyth GK.(2010) edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics, 26(1):139-40.

See Also

TranslatomeDataset DEGs Scatterplot Histogram CVplot MAplot SDplot

Examples

```
data(tRanslatomeSampleData)
computeDEGs(translatome.analysis, method= "limma", ,mult.cor=TRUE)
```

CVplot *CVplotDEGsHelpfile*

Description

The CV plot displays the coefficients of variation of the genes on the x-axis and the log2 fold changes on the y-axis. The upper plot represents a CV plot for the first level of gene expression analysis, whereas the other one refers to the second level. DEGs are color labeled. This function takes as input an object of class DEGs.

Usage

```
CVplot(object, outputformat="on screen",track="")
```

Arguments

object	an object of class DEGs.
outputformat	a character string specifying if the plot is saved in jpeg (jpeg), postscript (postscript), pdf (pdf) format, or it is simply displayed on the screen(on screen). By default this value is on screen.
track	a character vector of gene names that will be explicitly highlighted in the scatterplot, if they match any gene contained in the object of class DEGs. By default this vector is empty.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

DEGs TranslatomeDataset computeDEGs

Examples

```
data(tRanslatomeSampleData)
CVplot(limma.DEGs, outputformat="on screen",track=c("BLM"))
```

DEGs-class *Class* DEGs

Description

A class generated from the function computeDEGs() containing the result of the differential expression analysis at the two expression levels.

Slots

DEGs.table: Object of class matrix containing all the differential expression analysis. The matrix contains one row for each gene, while columns are organized as follows:

1. log2 FC(1st level), log2 fold change in the first level,
2. avg(1st level,control), average signal intensities of the control samples in the first level,
3. sd(1st level,control), standard deviation of the signal intensities of the control samples in the first level,
4. avg(1st level,treated), average signal intensities of the treated samples in the first level,
5. sd(1st level,treated), standard deviation of the signal intensities of the treated samples in the first level,
6. pv.METHOD(1st level), the differential expression p-value returned from the chosen method in the first level (the name of this column will change according to the selected method),
7. pv.METHOD.BH(1st level), the Benjamini-Hochberg corrected differential expression p-value returned from the chosen method in the first level (the name of this column will change according to the selected method),
8. log2 FC(2nd level), log2 fold change in the second level,
9. avg(2nd level,control), average signal intensities of the control samples in the second level,
10. sd(2nd level,control), standard deviation of the signal intensities of the control samples in the second level,
11. avg(2nd level,treated), average signal intensities of the treated samples in the second level,
12. sd(2nd level,treated), standard deviation of the signal intensities of the treated samples in the second level,

13. pv.METHOD(2nd level), the differential expression p-value returned from the chosen method in the second level (the name of this column will change according to the selected method),
14. pv.METHOD.BH(2nd level), the Benjamini-Hochberg corrected differential expression p-value returned from the chosen method in the second level (the name of this column will change according to the selected method),
15. level1Up, boolean value set to 1 if the gene is upregulated in the first level, otherwise set to 0,
16. level1Down, boolean value set to 1 if the gene is downregulated in the first level, otherwise set to 0,
17. level2Up, boolean value set to 1 if the gene is upregulated in the second level, otherwise set to 0,
18. level2Down, boolean value set to 1 if the gene is downregulated in the second level, otherwise set to 0,
19. DownDown, boolean value set to 1 if the gene is downregulated in both levels, otherwise set to 0,
20. DownUp, boolean value set to 1 if the gene is downregulated in the first level and up regulated in the second level, otherwise set to 0,
21. UpDown, boolean value set to 1 if the gene is upregulated in the first level and downregulated in the second level, otherwise set to 0,
22. UpUp, boolean value set to 1 if the gene is upregulated in both levels, otherwise set to 0

method**:** Object of class character specifying the statistical method used to detect DEGs.

significance.threshold**:** Object of class numeric specifying the significance threshold used to detect DEGs.

FC.threshold**:** Object of class numeric specifying the fold change threshold used to detect DEGs.

label.level.DEGs**:** Object of class character specifying the names of the two levels compared in the experiment.

label.condition**:** Object of class character specifying the names of the two conditions compared in the experiment.

Accessors

getDEGsMethod signature(object = "DEGs"): displays an object of class character specifying the statistical method used to detect DEGs.

significance.threshold signature(object = "DEGs"): displays an object of class numeric specifying the significance threshold used to detect DEGs.

FC.threshold signature(object = "DEGs"): displays an object of class numeric specifying the fold change threshold used to detect DEGs.

label.level.DEGs signature(object = "DEGs"): displays an object of class character specifying the names of the two levels compared in the experiment.

label.condition signature(object = "DEGs"): displays an object of class character specifying the names of the two conditions compared in the experiment.

DEGs.table signature(object = "DEGs"): displays an object of class matrix specifying the table of computed DEGs.

Methods

CVplot signature(object = "DEGs"): enables the generation of a plot where, for each level, signal coefficients of variation for each gene are displayed against their log2 fold changes. DEGs are color labeled.

GOEnrichment `signature(object = "DEGs")`: enables the GO enrichment analysis of the differentially expressed genes at each level.

RegulatoryEnrichment `signature(object = "DEGs")`: enables the enrichment analysis of post-transcriptional regulators (RBPs, miRNAs, ecc) in the DEGs gene list, by means of Fisher test.

Histogram `signature(object = "DEGs")`: enables the generation of a histogram displaying the number of DEGs belonging to different classes according to their expression behaviour at the two levels.

MAplot `signature(object = "DEGs")`: enables the generation of a plot where, for each level, average signal intensities for each gene are displayed against their log2 fold changes. DEGs are color labeled.

Scatterplot `signature(object = "DEGs")`: enables the generation of a plot where fold changes at the first level are displayed for each gene against fold changes at the second level. DEGs are color labeled.

SDplot `signature(object = "DEGs")`: enables the generation of a plot where, for each level, signal standard deviations for each gene are displayed against their log2 fold changes. DEGs are color labeled.

show `signature(object = "DEGs")`: displays all the six slots of the class.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

`TranslatomeDataset` `computeDEGs` `RegulatoryEnrichment` `GOEnrichment` `EnrichedSets` `GOsets`

Examples

```
showClass("DEGs")
```

`DEGs.table` *DEGsTableDEGsHelpfile*

Description

This function displays an object of class `numeric` specifying the significance threshold used to detect DEGs. It takes as input an object of class `DEGs`.

Usage

```
DEGs.table(object)
```

Arguments

`object`	an object of class `DEGs`.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

DEGs TranslatomeDataset computeDEGs

Examples

```
data(tRanslatomeSampleData)
DEGs.table(limma.DEGs)
```

---

enriched.table *enrichedtableenrichedSetsHelpfile*

Description

This function displays an object of class character specifying the names of the two levels compared in the experiment. It takes as input an object of class EnrichedSets.

Usage

```
enriched.table(object)
```

Arguments

object	an object of class EnrichedSets.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

EnrichedSets GOEnrichment

Examples

```
data(tRanslatomeSampleData)
enriched.table(CCEnrichment)
```

EnrichedSets-class *Class* EnrichedSets

Description

A class generated from the function RegulatoryEnrichment() containing the overrepresented post-transcriptional regulatory factors (RBPs, miRNAs, etc) from the lists of differentially expressed genes coming from the analysis of two different expression levels.

Slots

`enriched.table`**:** Object of class `data.frame` containing all the regulatory enrichment analysis. The data frame contains one row for each regulatory factor, while columns are organized as follows:

1. `level`, level of analysis on which the enrichment is calculated. The names of the two levels are taken from the object `DEGs` given as an input to the function `GOEnrichment`.
2. `ID`, ID of the regulatory factor.
3. `number`, number of genes in the human genome associated to the regulatory factor.
4. `list`, list of genes in the list of DEGs associated to the regulatory factor.
5. `pv.fisher`, enrichment p-value calculated with the Fisher test.
6. `pv.fisher.BH`, the Benjamini-Hochberg corrected enrichment p-value calculated with the Fisher test according to the chosen enrichment method.

`label.level.enriched`**:** Object of class `character` specifying the names of the two levels compared in the experiment.

Accessors

enriched.table `signature(object = "EnrichedSets")`: displays the slot `enriched.table`.

label.level.enriched `signature(object = "EnrichedSets")`: displays the slot label.level.enriched, slot of class `character` specifying the names of the two levels compared in the experiment.

Methods

Heatmap `signature(object = "EnrichedSets")`: enables the generation of a heatmap of the top enriched regulatory factors for the first and second level of analysis.

label.level.enriched `signature(object = "EnrichedSets")`: displays the slot `label.level.enriched`.

Radar `signature(object = "EnrichedSets")`: enables the generation of a radar plot of the top enriched regulatory factors for the first and second level of analysis.

show `signature(object = "EnrichedSets")`: displays all the two slots of the class.

Author(s)

Erik Dassi, Toma Tebaldi

References

Alexa A, Rahnenfuhrer J, Lengauer T. Improved scoring of functional groups from gene expression data by decorrelating go graph structure. Bioinformatics 2006, 22(13):1600-7. Dassi E et al(2012). AURA: Atlas of UTR Regulatory Activity. Bioinformatics. 28(1):142-4.

See Also

`GOsets` `DEGs`

Examples

```
showClass("EnrichedSets")
```

FC.threshold *FCThresholdDEGsHelpfile*

Description

This function displays an object of class numeric specifying the significance threshold used to detect DEGs. It takes as input an object of class DEGs.

Usage

```
FC.threshold(object)
```

Arguments

object	an object of class DEGs.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

DEGs TranslatomeDataset computeDEGs

Examples

```
data(tRanslatomeSampleData)
FC.threshold(limma.DEGs)
```

getConditionA *getConditionATranslatomeDatasetHelpfile*

Description

This function displays a character vector of column names (or a numeric vector of columns) belonging to expr.matrix. These columns contain the signal intensity data coming from the samples of the first level of the control condition. It takes as input an object of class TranslatomeDataset.

Usage

```
getConditionA(object)
```

Arguments

object	an object of class TranslatomeDataset.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

TranslatomeDataset computeDEGs

Examples

```
data(tRanslatomeSampleData)
getConditionA(translatome.analysis)
```

---

getConditionB *getConditionBTranslatomeDatasetHelpfile*

Description

This function displays a character vector of column names (or a numeric vector of columns) belonging to expr.matrix. These columns contain the signal intensity data coming from the samples of the first level of the treatment condition. It takes as input an object of class TranslatomeDataset.

Usage

```
getConditionB(object)
```

Arguments

object	an object of class TranslatomeDataset.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

TranslatomeDataset computeDEGs

Examples

```
data(tRanslatomeSampleData)
getConditionB(translatome.analysis)
```

getConditionC *getConditionCTranslatomeDatasetHelpfile*

Description

This function displays a character vector of column names (or a numeric vector of columns) belonging to expr.matrix. These columns contain the signal intensity data coming from the samples of the second level of the control condition. It takes as input an object of class TranslatomeDataset.

Usage

```
getConditionC(object)
```

Arguments

object	an object of class TranslatomeDataset.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

TranslatomeDataset computeDEGs

Examples

```
data(tRanslatomeSampleData)
getConditionC(translatome.analysis)
```

getConditionD *getConditionDTranslatomeDatasetHelpfile*

Description

This function displays a character vector of column names (or a numeric vector of columns) belonging to expr.matrix. These columns contain the signal intensity data coming from the samples of the second level of the treatment condition. It takes as input an object of class TranslatomeDataset.

Usage

```
getConditionD(object)
```

Arguments

object	an object of class TranslatomeDataset.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

TranslatomeDataset computeDEGs

Examples

```
data(tRanslatomeSampleData)
getConditionD(translatome.analysis)
```

---

getConditionLabels *getConditionLabelsTranslatomeDatasetHelpfile*

Description

This function displays an object of class character specifying the names given to the two conditions. By default, the vector is set to c("control","treated"), but the user can specify other names. It takes as input an object of class TranslatomeDataset.

Usage

```
getConditionLabels(object)
```

Arguments

object	an object of class TranslatomeDataset.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

TranslatomeDataset computeDEGs

Examples

```
data(tRanslatomeSampleData)
getConditionLabels(translatome.analysis)
```

getDataType *getDataTypeTranslatomeDatasetHelpfile*

Description

This function displays an object of class `character` specifying the type of data represented by `expr.matrix`. By default it is set to `array`, the other accepted value is `ngs`. It takes as input an object of class `TranslatomeDataset`.

Usage

```
getDataType(object)
```

Arguments

`object`	an object of class `TranslatomeDataset`.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

`TranslatomeDataset` `computeDEGs`

Examples

```
data(tRanslatomeSampleData)
getDataType(translatome.analysis)
```

getDEGs *getDEGsTranslatomeDatasetHelpfile*

Description

This function displays an object of class `DEGs` in which each gene is assigned an expression class: up- or down-regulated at the first level, up- or down-regulated at the second level, up-regulated at both levels, down-regulated at both levels, up-regulated at the first level and down-regulated at the second level and vice versa. It takes as input an object of class `TranslatomeDataset`.

Usage

```
getDEGs(object)
```

Arguments

`object`	an object of class `TranslatomeDataset`.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

TranslatomeDataset computeDEGs DEGs

Examples

```
data(tRanslatomeSampleData)
getDEGs(translatome.analysis)
```

---

getDEGsMethod *getDEGsMethodDEGsHelpfile*

Description

This function displays an object of class character specifying the method that the user employed in the differential expression analysis. It can have one the following values: limma, t-test, TE, RP, ANOTA, DESeq and none.By default, this value is set to limma. It takes as input an object of class DEGs.

Usage

```
getDEGsMethod(object)
```

Arguments

object	an object of class DEGs.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

DEGs computeDEGs TranslatomeDataset

Examples

```
data(tRanslatomeSampleData)
getDEGsMethod(limma.DEGs)
```

getExprMatrix *getExprMatrixTranslatomeDatasetHelpfile*

Description

This function displays a matrix that contains the normalized signal intensity data, each row representing a gene and each column representing a sample. Row names should correspond to gene names, column names should correspond to sample names. It takes as input an object of class TranslatomeDataset.

Usage

```
getExprMatrix(object)
```

Arguments

object	an object of class TranslatomeDataset.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

TranslatomeDataset computeDEGs

Examples

```
data(tRanslatomeSampleData)
getExprMatrix(translatome.analysis)
```

getLevelLabels *getLevelLabelsTranslatomeDatasetHelpfile*

Description

This function displays an object of class character specifying the names given to the two conditions. By default, the vector is set to c("control","treated"), but the user can specify other names. It takes as input an object of class TranslatomeDataset.

Usage

```
getLevelLabels(object)
```

Arguments

object	an object of class TranslatomeDataset.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

TranslatomeDataset computeDEGs

Examples

```
data(tRanslatomeSampleData)
getLevelLabels(translatome.analysis)
```

GOComparison *GOComparisonHelpfile*

Description

GOComparison is a function which takes as input an object of class GOsets, containing the results of a GO enrichment analysis applied to both levels, and returns an object of class GOsims, containing a variety of comparisons among the enriched GO terms.

Usage

```
GOComparison(object)
```

Arguments

object	an object of class GOsets

Value

An object of class GOsims

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

References

Wang et al.(2007) A new method to measure the semantic similarity of go terms Bioinformatics , 23:0 1274-81, May 2007.

See Also

GOEnrichment GOsets GOsims

Examples

```
data(tRanslatomeSampleData)
GOComparison(CCEnrichment)
```

GOEnrichment *GOEnrichmentHelpfile*

Description

GOEnrichment is a function which, given as input an object of class DEGs, identifies overrepresented GO terms among differentially expressed genes. The analysis can be applied to all the GO ontologies or restricted to GO terms specifically belonging to one ontology: molecular function, cellular component or biological process. Moreover the function can identify enriched GO terms for separate classes of genes of interest: only up-regulated genes, only down-regulated genes or both of them together. The output of the function is an object of class GOsets, containing the results of the enrichment analysis.

Usage

```
GOEnrichment(object, ontology="all", classOfDEGs="both",
    test.method="classic", test.threshold = 0.05, mult.cor=TRUE)
```

Arguments

object	an object of class DEGs
ontology	a character string specifying the GO ontology of interest: CC for Cellular Component, BP for Biological Process, MF for Molecular Function or all for all the three ontologies. The default is set to all.
classOfDEGs	a character string specifying the class of genes for which we want to detect enriched GO terms: up for considering only up-regulated genes, down for considering only down-regulated genes, both for considering all DEGs, independently from the direction of their changes. The default is set to both.
test.method	a character string specifying the statistical method to calculate the enrichment. By default it is set to classic (enrichment is measured with the classic Fisher exact test), but it can also be set to elim, weight, weight01 or parentchild. All these methods are implemented in the topGO Bioconductor package
test.threshold	a numeric value specifying the significance threshold upon which the GO terms are considered significantly over-represented. By default it is se to 0.05.
mult.cor	a boolean variable specifying whether the significance threshold is applied to the multiple test corrected or to the original p-values obtained from the selected enrichment method. By default it is set to TRUE.

Value

An object of class GOsets

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

References

Ashburner M et al(2000). Gene ontology: tool for the unification of biology. Nat. Genet. May 2000;25(1):25-9.

Alexa A, Rahnenfuhrer J, Lengauer T. Improved scoring of functional groups from gene expression data by decorrelating go graph structure. Bioinformatics 2006, 22(13):1600-7

See Also

GOComparison GOsets GOsims

Examples

```
data(tRanslatomeSampleData)
GOEnrichment(limma.DEGs,ontology="CC",classOfDEGs="up",
    test.method="classic", test.threshold = 0.05,mult.cor = TRUE)
```

GOsets-class *Class* GOsets

Description

A class generated from the function GOEnrichment() containing the overrepresented Gene Ontology terms from the lists of differentially expressed genes coming from the analysis of two different expression levels.

Slots

enriched.table: Object of class data.frame containing all the GO enrichment analysis. The data frame contains one row for each GO term, while columns are organized as follows:

1. ontology, GO ontology of the term, either BP,CC or MF.
2. level, level of analysis on which the enrichment is calculated. The names of the two levels are taken from the object DEGs given as an input to the function GOEnrichment.
3. GO.ID, Gene Ontology accession ID of the term.
4. term, Gene Ontology name of the term.
5. annotated, number of genes in the human genome associated to the GO term.
6. significant, number of genes in the list of DEGs associated to the GO term.
7. expected, number of genes in the list of DEGs expected to be associated to the GO term by chance.
8. expected, number of genes in the list of DEGs expected to be associated to the GO term by chance.
9. pv.fisher, enrichment p-value calculated with the Fisher test. The result is dependent on the method chosen to calculate enrichment(classic,elim, weight,weight01 or parentchild).
10. pv.fisher.BH, the Benjamini-Hochberg corrected enrichment p-value calculated with the Fisher test according to the chosen enrichment method.

label.level.enriched: Object of class character specifying the names of the two levels compared in the experiment.

Accessors

enriched.table `signature(object = "GOsets")`: displays the slot `enriched.table`.

label.level.enriched `signature(object = "GOsets")`: displays the slot `label.level.enriched`, slot of class `character` specifying the names of the two levels compared in the experiment.

Methods

GOComparison `signature(object = "GOsets")`: returns an object of class `GOsims`, containing identity and semantic similarity comparisons among the enriched GO terms.

Heatmap `signature(object = "GOsets")`: enables the generation of a heatmap of the top enriched GO terms for the first and second level of analysis.

Radar `signature(object = "GOsets")`: enables the generation of a radar plot of the top enriched GO terms for the first and second level of analysis.

show `signature(object = "GOsets")`: displays all the two slots of the class.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

References

Ashburner M et al(2000). Gene ontology: tool for the unification of biology. Nat. Genet. May 2000;25(1):25-9.

Alexa A, Rahnenfuhrer J, Lengauer T. Improved scoring of functional groups from gene expression data by decorrelating go graph structure. Bioinformatics 2006, 22(13):1600-7.

See Also

`EnrichedSets` `GOComparison` `DEGs`

Examples

```
showClass("GOsets")
```

GOsims-class *Class* GOsims

Description

A class generated from the function `GOComparison()` containing the result of the differential expression analysis at the two expression levels.

Slots

`similarity.matrix`**:** Object of class `matrix` containing the semantic similarity comparison between the GO terms enriched at the two levels of analysis. The matrix contains one row for each pairwise comparison between GO terms enriched in the two levels, while columns are organized as follows:

1. direction, direction of the comparison: "first level to second level" means that an enriched GO term from the first level is compared to the most similar GO term enriched in the second level, "second level to first level" means that an enriched GO term from the second level is compared to the most similar GO term enriched in the first level. Level names are taken from the object of class GOsets given as input to the function GOcomparison.
2. ontology, GO ontology of the two terms, either BP,CC or MF.
3. level, gene expression levels for which the term is enriched, either "first level only", "second level only" or "both levels". Level names are taken from the object of class GOsets given as input to the function GOcomparison.
4. start.GO.ID, Gene Ontology accession ID of the first term of the comparison.
5. start.term, Gene Ontology name of the first term of the comparison.
6. end.GO.ID, Gene Ontology accession ID of the second term of the comparison.
7. end.term, Gene Ontology name of the second term of the comparison.
8. similarity.score, semantic similarity value between the two campared GO terms.

identity.matrix**:** Object of class matrix containing the identity comparison between the GO terms enriched at the two levels of analysis. The matrix contains one row for each GO term, while columns are organized as follows:

1. ontology, GO ontology of the term, either BP,CC or MF.
2. level, gene expression levels for which the term is enriched, either "first level only", "second level only" or "both levels". Level names are taken from the object of class GOsets given as input to the function GOcomparison.
3. GO.ID, Gene Ontology accession ID of the term.
4. term, Gene Ontology name of the term.

average.similarity.scores**:** Object of class vector containing the general semantic similarity scores between the GO terms enriched at the two levels of analysis. One similarity score, ranging from 0 to 1, is produced for each GO ontology having at least one enriched term.

Accessors

identity.matrix signature(object = "GOsims"): displays the slot identity.matrix.

similarity.matrix signature(object = "GOsims"): displays the slot similarity.matrix.

average.similarity.scores signature(object = "GOsims"): displays the slot average.similarity.scores.

Methods

IdentityPlot signature(object = "GOsims"): enables the generation of a barplot where, for each GO ontology, the number of GO terms enriched at different levels are displayed.

SimilarityPlot signature(object = "GOsims"): enables the generation of a barplot where, for each GO ontology, the semantic similarity value between GO terms enriched at different levels is displayed.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

TranslatomeDataset computeDEGs GOEnrichment GOsets

Examples

```
showClass("GOsims")
```

`Heatmap` *HeatmapEnrichedHelpfile*

Description

Heatmap is a function that plots the top enriched regulatory element or GO terms for the first and second level of analysis in a heatmap. The number of terms to be displayed can be set.

Usage

```
Heatmap(object, outputformat="on screen", n.nodes.1stlevel="5",
    n.nodes.2ndlevel="5", mult.cor=TRUE, ...)
```

Arguments

`object`	an object of class `GOsets`.
`outputformat`	a character string specifying if the plot is saved in jpeg (`jpeg`), postscript (`postscript`), pdf (`pdf`) format, or it is simply displayed on the screen(`on screen`). By default this value is `on screen`.
`n.nodes.1stlevel`	a numeric value specifying the number of top enriched GO terms, from the first level, that will be represented on the plot. By default the value is set to `5`.
`n.nodes.2ndlevel`	a numeric value specifying the number of top enriched GO terms, from the second level, that will be represented on the plot. By default the value is set to `5`.
`mult.cor`	a boolean variable specifying whether the displayed significance values are multiple test corrected or the original p-values obtained from the selected enrichment method. By default it is set to `TRUE`.
`...`	if the class of object is `GOsets`, it accepts a character string named `ontology` selecting the GO ontology of interest, either `CC`,`BP` or `MF`.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

`EnrichedSets` `RegulatoryEnrichment` `GOsets` `GOEnrichment` `GOsims`

Examples

```
data(tRanslatomeSampleData)
Heatmap(CCEnrichment, ontology="MF")
```

Histogram *HistogramHelpfile*

Description

This function shows in a histogram the number of genes showing differential expression at both levels, or up/down regulated only at one level.

Usage

```
Histogram(object, plottype ="summary", outputformat="on screen")
```

Arguments

object	an object of class DEGs
plottype	a character string specifying whether the histogram should display a summary of DEGs classes (summary) or detailed classes taking into account the number of genes up or down regulated in the first or second level (detailed).
outputformat	a character string specifying if the plot is saved in jpeg (jpeg), postscript (postscript), pdf (pdf) format, or it is simply displayed on the screen(on screen). By default this value is on screen.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

DEGs TranslatomeDataset computeDEGs

Examples

```
data(tRanslatomeSampleData)
Histogram(limma.DEGs,plottype ="summary", outputformat="on screen")
```

identity.matrix *labelLevelsGOSetsHelpfile*

Description

This function displays an object of class character specifying the names of the two levels compared in the experiment. It takes as input an object of class GOsims.

Usage

```
identity.matrix(object)
```

Arguments

object	an object of class GOsims.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

GOsims GOComparison

Examples

```
data(tRanslatomeSampleData)
identity.matrix(CCComparison)
```

IdentityPlot *IdentityPlotHelpfile*

Description

This function displays in a barplot, for each GO ontology, the number of GO terms showing enrichment at both levels or only at one level.

Usage

```
IdentityPlot(object, outputformat="on screen")
```

Arguments

object	an object of class GOsims
outputformat	a character string specifying if the plot is saved in jpeg (jpeg), postscript (postscript), pdf (pdf) format, or it is simply displayed on the screen(on screen). By default this value is on screen.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

GOComparison GOsims GOsets GOEnrichment

Examples

```
data(tRanslatomeSampleData)
IdentityPlot(CCComparison, outputformat="on screen")
```

label.condition *labelConditionDEGsHelpfile*

Description

This function displays an object of class numeric specifying the significance threshold used to detect DEGs. It takes as input an object of class DEGs.

Usage

```
label.condition(object)
```

Arguments

object	an object of class DEGs.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

DEGs TranslatomeDataset computeDEGs

Examples

```
data(tRanslatomeSampleData)
label.condition(limma.DEGs)
```

label.level.DEGs *labelLevelDEGsHelpfile*

Description

This function displays an object of class character specifying the names of the two levels compared in the experiment. It takes as input an object of class DEGs.

Usage

```
label.level.DEGs(object)
```

Arguments

object	an object of class DEGs.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

DEGs TranslatomeDataset computeDEGs

Examples

```
data(tRanslatomeSampleData)
label.level.DEGs(limma.DEGs)
```

`label.level.enriched` *labelLevelsEnrichedSetsHelpfile*

Description

This function displays an object of class `character` specifying the names of the two levels compared in the experiment. It takes as input an object of class `EnrichedSets` or `GOsets`.

Usage

```
label.level.enriched(object)
```

Arguments

`object`	an object of class `EnrichedSets` or `GOsets`.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

EnrichedSets RegulatoryEnrichment GOsets GOEnrichment

Examples

```
data(tRanslatomeSampleData)
label.level.enriched(CCEnrichment)
```

`MAplot` *MAplotHelpfile*

Description

The MA plot displays the average log2 signal of the genes on the x-axis and the log2 fold changes on the y-axis. The upper plot represents a MA plot for the first level of gene expression analysis, whereas the other one refers to the second level. DEGs are color labeled. This function takes as input an object of class `DEGs`.

Usage

```
MAplot(object, outputformat="on screen",track="")
```

Arguments

object	an object of class DEGs.
outputformat	a character string specifying if the plot is saved in jpeg (jpeg), postscript (postscript), pdf (pdf) format, or it is simply displayed on the screen(on screen). By default this value is on screen.
track	a character vector of gene names that will be explicitly highlighted in the scatterplot, if they match any gene contained in the object of class DEGs. By default this vector is empty.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

References

Dudoit, S, Yang, YH, Callow, MJ, Speed, TP. (2002). Statistical methods for identifying differentially expressed genes in replicated cDNA microarray experiments. Stat. Sin. 12:1 111-139

See Also

DEGs TranslatomeDataset computeDEGs

Examples

```
data(tRanslatomeSampleData)
MAplot(limma.DEGs, outputformat="on screen",track="")
```

---

newTranslatomeDataset *newTranslatomeDatasetHelpfile*

Description

This function takes as input a normalized data matrix coming from an high-throughput experiment, along with vectors of column names (or numeric vectors of columns) defining the samples for each of the conditions. It takes as an input a character label specifying the data type(array, ngs) and returns an object of the class TranslatomeDataset, which stores the matrix and the samples definitions and allow to call the computeDEGs function on it to compute differentially expressed genes in the various conditions.

Usage

```
newTranslatomeDataset(expr.matrix, cond.a, cond.b, cond.c, cond.d,
    data.type="array", label.level=c("1st level","2nd level"),
    label.condition=c("control","treated"))
```

Arguments

expr.matrix	a matrix that contains the normalized signal intensity data, each row representing a gene and each column representing a sample. Row names should correspond to gene names, column names should correspond to sample names,
cond.a	a character vector of column names (or a numeric vector of columns) belonging to expr.matrix. These columns contain the signal intensity data coming from the samples of the first level of the control condition (in our example: total RNA, undifferentiated cells),
cond.b	a character vector of column names (or a numeric vector of columns) belonging to expr.matrix. These columns contain the signal intensity data coming from the samples of the first level of the treatment condition(in our example: total RNA, differentiated cells),
cond.c	a character vector of column names (or a numeric vector of columns) belonging to expr.matrix. These columns contain the signal intensity data coming from the samples of the second level of the control condition (in our example: polysomal RNA, undifferentiated cells),
cond.d	a character vector of column names (or a numeric vector of columns) belonging to expr.matrix. These columns contain the signal intensity data coming from the samples of the second level of the treatment group (in our example: polysomal RNA, differentiated cells),
data.type	a character specifying the type of data represented by expr.matrix. By default it is set to array, the other accepted value is ngs,
label.level	a character vector specifying the names given to the two levels. By default the vector is set to c("1st level","2nd level"), but the user can specify other names: in our example the two levels are named "transcriptome" and "translatome",
label.condition	a character vector specifying the names given to the two conditions. By default, the vector is set to c("control","treated"), but the user can specify other names: in our example the two levels are named "undifferentiated" and "differentiated",

Details

Signals contained in expr.matrix should be previously normalized with standard methods (quantile, percentile shift, ...) when data is coming from microarrays or in the appropriate cases when it is coming from sequencing experiments.

Value

An object of class TranslatomeDataset

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

TranslatomeDataset computeDEGs DEGs Scatterplot Histogram CVplot MAplot SDplot

Examples

```
data(tRanslatomeSampleData)
translatome.analysis <-
    newTranslatomeDataset(expressionMatrix,
        c("tot.undiff.a","tot.undiff.b","tot.undiff.c"),
        c("tot.diff.a","tot.diff.b","tot.diff.c"),
        c("pol.undiff.a","pol.undiff.b","pol.undiff.c"),
        c("pol.diff.a","pol.diff.b","pol.diff.c"),
        data.type="array", label.level=c("transcriptome","translatome"),
        label.condition=c("undifferentiated","differentiated"))
```

Radar *RadarEnrichedHelpfile*

Description

Radar is a function that plots the top enriched GO terms for the first and second level of analysis in a radar plot. The number of terms to be displayed can be set.

Usage

```
Radar(object ,outputformat="on screen",n.nodes.1stlevel="5",
    n.nodes.2ndlevel="5",mult.cor=TRUE, ...)
```

Arguments

`object`	an object of class `GOsets`.
`outputformat`	a character string specifying if the plot is saved in jpeg (`jpeg`), postscript (`postscript`), pdf (`pdf`) format, or it is simply displayed on the screen(`on screen`). By default this value is `on screen`.
`n.nodes.1stlevel`	a numeric value specifying the number of top enriched GO terms, from the first level, that will be represented on the plot. By default the value is set to `5`.
`n.nodes.2ndlevel`	a numeric value specifying the number of top enriched GO terms, from the second level, that will be represented on the plot. By default the value is set to `5`.
`mult.cor`	a boolean variable specifying whether the displayed significance values are multiple test corrected or the original p-values obtained from the selected enrichment method. By default it is set to `TRUE`.
`...`	if the class of object is `GOsets`, it accepts a character string named `ontology` selecting the GO ontology of interest, either `CC`,`BP` or `MF`.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

`EnrichedSets` `RegulatoryEnrichment` `GOsets` `GOEnrichment` `GOsims`

Examples

```
data(tRanslatomeSampleData)
Radar(CCEnrichment)
```

`RegulatoryEnrichment` *RegulatoryEnrichmentHelpfile*

Description

RegulatoryEnrichment is a function which, given as input an object of class DEGs, identifies over-represented post-transcriptional regulators (RNA-binding proteins, microRNA, etc) controlling differentially expressed genes. The analysis is by default applied to a dataset of experimentally determined post-transcriptional interactions (i.e. regulator-UTR interaction) extracted from AURA (http://aura.science.unitn.it). However, the user can specify a custom dataset onto which the analysis can be performed (see arguments for details). Moreover, the function can identify enriched regulators for separate classes of genes of interest: only up-regulated genes, only down-regulated genes or both of them together. The method works by exploiting two lists: one containing all genes regulated by each of the post-transcriptional regulators, and the other containing the number of regulated and non-regulated genes for each of these post-transcriptional regulators in the backgroung gene set (usually the whole genome). By means of these two lists it is possible to compute a Fisher enrichment p-value indicating whether a significant group of genes in the DEGs list is likely to be regulated by one or more of these post-transcriptional regulators. The output of the function is an object of class EnrichedSets, containing the results of the enrichment analysis.

Usage

```
RegulatoryEnrichment(object, classOfDEGs="both",
   significance.threshold = 0.05, mult.cor=TRUE, regulated.identities=NULL, regulated.counts=NULL)
```

Arguments

`object`	an object of class DEGs
`classOfDEGs`	a character string specifying the class of genes for which we want to detect enriched regulators: `up` for considering only up-regulated genes, `down` for considering only down-regulated genes, `both` for considering all DEGs, independently from the direction of their changes. The default is set to `both`.
`significance.threshold`	a numeric value specifying the significance threshold upon which the regulators are considered significantly over-represented. By default it is se to `0.05`.
`mult.cor`	a boolean variable specifying whether the significance threshold is applied to the multiple test corrected or to the original p-values obtained from the selected enrichment method. By default it is set to `TRUE`.
`regulated.identities`	a matrix containing two columns (RegulatoryElement, RegulatedGenes) specyifing, for each row, a regulatory element name and the comma-separated list of genes it regulates. The user can use the regulatory.elements.regulated table in the tRanslatomeSampleData dataset as a template. By default this argument is NULL, which implies the dataset obtained from AURA will be used.

`regulated.counts`
: a matrix containing three columns (RegulatoryElement, RegulatedGenes, Non-RegulatedGenes) specyifing, for each row, a regulatory element name, the number of genes it regulates in the background gene set and the number of genes it does not regulate in the background gene set. The user can use the regulatory.elements.counts table in the tRanslatomeSampleData dataset as a template. By default this argument is NULL, which implies the dataset obtained from AURA will be used.

Value

An object of class `EnrichedSets`

Author(s)

Erik Dassi, Toma Tebaldi

References

Dassi E et al(2012). AURA: Atlas of UTR Regulatory Activity. Bioinformatics. 28(1):142-4.

See Also

`TranslatomeDataset` `computeDEGs` `DEGs`

Examples

```
data(tRanslatomeSampleData)
RegulatoryEnrichment(limma.DEGs, significance.threshold = 0.05)
```

Scatterplot *ScatterplotHelpfile*

Description

This plot shows each gene as dot uniquely determined by its log2 fold change at the first level (represented on the x-axis) and the fold change at the second level (represented on the y-axis).

Usage

```
Scatterplot(object, outputformat="on screen",track="")
```

Arguments

`object`	an object of class `DEGs`.
`outputformat`	a character string specifying if the plot is saved in jpeg (`jpeg`), postscript (`postscript`), pdf (`pdf`) format, or it is simply displayed on the screen(`on screen`). By default this value is `on screen`.
`track`	a character vector of gene names that will be explicitly highlighted in the scatterplot, if they match any gene contained in the object of class `DEGs`. By default this vector is empty.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

DEGs TranslatomeDataset computeDEGs

Examples

```
data(tRanslatomeSampleData)
Scatterplot(limma.DEGs, outputformat="on screen",track="")
```

SDplot *SDplotHelpfile*

Description

The SD plot displays the standard deviations of the genes on the x-axis and the log2 fold changes on the y-axis. The upper plot represents a SD plot for the first level of gene expression analysis, whereas the other one refers to the second level. DEGs are color labeled. This function takes as input an object of class DEGs.

Usage

```
SDplot(object, outputformat="on screen",track="")
```

Arguments

object	an object of class DEGs.
outputformat	a character string specifying if the plot is saved in jpeg (jpeg), postscript (postscript), pdf (pdf) format, or it is simply displayed on the screen(on screen). By default this value is on screen.
track	a character vector of gene names that will be explicitly highlighted in the scatterplot, if they match any gene contained in the object of class DEGs. By default this vector is empty.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

DEGs TranslatomeDataset computeDEGs

Examples

```
data(tRanslatomeSampleData)
SDplot(limma.DEGs, outputformat="on screen",track="")
```

significance.threshold *significanceThresholdDEGsHelpfile*

Description

This function displays an object of class numeric specifying the significance threshold used to detect DEGs. It takes as input an object of class DEGs.

Usage

```
significance.threshold(object)
```

Arguments

object	an object of class DEGs.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

DEGs TranslatomeDataset computeDEGs

Examples

```
data(tRanslatomeSampleData)
significance.threshold(limma.DEGs)
```

similarity.matrix *similaritymatrixGOSetsHelpfile*

Description

This function displays an object of class character specifying the names of the two levels compared in the experiment. It takes as input an object of class GOsims.

Usage

```
similarity.matrix(object)
```

Arguments

object	an object of class GOsims.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

GOsims GOComparison

Examples

```
data(tRanslatomeSampleData)
similarity.matrix(CCComparison)
```

SimilarityPlot *SimilarityPlotHelpfile*

Description

This function displays in a barplot, for each GO ontology, the average semantic similarity value between GO terms showing enrichment at the first or at the second level of analysis.

Usage

```
SimilarityPlot(object, outputformat="on screen")
```

Arguments

object	an object of class GOsims
outputformat	a character string specifying if the plot is saved in jpeg (jpeg), postscript (postscript), pdf (pdf) format, or it is simply displayed on the screen(on screen). By default this value is on screen.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

GOComparison GOsets GOsims

Examples

```
data(tRanslatomeSampleData)
SimilarityPlot(CCComparison, outputformat="on screen")
```

TranslatomeDataset-class
Class TranslatomeDataset

Description

A class generated from the function newTranslatomeDataset() containing the input matrix, the condition vectors and labels and the result of the differential expression analysis at the two expression levels. This class represents an analysis in its entirety, containing all data from input parameters to output results.

Slots

expr.matrix: Object of class matrix specifying the normalized signal intensity data, each row representing a gene and each column representing a sample.

cond.a: Object of class character specifying a vector of column names belonging to expression matrix. These columns contain the signal intensity data coming from the samples of the first expression level of the control condition.

cond.b: Object of class character specifying a vector of column names belonging to expression matrix. These columns contain the signal intensity data coming from the samples of the first expression level of the treatment condition.

cond.c: Object of class character specifying a vector of column names belonging to expression matrix. These columns contain the signal intensity data coming from the samples of the second expression level of the control condition.

cond.d: Object of class character specifying a vector of column names belonging to expression matrix. These columns contain the signal intensity data coming from the samples of the second expression level of the treatment condition.

data.type: Object of class character specifying the type of the data contained in exprMatrix. The default is "array" and the alternative is "ngs".

label.condition: Object of class character specifying the names given to the two conditions. By default, these values are "control" and "treated", but user can specify others.

label.level: Object of class character specifying the names given to the two levels. By default levels are named "1st level" and "2nd level", but the user can specify others.

DEGs: Object of class DEGs specifying the result of the differential expression analysis at the two expression levels.

Accessors

getExprMatrix signature(object = "TranslatomeDataset"): displays anObject of class matrix specifying the normalized signal intensity data, each row representing a gene and each column representing a sample.

getConditionA signature(object = "TranslatomeDataset"): displays anObject of class character specifying a vector of column names belonging to expression matrix. These columns contain the signal intensity data coming from the samples of the first expression level of the control condition.

getConditionB signature(object = "TranslatomeDataset"): displays anObject of class character specifying a vector of column names belonging to expression matrix. These columns contain the signal intensity data coming from the samples of the first expression level of the treatment condition.

getConditionC `signature(object = "TranslatomeDataset")`: displays anObject of class `character` specifying a vector of column names belonging to expression matrix. These columns contain the signal intensity data coming from the samples of the second expression level of the control condition.

getConditionD `signature(object = "TranslatomeDataset")`: displays anObject of class `character` specifying a vector of column names belonging to expression matrix. These columns contain the signal intensity data coming from the samples of the second expression level of the treatment condition.

getDataType `signature(object = "TranslatomeDataset")`: displays anObject of class `character` specifying the type of the data contained in `exprMatrix`. The default is "array" and the alternative is "ngs".

getConditionLabels `signature(object = "TranslatomeDataset")`: displays anObject of class `character` specifying the names given to the two conditions. By default, these values are "control" and "treated", but user can specify others.

getLevelLabels `signature(object = "TranslatomeDataset")`: displays anObject of class `character` specifying the names given to the two levels. By default levels are named "1st level" and "2nd level", but the user can specify others.

getDEGs `signature(object = "TranslatomeDataset")`: displays anObject of class `DEGs` specifying the result of the differential expression analysis at the two expression levels.

Methods

computeDEGs `signature(object = "TranslatomeDataset")`: compute the differentially expressed genes at the two levels by means of the chosen method, returning anObject of class `DEGs`.

show `signature(object = "TranslatomeDataset")`: displays all the slots of the class.

Author(s)

Toma Tebaldi, Erik Dassi, Galena Kostoska

See Also

`TranslatomeDataset` `newTranslatomeDataset` `computeDEGs`

Examples

```
showClass("TranslatomeDataset")
```

---

tRanslatomeSampleData *Sample data set for tRanslatome*

Description

matrix with data coming from differentiated and undifferentiated human HepaRG cells

Usage

```
data(tRanslatomeSampleData)
```

Format

tRanslatomeSampleData is a list that has 7 components :

expressionMatrix is a matrix of 12 columns and 1000 rows containing the microarray signals used as input of the function GetDEGs. Columns are organized in biological triplicates as follows: tot.diff.a, tot.diff.b, tot.diff.c contain the signals coming from the transcriptome of the differentiated cell line.

tot.undiff.a, tot.undiff.b, tot.undiff.c contain the signals coming from the transcriptome of the undifferentiated cell line.

pol.diff.a, pol.diff.b, pol.diff.c contain the signals coming from the translatome of the differentiated cell line.

pol.undiff.a, pol.undiff.b, pol.undiff.c contain the signals coming from the translatome of the undifferentiated cell line.

translatome.analysis is an object of class `TranslatomeDataset` generated calling `newTranslatomeDataset()` on `expressionMatrix` (see the examples section for the exactcall).

limma.DEGs is an object of class `DEGs` generated calling `getDEGs()` on `expressionMatrix` (see the examples section for the exact call).

CCEnrichment is an object of class `GOsets` generated calling `GOEnrichment()` on `limma.DEGs` (see the examples section for the exact call).

CCComparison is an object of class `GOsims` generated calling `GOComparison()` on `CCEnrichment` (see the examples section for the exact call).

regulatory.elements.counts is an object of class `data.frame` containing the background numbers of regulated and non-regulated genes for each post-transcriptional regulatory factor considered by function `RegulatoryEnrichment`.

regulatory.elements.regulated is an object of class `data.frame` containing the list of regulated genes for each post-transcriptional regulatory factor considered by function `RegulatoryEnrichment`

Source

Parent R, Kolippakkam D, Booth G, Beretta L. Mammalian target of rapamycin activation impairs hepatocytic differentiation and targets genes moderating lipid homeostasis and hepatocellular growth. Cancer Res. 2007;67(9):4337-4345

See Also

`TranslatomeDataset` `computeDEGs` `DEGs` `RegulatoryEnrichment` `EnrichedSets` `GOEnrichment` `GOsets` `GOComparison` `GOsims`

Examples

```
##load the tRanslatome sample data
data(tRanslatomeSampleData)
```

Index