Package ‘eegc’

October 16, 2019

Type Package

Title Engineering Evaluation by Gene Categorization (eegc)

Version 1.10.0

Date 2018-12-21

Author Xiaoyuan Zhou, Guofeng Meng, Christine Nardini, Hongkang Mei

Maintainer Xiaoyuan Zhou `<zhouxiaoyuan@picb.ac.cn>`

Description This package has been developed to evaluate cellular engineering processes for direct differentiation of stem cells or conversion (transdifferentiation) of somatic cells to primary cells based on high throughput gene expression data screened either by DNA microarray or RNA sequencing. The package takes gene expression profiles as inputs from three types of samples: (i) somatic or stem cells to be (trans)differentiated (input of the engineering process), (ii) induced cells to be evaluated (output of the engineering process) and (iii) target primary cells (reference for the output). The package performs differential gene expression analysis for each pair-wise sample comparison to identify and evaluate the transcriptional differences among the 3 types of samples (input, output, reference). The ideal goal is to have induced and primary reference cell showing overlapping profiles, both very different from the original cells.

VignetteBuilder knitr

Depends R (>= 3.4.0)

Imports R.utils, gplots, sna, wordcloud, igraph, pheatmap, edgeR, DESeq2, clusterProfiler, S4Vectors, ggplot2, org.Hs.eg.db, org.Mm.eg.db, limma, DOSE, AnnotationDbi

Suggests knitr

biocViews ImmunoOncology, Microarray, Sequencing, RNASeq, DifferentialExpression, GeneRegulation, GeneSetEnrichment, GeneExpression, GeneTarget

License GPL-2

LazyData TRUE

RoxygenNote 5.0.1

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

git_branch RELEASE_3_9

git_last_commit 10f55cc

git_last_commit_date 2019-05-02

Date/Publication 2019-10-15

R topics documented:

`barplotEnrich` *Barplot the enrichResult*

Description

This function is revised from the `barplot.enrichResult` function DOSE package, and used to perform a barplot of the enrichResult object.

Usage

```
barplotEnrich(height, x = "Count", colorBy = "p.adjust", top = 5,
font.size = 12, title = "", color = NULL,...)
```

Arguments

`height`	enrichResult object, alternatively output from `functionEnrich` function.
`x`	one of "Count" and "GeneRatio" to specify the x axis.
`colorBy`	one of ’pvalue’, ’p.adjust’, ’qvalue’.
`top`	number of top categories to show.

`font.size, title`	font size and title.
`color`	the color of the bar.
`...`	see parameters in `fortify` function.

Value

bar plot of enrichment results

Examples

```
## Not run:
  # plot the "enrichResult" of Inactive category
  inactive = goenrichraw[[2]]
  barplotEnrich(inactive, top =5, color ="#2c7bb6", title = "Inactive")

## End(Not run)
```

`cate.gene` *Categorized genes*

Description

A list with five gene categories.

Usage

```
data(cate.gene)
```

Format

A list

Value

A list with five gene catogories.

`cate.ratio` *Expression Difference ratios of categorized genes*

Description

A list with five data frames of gene ED ratios

Usage

```
data(cate.ratio)
```

Format

A list

Value

A list with ED ratios for five gene catogories.

`categorizeGene` *Gene Categorization*

Description

This function categorizes differential genes of each pair-wise comparison among e.g. initiating A, derived B and primary C samples during a cellular engineering, into five categories named from *Reversed*,*Inactive*,*Insufficient*, *Successful* and *Over* representing the gene reprogrammed states, and calculates the ratio of expression difference (ED) between B and A to the ED between C and A.

Usage

```
categorizeGene(expr, diffGene, from.sample,to.sample, target.sample)
```

Arguments

`expr`	a data frame with expression for all genes in `diffGene`, alternatively output from `diffGene` function.
`diffGene`	a list of differential genes in three comparisons, alternatively output by `diffGene` function.
`from.sample, to.sample, target.sample`	character to specify the name of initiating sample, derived sample and primary sample during a cellular engineering, must be consistent with sample names in the `expr` data frame.

Details

Gene (g) categorization is achieved by considering the pair-wise comparisons (Expression Difference, ED eq. 1) among the three types of samples and the ratio of such differences (EDg ratio, eq. 2). $EDg(B, A) = averageexpressionofginBminusaverageexpressionofginA$ (1) $EDgratio = EDg(B, A)/EDg(C, A)$ (2)

Reversed, EDg(B,A) and EDg(B,C) are significantly differential, while EDg(C,A) is not limited by differential or not, EDg ratio is smaller than 0; *Inactive*, EDg(C,A) and EDg(B,C) are significantly differential, while EDg(B,A) is not differential; *Insufficient*, EDg(B,A) ,EDg(C,A) and EDg(B,C) are all significantly differential, EDg ratio is between 0 and 1; *Successful*, EDg(B,A) and EDg(C,A) are significantly differential, while EDg(B,C) is not differential; *Over*, EDg(B,A) and EDg(B,C) are significantly differential, while EDg(C,A) is not limited by differential or not, EDg ratio is greater than 1. For the *Inactive* and *Successful* categories, genes with bottom and top 5 percentage ED ratios are removed to avoid the ambiguous categorization with *Reversed*, *Insufficient* or *Over* categories.

Value

A list with components: a list of the five gene categories a list of the ED ratios for the five gene categories.

Examples

```
data(expr.filer)
data(diffgene.genes)

category = categorizeGene(expr = expr.filter,diffGene = diffgene.genes,
           from.sample="DMEC", to.sample="rEChMPP", target.sample="CB")
cate.gene = category[[1]]
cate.ratio = category[[2]]
```

`densityPlot` *Quantify Genes and Corresponding ED ratios in Each Category*

Description

Quantify genes in each gene category and their expression difference (ED) ratios in a density plot.

Usage

```
densityPlot(ratio, color = NULL, main = NA, xlab = NA, ylab = "Density",
legend.labels = NULL, shade = TRUE, transparency = TRUE, proportion = TRUE,
out.file = NULL, ...)
```

Arguments

`ratio`	a list of ED ratios for five gene categories, alternatively output by `categorizeGene`.
`color`	vector of colors for the five gene categories.
`main, xlab, ylab`	the overall title, tile for x axis, and title for y axis, see `title`.
`legend.labels`	vector of labels for the legend.
`shade`	logical to determine if the five categories are filled with shades.
`transparency`	logical to determine if the density plot is transparent.
`proportion`	logical to determine whether the proportion of each category genes over the all genes is drawn on the density plot.
`out.file`	a character string naming the output file with density plot.
`...`	parameters in `plot`.

Value

a density plot

Examples

```
data(cate.ratio)
names(cate.ratio)
# make the extreme ED ratios in Reversed and Over categories to the median values
reverse = cate.ratio[[1]]
over = cate.ratio[[5]]
reverse[reverse[,1] <= median(reverse[,1]), 1]  = median(reverse[,1])
over[over[,1] >= median(over[,1]),1] = median(over[,1])
cate.ratio[[1]] = reverse
```

```
cate.ratio[[5]] = over

# densityPlot(cate.ratio, xlab = "ED ratio", ylab = "Density", proportion = TRUE)
```

`diffGene` *Gene Expression Differential Analysis*

Description

Identify the differentially expressed genes for each pair-wise comparison of given three types of samples.

Usage

```
diffGene(expr, array = TRUE, fpkm = FALSE, counts =FALSE, method =c("limma","DESeq2"),
from.sample, to.sample, target.sample, filter = FALSE, filter.perc = 0.4,
padjust ="fdr", signif = TRUE, pvalue = 0.05)
```

Arguments

`expr`	a data frame with gene expression data.
`array, fpkm, counts`	logical, specifying the type of input gene expression data.
`method`	differential analysis method, alternatively to "limma" and "DESeq2", default to "limma". "DESeq2" can be chosen only when `counts` is `TRUE`.
`from.sample, to.sample, target.sample`	character to specify the name of initiating sample, derived sample and primary sample during a cellular engineering.
`filter`	logical to indicate whether the genes need to be filtered when match the parameter `filter.perc` , only applied to fpkm and counts data.
`filter.perc`	a 0 to 1 number to specify the gene filter criteria by the percentage of samples with non-zero expression. Only used to fpkm and counts data when `filter` is `TRUE`, and filter the genes with non-zero expression in less than filter.perc samples.
`padjust`	indicate the method to do p.value correction, default to "fdr". See `p.adjust`.
`signif`	logical to indicate whether only the significantly differential genes are output, default to FALSE.
`pvalue`	a cutoff p.value for the significant genes, default to 0.05, only used when `signif` is TRUE.

Details

This function can be applied on both microarray and RNA-seq data for differential analysis when one of the "array", "fpkm", or "counts" is specified. It does differential analysis to each pair-wise sample comparison among the from.sample, to.sample and target.sample.

Value

A list with components : a list with differential analysis result for each pair-wise comparison; a list with differential gene names for each pair-wise comparison; a data frame with filtered/unfiltered gene expression.

Examples

```
data(SandlerFPKM)

# differential expression analysis:
diffgene = diffGene(expr = SandlerFPKM, array=FALSE,  fpkm=TRUE,  counts=FALSE,
                   from.sample="DMEC", to.sample="rEChMPP", target.sample="CB",
                   filter=TRUE, filter.perc =0.4, pvalue = 0.05 )

# differential analysis results
diffgene.result = diffgene[[1]]
# differential genes
diffgene.genes = diffgene[[2]]
# filtered expression data
expr.filter = diffgene[[3]]
```

`diffgene.genes` *Differential genes for three comparisons*

Description

A list of three vectors with significantly differential genes among three comparisons.

Usage

```
data(diffgene.genes)
```

Format

A list

Value

A list with differential genes in three comparisons

`dotPercentage` *Percentage Calculation and Visualization*

Description

This function calculate the percentage of genes in each category over given annotated gene sets and plot the percentages.

Usage

```
dotPercentage(cate.gene, annotated.gene, order.by = NULL, type = "l", lty = 1,
pch = NULL, col = NULL, srt = 50, font = 1, adj = c(1,1), cex = 1, add.line = TRUE,
legend = TRUE, legend.label = NULL, ...)
```

Arguments

`cate.gene`	a list of the five gene categories, alternatively output by `categorizeGene`.
`annotated.gene`	a list of the annotated gene sets which the `cate.gene` are proportioned in.
`order.by`	one character out of of "Reversed","Inactive","Insufficient","Successful" and "Over" to specify a gene category the percentage is ordered by.
`type, lty, pch, col`	parameters for the plotting, specifying the type of plotting; the line type when `type` is "l"; the symbol of points on the line; and the color of lines, see graphic parameters in `par`().
`srt, font, cex, adj`	parameters for the text labeled on x-axis, specifying the string rotation in degrees; the font of text; the text size, see graphic parameters in `par`().
`add.line`	logical to determine if to add lines on the dots, logical to `TRUE`.
`legend`	logical to determine whether the legend is added on the figure, default to `TRUE`.
`legend.label`	labels of the legend, applied only when `legend` is `TRUE`.
`...`	other parameters see `plot`.

Value

a data frame with the percentage of `cate.gene` in the `annotated.gene`.

Examples

```
# load the C/T-specific genes in 16 cells/tissues
data(human.gene)
data(cate.gene)
# perc = dotPercentage(cate.gene = cate.gene, annotated.gene = human.gene,
#                      order.by = "Successful")
```

`eegc` *eegc*

Description

eegc

`enrichment` *Enrichment Analysis by Hypergeometric Distribution Test*

Description

Enrichment Analysis by Hypergeometric Distribution Test

Usage

```
enrichment(cate.gene, annotated.gene, background.gene, padjust.method = "fdr" )
```

Arguments

`cate.gene`	a list of the five gene categories, alternatively output by `categorizeGene`.
`annotated.gene`	a list of annotated gene sets which the `cate.gene` are enriched in.
`background.gene`	vector of background genes, e.g. all genes screened by microarray or RNA-sequencing.
`padjust.method`	correction method for enrichment p-values, one of "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none", default to "fdr", see details in `p.adjust`.

Value

A list of enrichment results for the five gene categories.

Examples

```
# load the cell/tissue-specific genes
data(tissueGenes)
# load the mapping file of cells/tissues to grouped cells/tissues
data(tissueGroup)

# get the background genes
data(expr.filter)
genes = rownames(expr.filter)
# enrichment analysis for the five gene categories
data(cate.gene)
tissueenrich = enrichment(cate.gene = cate.gene, annotated.gene = tissueGenes,
                         background.gene = genes, padjust.method = "fdr")
# select a group of cells/tissues
tissueGroup.selec = c("stem cells","B cells","T cells","Myeloid","Endothelial CD105+")
tissues.selec = tissueGroup[tissueGroup[,"Group"] %in% tissueGroup.selec,c(2,3)]
# tissuetable = heatmapPlot(tissueenrich, terms = tissues.selec, GO=FALSE,
#                            annotated_row = TRUE,annotation_legend = TRUE,
#                            main = "Tissue-specific enrichment")
```

`expr.filter` *Filtered expression data*

Description

A data frame with filtered RNASeq FPKM data.

Usage

```
data(expr.filter)
```

Format

A data frame

Value

A data frame with filtered gene expression

`functionEnrich` *Funtional Enrichment Analysis*

Description

This function performs Gene Ontology (GO) and KEGG pathways functional enrichment for the five gene categories by calling **clusterProfiler** package.

Usage

```
functionEnrich(cate.gene, organism = "human", convert = TRUE, from = "SYMBOL", ont = "BP",
pAdjustMethod = "bonferroni", GO = TRUE, KEGG = FALSE, enrichResult = FALSE)
```

Arguments

`cate.gene`	a list of the five gene categories, alternatively output by `categorizeGene`.
`organism`	a character of organism "human" or "mouse" to indicate the species of background genes.
`convert`	logical to determine whether the gene ID should be converted to "ENTREZID", default to `TRUE`.
`from`	the gene id type of input data, see the key types of `org.Hs.eg.db`.
`ont`	One of "MF", "BP", and "CC" subontologies, see `enrichGO`.
`pAdjustMethod`	correction method for p-value, one of "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none", default to "fdr", see details in `p.adjust`.
`GO`	logical to determine whether the functional enrichment is performed on Gene Ontology, default to `TRUE`.
`KEGG`	logical to determine whether the functional enrichment is performed on KEGG pathways, default to `FALSE`.
`enrichResult`	logical to determine if the "enrichResult" is output, default to `FALSE` to output a summary of the "enrichResult".

Value

Function enrichment analysis results.

Examples

```
data(cate.gene)
# result in "enrichResult" class by specifying TRUE to enrichResult parameter
goenrichraw = functionEnrich(cate.gene, organism = "human", pAdjustMethod = "fdr",
                             GO = TRUE, KEGG = FALSE, enrichResult = TRUE)

# result of the summary of "enrichResult" by specifying FALSE to enrichResult parameter
# GO enrichment
goenrich = functionEnrich(cate.gene, organism = "human", pAdjustMethod = "fdr",
                          GO = TRUE, KEGG = FALSE, enrichResult = FALSE)

# KEGG enrichment
keggenrich = functionEnrich(cate.gene, organism = "human", pAdjustMethod = "fdr",
                            GO = FALSE, KEGG = TRUE, enrichResult = FALSE)
```

`goenrich` *GO enrichment results*

Description

A list of 5 data frames with the Gene ontology enrichment results for the five gene categories.

Usage

```
data(goenrich)
```

Format

A list with 5 data frames

Value

A list of GO enrichment results

`grnPlot` *Gene Regulatory Network Plot*

Description

This function plot the a cell/tissue-specific gene regulatory network with genes in the five categories, alterntively reducing the network size by plotting given specific genes as nodes.

Usage

```
grnPlot(grn.data, cate.gene, filter = TRUE, nodes = NULL, centrality.score,
col = NULL, main= NULL, vertex.label =NULL, vertex.label.dist = 0, vertex.label.font = 1,
vertex.label.cex = 0.5, vertex.label.color="black", edge.arrow.size = 0.4,
edge.color = "grey", layout ="layout_with_lgl", legend.labels = NULL, ...)
```

Arguments

`grn.data`	a data frame with two columns named "TF" and "TG" to specify the genes as transcription regulators (TF) and target genes being regulated (TG).
`cate.gene`	a list of the five gene categories as nodes in the network, alternatively output by `categorizeGene`.
`filter`	logical to specify if the network is reduced to less nodes by a filter, logical to `TRUE` for a clear visualization.
`nodes`	a character vector of genes to kept in the network, only applied when `filter` is `TRUE`.
`centrality.score`	a vector or data frame of centrality scores for genes in the network, alternatively calculated by `networkAnalyze` function.
`col`	colors of gene vertex in each `cate.gene`.

`main`	title of the plot.
`vertex.label, vertex.label.dist, vertex.label.font, vertex.label.cex, vertex.label.color`	parameters for vertex labels, to specify the labels of vertex, the position of labels on the vertex, font size, cex and color for label, see details in `igraph.plotting`.
`edge.arrow.size, edge.color`	parameters for edge to specify the arrow size and color of edge, see details in `igraph.plotting`.
`layout`	the layout of network plot, see details in `layout`.
`legend.labels`	vector of label names for each `cate.gene`.
`...`	other parameters used in `igraph.plotting`.

Value

a igraph plot for gene regulatory network.

Examples

```
## Not run:
  # select genes to shown their regulation with others
  node.genes = c("ZNF641", "BCL6")
  # enlarge the centrality
  centrality.score = degree$centrality*100
  names(centrality.score) = degree$Gene
  par(mar = c(2,2,3,2))
  grnPlot(grn.data = human.grn[[tissue]], cate.gene = cate.gene, filter = TRUE,
         nodes = node.genes, centrality.score = centrality.score,
         main = "Gene regulatory network")

## End(Not run)
```

`heatmapPlot` *Heatmap Plot of Enriched Terms*

Description

This function plot the significantly enriched terms in a heatmap by calling **pheatmap** package.

Usage

```
heatmapPlot(enrichresult, GO = FALSE, terms = NULL, padjust = TRUE, pvalue = 0.05,
top= NA, filter = FALSE, main = NA, annotation = NULL, annotation_col = NULL,
annotated_row = FALSE, annotation_row = NULL,annotation_colors = NA,
display_numbers = FALSE, annotation_legend = FALSE,...)
```

Arguments

`enrichresult`	a list of data frames with enrichment results, alternatively output by `functionEnrich` or `enrichment`.
`GO`	logical to determine whether the terms are Gene Ontology(GO) terms enriched by `functionEnrich`.

`terms`	a character vector to specify the terms chosen to be listed in the heatmap, selected from the `enrichment` result, or a data frame with terms and corresponding term annotations used for `annotation_row`.
`padjust`	logical to determine whether the significantly enriched terms were selected by adjusted p.value rather than the p.value, default to `TRUE`.
`pvalue`	a cutoff value to select the significantly enriched terms.
`top`	a number to specify the most significantly enriched terms to be drawn in each category, default to `NA` without specifying.
`filter`	logical to specify whether the enriched terms need to be filered with the ones which are significant among the first four categories.
`main`	a character of main title on the heatmap plot.
`annotation, annotation_row, annotation_col, annotation_colors, annotation_legend, ...`	see details in `pheatmap`.
`annotated_row`	logical to determine whether the the rows are annotated by `annotation_row`, default to `FALSE`. When it's `TRUE`, `annotation_row` should be specified or using annotations in a data frame of `terms`.
`display_numbers`	logical to determine whether the gene counts number is shown on the heatmap.

Value

heatmap plot and the terms with p.values for the heatmap

Examples

```
# plot the enrichment results by the five gene categories
data(goenrich)
heatmaptable = heatmapPlot(goenrich, GO = TRUE, top = 5, filter = FALSE,
                           main = "Gene ontology enrichment",
                           display_numbers =  FALSE)
```

`human.gene` *Gene regulatory network based human cell/tissue-specific gene sets*

Description

A list of 16 human cells/tissues-specific gene sets summarized from the gene regulatory network downloaded from the CellNet website.

Usage

```
data(human.gene)
```

Format

A list

Value

A list with 16 human cells/tissues-specific gene sets from CellNet.

`human.grn` *Human cell/tissue-specific gene-gene regulation*

Description

A list of 16 data frames (cells/tissues) with transcription factors (TF) to target genes (TG) regulation information.

Usage

```
data(human.grn)
```

Format

A list with 16 data frames

Value

A list of human gene regulatory information.

`human.tf` *Gene regulatory network based human cell/tissue-specific transcription factor (TF) regulated gene sets*

Description

A list of 1455 human cells/tissues-specific TF regulated gene sets summarized from the gene regulatory network downloaded from the CellNet website.

Usage

```
data(human.tf)
```

Format

A list

Value

A list with 1455 human cells/tissues-specific TF regulated gene sets

`markers` *Marker genes*

Description

A vector containing 65 genes specific in endothelial and haematopoietic cells as listed in Sandler's paper.

Usage

```
data(markers)
```

Format

A vector

Value

A vector of marker genes

`markerScatter` *Scatter Plot for Gene Expression*

Description

Generates an expression profile of each gene catetory in one sample against another, alternatively plot the regression line from linear modeling fitting.

Usage

```
markerScatter(expr, log = FALSE, samples, cate.gene, markers = NULL, pch = 19, cex = 0.5,
col = NULL, xlab = NULL, ylab = NULL, main = NULL, add.line = TRUE, text.cex = 1, legend.labels = NULL
...  )
```

Arguments

`expr`	a data frame with gene expression.
`log`	logical to determine if the gene expression data is log converted (add a small constant 2), default to FALSE.
`samples`	a vector of samples to compare on the x axis and y axis.
`cate.gene`	a list of the gene categories, alternatively output by `categorizeGene`.
`markers`	vector of marker genes to be highlighted in the plot. No gene is highlighted when it's `NULL`.
`pch, cex, col, xlab, ylab, main`	plot parameters, see details in `par`.
`add.line`	logical to determine if the linear model fitting line is added on the figure.
`text.cex`	font size for the text on `markers`, see details in `text`.
`legend.labels`	vector of labels for the marker legend.
`...`	other parameters in `plot`.

Details

Visualization of gene expression in the five categories under each pair-wised comparison.

Value

plot with gene expression profile.

Examples

```
#load the marker genes of somatic and primary cells
data(markers)
data(expr.filter)
#scatterplot
col = c("#abd9e9", "#2c7bb6", "#fee090", "#d7191c", "#fdae61")
markerScatter(expr = expr.filter, log = TRUE, samples = c("CB", "DMEC"),
              cate.gene = cate.gene[2:4], markers = markers, col = col[2:4],
              xlab = expression('log'[2]*' expression in CB (target)'),
             ylab = expression('log'[2]*' expression in DMEC (input)'),main = "")
```

`mouse.gene` *Gene regulatory network based mouse cell/tissue-specific gene sets*

Description

A list of 20 mouse cells/tissues-specific gene sets summarized from the gene regulatory network downloaded from the CellNet website.

Usage

```
data(mouse.gene)
```

Format

A list

Value

A list of 20 mouse cells/tissues-specific gene sets

`mouse.grn` *Mouse cell/tissue-specific gene-gene regulation*

Description

A list of 20 data frames (cells/tissues) with transcription factors (TF) to target genes (TG) regulation information.

Usage

```
data(mouse.grn)
```

Format

A list with 20 data frames

Value

A list of mouse gene regulatory information.

`mouse.tf` *Gene regulatory network based mouse cell/tissue-specific transcription factor (TF) regulated gene sets*

Description

A list of 1744 mouse cells/tissues-specific TF regulated gene sets summarized from the gene regulatory network downloaded from the CellNet website.

Usage

```
data(mouse.tf)
```

Format

A list

Value

A list of 1744 mouse cells/tissues-specific TF regulated gene sets.

`networkAnalyze` *Network Topological Analysis*

Description

This function analyzes the topological structure of gene regulation network (GRN) by calculating the "degree", "betweenness", "closeness" and "stress" parameters, and output the centrality values for given genes in each gene categories.

Usage

```
networkAnalyze(grn.data, cate.gene, centrality = c("degree", "betweenness",
"stress", "closeness"), mode = c("all","in", "out", "total"))
```

Arguments

`grn.data`	a data frame with two columns named "TF" and "TG" to specify the genes as transcription regulators (TF) and target genes (TG) being regulated.
`cate.gene`	a list of the five gene categories as nodes in the network, alternatively output by `categorizeGene`.
`centrality`	charactor string of "degree", "betweenness", "closeness" and "stress" to calculate the centrality of network built from input `grn.data`, see `degree`. `betweenness`, `closeness`, and `stresscent` for details.
`mode`	character string of "all", "in", "out" and "total", only used when `centrality` is "degree" or "closeness", see `degree`, `closeness` for details.

Value

data frame with genes and centrality scores.

Examples

```
# load the CellNet GRN and gene categories
data(human.grn)
data(cate.gene)

# specify a tissue-specifc network
tissue = "Hspc"
degree = networkAnalyze(human.grn[[tissue]], cate.gene = cate.gene,
                        centrality = "degree", mode ="all")
```

`SandlerFPKM` *RNA-seq data in FPKM A data frame containing 16692 gene in 7 samples from paper published by Sandler et.al. in 2014., samples are named from somatic dermal microvascular endothelial cell (DMEC) to induced multipotent haematopoietic progenitor (rEChMPP) cells and primary cord blood (CB) cells.*

Description

RNA-seq data in FPKM A data frame containing 16692 gene in 7 samples from paper published by Sandler et.al. in 2014., samples are named from somatic dermal microvascular endothelial cell (DMEC) to induced multipotent haematopoietic progenitor (rEChMPP) cells and primary cord blood (CB) cells.

Usage

```
data(SandlerFPKM)
```

Format

A data frame with 16692 rows and 7 columns specifying for genes and samples.

Value

A data frame with gene expression data.

`tissueGenes` *Cell/Tissue specific gene sets*

Description

A list of 126 cells/tissues-specific gene sets identified by SpeCond algorithm from 126 cells/tissues in Gene Enrichment Profiler database.

Usage

```
data(tissueGenes)
```

Format

A list

Value

A list with 126 cells/tissues-specific gene sets

`tissueGroup` *Tissue mapping to groups*

Description

A data frame with 126 cells/tissues and 30 C/T groups they belong to.

Usage

```
data(tissueGroup)
```

Format

A data frame with 126 rows (tissues) and 3 columns (Tissue, Tissue_abbr, Group)

Value

A data frame with 126 cells/tissues and 30 C/T groups they mapped to

Index