Package 'rgsepd'

June 29, 2022

Type Package

Title Gene Set Enrichment / Projection Displays

Version 1.29.0

Date 2022-03-22

Author Karl Stamm

Maintainer Karl Stamm <karl.stamm@gmail.com>

Description R/GSEPD is a bioinformatics package for R to help disambiguate transcriptome samples (a matrix of RNA-Seq counts at transcript IDs) by automating differential expression (with DESeq2), then gene set enrichment (with GOSeq), and finally a N-dimensional projection to quantify in which ways each sample is like either treatment group.

Depends R (>= 4.2.0), DESeq2, goseq (>= 1.28)

Imports gplots, biomaRt, org.Hs.eg.db, GO.db, SummarizedExperiment, AnnotationDbi

Suggests boot, tools, BiocGenerics, knitr, xtable

License GPL-3

biocViews ImmunoOncology, Software, DifferentialExpression, GeneSetEnrichment, RNASeq

VignetteBuilder knitr

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

git_branch master

git_last_commit 89afd22

git_last_commit_date 2022-04-26

Date/Publication 2022-06-29

R topics documented:

`rgsepd-package` *R/Gene Set Enrichment and Projection Displays*

Description

R/GSEPD is a Bioinformatics package for the R programming environment that helps you disambiguate transcriptome samples (Human RNA-Seq at RefSeq IDs) by automating differential expression (DESeq), then gene set enrichment (GOSeq hg19), and finally a N-dimensional projection to quantify in which ways each sample is like either treatment group. Many exploratory tables and plots are generated for you to browse the behavior of your samples in various gene-sets (defined by GO). Sets which significantly segregate your sample conditions by bootstrapped k-means are further explored.

See the Vignette for usage examples, and minimal examples within each function's reference.

Details

Package:	rgsepd
Type:	Package
Version:	1.15
Date:	2019-01-05
License:	GPL-3

Author(s)

Karl D. Stamm PhD <karl.stamm@gmail.com>

References

Initially a wrapper for DESeq2 and GOSeq, late-stage processing continues with a unique geneset based sample clustering.

See Also

Example data is public human RNA-Seq from Illumina Human Bodymap2 project, aligned to GRCh37 by Ensembl and downloaded from ftp://ftp.ensembl.org/pub/release-70/bam/homo_sapiens/genebuild/ then read counts are collected by coverageBed using RefSeq.GTF. We downloaded Adipose,Blood,Heart and Skeletal Muscle, and downsampled each to one third to create artificial replicates.

See the Vignette for usage walkthrough and results summaries.

Examples

```
data("IlluminaBodymap")
data("IlluminaBodymapMeta")
set.seed(1000) #fixed randomness
isoform_ids <- Name_to_RefSeq(c("HIF1A","EGFR","MYH7","CD33","BRCA2"))
rows_of_interest <- unique( c( isoform_ids ,
                               sample(rownames(IlluminaBodymap),
                                      size=2000,replace=FALSE)))
G <- GSEPD_INIT(Output_Folder="OUT",
              finalCounts=round(IlluminaBodymap[rows_of_interest , ]),
              sampleMeta=IlluminaBodymapMeta,
              COLORS=c("green","black","red"))
G <- GSEPD_ChangeConditions( G, c("A","B"))
# G <- GSEPD_Process( G ) #would run DESeq2 and GOSeq and GSEPD comparing conditions A and B
```

`DisplayName` *DisplayName*

Description

Convert a transcript id number to the corresponding gene name, where available.

Usage

```
DisplayName(txid)
```

Arguments

`txid`	The transcript id number, or a vector thereof.

Value

The gene's human-readable name.

Note

Uses org.Hs.eg.db and pulls the first Entrez Gene ID, then that ID's associated HGNC.

Examples

```
DisplayName("NM_005228")
```

`ExtractProjection` *ExtractProjection*

Description

This function takes a completed GSEPD object with sample data, and a set of gene identifiers and produces the projection of sample expression in the sub-space.

Usage

```
ExtractProjection(GSEPD, txids, DRAWING=FALSE, GN=c(1,2), PRINTING=FALSE, plotTitle="")
```

Arguments

`GSEPD`	The GSEPD parameter object. Must be post-Process.
`txids`	The transcript IDs, generally REFSEQ identifiers corresponding to rows of the counts table for this a projection is desired. In normal usage these are based on a GO Term.
`DRAWING`	Boolean flag to draw a plot of the projection.
`GN`	The gene numbers: which items of the 'txids' list are to be drawn. Only the first two are used. If Drawing=FALSE, this parameter is irrelevant.
`PRINTING`	Boolean flag to print some debug information.
`plotTitle`	A name for this set of genes, serves as the plot's main title.

Details

Primary gene set projection tool. This function calculates the vector projection and axis in a N-dimensional space of gene expression for a set of samples. When DRAWING=TRUE you will get some diagrams of the expression normalized counts.

Value

Returns a list object with four values for each sample.

`alpha`	Distance along the axis from group1 to group2, generally 0-1, as in percent. Samples within group 1 should average zero, and samples in group 2 should average one.

beta	Distance from the samples to the axis. This is a measure of goodness of fit, when the value is zero it means the sample is a linear interpolation between the comparison groups. When the value is high, the sample is not along the n-dimensional axis.
gamma1	Distance from the samples to the center of group1
gamma2	Distance from the samples to the center of group2
Validity.Score	A score, 0% through 100%, of the segregation validity for this gene set among the two sample test groups.
Validity.P	The validity score's associated p-value, empirically calculated chance of a random sample assignment creating such a strong score.

Examples

```
data("IlluminaBodymap")
data("IlluminaBodymapMeta")
set.seed(1000) #fixed randomness
isoform_ids <- Name_to_RefSeq(c("HIF1A","EGFR","MYH7","CD33","BRCA2"))
rows_of_interest <- unique( c( isoform_ids ,
                               sample(rownames(IlluminaBodymap),
                                      size=2000,replace=FALSE)))
G <- GSEPD_INIT(Output_Folder="OUT",
              finalCounts=round(IlluminaBodymap[rows_of_interest , ]),
              sampleMeta=IlluminaBodymapMeta,
              COLORS=c("green","black","red"))

G <- GSEPD_ChangeConditions( G, c("A","B")) #set testing groups first!
G <- GSEPD_Process( G ) #have to have processed results to plot them

 # looking at genes 2 and 3 will show us a view in dimensions "EGFR" and "MYH7"
 # and an axis through five dimensional space.
ExtractProjection(GSEPD=G, txids=isoform_ids,
  DRAWING=TRUE, PRINTING=TRUE, GN=c(2,3))
```

GSEPD_ChangeConditions

GSEPD_ChangeConditions

Description

This function is an interface to set which samples are the test conditions. Don't forget to GSEPD_Process() after changing settings. If you want to systematically try each condition pairing, try GSEPD_ProcessAll()

Usage

```
GSEPD_ChangeConditions(GSEPD, newConditions)
```

Arguments

GSEPD	Parameters object.
newConditions	a two-item vector matching some of your sampleMeta$Conditions

Details

Interface will check if the conditions are known, then set the C2T value.

Value

Returns the GSEPD parameter object with its mode set via the C2T and Conditions element of the named list. These tell later steps which sample conditions you intend on comparing.

See Also

GSEPD_Example

Examples

```
data("IlluminaBodymap")
data("IlluminaBodymapMeta")
set.seed(1000) #fixed randomness
isoform_ids <- Name_to_RefSeq(c("HIF1A","EGFR","MYH7","CD33","BRCA2"))
rows_of_interest <- unique( c( isoform_ids ,
                               sample(rownames(IlluminaBodymap),
                                      size=2000,replace=FALSE)))
G <- GSEPD_INIT(Output_Folder="OUT",
              finalCounts=round(IlluminaBodymap[rows_of_interest , ]),
              sampleMeta=IlluminaBodymapMeta,
              COLORS=c("green","black","red"))
     ConditionsToTest <- c("A","B")
G <- GSEPD_ChangeConditions( G, ConditionsToTest  )
#G <- GSEPD_Process( G ) #would test samples A vs samples B
G <- GSEPD_ChangeConditions( G, c("A","C"))
#G <- GSEPD_Process( G ) #would test samples A vs samples C
```

GSEPD_ChangeOutput *GSEPD_ChangeOutput*

Description

Update the stored output folder designation, and create it if necessary. This is useful if you want to change some LIMIT parameters and re-run the pipeline. Don't forget to GSEPD_Process() after changing settings.

Usage

```
GSEPD_ChangeOutput(GSEPD, newFolder)
```

Arguments

GSEPD	The initial GSEPD parameter object to update the output folder of.
newFolder	The new output folder to be created.

Value

Returns the updated GSEPD parameter object.

Author(s)

karl.stamm@gmail.com

Examples

```
data("IlluminaBodymap")
data("IlluminaBodymapMeta")
set.seed(1000) #fixed randomness
isoform_ids <- Name_to_RefSeq(c("HIF1A","EGFR","MYH7","CD33","BRCA2"))
rows_of_interest <- unique( c( isoform_ids ,
                               sample(rownames(IlluminaBodymap),
                                      size=2000,replace=FALSE)))
G <- GSEPD_INIT(Output_Folder="OUT",
              finalCounts=round(IlluminaBodymap[rows_of_interest , ]),
              sampleMeta=IlluminaBodymapMeta,
              COLORS=c("green","black","red"))
G <- GSEPD_ChangeConditions( G, c("A","B")) #set testing groups first!

G<- GSEPD_ChangeOutput(G, "Output2")
#G <- GSEPD_Process( G ) #would output to folder Output2
#now tweak some settings and re-do
G$LIMIT$LFC <- 0.25 #lower than default log-fold-change limit
G<- GSEPD_ChangeOutput(G, "Output-Low")
#G <- GSEPD_Process( G ) #would output to folder Output-Low
```

GSEPD_DEGHeatmap *Differentially Expressed Genes Heatmap*

Description

Generates a gene-by-subject heatmap plot of differentially expressed genes.

Usage

```
GSEPD_DEGHeatmap(G)
```

Arguments

G	The GSEPD master object carries sample information and gene expression data. It should have already run Process() to be eligible. Parameters regarding differential expression limits are set within the G$LIMIT list object.

Details

After GSEPD_Process() has created differential expression tables with known filenames, this function can read those tables and make heatmap plots for a subset of genes. We use the N most significant genes, specified by the `MAX_Genes_for_Heatmap` parameter of the passed GSEPD object.

Value

This function doesn't return anything. If successful, four PDF files are created. HM and HM- are all subjects from sampleMeta and finalCounts, HMS and HMS- are only those in the test groups. The hyphen indicates a smaller unlabeled figure. In each case the data is manipulated as in `GSEPD_Heatmap()` such that complete linkage clustering is performed on z-score normalized genes using the normalized counts directly from DESeq2::varianceStabilizingTransformation, which are displayed in the labeled figures.

Examples

```
data("IlluminaBodymap")
data("IlluminaBodymapMeta")
set.seed(1000) #fixed randomness
isoform_ids <- Name_to_RefSeq(c("HIF1A","EGFR","MYH7","CD33","BRCA2"))
rows_of_interest <- unique( c( isoform_ids ,
                               sample(rownames(IlluminaBodymap),
                                      size=2000,replace=FALSE)))
G <- GSEPD_INIT(Output_Folder="OUT",
              finalCounts=round(IlluminaBodymap[rows_of_interest , ]),
              sampleMeta=IlluminaBodymapMeta,
              COLORS=c("green","black","red"))
G <- GSEPD_ChangeConditions( G, c("A","B")) #set testing groups first!
G <- GSEPD_Process( G ) #have to have processed results to plot them
GSEPD_DEGHeatmap(G) # all parameters automatic
```

GSEPD_Export_DESeq *Export DESeqDataSet object*

Description

Converts from the internal matrices to a DESeq standard object.

Usage

```
GSEPD_Export_DESeq(G)
```

Arguments

G	The GSEPD list object to extract a DeseqDataSet from.

Details

Using the given GSEPD object's finalCounts and sampleMeta, a simple DESeqDataSet object is created with the default design matrix. Provided for interoperability with other analysis packages.

Value

an object of class DESeqDataSet

References

DESeq2

Examples

```
data("IlluminaBodymap")
data("IlluminaBodymapMeta")
set.seed(1000) #fixed randomness
isoform_ids <- Name_to_RefSeq(c("HIF1A","EGFR","MYH7","CD33","BRCA2"))
rows_of_interest <- unique( c( isoform_ids ,
                               sample(rownames(IlluminaBodymap),
                                      size=2000,replace=FALSE)))
G <- GSEPD_INIT(Output_Folder="OUT",
              finalCounts=round(IlluminaBodymap[rows_of_interest , ]),
              sampleMeta=IlluminaBodymapMeta,
              COLORS=c("green","black","red"))
G <- GSEPD_ChangeConditions( G, c("A","B")) #set testing groups first!
dds <- GSEPD_Export_DESeq(G)
print(dds)
```

GSEPD_Heatmap *GSEPD_Heatmap*

Description

Plots the heatmap to the standard display. Uses heatmap.2 from gplots to display selected genes' expression level.

Usage

```
GSEPD_Heatmap(G,genes,cap_range=3,cellnote="log10")
```

Arguments

G	The GSEPD parameter object. Must be post Process.
genes	rownames of finalCounts, usually isoform ID#s.
cap_range	z-score of most extreme color
cellnote	display the log10 values in each cell. No other options are supported.

Details

Will use GSEPD$COLORFUNCTION scaled between samples of type GSEPD$Conditions in GSEPD$sampleMeta, including others in the mix. The heatmap's dendrograms (margin trees) are computed by the heatmap.2() function's default method hclust() on the supplied data, resulting in complete linkage heirarchical clustering. Because the magnitude of gene expression varies across a wide range, and we're interested in patterns more than scale, we first normalize each gene(row) by subtracting the mean, dividing by the standard deviation, and capping the min and max to the parameter cap_range=3. The heatmap function is run with no further scaling, ensuring genes with similar differential expression profiles are clustered together. The numbers written in each cell of the heatmap are simply the normalized counts directly from DESeq2::varianceStabilizingTransformation.

Value

No return value: generates a figure.

Examples

```
data("IlluminaBodymap")
data("IlluminaBodymapMeta")
set.seed(1000) #fixed randomness
isoform_ids <- Name_to_RefSeq(c("HIF1A","EGFR","MYH7","CD33","BRCA2"))
rows_of_interest <- unique( c( isoform_ids ,
                               sample(rownames(IlluminaBodymap),
                                      size=2000,replace=FALSE)))
G <- GSEPD_INIT(Output_Folder="OUT",
              finalCounts=round(IlluminaBodymap[rows_of_interest , ]),
              sampleMeta=IlluminaBodymapMeta,
              COLORS=c("green","black","red"))

G <- GSEPD_ChangeConditions( G, c("A","B")) #set testing groups first!
G <- GSEPD_Process( G ) #have to have processed results to plot them

GSEPD_Heatmap(G, genes=sample(rownames(G$finalCounts),8) )
```

GSEPD_INIT *Initialization*

Description

Initializes the system, here you will pass in the count dataset and the sample metadata, before any GSEPD processing. Return value is a named list holding configurable parameters.

Usage

```
GSEPD_INIT(Output_Folder = "OUT", finalCounts = NULL, sampleMeta = NULL,
DESeqDataSet = NULL, renormalize = TRUE, vstBlind=TRUE,
  COLORS = c("green", "gray", "red"),
  C2T = "x" )
```

Arguments

Output_Folder	Specify the subdirectory to hold output/generated files. Defaults to "OUT".
finalCounts	This must be a matrix of count data, rows are transcript IDs and columns are samples.
sampleMeta	The sampleMeta matrix must be passed here. It is a data frame with a row for each sample in the finalCounts matrix. Some required columns are SHORTNAME= sample nicknames; Condition= treatment group for differential expression; and Sample are the column names of finalCounts. Other columns are permitted to facilitate subsetting (not automatically supported).
DESeqDataSet	Data may also be included in the format of a DESeqDataSet object, this is mutually exclusive of the finalCounts/sampleMeta scheme.
renormalize	Boolean performance flag. Default is TRUE, which causes a normalized counts table to be computed from your given raw reads 'finalCounts'. If you set this to FALSE, then the normCounts table is preloaded with the given finalCounts input matrix, short circuiting the built-in DESeq VST, and allowing the user to specify some sort of pre-normalized dataset.
vstBlind	Exposes the option from DESeq2 to change the way varianceStabilizingTransformation works. According to the DESeq manual: blind=TRUE should be used for comparing samples in an manner unbiased by prior information on samples ... If many of genes have large differences in counts due to the experimental design, it is important to set blind=FALSE for downstream analysis.
COLORS	A three element vector of colors to make the heatmaps, the first element is the under-expressed genes, and the third element is the over-expressed genes. Defaults to green-red through gray.
C2T	This symbol is used in the filenames to delimit sample groups.

Details

This function sets up the master parameter object, and therefore must be called first. This object includes all configurable parameters you can change before running the pipeline. Count data should be provided in the finalCounts matrix, with phenotype and sample data in the sampleMeta matrix. Optionally, these data may be packages in a DESeqDataSet instead. Rows with no expression are dropped at the point of loading.

Value

Returns the GSEPD named list master object, to be used in subsequent function calls.

See Also

GSEPD_Process

Examples

```
data("IlluminaBodymap")
data("IlluminaBodymapMeta")
isoform_ids <- Name_to_RefSeq(c("HIF1A","EGFR","MYH7","CD33","BRCA2"))
rows_of_interest <- unique( c( isoform_ids ,
                               sample(rownames(IlluminaBodymap),
                                      size=2000,replace=FALSE)))
G <- GSEPD_INIT(Output_Folder="OUT",
              finalCounts=round(IlluminaBodymap[rows_of_interest , ]),
              sampleMeta=IlluminaBodymapMeta,
              COLORS=c("green","black","red"))
#now ready to run:
# G<-GSEPD_ProcessAll(G);
```

GSEPD_PCA_Plot *Principle Components Analysis figure generation*

Description

After processing the pipeline, users may want to have further PCA figures generated. This function takes a completed GSEPD object and generates informative figures, based on the differentially expressed genes.

Usage

```
GSEPD_PCA_Plot(GSEPD, customColors=FALSE)
```

Arguments

GSEPD	The master object, it should have already been run through GSEPD_Process().
customColors	a boolean value, when FALSE, default behavior is to color points to match the test conditions. When TRUE, use sampleMeta$CustomColor column for a sample-by-sample user specification. This behavior disables the built-in legend.

Value

No return value. Generates files.

See Also

GSEPD_PCA_Spec

Examples

```
data("IlluminaBodymap")
data("IlluminaBodymapMeta")
set.seed(1000) #fixed randomness
isoform_ids <- Name_to_RefSeq(c("HIF1A","EGFR","MYH7","CD33","BRCA2"))
rows_of_interest <- unique( c( isoform_ids ,
                               sample(rownames(IlluminaBodymap),
                                      size=2000,replace=FALSE)))
G <- GSEPD_INIT(Output_Folder="OUT",
              finalCounts=round(IlluminaBodymap[rows_of_interest , ]),
              sampleMeta=IlluminaBodymapMeta,
              COLORS=c("green","black","red"))
G <- GSEPD_ChangeConditions( G, c("A","B")) #set testing groups first!
G <- GSEPD_Process( G ) #have to have processed results to plot them
GSEPD_PCA_Plot(G)
```

GSEPD_PCA_Spec *Specialized PCA Plot*

Description

After processing the pipeline, users may want to have further PCA figures generated. This function takes a completed GSEPD object and generates informative figures. This function includes parameters to specify a particular GO-Term of interest.

Usage

```
GSEPD_PCA_Spec(GSEPD, GOT, MDATA = NULL, customColors=FALSE)
```

Arguments

GSEPD	The master GSEPD object, post-processed.
GOT	The GO-Term you'd like to specifically analyse. It should be found in the .MERGE file.
MDATA	Optionally, pass in the .MERGE dataset, if missing, we'll try to read the already-processed file from the output directory. This option exists because reading that file repeatedly is quite slow, so you're recommended to read it in once in advance if you intend on making more than a couple GO-Term specific plots.
customColors	a boolean value, when FALSE, default behavior is to color points to match the test conditions. When TRUE, use sampleMeta$CustomColor column for a sample-by-sample user specification. This behavior disables the built-in legend.

Value

No return value. Generates files.

Note

This function uses either princomp() or prcomp() as neccesary, depending on sample count vs gene count.

See Also

GSEPD_PCA_Plot

Examples

```
data("IlluminaBodymap")
data("IlluminaBodymapMeta")
set.seed(1000) #fixed randomness
isoform_ids <- Name_to_RefSeq(c("HIF1A","EGFR","MYH7","CD33","BRCA2"))
rows_of_interest <- unique( c( isoform_ids ,
                               sample(rownames(IlluminaBodymap),
                                      size=2000,replace=FALSE)))
G <- GSEPD_INIT(Output_Folder="OUT",
              finalCounts=round(IlluminaBodymap[rows_of_interest , ]),
              sampleMeta=IlluminaBodymapMeta,
              COLORS=c("green","black","red"))
G <- GSEPD_ChangeConditions( G, c("A","B")) #set testing groups first!
G <- GSEPD_Process( G ) #have to have processed results to plot them

GOT <- "GO:0012345" # specify a GO Term you'd like to review

#it should be present in the MERGE file.
MergeFile <- list.files(G$Output_Folder, pattern="MERGE")[1]
MDATA<-read.csv(sprintf("%s%s%s", G$Output_Folder, .Platform$file.sep, MergeFile),
    as.is=TRUE,header=TRUE)

GOT=MDATA$category[1] #choose a GO term that is definitely in the output data.

GSEPD_PCA_Spec(G, GOT,MDATA=MDATA)
```

GSEPD_Process *Processing*

Description

Primary interface, use this function to kick off the pipeline.

Usage

```
GSEPD_Process(GSEPD)
```

Arguments

GSEPD	The initialized GSEPD master object to operate on.

Details

Runs the pipeline. If any files are already present matching the generated filenames, they will be reused. If you changed a parameter that would alter the generated filenames, new ones are created. If a customization parameter is not part of the filename (like a p-value cutoff), you should change the output folder to keep new files separate.

Value

Returns the GSEPD object post-processed, for use in further plotting functions. Optional.

See Also

GSEPD_INIT

Examples

```
data("IlluminaBodymap")
data("IlluminaBodymapMeta")
set.seed(1000) #fixed randomness
isoform_ids <- Name_to_RefSeq(c("HIF1A","EGFR","MYH7","CD33","BRCA2"))
rows_of_interest <- unique( c( isoform_ids ,
                               sample(rownames(IlluminaBodymap),
                                      size=2000,replace=FALSE)))
G <- GSEPD_INIT(Output_Folder="OUT",
              finalCounts=round(IlluminaBodymap[rows_of_interest , ]),
              sampleMeta=IlluminaBodymapMeta,
              COLORS=c("green","black","red"))
G <- GSEPD_ChangeConditions( G, c("A","B")) #set testing groups first!
#G <- GSEPD_Process( G ) #would run DESeq2 and GOSeq and GSEPD comparing conditions A and B
```

GSEPD_ProcessAll *GSEPD_ProcessAll*

Description

Runs each pairing within GSEPD$sampleMeta$Conditions.

Usage

```
GSEPD_ProcessAll(G)
```

Arguments

G	The GSEPD object from GSEPD_INIT()

Details

Set your GSEPD$LIMIT before running each pairwise comparison.

Value

Returns the last GSEPD object.

See Also

GSEPD_Process

Examples

```
  data("IlluminaBodymap")
  data("IlluminaBodymapMeta")
  head(IlluminaBodymap)
  set.seed(1000) #fixed randomness
  isoform_ids <- Name_to_RefSeq(c("HIF1A","EGFR","MYH7","CD33","BRCA2"))
  rows_of_interest <- unique( c( isoform_ids ,
                                 sample(rownames(IlluminaBodymap),
                                        size=2000,replace=FALSE)))
  G <- GSEPD_INIT(Output_Folder="OUT",
                  finalCounts=round(IlluminaBodymap[rows_of_interest , ]),
                  sampleMeta=IlluminaBodymapMeta,
                  COLORS=c("green","black","red"))

#  G <- GSEPD_ProcessAll( G ) #would run across all pairs of G$Condition
```

GSEPD_PullDEG *Pull Differentially Expressed Genes*

Description

After processing, if you want to easily access the differentially expressed transcript listing, this function will read in the default generated files, and apply filters as specified by the GSEPD master object (default p-values).

Usage

```
GSEPD_PullDEG(GSEPD, PTHRESH)
```

Arguments

GSEPD	The master object should have been processed already such that differentially expressed genes are readily available.
PTHRESH	Specify the degree of stringency.

Value

Returns a vector of ID#, suitable to row-subsetting of the finalCounts table.

Examples

```
data("IlluminaBodymap")
data("IlluminaBodymapMeta")
set.seed(1000) #fixed randomness
isoform_ids <- Name_to_RefSeq(c("HIF1A","EGFR","MYH7","CD33","BRCA2"))
rows_of_interest <- unique( c( isoform_ids ,
                               sample(rownames(IlluminaBodymap),
                                      size=2000,replace=FALSE)))
G <- GSEPD_INIT(Output_Folder="OUT",
              finalCounts=round(IlluminaBodymap[rows_of_interest , ]),
              sampleMeta=IlluminaBodymapMeta,
              COLORS=c("green","black","red"))
G <- GSEPD_ChangeConditions( G, c("A","B")) #set testing groups first!
G <- GSEPD_Process( G ) #have to have processed results to plot them

Significant_Genes <- GSEPD_PullDEG(G, PTHRESH=0.0250)
#then do more with these identifiers:
print(Significant_Genes)
# GSEPD_Heatmap(G, genes= Significant_Genes )
```

IlluminaBodymap *Sample RNA-Seq Counts data*

Description

A collection of counts datasets from Illumina Human Bodymap 2.0, one sample each for adipose, blood, heart and skeletal_muscle. Four technical replicates are created by downsampling the original Illumina data. Alignment was performed by Ensembl, so the source of this dataset is ftp://ftp.ensembl.org/pub/release-70/bam/homo_sapiens/genebuild . Each of the four Human Bodymap samples are downsampled four times. Read counts are collected with Bedtools CoverageBed and a RefSeq exon annotation.

Usage

```
data(IlluminaBodymap)
```

Format

A data frame with 37653 observations on the following 16 variables.

`adipose.1` Illumina Human Bodymap 2 'Adipose' sample, downsampled to one-third.

`adipose.2` Illumina Human Bodymap 2 'Adipose' sample, downsampled to one-third.

`adipose.3` Illumina Human Bodymap 2 'Adipose' sample, downsampled to one-third.

`adipose.4` Illumina Human Bodymap 2 'Adipose' sample, downsampled to one-third.

`blood.1` Illumina Human Bodymap 2 'Blood' sample, downsampled to one-third.

`blood.2` Illumina Human Bodymap 2 'Blood' sample, downsampled to one-third.

`blood.3` Illumina Human Bodymap 2 'Blood' sample, downsampled to one-third.

`blood.4` Illumina Human Bodymap 2 'Blood' sample, downsampled to one-third.

`heart.1` Illumina Human Bodymap 2 'Heart' sample, downsampled to one-third.

`heart.2` Illumina Human Bodymap 2 'Heart' sample, downsampled to one-third.

`heart.3` Illumina Human Bodymap 2 'Heart' sample, downsampled to one-third.

`heart.4` Illumina Human Bodymap 2 'Heart' sample, downsampled to one-third.

`skeletal_muscle.1` Illumina Human Bodymap 2 'Skeletal Muscle' sample, downsampled to one-third.

`skeletal_muscle.2` Illumina Human Bodymap 2 'Skeletal Muscle' sample, downsampled to one-third.

`skeletal_muscle.3` Illumina Human Bodymap 2 'Skeletal Muscle' sample, downsampled to one-third.

`skeletal_muscle.4` Illumina Human Bodymap 2 'Skeletal Muscle' sample, downsampled to one-third.

Value

A numeric matrix of read-counts from RNA-Seq, measured at transcripts by coverageBed.

Source

http://www.ebi.ac.uk/arrayexpress/experiments/E-MTAB-513/

References

Illumina Human Bodymap 2.0. Ensembl etc.

Examples

```
data(IlluminaBodymap)
head(IlluminaBodymap,30)
```

IlluminaBodymapMeta *Metadata table for the included sample data*

Description

The metadata table required to inform GSEPD of the sample/condition and abbreviated names for each column of the included 'counts' dataset. You should mirror this table's structure for your dataset.

Usage

```
data(IlluminaBodymapMeta)
```

Format

A data frame with 16 observations on the following 3 variables.

`Sample` A vector of the column names in your counts table, for the included sample data, it's four tissue types repeated four times each. For your data this must correspond to the column labels in the counts table.

`Condition` The sample categorizations for use in differential expression, this should also be a vector the same length as the number of columns in your counts table. Here we have 'A' for each Adipose, 'B' for each muscle type, and 'C' for the blood samples.

`SHORTNAME` Abbreviated names for each sample to appear in plots.

Value

A dataframe of sample identifiers for the rgsepd::IlluminaBodymap matrix.

Examples

```
data(IlluminaBodymapMeta)
str(IlluminaBodymapMeta)
```

`Name_to_RefSeq` *Name to RefSeq*

Description

Lookup a HGNC symbol and return an appropriate NM##.

Usage

```
Name_to_RefSeq(x)
```

Arguments

`x`	The HGNC symbol(s) you wish to convert.

Details

Not found gene symbols will return NA or the empty string.

Value

The NM_### id numbers corresponding to the input gene names (HGNC symbols.)

Note

This routine relies on bioconductor annotation package org.Hs.eg.db to ensure the most up to date mappings.

Examples

```
Name_to_RefSeq("LSMEM2")
#should return NM_153215
```

Index