

Package ‘bioEAT’

June 23, 2021

Title bioExploreAnnotateTranslate

Version 1.0.0

Description The aim of bioEAT is to provide functionality to explore data, make gene annotation exhaustive and simplify the gene IDs translation between different organisms.

For the data exploration:

- find the non-overlapping elements of two vectors with `outersect()`
- get main n pca components with `get_main_pca()`
- calculate the logFold change between two dataframe columns and order the result with `logfc()`

For the data annotation:

- use OrgDB and maRt in one function with `conv_ids_full()`

License MIT + file LICENSE

BugReports <https://github.com/Troshchk/bioEAT/issues>

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

biocViews

Imports openxlsx, clusterProfiler, Biobase, biomaRt, factoextra, gtools, plyr, reshape2, ggplot2

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Ksenia Troshchenkova [aut, cre]

Maintainer Ksenia Troshchenkova <ksenia.trs@gmail.com>

R topics documented:

<code>conv_ids_cp</code>	2
<code>conv_ids_full</code>	2
<code>conv_ids_mart</code>	3
<code>get_main_pca</code>	4
<code>groupGO_all_groups</code>	4
<code>logfc_cols</code>	5
<code>outersect</code>	6
<code>sh_entropy</code>	7
<code>test_input_type</code>	8

Index**9**

conv_ids_cp	<i>Converting gene IDS using orgDB</i>
-------------	--

Description

This function returns a dataframe with the translated IDs.

Usage

```
conv_ids_cp(input, from, to, db)
```

Arguments

input	vector of the IDs to convert
from	input ID type
to	output ID type or vector of output ID types
db	annotation database

Details

This function takes a vector of IDs and translated them from the input format to the output format using the selected orgDB. In case the orgDB is not installed before launching the function, the function will exit with error.

Value

dataframe with the translated IDs

Examples

```
conv_ids_cp(rownames(df), "ENSEMBL", c("ENTREZID", "GENENAME"), "org.Mmu.eg.db")
conv_ids_cp(c("IFNA1", "IFNA13", "SLC2A3"), "SYMBOL", "ENSEMBL", "org.Mmu.eg.db")
```

conv_ids_full	<i>Converting gene IDS using orgDB and maRt</i>
---------------	---

Description

This function returns a dataframe with translated IDs and NA row for the not annotated genes.

Usage

```
conv_ids_full(
  input,
  db_cluster_profiler,
  from_cluster_profiler,
  mart,
  from_mart
)
```

Arguments

input	vector of IDs
db_cluster_profiler	annotation clusterProfiler database
from_cluster_profiler	input clusterProfiler ID type: ENSEMBL, ENTREZID, SYMBOL
from_mart	annotation mart

Details

This function takes a vector of IDs and translated them from the input format to the output format using the selected orgDB. The IDs not found by OrgDB are further annotated by maRt. If the ID was not found it will be at the end of the output dataframe with NAs. This function works only with the IDs: ENSEMBL ID, SYMBOL, ENTREZID In case the orgDB is not installed before launching the function, the function will exit with error.

Value

dataframe with the ENSEMBL ID, SYMBOL, ENTREZID, GENENAME(description); NAs are not dropped

Examples

```
conv_ids_full(rownames(dataNorm_df), "org.Mmu.eg.db", "ENSEMBL",
"mmulatta_gene_ensembl", "ensembl_gene_id")
conv_ids_full(c("IFNA13", "SLC2A3", "CD45RA", "CDY2A", "IGHM", "IGKC"), "org.Mmu.eg.db",
"SYMBOL", "mmulatta_gene_ensembl", "external_gene_name")
```

conv_ids_mart

*Converting gene IDS using maRt***Description**

This function returns a dataframe with the translated IDs. #’@details This function takes a vector of IDs and translated them from the input format to the output format using the selected maRt.

Usage

```
conv_ids_mart(input, mart, from, to)
```

Arguments

input	vector of IDs
mart	annotation maRt
from	input ID type
to	output ID type or vector of output ID types

Value

deataframe with translated IDs

Examples

```
conv_ids_mart(c("CDY2A", "IGHM", "IGKC"), "mmulatta_gene_ensembl", "external_gene_name",
c("entrezgene_id", "ensembl_gene_id"))
```

get_main_pca

Getting the main n PCA components

Description

This function returns a vector of top n contributors

Usage

```
get_main_pca(df, n)
```

Arguments

df dataframe with normalized counts (column = sample, row = transcript)
n number of top contributors

Value

vector of top n contributors

Examples

```
get_main_pca(df, 3)
```

groupGO_all_groups

GO analysis and plots for BP, CC, MF

Description

This function returns table for GO results for BP, CC, MF with gene symbols. The function may also create excel with the GO result tables and the PDFs with the barplots of the results.

Usage

```
groupGO_all_groups(
  gene_entrezIDs,
  orgDB,
  name = "",
  bp = 2,
  cc = 3,
  mf = 3,
  xls = TRUE,
  pdf = TRUE
)
```

Arguments

gene_entrezIDs	vector of entrezIDs
orgDB	db to use for GO
name	name of the output file
bp	level of the biological process with default 2
cc	level of the cellular component with default 3
mf	level of the molecular function with default 3
xls	logical parameter to create excel with with default TRUE
pdf	logical parameter to create pdf with with default TRUE

Value

table with groupGO result arranged as list of dataframes

excel file with groupGO result and the pdf with the groupGO result barplot per biological process, cellular component and molecular function

Examples

```
groupGO_all_groups(entrez_ids_list, "org.Mmu.eg.db", "out_name", mf = 1, xls = FALSE, pdf = FALSE)
```

logfc_cols	<i>Calculating logFC between two columns of a dataframe</i>
------------	---

Description

This function returns dataframe with 3 columns col1, col2, log2FC. Mind the column names order: the logFC is always calculated between the first and the second input.

Usage

```
logfc_cols(data, col1, col2)
```

Arguments

col1	name or number of the column in df
col2	name or number the column in df
df	dataframe with normalized counts

Details

As an example, when this function is useful is lack of data. In order to find the differentially expressed genes more samples are needed. In case of fewer samples an alternative approach is to manually calculate the expected behaviour of the desired output genes. For example, we expect a gene to have lfc > 3 for samples 1vs2 and 5vs6. In this case the logfc_cols() function could be used to calculate and order individual logFold changes, which could be further used to select the genes with the desired behaviour.

Value

dataframe with 3 columns col1, col2, log2FC

Examples

```
logfc_cols(dataNorm_df, "column_name", "5")  
logfc_cols(dataNorm_df, "column_name", 5)  
logfc_cols(dataNorm_df, 1, 5)
```

outersect	<i>Outersect</i>
-----------	------------------

Description

This function returns the non-overlapping values from the two input vectors.

Usage

```
outersect(x, y)
```

Arguments

x	vector
y	vector

Details

It checks for values from the first inout not appearing in the second input and for the values from the second inout not appearing in the first input.

Value

sorted vector of non-overlapping values from lists x and y

Examples

```
outersect(df_1[,2],df_2[,2])  
outersect(1:5, 4:7)
```

`sh_entropy`*Calculating Shannon entropy of each sample, Shannon entropy values and transcript frequency visualization*

Description

This function returns a list, containing dataframe with the entropy values per sample, entropy plot and transcript frequency plot

Usage

```
sh_entropy(  
  df,  
  name = "Shannon entropy",  
  pdf = TRUE,  
  relevel = FALSE,  
  levels = NULL  
)
```

Arguments

<code>df</code>	input dataframe with transcripts in rows and samples in columns
<code>name</code>	name of the output pdf with default "Shannon entropy", used when pdf is TRUE
<code>pdf</code>	logical parameter to create pdf with with default TRUE
<code>relevel</code>	logical parameter to reorder the samples order in the plots with default FALSE
<code>levels</code>	vector with the correct order of the samples with default NULL; will be used only if relevel is TRUE

Details

This function takes a dataframe with transcripts in rows and samples in columns as an input and calculates the Shannon entropy for each sample and the frequency of the transcript in the sample. Further it visualizes the entropy levels and the frequencies. The transcripts are reordered by their frequencies. Function can also print the plots to pdf if pdf parameter is TRUE.

Value

list with the entropy levels and plots

Examples

```
sh_entropy(df, "name", relevel = TRUE, levels = c(seq(1, ncol(df), 1)))  
sh_entropy(df)  
sh_entropy(df, pdf = FALSE)
```

test_input_type	<i>Help function for logfc_cols()</i>
-----------------	---------------------------------------

Description

This function checks whether the input exists in the df:

- for strings checks the column names
- for numbers checks whether it is smaller than the number of columns

Usage

```
test_input_type(data, input)
```

Arguments

data	input df
input	column number or column to check

Value

error or pass

Index

conv_ids_cp, [2](#)
conv_ids_full, [2](#)
conv_ids_mart, [3](#)

get_main_pca, [4](#)
groupG0_all_groups, [4](#)

logfc_cols, [5](#)

outersect, [6](#)

sh_entropy, [7](#)

test_input_type, [8](#)