Package ‘func2vis’

March 16, 2023

Type Package

Title Clean and Visualize Over Expression Results from 'ConsensusPathDB'

Version 1.0-3

Date 2023-03-16

Author Raghvendra Mall [aut, cre]

Maintainer Raghvendra Mall <raghvendra5688@gmail.com>

Repository CRAN

Description
Provides functions to have visualization and clean-up of enriched gene ontologies (GO) terms, protein complexes and pathways (obtained from multiple databases) using 'ConsensusPathDB' from gene set over-expression analysis. Performs clustering of pathway based on similarity of over-expressed gene sets and visualizations similar to Ingenuity Pathway Analysis (IPA) when up and down regulated genes are known. The methods are described in a paper currently submitted by Orecchioni et al, 2020 in Nanoscale.

License GPL (>= 3)

LazyLoad true

Depends ggplot2, igraph, devtools, ggrepel, grDevices, randomcoloR, R (>= 4.0)

NeedsCompilation yes

Date/Publication 2023-03-16 17:30:02 UTC

RoxygenNote 7.1.1

R topics documented:

`clean_go_terms` *Clean Gene Ontologies (GO) Terms*

Description

Clean set of enriched goterms obtained from 'ConsensusPathDB' for gene set overexpression analysis. We also append two columns indicating the number of up-regulated and number of down-regulated genes based on fold change information available in data frame case_vs_ctrl.

Usage

```
clean_go_terms(df_case_vs_ctrl, df_goterms)
```

Arguments

`df_case_vs_ctrl`
: Data frame which has at least 2 columns: <gene,fc>. Here gene represents the set of genes which are differentially expressed between case and control. Here fc represents the fold-change value for each gene.

`df_goterms`
: The tab-separated data frame with the goterms information obtained after performing gene set overexpression analysis using 'ConsensusPathDB'.

Value

Returns clean enriched GO terms data frame.

Note

rmall@hbku.edu.qa

Author(s)

Raghvendra Mall

See Also

See Also as `clean_pc`, `plot_go_terms`

Examples

```
data("t.tests.treatment.sign")
data("enriched_goterms")
revised_goterms <- clean_go_terms(df_case_vs_ctrl=t.tests.treatment.sign,
                                  df_goterms = enriched_goterms)
print(head(revised_goterms))
```

---

clean_pathways *Clean Enriched Pathways*

Description

Clean set of enriched pathways obtained from 'ConsensusPathDB' for gene set overexpression analysis. We also append two columns indicating the number of up-regulated and number of down-regulated genes based on fold change information available in data frame case_vs_ctrl. We cluster pathways based on similarity of gene set using igraph's walktrap clustering algorithm. Within each cluster, pathways are ordered by most to least significant pathway in terms of p-values.

Usage

```
clean_pathways(df_case_vs_ctrl, df_pathway)
```

Arguments

`df_case_vs_ctrl`
: Data frame which has at least 2 columns: <gene,fc>. Here gene represents the set of genes which are differentially expressed between case and control. Here fc represents the fold-change value for each gene.

`df_pathway`
: The tab-separated data frame with the pathways information obtained after performing gene set overexpression analysis using 'ConsensusPathDB'.

Value

Returns clean enriched pathways data frame. The data frame has an additional column clusters highlighting the cluster to which each enriched pathway belongs.

Note

rmall@hbku.edu.qa

Author(s)

Raghvendra Mall

See Also

clean_go_terms, clean_pc

Examples

```
data("t.tests.treatment.sign")
data("enriched_pathways")
revised_pathway <- clean_pathways(df_case_vs_ctrl=t.tests.treatment.sign,
                                  df_pathway = enriched_pathways)
print(head(revised_pathway))
```

---

clean_pc *Clean Enriched Protein Complexes*

Description

Clean set of enriched protein complexes obtained from 'ConsensusPathDB' for gene set overexpression analysis. We also append two columns indicating the number of up-regulated and number of down-regulated genes based on fold change information available in data frame case_vs_ctrl.

Usage

```
clean_pc(df_case_vs_ctrl,df_pc)
```

Arguments

`df_case_vs_ctrl`
Data frame which has at least 2 columns: <gene,fc>. Here gene represents the set of genes which are differentially expressed between case and control. Here fc represents the fold-change value for each gene.

`df_pc` The tab-separated data frame with the protein complexes information obtained after performing gene set overexpression analysis using 'ConsensusPathDB'.

Value

Returns clean enriched protein complexes data frame.

Note

rmall@hbku.edu.qa

Author(s)

Raghvendra Mall

See Also

See Also as `clean_go_terms`, `plot_go_terms`

Examples

```
data("t.tests.treatment.sign")
data("enriched_pc")
revised_pc <- clean_pc(df_case_vs_ctrl=t.tests.treatment.sign,
                                  df_pc = enriched_pc)
print(head(revised_pc))
```

---

enriched_goterms *Sample Enriched Gene Ontologies (GO) Terms*

Description

This dataset highlights enriched gene ontologies (GO) terms identified by using ConsensusPathDB while performing overexpression analysis for a sample set of genes.

Usage

```
data("enriched_goterms")
```

References

Kamburov, A., Stelzl, U., Lehrach, H. and Herwig, R., 2013. The ConsensusPathDB interaction database: 2013 update. Nucleic acids research, 41(D1), pp.D793-D800.

Examples

```
data(enriched_goterms)
## maybe str(enriched_goterms) ;
```

---

enriched_pathways *Sample Enriched Pathways*

Description

This dataset highlights enriched pathways identified by using ’ConsensusPathDB’ while performing overexpression analysis for a sample set of genes.

Usage

```
data("enriched_pathways")
```

References

Kamburov, A., Stelzl, U., Lehrach, H. and Herwig, R., 2013. The ConsensusPathDB interaction database: 2013 update. Nucleic acids research, 41(D1), pp.D793-D800.

Examples

```
data(enriched_pathways)
## maybe str(enriched_pathways) ;
```

enriched_pc *Sample Enriched Protein Complexes*

Description

This dataset highlights protein complexes identified by using 'ConsensusPathDB' while performing overexpression analysis for a sample set of genes.

Usage

```
data("enriched_pc")
```

References

Kamburov, A., Stelzl, U., Lehrach, H. and Herwig, R., 2013. The ConsensusPathDB interaction database: 2013 update. Nucleic acids research, 41(D1), pp.D793-D800.

Examples

```
data(enriched_pc)
## maybe str(enriched_pc) ;
```

plot_go_terms *Bupple Plot for GO Terms*

Description

Make a bubble plot for significantly enriched Gene Ontologies (GO) Terms obtained after performing gene set overexpression analysis using 'ConsensusPathDB'.

Usage

```
plot_go_terms(df_goterms, total_no_background_genes,
              negative_log_10_p_value_cutoff, max_overlap)
```

Arguments

`df_goterms` The tab-separated data frame with the GO terms information obtained after performing gene set overexpression analysis using 'ConsensusPathDB'.

`total_no_background_genes` Total no of genes in the background set.

`negative_log_10_p_value_cutoff` The threshold on -log10(pvalue) to be used to identify the GO terms to be highlighted in the plot.

`max_overlap` To prevent overlapping text, set this paramater to a number >= 20.

Details

Plots the significantly enriched molecular function (m), cellular components (c) and biological processes (b) obtained via ConsensusPathDB.

Value

Returns a bubble plot of type ggplot.

Note

rmall@hbku.edu.qa

Author(s)

Raghvendra Mall

Examples

```
data("enriched_goterms")
g <- plot_go_terms(df_goterms = enriched_goterms, negative_log_10_p_value_cutoff=17)
g
```

`plot_pathways` *Plot clean enriched pathways as a bubble plot*

Description

Make a bubble plot of clean enriched pathways obtained from 'ConsensusPathDB' by performing gene set overexpression analysis. Colours represent the clusters to which each pathway belongs. You need to run the function `clean_pathways` to obtain the input data frame.

Usage

```
plot_pathways(final_df_pathway, total_no_background_genes, fontsize)
```

Arguments

`final_df_pathway`
: Clean and clustered pathways obtained using clean_pathways.

`total_no_background_genes`
: Total no of genes in the background set.

`fontsize`
: Font size of the pathways to be displayed on y-axis.

Value

Returns a bubble plot of type ggplot. Colours represent the clusters to which each pathway belongs.

Note

rmall@hbku.edu.qa

Author(s)

Raghvendra Mall

See Also

See Also as `clean_pathways`, `plot_pathways_stacked_barplot`, `plot_go_terms`

Examples

```
data("t.tests.treatment.sign")
data("enriched_pathways")
revised_pathway <- clean_pathways(df_case_vs_ctrl=t.tests.treatment.sign,
                                  df_pathway = enriched_pathways)
p <- plot_pathways(revised_pathway)
p
```

---

plot_pathways_stacked_barplot

Stacked Barplot of Cleaned Pathways

Description

Make a stacked barplot like the one available in Ingenuity Pathway Analysis highlighting percentage of up, down and non-differentially expressed genes in the set of clean enriched pathways obtained from 'ConsensusPathDB' by performing gene set overexpression analysis. You need to run the function `clean_pathways` to obtain the input data frame

Usage

```
plot_pathways_stacked_barplot(final_df_pathway)
```

Arguments

`final_df_pathway`
Clean and clustered pathways obtained using `clean_pathways`.

Value

Returns a stacked barplot of type ggplot.

Note

rmall@hbku.edu.qa

Author(s)

Raghvendra Mall

See Also

`clean_pathways`, `plot_go_terms`

Examples

```
data("t.tests.treatment.sign")
data("enriched_pathways")
revised_pathway <- clean_pathways(df_case_vs_ctrl=t.tests.treatment.sign,
                                  df_pathway = enriched_pathways)
p <- plot_pathways_stacked_barplot(revised_pathway)
p
```

`t.tests.treatment.sign` *List of differentially expressed genes*

Description

Consist of list of differentially expressed genes (DEG) with fold-change information i.e. up and down regulated genes between case and control.

Usage

```
data("t.tests.treatment.sign")
```

Format

A data frame with 1820 observations on the following 8 variables.

`gene` a character vector

`p.value` a numeric vector

`p.value.fdr` a numeric vector

`fc` a numeric vector

`mean.A` a numeric vector

`mean.B` a numeric vector

`sd.A` a numeric vector

`sd.B` a numeric vector

Examples

```
data(t.tests.treatment.sign)
## maybe str(t.tests.treatment.sign) ;
```

Index