Package ‘HybridExpress’

May 29, 2024

**Title** Comparative analysis of RNA-seq data for hybrids and their progenitors

**Version** 1.0.0

**Date** 2023-10-17

**Description** HybridExpress can be used to perform comparative transcriptomics analysis of hybrids (or allopolyploids) relative to their progenitor species. The package features functions to perform exploratory analyses of sample grouping, identify differentially expressed genes in hybrids relative to their progenitors, classify genes in expression categories (N = 12) and classes (N = 5), and perform functional analyses. We also provide users with graphical functions for the seamless creation of publication-ready figures that are commonly used in the literature.

**License** GPL-3

**URL** [https://github.com/almeidasilvaf/HybridExpress](https://github.com/almeidasilvaf/HybridExpress)

**BugReports** [https://support.bioconductor.org/tag/HybridExpress](https://support.bioconductor.org/tag/HybridExpress)

**biocViews** Software, FunctionalGenomics, GeneExpression, Transcriptomics, RNASeq, Classification, DifferentialExpression

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.1

**Imports** ggplot2, patchwork, rlang, DESeq2, SummarizedExperiment, stats, methods, RColorBrewer, ComplexHeatmap, grDevices, BiocParallel

**Suggests** BiocStyle, knitr, sessioninfo, testthat (>= 3.0.0)

**Config/testthat/edition** 3

**VignetteBuilder** knitr

**Depends** R (>= 4.3.0)

**LazyData** false

**git_url** [https://git.bioconductor.org/packages/HybridExpress](https://git.bioconductor.org/packages/HybridExpress)
HybridExpress-package

HybridExpress: Comparative analysis of RNA-seq data for hybrids and their progenitors

Description

HybridExpress can be used to perform comparative transcriptomics analysis of hybrids (or allopolyploids) relative to their progenitor species. The package features functions to perform exploratory analyses of sample grouping, identify differentially expressed genes in hybrids relative to their progenitors, classify genes in expression categories (N = 12) and classes (N = 5), and perform functional analyses. We also provide users with graphical functions for the seamless creation of publication-ready figures that are commonly used in the literature.
Author(s)

Maintainer: Fabricio Almeida-Silva <fabricio_almeidasilva@hotmail.com> (ORCID)
Authors:
  • Lucas Prost-Boxoen (ORCID)
  • Yves Van de Peer <yves.vandepeer@psb.vib-ugent.be> (ORCID)

See Also

Useful links:
  • https://github.com/almeidasilvaf/HybridExpress
  • Report bugs at https://support.bioconductor.org/tag/HybridExpress

add_midparent_expression

Add midparent expression to SummarizedExperiment object

Description

Add midparent expression to SummarizedExperiment object

Usage

add_midparent_expression(
  se,
  coldata_column = "Generation",
  parent1 = "P1",
  parent2 = "P2",
  method = "mean",
  weights = c(1, 1)
)

Arguments

se A SummarizedExperiment object with a count matrix and sample metadata.
coldata_column Character indicating the name of column in colData(se) where information on
the generation are stored. Default: "Generation".
parent1 Character indicating which level of the variable coldata_column represents parent 1. Default: "P1".
parent2 Character indicating which level of the variable coldata_column represents parent 2. Default: "P2".
method Character indicating the method to use to create midparent values. One of 'mean' (default), 'sum', or 'weightedmean'.
weights Numeric vector of length 2 indicating the weights to give to parents 1 and 2 (respectively) if method == "weightedmean". Setting method == "weightedmean" is used sometimes when parents have different ploidy levels. In such cases, the ploidy levels of parents 1 and 2 can be passed in a vector. Default: c(1, 2).
Value

A SummarizedExperiment object.

Examples

data(se_chlamy)
new_se <- add_midparent_expression(se_chlamy)

add_size_factors(se, spikein = FALSE, spikein_pattern = "ERCC")

Arguments

se A SummarizedExperiment object with a count matrix and sample metadata.
spikein Logical indicating whether or not to normalize data using spike-ins. If FALSE, data will be normalized by library size. Default: FALSE.
spikein_pattern Character with the pattern (regex) to use to identify spike-in features in the count matrix. Only valid if spikein_norm = TRUE.

Value

A SummarizedExperiment object as in se, but with an extra column in the colData slot named "sizeFactor". This column contains size factors that will be used by DESeq2 when performing differential expression analyses.

Examples

data(se_chlamy)
se_norm <- add_size_factors(se_chlamy)
### deg_counts

*Data frame with frequencies (absolute and relative) of DEGs per contrast*

**Description**

This object was obtained with `get_deg_counts()` using the example data set `deg_list`.

**Usage**

```r
data(deg_counts)
```

**Format**

A data frame with the frequencies (absolute and relative) of up- and down-regulated genes in each contrast. Relative frequencies are calculated relative to the total number of genes in the count matrix used for differential expression analysis.

**Examples**

```r
data(deg_counts)
```

### deg_list

*List of differentially expressed genes for all contrasts*

**Description**

This object was obtained with `get_deg_list()` using the example data set `se_chlamy`.

**Usage**

```r
data(deg_list)
```

**Format**

A list of data frames with gene-wise test statistics for differentially expressed genes for each contrast. Contrasts are "P2_vs_P1", "F1_vs_P1", "F1_vs_P2", and "F1_vs_midparent", where the ID before 'vs' represents the numerator, and the ID after 'vs' represents the denominator.

**Examples**

```r
data(deg_list)
```
expression_partitioning

*Partition genes in groups based on their expression patterns*

**Description**

Partition genes in groups based on their expression patterns

**Usage**

```r
expression_partitioning(deg_list)
```

**Arguments**

- `deg_list`: A list of data frames with gene-wise test statistics for differentially expressed genes as returned by `get_deg_list()`.

**Value**

A data with the following variables:

- **Gene**: Character, gene ID.
- **Category**: Factor, expression group. Category names are numbers from 1 to 12.
- **Class**: Factor, expression group class. One of "UP" (transgressive up-regulation), "DOWN" (transgressive down-regulation), "ADD" (additivity), "ELD_P1" (expression-level dominance toward the parent 1), or "ELD_P2" (expression-level dominance toward the parent 2).

**Examples**

```r
data(deg_list)
exp_partitions <- expression_partitioning(deg_list)
```

---

`get_deg_counts`

*Get a count table of differentially expressed genes per contrast*

**Description**

Get a count table of differentially expressed genes per contrast

**Usage**

```r
get_deg_counts(deg_list)
```

**Arguments**

- `deg_list`: A list of data frames with gene-wise test statistics for differentially expressed genes as returned by `get_deg_list()`.
get_deg_list

Value
A data frame with the following variables:

- **contrast**: Character, contrast name.
- **up**: Numeric, number of up-regulated genes.
- **down**: Numeric, number of down-regulated genes.
- **total**: Numeric, total number of differentially expressed genes.
- **perc_up**: Numeric, percentage of up-regulated genes.
- **perc_down**: Numeric, percentage of down-regulated genes.
- **perc_total**: Numeric, percentage of differentially expressed genes.

Examples

```r
data(deg_list)
deg_counts <- get_deg_counts(deg_list)
```

get_deg_list

*Get a table of differential expression expression statistics with DESeq2*

Description
Get a table of differential expression expression statistics with DESeq2

Usage

```r
get_deg_list(
  se,
  coldata_column = "Generation",
  parent1 = "P1",
  parent2 = "P2",
  offspring = "F1",
  midparent = "midparent",
  lfcThreshold = 0,
  alpha = 0.01,
  ...
)
```

Arguments

- **se**: A SummarizedExperiment object with a count matrix and sample metadata.
- **coldata_column**: Character indicating the name of column in colData(se) where information on the generation are stored. Default: "Generation".
- **parent1**: Character indicating which level of the variable coldata_column represents parent 1. Default: "P1".
get_deg_list

parent2  Character indicating which level of the variable coldata_column represents parent 2. Default: "P2".

offspring  Character indicating which level of the variable coldata_column represents the offspring (hybrid or allopolyploid). Default: "F1"

midparent  Character indicating which level of the variable coldata_column represents the midparent value. Default: "midparent", as returned by add_midparent_expression().

lfcThreshold  Numeric indicating the log2 fold-change threshold to use to consider differentially expressed genes. Default: 0.

alpha  Numeric indicating the adjusted P-value threshold to use to consider differentially expressed genes. Default: 0.01.

...  Additional arguments to be passed to DESeq2::results().

Value

A list of data frames with DESeq2’s gene-wise tests statistics for each contrast. Each data frame contains the same columns as the output of DESeq2::results(). Contrasts (list names) are:

P2_vs_P1  Parent 2 (numerator) versus parent 1 (denominator).
F1_vs_P1  Offspring (numerator) versus parent 1 (denominator).
F1_vs_P2  Offspring (numerator) versus parent 2 (denominator).
F1_vs_midparent  Offspring (numerator) versus midparent (denominator).

The data frame with gene-wise test statistics in each list element contains the following variables:

baseMean  Numeric, base mean.
log2FoldChange  Numeric, log2-transformed fold changes.
lfcSE  Numeric, standard error of the log2-transformed fold changes.
stat  Numeric, observed test statistic.
pvalue  Numeric, p-value.
padj  Numeric, P-value adjusted for multiple testing.

The list contains two additional attributes named ngenes (numeric, total number of genes), and plot-data, which is a 3-column data frame with variables "gene" (character, gene ID), "lFC_F1_vs_P1" (numeric, log2 fold change between F1 and P1), and "lFC_F1_vs_P2" (numeric, log2 fold change between F1 and P2).

Examples

data(se_chlamy)
se <- add_midparent_expression(se_chlamy)
se <- add_size_factors(se, spikein = TRUE)
deg_list <- get_deg_list(se)
go_chlamy

Data frame with GO terms annotated to each gene of *Chlamydomonas reinhardtii*

Description

Data were obtained from Phytozome and processed so that each row contains only one GO term (long format).

Usage

data(go_chlamy)

Format

A 2-column data frame with columns `gene` (character, gene ID), and `GO` (character, name of GO term.)

Examples

data(go_chlamy)

ora

Perform overrepresentation analysis for a set of genes

Description

Perform overrepresentation analysis for a set of genes

Usage

ora(
    genes,
    annotation,
    column = NULL,
    background,
    correction = "BH",
    alpha = 0.05,
    min_setsize = 5,
    max_setsize = 500,
    bp_param = BiocParallel::SerialParam()
)
Arguments

genes  Character vector containing genes for overrepresentation analysis.
annotation  Annotation data frame with genes in the first column and functional annotation in the other columns. This data frame can be exported from Biomart or similar databases.
column  Column or columns of annotation to be used for enrichment. Both character or numeric values with column indices can be used. If users want to supply more than one column, input a character or numeric vector. Default: all columns from annotation.
background  Character vector of genes to be used as background for the overrepresentation analysis.
correction  Multiple testing correction method. One of "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr" or "none". Default is "BH".
alpha  Numeric indicating the adjusted P-value threshold for significance. Default: 0.05.
min_setsize  Numeric indicating the minimum gene set size to be considered. Gene sets correspond to levels of each variable in annotation. Default: 5.
max_setsize  Numeric indicating the maximum gene set size to be considered. Gene sets correspond to levels of each variable in annotation. Default: 500.
bp_param  BiocParallel back-end to be used. Default: BiocParallel::SerialParam()

Value

A data frame of overrepresentation results with the following variables:

  term  Character, functional term ID/name.
  genes  Numeric, intersection length between input genes and genes in a particular functional term.
  all  Numeric, number of all genes in a particular functional term.
  pval  Numeric, P-value for the hypergeometric test.
  padj  Numeric, P-value adjusted for multiple comparisons using the method specified in parameter adj.
  category  Character, name of the grouping variable (i.e., column name of annotation).

Examples

```r
data(se_chlamy)
data(go_chlamy)
data(deg_list)

# Perform ORA for up-regulated genes in contrast F1_vs_P1
up_genes <- deg_list$F1_vs_P1
up_genes <- rownames(up_genes[up_genes$log2FoldChange > 0, ])
background <- rownames(se_chlamy)
ora(up_genes, go_chlamy, background = background)
```
Perform a principal component analysis (PCA) and plot PCs

**Usage**

```r
pca_plot(
  se,
  PCs = c(1, 2),
  ntop = 500,
  color_by = NULL,
  shape_by = NULL,
  add_mean = FALSE,
  palette = NULL
)
```

**Arguments**

- **se**: A `SummarizedExperiment` object with a count matrix and sample metadata.
- **PCs**: Numeric vector indicating which principal components to show in the x-axis and y-axis, respectively. Default: `c(1, 2)`.
- **ntop**: Numeric indicating the number of top genes with the highest variances to use for the PCA. Default: 500.
- **color_by**: Character with the name of the column in `colData(se)` to use to group samples by color. Default: `NULL`.
- **shape_by**: Character with the name of the column in `colData(se)` to use to group samples by shape. Default: `NULL`.
- **add_mean**: Logical indicating whether to add a diamond symbol with the mean value for each level of the variable indicated in `color_by`. Default: `FALSE`.
- **palette**: Character vector with colors to use for each level of the variable indicated in `color_by`. If `NULL`, a default color palette will be used.

**Value**

A `ggplot` object with a PCA plot showing 2 principal components in each axis along with their % of variance explained.

**Examples**

```r
data(se_chlamy)
se <- add_midparent_expression(se_chlamy)
se$Ploidy[is.na(se$Ploidy)] <- "midparent"
se$Generation[is.na(se$Generation)] <- "midparent"
pca_plot(se, color_by = "Generation", shape_by = "Ploidy", add_mean = TRUE)
```
plot_expression_partitions

Plot expression partitions

Description

Plot expression partitions

Usage

plot_expression_partitions(
  partition_table,
  group_by = "Category",
  palette = NULL,
  labels = c("P1", "F1", "P2")
)

Arguments

partition_table
  A data frame with genes per expression partition as returned by expression_partitioning().

group_by
  Character indicating the name of the variable in partition_table to use to group
genes. One of "Category" or "Class". Default: "Category".

palette
  Character vector with color names to be used for each level of the variable specified in group_by. If group_by = "Category", this must be a vector of length 12. If group_by = "Class", this must be a vector of length 5. If NULL, a default color palette will be used.

labels
  A character vector of length 3 indicating the labels to be given for parent 1, offspring, and parent 2. Default: c("P1", "F1", "P2").

Value

A ggplot object with a plot showing genes in each expression partition.

Examples

data(deg_list)
partition_table <- expression_partitioning(deg_list)
plot_expression_partitions(partition_table)
**plot_expression_triangle**

*Plot a triangle of comparisons of DEG sets among generations*

---

**Description**

Plot a triangle of comparisons of DEG sets among generations

**Usage**

```r
plot_expression_triangle(deg_counts, palette = NULL, box_labels = NULL)
```

**Arguments**

- `deg_counts`: Data frame with number of differentially expressed genes per contrast as returned by `get_deg_counts`.
- `palette`: Character vector of length 4 indicating the colors of the boxes for P1, P2, F1, and midparent, respectively. If NULL, a default color palette will be used.
- `box_labels`: Character vector of length 4 indicating the labels of the boxes for P1, P2, F1, and midparent, respectively. Default: NULL, which will lead to labels "P1", "P2", "F1", and "Midparent", respectively.

**Details**

The expression triangle plot shows the number of differentially expressed genes (DEGs) for each contrast. Numbers in the center of the lines (in bold) indicate the total number of DEGs, while numbers near boxes indicate the number of up-regulated genes in each generation of the triangle.

**Value**

A ggplot object with an expression triangle.

**Examples**

```r
data(deg_counts)
plot_expression_triangle(deg_counts)
```


\textbf{plot_partition_frequencies}

*Plot a barplot of gene frequencies per expression partition*

\section*{Description}

Plot a barplot of gene frequencies per expression partition

\section*{Usage}

\begin{verbatim}
plot_partition_frequencies(
    partition_table,
    group_by = "Category",
    palette = NULL,
    labels = c("P1", "F1", "P2")
)
\end{verbatim}

\section*{Arguments}

\begin{itemize}
  \item \textbf{partition_table} A data frame with genes per expression partition as returned by \texttt{expression_partitioning()}.  
  \item \textbf{group_by} Character indicating the name of the variable in \texttt{partition_table} to use to group genes. One of "Category" or "Class". Default: "Category".  
  \item \textbf{palette} Character vector with color names to be used for each level of the variable specified in \texttt{group_by}. If \texttt{group_by = "Category"}, this must be a vector of length 12. If \texttt{group_by = "Class"}, this must be a vector of length 5. If NULL, a default color palette will be used.  
  \item \textbf{labels} A character vector of length 3 indicating the labels to be given for parent 1, offspring, and parent 2. Default: c("P1", "F1", "P2").
\end{itemize}

\section*{Value}

A ggplot object with a barplot showing gene frequencies per partition next to explanatory line plots depicting each partition.

\section*{Examples}

\begin{verbatim}
data(deg_list)
partition_table <- expression_partitioning(deg_list)
plot_partition_frequencies(partition_table)
\end{verbatim}
**plot_samplecor**

*Plot a heatmap of pairwise sample correlations with hierarchical clustering*

---

### Description

Plot a heatmap of pairwise sample correlations with hierarchical clustering.

### Usage

```r
plot_samplecor(
  se,
  coldata_cols = NULL,
  rowdata_cols = NULL,
  ntop = 500,
  cor_method = "pearson",
  palette = "Blues",
  ...
)
```

### Arguments

- **se**: A `SummarizedExperiment` object with a count matrix and sample metadata in the `colData` slot. If a `rowData` slot is available, it can also be used for clustering rows.
- **coldata_cols**: A vector (either numeric or character) indicating which columns should be extracted from `colData(se)`.
- **rowdata_cols**: A vector (either numeric or character) indicating which columns should be extracted from `rowData(se)`.
- **ntop**: Numeric indicating the number of top genes with the highest variances to use for the PCA. Default: 500.
- **cor_method**: Character indicating the correlation method to use. One of "pearson" or "spearman". Default: "pearson".
- **palette**: Character indicating the name of the color palette from the RColorBrewer package to use. Default: "Blues".
- **...**: Additional arguments to be passed to `ComplexHeatmap::pheatmap()`. These arguments can be used to control heatmap aesthetics, such as show/hide row and column names, change font size, activate/deactivate hierarchical clustering, etc. For a complete list of the options, see `?ComplexHeatmap::pheatmap()`.

### Value

A heatmap of hierarchically clustered pairwise sample correlations.
Examples

```r
data(se_chlamy)
se <- add_midparent_expression(se_chlamy)
se$Ploidy[is.na(se$Ploidy)] <- "midparent"
se$Generation[is.na(se$Generation)] <- "midparent"
plot_samplecor(se, ntop = 500)
```

**se_chlamy**  
Expression data (in counts) for 3 Chlamydomonas lines (P1, P2, and F1)

Description

Two lines (referred to as parent 1 and parent 2) with different ploidy levels were crossed to generate an allopolyploid (F1).

Usage

```r
data(se_chlamy)
```

Format

A SummarizedExperiment object with an assay (count) and colData.

Examples

```r
data(se_chlamy)
```
Index

* **datasets**
  - deg_counts, 5
  - deg_list, 5
  - go_chlamy, 9
  - se_chlamy, 16

* **internal**
  - HybridExpress-package, 2

  - add_midparent_expression, 3
  - add_size_factors, 4

  - deg_counts, 5
  - deg_list, 5

  - expression_partitioning, 6

  - get_deg_counts, 6
  - get_deg_list, 7
  - go_chlamy, 9

HybridExpress (HybridExpress-package), 2
HybridExpress-package, 2

ora, 9

pca_plot, 11
plot_expression_partitions, 12
plot_expression_triangle, 13
plot_partition_frequencies, 14
plot_samplecor, 15

se_chlamy, 16