Package ‘myTAI’

November 7, 2025

Type Package

Title Evolutionary Transcriptomics

Version 2.3.3

Date 2025-11-04

Maintainer Hajk-Georg Drost <hajk-georg.drost@tuebingen.mpg.de>

Description Investigate the evolution of biological processes by capturing evolutionary signatures in transcriptomes (Drost et al. (2018) <doi:10.1093/bioinformatics/btx835>). This package aims to provide a transcriptome analysis environment to quantify the average evolutionary age of genes contributing to a transcriptome of interest.

NeedsCompilation yes

License GPL-2

Depends R (>= 4.1)

Imports S7, patchwork, purrr, tidyr, Rcpp, memoise, fitdistrplus (>= 1.1-5), dplyr (>= 0.3.0), RColorBrewer (>= 1.1-2), ggplot2 (>= 1.0.1), ggforce, ggridges, ggtext, readr (>= 0.2.2), tibble, ggplotify, ggrepel, Matrix, pheatmap

Suggests knitr (>= 1.6), rmarkdown (>= 0.3.3), testthat (>= 0.9.1), mgcv, Seurat, SeuratObject, uwot, decor, DESeq2, gganimate, taxize

LinkingTo RcppArmadillo, RcppThread, Rcpp

URL https://drostlab.github.io/myTAI/

BugReports https://github.com/drostlab/myTAI/issues

RoxygenNote 7.3.3

Encoding UTF-8

LazyData true

LazyDataCompression xz

VignetteBuilder knitr

Collate 'globals.R' 'utils_S7.R' 'utils_math.R' 'phyloset_base.R' 'phyloset_bulk.R' 'phyloset_sc.R' 'stat_distributions.R' 'stat_test_result.R' 'RcppExports.R' 'age.apply.R' 'datasets.R' 'destroy_pattern.R' 'gatai_convergence_plots.R' 'genes_filter.R' 'genes_patterns.R' 'genes_transforms.R' 'myTAI-package.R' 'omit_matrix.R' 'plot_contribution.R' 'plot_distribution_expression.R' 'plot_distribution_pTAI.R' 'plot_distribution_strata.R' 'plot_gene_heatmap.R' 'plot_gene_profiles.R' 'plot_gene_space.R' 'plot_mean_var.R' 'plot_relative_expression.R' 'plot_sample_space.R' 'plot_signature.R' 'plot_signature_gene_quantiles.R' 'plot_signature_multiple.R' 'plot_signature_transformed.R' 'plot_strata_expression.R' 'plot_utils.R' 'stat_ci_estimation.R' 'stat_early_conservation_test.R' 'stat_flatline_test.R' 'stat_generic_conservation_test.R' 'stat_late_conservation_test.R' 'stat_null_conservation_txis.R' 'stat_pairwise_test.R' 'stat_reductive_hourglass_test.R' 'stat_reverse_hourglass_test.R' 'taxid.R' 'tf_PS.R' 'tf_stability.R' 'zzz.R'

Author Hajk-Georg Drost [aut, cre] (ORCID: <https://orcid.org/0000-0002-1567-306X>),
Stefan Manolache [aut, ctb] (ORCID: <https://orcid.org/0009-0006-3326-985X>),
Jaruwatana Sodai Lotharukpong [aut, ctb] (ORCID: <https://orcid.org/0000-0002-3475-0980>),
Nikola Kalábová [aut, ctb] (ORCID: <https://orcid.org/0009-0007-6384-8809>),
Filipa Martins Costa [aut, ctb],
Kristian K Ullrich [aut, ctb] (ORCID: <https://orcid.org/0000-0003-4308-9626>)

Repository CRAN

Date/Publication 2025-11-07 14:10:02 UTC

Contents

`age.apply` *Age Category Specific apply Function*

Description

This function performs the split-apply-combine methodology on Phylostrata or Divergence Strata stored within the input PhyloExpressionSet.

This function is very useful to perform any phylostratum or divergence-stratum specific analysis.

Usage

```
age.apply(phyex_set, FUN, ..., as.list = FALSE)
```

Arguments

`phyex_set`	a standard PhyloExpressionSet object.
`FUN`	a function to be performed on the corresponding expression matrix of each phylostratum or divergence-stratum.
`...`	additional arguments of FUN.
`as.list`	a boolean value specifying whether the output format shall be a matrix or a list object.

Details

This function uses the `split` function to subset the expression matrix into phylostratum specific sub-matrices. Internally using `lapply`, any function can be performed to the sub-matrices. The return value of this function is a numeric matrix storing the return values by `FUN` for each phylostratum and each developmental stage s. Note that the input `FUN` must be an function that can be applied to a matrix (e.g., `colMeans`). In case you use an anonymous function you could use `function(x) apply(x , 2 , var)` as an example to compute the variance of each phylostratum and each developmental stage s.

Value

Either a numeric matrix storing the return values of the applied function for each age class or a numeric list storing the return values of the applied function for each age class in a list.

Author(s)

Hajk-Georg Drost

See Also

`split`, `tapply`, `lapply`

Examples

```
# source the example dataset
data(example_phyex_set)

# Example 1
# get the relative expression profiles for each phylostratum
age.apply(example_phyex_set, relative_expression)

# this is analogous to
rel_exp_matrix(example_phyex_set)
# Example 2
# compute the mean expression profiles for each phylostratum
age.apply(example_phyex_set, colMeans)

# Example 3
# compute the variance profiles for each phylostratum
age.apply(example_phyex_set, function(x) apply(x , 2 , var))
```

```
# Example 4
# compute the range for each phylostratum
# Note: in this case, the range() function returns 2 values for each phylostratum
# and each developmental stage, hence one should use the argument 'as.list = TRUE'
# to make sure that the results are returned properly
age.apply(example_phyex_set, function(x) apply(x , 2 , range), as.list = TRUE)
```

as_BulkPhyloExpressionSet

as_BulkPhyloExpressionSet

Description

This function is an alias for `BulkPhyloExpressionSet_from_df`. Please refer to the documentation for `BulkPhyloExpressionSet_from_df` for usage details, arguments, and examples.

Usage

```
as_BulkPhyloExpressionSet(
  data,
  groups = colnames(data[, 3:ncol(data)]),
  name = deparse(substitute(data)),
  strata_legend = NULL,
  ...
)
```

Arguments

`data`	A data frame with phylostratum assignments and gene expression data
`groups`	Vector of group labels for the samples/replicates
`name`	Character string to name the dataset
`strata_legend`	Optional data frame mapping phylostratum numbers to labels
`...`	Additional arguments passed to BulkPhyloExpressionSet_from_df

Value

A BulkPhyloExpressionSet object

See Also

`BulkPhyloExpressionSet_from_df`

as_data_frame — *Convert BulkPhyloExpressionSet to Data Frame*

Description

Convert a BulkPhyloExpressionSet object back to the original data frame format with phylostratum, gene ID, and expression data as columns.

Usage

```
as_data_frame(phyex_set, use_collapsed = FALSE)
```

Arguments

`phyex_set`	A BulkPhyloExpressionSet object
`use_collapsed`	Logical indicating whether to use collapsed expression data (default: FALSE)

Value

A data frame where column 1 contains phylostratum information, column 2 contains gene IDs, and columns 3+ contain expression data

Examples

```
# Convert BulkPhyloExpressionSet back to data frame
df <- as_data_frame(example_phyex_set)
df_collapsed <- as_data_frame(example_phyex_set, use_collapsed = TRUE)
```

BulkPhyloExpressionSet — *Bulk PhyloExpressionSet Class*

Description

S7 class for bulk RNA-seq phylotranscriptomic expression data. This class handles expression data with biological replicates and provides bootstrapping functionality for statistical analysis.

Usage

```
BulkPhyloExpressionSet(
  strata = stop("@strata is required"),
  strata_values = stop("@strata_values is required"),
  expression = stop("@expression is required"),
  groups = stop("@groups is required"),
  name = "Phylo Expression Set",
  species = character(0),
  index_type = "TXI",
  identities_label = "Identities",
  gene_ids = character(0),
  null_conservation_sample_size = 5000L,
  .null_conservation_txis = NULL,
  .bootstrapped_txis = NULL
)
```

Arguments

`strata`	Factor vector of phylostratum assignments for each gene
`strata_values`	Numeric vector of phylostratum values used in TXI calculations
`expression`	Matrix of expression counts with genes as rows and samples as columns
`groups`	Factor vector indicating which identity each sample belongs to
`name`	Character string naming the dataset (default: "Phylo Expression Set")
`species`	Character string specifying the species (default: NULL)
`index_type`	Character string specifying the transcriptomic index type (default: "TXI")
`identities_label`	Character string labeling the identities (default: "Stages")
`gene_ids`	Character vector of gene identifiers (default: character(0), auto-generated from expression rownames if not provided)
`null_conservation_sample_size`	Numeric value for null conservation sample size (default: 5000)
`.null_conservation_txis`	Precomputed null conservation TXI values (default: NULL)
`.bootstrapped_txis`	Precomputed bootstrapped TXI values (default: NULL)

Details

The BulkPhyloExpressionSet class is designed for bulk RNA-seq data with biological replicates. It extends the base PhyloExpressionSetBase class with bulk-specific functionality.

Replicate Handling: Expression data across biological replicates is collapsed by taking row means within each experimental condition or developmental stage.

Computed Properties: In addition to inherited computed properties from the base class, this class provides:

- `expression_collapsed` - Matrix of expression data collapsed across replicates (genes x identities)
- `bootstrapped_txis` - Matrix of bootstrapped TXI values for statistical inference (500 bootstrap samples x identities)

Inherited computed properties from PhyloExpressionSetBase include:

- `gene_ids` - Character vector of gene identifiers
- `identities` - Character vector of identity labels
- `sample_names` - Character vector of sample names
- `num_identities` - Integer count of unique identities
- `num_samples` - Integer count of total samples
- `num_genes` - Integer count of genes
- `num_strata` - Integer count of phylostrata
- `index_full_name` - Full name of the transcriptomic index type
- `group_map` - List mapping identity names to sample names
- `TXI` - Numeric vector of TXI values for each identity
- `TXI_sample` - Numeric vector of TXI values for each sample
- `null_conservation_txis` - Matrix of null conservation TXI values for statistical testing

Statistical Analysis: The class supports confidence interval estimation and standard deviation calculation through bootstrapped TXI values, enabling robust statistical analysis of developmental or experimental patterns.

Value

A BulkPhyloExpressionSet object

`BulkPhyloExpressionSet_from_df`

Convert Data to BulkPhyloExpressionSet

Description

Convert a data frame with phylostratum, gene ID, and expression data into a BulkPhyloExpressionSet object.

Usage

```
BulkPhyloExpressionSet_from_df(
  data,
  groups = colnames(data[, 3:ncol(data)]),
  name = deparse(substitute(data)),
  strata_legend = NULL,
  ...
)
```

Arguments

`data`	A data frame where column 1 contains phylostratum information, column 2 contains gene IDs, and columns 3+ contain expression data
`groups`	A factor or character vector indicating which group each sample belongs to. Default uses column names from expression data
`name`	A character string naming the dataset. Default uses the variable name
`strata_legend`	A data frame with two columns: phylostratum assignments and name of each stratum. If NULL, no labels will be added (default: NULL) If NULL, uses sorted unique values from column 1
`...`	Additional arguments passed to BulkPhyloExpressionSet constructor

Value

A BulkPhyloExpressionSet object

`check_BulkPhyloExpressionSet`
Check if object is a BulkPhyloExpressionSet

Description

Checks if the input is a PhyloExpressionSet S7 object and throws an error if not.

Usage

```
check_BulkPhyloExpressionSet(phyex_set)
```

Arguments

`phyex_set`	An object to check

Value

Invisibly returns TRUE if check passes, otherwise throws an error

`check_PhyloExpressionSet` *Check if object is a PhyloExpressionSet*

Description

Checks if the input is a PhyloExpressionSet S7 object and throws an error if not.

Usage

```
check_PhyloExpressionSet(phyex_set)
```

Arguments

`phyex_set`	An object to check

Value

Invisibly returns TRUE if check passes, otherwise throws an error

`check_ScPhyloExpressionSet` *Check if object is a ScPhyloExpressionSet*

Description

Checks if the input is a PhyloExpressionSet S7 object and throws an error if not.

Usage

```
check_ScPhyloExpressionSet(phyex_set)
```

Arguments

`phyex_set`	An object to check

Value

Invisibly returns TRUE if check passes, otherwise throws an error

collapse *Collapse PhyloExpressionSet Replicates*

Description

Convert a PhyloExpressionSet with replicates to one with collapsed expression data.

Usage

```
collapse(phyex_set, ...)
```

Arguments

`phyex_set`	A PhyloExpressionSet object
`...`	Additional arguments passed to methods

Value

A new PhyloExpressionSet object with collapsed expression data

Examples

```
# Collapse replicates in a PhyloExpressionSet
collapsed_set <- collapse(example_phyex_set)
```

ConservationTestResult *Conservation Test Result S7 Class*

Description

S7 class extending TestResult for conservation-specific test results, including TXI profiles and null distributions.

Usage

```
ConservationTestResult(
  method_name = stop("@method_name is required"),
  test_stat = stop("@test_stat is required"),
  fitting_dist = stop("@fitting_dist is required"),
  params = stop("@params is required"),
  alternative = "two-sided",
  null_sample = stop("@null_sample is required"),
  data_name = character(0),
  p_label = "p_val",
  test_txi = stop("@test_txi is required"),
  null_txis = stop("@null_txis is required"),
  modules = list()
)
```

Arguments

`method_name`	Character string specifying the statistical test method
`test_stat`	Numeric value of the test statistic
`fitting_dist`	Character string specifying the fitted distribution
`params`	Named list of distribution parameters
`alternative`	Character string specifying the alternative hypothesis
`null_sample`	Numeric vector of null distribution values
`data_name`	Character string describing the data
`p_label`	Character string for p-value label
`test_txi`	Numeric vector of observed TXI values
`null_txis`	Matrix of null TXI distributions from permutations
`modules`	Optional list of developmental modules used in the test

Details

ConservationTestResult extends TestResult with phylotranscriptomic-specific information including the observed TXI profile and null TXI distributions generated by permutation testing.

Value

A ConservationTestResult object

`COUNT_TRANSFORMS` *Count Transformation Functions*

Description

Predefined list of transformation functions for count data normalization.

Usage

```
COUNT_TRANSFORMS
```

Format

A named list of transformation functions:

none Identity transformation (no change)

sqrt Square root transformation

log2 log2(x+1) transformation

vst Variance stabilizing transformation (DESeq2)

rlog Regularized log transformation (DESeq2)

rank Rank transformation within each sample

Details

This object provides a list of predefined transformation functions for gene expression matrix. For ‘rlog()‘ and ‘vst()‘ transformations (from ‘DESeq2‘), the expression matrix is multiplied by 2 prior to rounding. This is done to preserve more information for the lower expression values (especially for TPM).

destroy_pattern *Destroy Phylotranscriptomic Pattern Using GATAI*

Description

Apply the GATAI algorithm to identify and remove genes that contribute to phylotranscriptomic patterns.

Usage

```
destroy_pattern(
  phyex_set,
  num_runs = 20,
  runs_threshold = 0.5,
  analysis_dir = NULL,
  plot_results = TRUE,
  max_generations = 10000,
  seed = 1234,
  extended_analysis = FALSE,
  min_pval = 0.05,
  always_return_genes = FALSE,
  ...
)
```

Arguments

`phyex_set`	A PhyloExpressionSet object (bulk or single cell, the latter which will get pseudo-bulked)
`num_runs`	Number of GATAI runs to perform (default: 20)
`runs_threshold`	Threshold for gene removal consistency across runs (default: 0.5)
`analysis_dir`	Directory to store GATAI analysis results (default: NULL)
`plot_results`	Whether to plot the results. If analysis dir is given, this will be ignored.
`max_generations`	Integer. Maximum number of generations (iterations) for the genetic algorithm (default 10000).
`seed`	Random seed for reproducibility (default: 1234)
`extended_analysis`	Whether to show the multiple runs and convergence plots (default: FALSE)
`min_pval`	Minimum p-value for which the pattern is considered destroyed (default: 0.05).
`always_return_genes`	Whether to return genes even when the pattern is not destroyed (default: FALSE).
`...`	Additional arguments passed to gataiR::gatai

Details

This function requires the gataiR package to be installed. GATAI systematically removes genes that contribute to phylotranscriptomic patterns by iteratively testing gene removal and evaluating the impact on the overall transcriptomic signature.

Value

A list containing GATAI results including identified genes that contribute to the pattern

Author(s)

Filipa Martins Costa

`diagnose_test_robustness` *Diagnose Test Robustness*

Description

Evaluate the robustness of conservation tests across different sample sizes for null distribution generation.

Usage

```
diagnose_test_robustness(
  test,
  phyex_set,
  sample_sizes = c(500, 1000, 5000, 10000),
  plot_result = TRUE,
  num_reps = 5,
  ...
)
```

Arguments

`test`	Function representing the conservation test to evaluate
`phyex_set`	A PhyloExpressionSet object
`sample_sizes`	Numeric vector of sample sizes to test (default: c(500, 1000, 5000, 10000))
`plot_result`	Logical indicating whether to plot results (default: TRUE)
`num_reps`	Number of replicates for each sample size (default: 5)
`...`	Additional arguments passed to the test function

Details

This function assesses how consistent test results are across different sample sizes for null distribution generation, helping to determine appropriate sample sizes for reliable testing.

Value

A data frame with test results across different sample sizes

Examples

```
# Diagnose flatline test robustness
robustness <- diagnose_test_robustness(stat_flatline_test,
                                       example_phyex_set,
                                       sample_sizes=c(10,20),
                                       plot_result=FALSE,
                                       num_reps=3)
```

`Distribution` *Distribution S7 Class*

Description

S7 class for representing probability distributions used in statistical testing, including PDF, CDF, quantile functions, and fitting procedures.

Usage

```
Distribution(
  name = stop("@name is required"),
  pdf = stop("@pdf is required"),
  cdf = stop("@cdf is required"),
  quantile_function = stop("@quantile_function is required"),
  fitting_function = stop("@fitting_function is required"),
  param_names = stop("@param_names is required")
)
```

Arguments

`name`	Character string identifying the distribution
`pdf`	Function for probability density function
`cdf`	Function for cumulative distribution function
`quantile_function`	Function for quantile calculations
`fitting_function`	Function to fit distribution parameters from data
`param_names`	Character vector of parameter names

Details

The Distribution class provides a unified interface for different probability distributions used in phylotranscriptomic testing. Each distribution includes the necessary functions for statistical inference.

Examples

```
# Access predefined distributions
normal_dist <- distributions$normal
gamma_dist <- distributions$gamma
```

`distributions` *Predefined Distribution Objects*

Description

List of predefined Distribution objects for use in statistical testing.

Usage

```
distributions
```

Format

A named list containing Distribution objects:

normal Normal distribution with mean and sd parameters

gamma Gamma distribution with shape and rate parameters

Details

This list provides ready-to-use Distribution objects for common statistical tests. Each distribution includes appropriate fitting functions and statistical functions for hypothesis testing.

`downsample` *Downsample ScPhyloExpressionSet*

Description

Create a downsampled copy of a ScPhyloExpressionSet object with fewer cells per identity.

Usage

```
downsample(phyex_set, downsample = 10)
```

Arguments

`phyex_set`	A ScPhyloExpressionSet object
`downsample`	Integer, number of cells to keep per identity (default: 10)

Details

This function creates a new ScPhyloExpressionSet with a subset of cells, maintaining the same proportional representation across identities. The sampling is stratified by the current `selected_idents` grouping. All metadata and reductions are filtered to match the selected cells.

Value

A new ScPhyloExpressionSet object with downsampled cells

Examples

```
# Downsample to 20 cells per identity
small_set <- downsample(example_phyex_set_sc, downsample = 20)

# Change grouping and downsample
example_phyex_set_sc@selected_idents <- "day"
treatment_set <- downsample(example_phyex_set_sc, downsample = 15)
```

`downsample_expression` *Downsample Expression Matrix by Groups*

Description

Downsample an expression matrix by randomly selecting a specified number of samples from each group.

Usage

```
downsample_expression(expression_matrix, groups, downsample = 10)
```

Arguments

`expression_matrix`	Expression matrix with genes as rows and samples as columns
`groups`	Factor vector indicating which group each sample belongs to
`downsample`	Integer, number of samples to keep per group (default: 10)

Details

This function randomly samples up to `downsample` samples from each group level. The returned expression matrix is converted to dense format and maintains column names from the original matrix. Useful for creating balanced subsets for visualization or when memory is limited.

Value

A dense expression matrix (genes x downsampled samples)

`example_phyex_set` *Example phyex set*

Description

Arabidopsis thaliana embryogenesis dataset from Hoffman et al. 2019. check the phyexSet package for more details

Usage

```
example_phyex_set
```

Format

A BulkPhyloExpressionSet with 8 developmental stages (3 reps each) and 27520 genes

Source

phyexSets::Athaliana.embryogenesis_2019 matched with the Arabidopsis thaliana phylomap from phylomapr

---

`example_phyex_set_old` *Example phyex set old*

Description

Arabidopsis thaliana embryogenesis dataset from Xiang et al. 2011. check the phyexSet package for more details

Usage

```
example_phyex_set_old
```

Format

A BulkPhyloExpressionSet with 7 developmental stages (1 rep each) and 25096 genes

Source

phyexSets::Athaliana.embryogenesis_2011 matched with the Arabidopsis thaliana phylomap from phylomapr

---

`example_phyex_set_sc` *Load Example Single-Cell PhyloExpressionSet*

Description

Creates and returns an example ScPhyloExpressionSet object for testing and examples.

Usage

```
example_phyex_set_sc
```

Format

A ScPhyloExpressionSet with 1000 cells and 1000 genes

Source

Synthetically generated data

`exp_p` *Format P-Value for Scientific Notation*

Description

Format p-values in scientific notation for plot annotations.

Usage

```
exp_p(p, sci_thresh = 4)
```

Arguments

`p`	Numeric p-value
`sci_thresh`	Numeric threshold for using scientific notation (number of decimal places)

Details

This function formats p-values in scientific notation using the format "p = a × 10^b" which is suitable for ggplot2 annotations and maintains proper mathematical formatting.

Value

Expression object for use in plot annotations

Examples

```
# Format p-value for plotting
expr <- exp_p(0.001)
```

`gatai_animate_destruction` *Animate GATAI Destruction Process*

Description

Create an animation showing how the transcriptomic signature changes during the GATAI gene removal process across generations.

Usage

```
gatai_animate_destruction(
  phyex_set,
  save_file = NULL,
  fps = 10,
  width = 1000,
  height = 800,
  ...
)
```

Arguments

`phyex_set`	A PhyloExpressionSet object containing the original gene expression data.
`save_file`	Optional file path to save the animation (default: NULL, returns animation object).
`fps`	Frames per second for the animation (default: 20).
`width`	Width of the animation in pixels (default: 1000).
`height`	Height of the animation in pixels (default: 800).
`...`	Additional arguments passed to `gataiR::gatai`.

Details

This function runs GATAI for a single run while saving intermediate TAI values at each generation. It then creates an animated plot showing how the transcriptomic signature evolves as genes are progressively removed. The animation shows: - The original signature (generation 0) - Progressive changes through each generation - Final signature after convergence

The intermediate file format contains generation numbers in the first column and TAI values for each developmental stage in subsequent columns.

Value

If save_file is NULL, returns a gganimate animation object. If save_file is specified, saves the animation and returns the file path invisibly.

`genes_lowly_expressed` *Select Lowly Expressed Genes*

Description

Select genes with mean expression below a specified threshold.

Usage

```
genes_lowly_expressed(phyex_set, threshold = 1)
```

Arguments

`phyex_set`	A PhyloExpressionSet object
`threshold`	Mean expression threshold (default: 1)

Details

This function identifies genes with low mean expression levels, which might be candidates for filtering or separate analysis.

Value

Character vector of gene IDs with mean expression <= threshold

Examples

```
# Select genes with mean expression <= 1
low_expr_genes <- genes_lowly_expressed(example_phyex_set, threshold = 1)
```

`genes_top_expr` *Gene Expression Filtering Functions*

Description

Collection of functions for filtering genes based on expression patterns in PhyloExpressionSet objects.

Generic function to select genes with the highest values for a given expression metric.

Usage

```
genes_top_expr(phyex_set, FUN = rowMeans, top_p = 0.99, top_k = NULL, ...)
```

Arguments

`phyex_set`	A PhyloExpressionSet object
`FUN`	Function to calculate gene-wise expression metric (default: rowMeans)
`top_p`	Quantile threshold for gene selection (default: 0.99). Ignored if top_k is specified.
`top_k`	Absolute number of top genes to select (default: NULL). Takes precedence over top_p.
`...`	Additional arguments passed to FUN

Details

This function applies the specified function to calculate a metric for each gene across samples, then selects genes above the specified quantile threshold or the top k genes by absolute count. If both top_p and top_k are specified, top_k takes precedence.

Value

Character vector of gene IDs with metric values >= top_p quantile or top top_k genes

Examples

```
# Select top 1% most expressed genes by mean
high_expr_genes <- genes_top_expr(example_phyex_set, function(x) apply(x, 1, mean), top_p = 0.99)

# Select top 100 most expressed genes
top_100_genes <- genes_top_expr(example_phyex_set, function(x) apply(x, 1, mean), top_k = 100)
```

`genes_top_mean` *Select Top Mean Expressed Genes*

Description

Select genes with the highest mean expression across samples.

Usage

```
genes_top_mean(phyex_set, top_p = 0.99, top_k = NULL)
```

Arguments

`phyex_set`	A PhyloExpressionSet object
`top_p`	Quantile threshold for gene selection (default: 0.99). Ignored if top_k is specified.
`top_k`	Absolute number of top genes to select (default: NULL). Takes precedence over top_p.

Details

This function identifies genes with the highest mean expression levels, which are often the most reliably detected and functionally important.

Value

Character vector of gene IDs with mean expression >= top_p quantile or top top_k genes

Examples

```
# Select top 1% most expressed genes by mean
high_expr_genes <- genes_top_mean(example_phyex_set, top_p = 0.99)

# Select top 1000 most expressed genes
top_1000_genes <- genes_top_mean(example_phyex_set, top_k = 1000)
```

`genes_top_variance` *Select Top Variable Genes*

Description

Select genes with the highest variance across samples.

Usage

```
genes_top_variance(phyex_set, top_p = 0.99, top_k = NULL)
```

Arguments

`phyex_set`	A PhyloExpressionSet object
`top_p`	Quantile threshold for gene selection (default: 0.99). Ignored if top_k is specified.
`top_k`	Absolute number of top genes to select (default: NULL). Takes precedence over top_p.

Details

This function identifies genes with the highest variance across samples, which are often the most informative for downstream analyses.

Value

Character vector of gene IDs with variance >= top_p quantile or top top_k genes

Examples

```
# Select top 1% most variable genes
high_var_genes <- genes_top_variance(example_phyex_set, top_p = 0.99)

# Select top 500 most variable genes
top_500_var_genes <- genes_top_variance(example_phyex_set, top_k = 500)
```

match_map — *Match Gene Expression Data with Phylostratum Map*

Description

Join gene expression data with a phylostratum mapping to create a BulkPhyloExpressionSet object.

Usage

```
match_map(
  data,
  phylomap,
  groups = colnames(data[, 2:ncol(data)]),
  name = NULL,
  ...
)
```

Arguments

`data`	A data frame where column 1 contains gene IDs and columns 2+ contain expression data
`phylomap`	A data frame with two columns: phylostratum assignments and gene IDs
`groups`	A factor or character vector indicating which group each sample belongs to. Default uses column names from expression data
`name`	A character string naming the dataset. Default uses the variable name
`...`	Additional arguments passed to as_BulkPhyloExpressionSet

Value

A BulkPhyloExpressionSet object

Examples

```
# Match expression data with phylostratum map
# bulk_set <- match_map(expression_data, phylo_map,
#                       groups = c("stage1", "stage2", "stage3"),
#                       name = "Matched Dataset")
```

`match_map_sc_matrix` *Match Expression Matrix with Phylostratum Map*

Description

Join single-cell gene expression matrix with a phylostratum mapping to create a ScPhyloExpressionSet object.

Usage

```
match_map_sc_matrix(
  expression_matrix,
  metadata,
  phylomap,
  strata_legend = NULL,
  ...
)
```

Arguments

`expression_matrix`	Expression matrix with genes as rows and cells as columns
`metadata`	Data frame with cell metadata, rownames should match colnames of expression_matrix
`phylomap`	A data frame with two columns: phylostratum assignments and gene IDs
`strata_legend`	A data frame with two columns: phylostratum assignments and name of each stratum. If NULL, numeric labels will be used (default: NULL)
`...`	Additional arguments passed to ScPhyloExpressionSet_from_matrix

Details

This function combines phylostratum mapping with expression matrix and metadata to create a ScPhyloExpressionSet. Only genes present in both the expression matrix and phylomap will be retained. All discrete metadata columns are converted to factors automatically.

Value

A ScPhyloExpressionSet object

match_map_sc_seurat *Match Single-Cell Expression Data with Phylostratum Map (Seurat)*

Description

Join single-cell gene expression data (from a Seurat object) with a phylostratum mapping to create a ScPhyloExpressionSet object. Automatically extracts dimensional reductions and metadata.

Usage

```
match_map_sc_seurat(
  seurat,
  phylomap,
  layer = "counts",
  strata_legend = NULL,
  ...
)
```

Arguments

`seurat`	A Seurat object containing single-cell expression data
`phylomap`	A data frame with two columns: phylostratum assignments and gene IDs
`layer`	Character string specifying which layer to use from the Seurat object (default: "counts")
`strata_legend`	A data frame with two columns: phylostratum assignments and name of each stratum. If NULL, numeric labels will be used (default: NULL)
`...`	Additional arguments passed to ScPhyloExpressionSet_from_seurat

Details

This is a convenience function that combines phylostratum mapping with Seurat object conversion. Only genes present in both the expression data and phylomap will be retained. The function extracts all metadata and dimensional reductions from the Seurat object.

Value

A ScPhyloExpressionSet object

`normalise_stage_expression` *Normalise Stage Expression Data*

Description

Normalise expression data to a specified total expression level per sample.

Usage

```
normalise_stage_expression(phyex_set, total = 1e+06)
```

Arguments

`phyex_set`	A PhyloExpressionSet object
`total`	Numeric value to normalise each sample to (default: 1e6)

Value

A PhyloExpressionSet object with normalised expression data

Examples

```
# Normalise to 1 million total expression per sample
normalised_set <- normalise_stage_expression(example_phyex_set, total = 1e6)
```

`omit_matrix` *Compute TXI Profiles Omitting Each Gene*

Description

For each gene i, exclude the corresponding gene i from the PhyloExpressionSet and compute the TXI profile for the dataset with gene i excluded.

This procedure results in a TXI profile matrix storing the TXI profile for each omitted gene i.

Usage

```
omit_matrix(phyex_set)
```

Arguments

`phyex_set`	A PhyloExpressionSet object

Details

This function systematically removes each gene and recalculates the transcriptomic index profile to assess the contribution of individual genes to the overall pattern. This is useful for identifying genes that have a large influence on the phylotranscriptomic signature.

Value

A numeric matrix storing TXI profiles for each omitted gene i

Author(s)

Hajk-Georg Drost

Examples

```
# Compute omit matrix for a PhyloExpressionSet
omit_mat <- omit_matrix(example_phyex_set)
```

`permute_PS` *Permute Strata in PhyloExpressionSet*

Description

Returns a copy of the PhyloExpressionSet with permuted strata and corresponding strata values.

Usage

```
permute_PS(phyex_set)
```

Arguments

`phyex_set`	A PhyloExpressionSet object

Value

A new PhyloExpressionSet object with permuted strata and strata_values

PhyloExpressionSetBase

PhyloExpressionSet Base Class

Description

Abstract S7 base class for storing and manipulating phylotranscriptomic expression data. This class provides the common interface for both bulk and single-cell phylotranscriptomic data.

Usage

```
PhyloExpressionSetBase(
  strata = stop("@strata is required"),
  strata_values = stop("@strata_values is required"),
  expression = stop("@expression is required"),
  groups = stop("@groups is required"),
  name = "Phylo Expression Set",
  species = character(0),
  index_type = "TXI",
  identities_label = "Identities",
  gene_ids = character(0),
  null_conservation_sample_size = 5000L,
  .null_conservation_txis = NULL
)
```

Arguments

`strata`	Factor vector of phylostratum assignments for each gene
`strata_values`	Numeric vector of phylostratum values used in TXI calculations
`expression`	Matrix of expression counts with genes as rows and samples as columns
`groups`	Factor vector indicating which identity each sample belongs to
`name`	Character string naming the dataset (default: "Phylo Expression Set")
`species`	Character string specifying the species (default: NULL)
`index_type`	Character string specifying the transcriptomic index type (default: "TXI")
`identities_label`	Character string labeling the identities (default: "Identities")
`gene_ids`	Character vector of gene identifiers (default: character(0), auto-generated from expression rownames if not provided)
`null_conservation_sample_size`	Numeric value for null conservation sample size (default: 5000)
`.null_conservation_txis`	Precomputed null conservation TXI values (default: NULL)

Details

The PhyloExpressionSetBase class serves as the foundation for phylotranscriptomic analysis, providing shared functionality for both bulk and single-cell data types.

Abstract Properties: Subclasses must implement the `expression_collapsed` property to define how expression data should be collapsed across replicates or cells.

Computed Properties: Several properties are computed automatically when accessed:

- `gene_ids` - Character vector of gene identifiers (rownames of expression matrix)
- `identities` - Character vector of identity labels (colnames of collapsed expression)
- `sample_names` - Character vector of sample names (colnames of expression matrix)
- `num_identities` - Integer count of unique identities
- `num_samples` - Integer count of total samples
- `num_genes` - Integer count of genes
- `num_strata` - Integer count of phylostrata
- `index_full_name` - Full name of the transcriptomic index type
- `group_map` - List mapping identity names to sample names
- `TXI` - Numeric vector of TXI values for each identity (computed from collapsed expression)
- `TXI_sample` - Numeric vector of TXI values for each sample (computed from raw expression)
- `null_conservation_txis` - Matrix of null conservation TXI values for statistical testing

Validation: The class ensures consistency between expression data, phylostratum assignments, and groupings. All gene-level vectors must have matching lengths, and sample groupings must be consistent.

Value

A PhyloExpressionSetBase object

`plot_contribution` *Plot Phylostratum Contribution to Transcriptomic Index*

Description

Create a stacked area plot showing the contribution of each phylostratum to the overall transcriptomic index across developmental stages or cell types.

Usage

```
plot_contribution(phyex_set, type = c("stacked", "line"))
```

Arguments

`phyex_set`	A PhyloExpressionSet object (BulkPhyloExpressionSet or ScPhyloExpression-Set)
`type`	One of "stacked" or "line". If stacked, will show the lines stacked in a cumulative area plot, otherwise they will not be stacked.

Details

This function visualizes how different phylostrata contribute to the overall transcriptomic index pattern across developmental stages (bulk data) or cell types (single-cell data). Each area represents the contribution of a specific phylostratum, with older strata typically shown in darker colors and younger strata in lighter colors.

The plot uses the PS_colours function to create a consistent color scheme that matches other myTAI visualizations.

Value

A ggplot2 object showing phylostratum contributions as a stacked area plot

Examples

```
# Create contribution plot for bulk data
contrib_plot <- plot_contribution(example_phyex_set)

# Create contribution plot for single-cell data
contrib_plot_sc <- plot_contribution(example_phyex_set_sc)
```

`plot_cullen_frey` *Plot Cullen-Frey Diagram for Distribution Assessment*

Description

Create a Cullen-Frey diagram to assess which distribution family best fits the null sample data.

Usage

```
plot_cullen_frey(test_result)
```

Arguments

`test_result`	A TestResult object

Details

The Cullen-Frey diagram plots skewness vs. kurtosis to help identify appropriate distribution families for the null sample data.

Value

A Cullen-Frey plot from the fitdistrplus package

`plot_distribution_expression`
Comparing expression levels distributions across developmental stages

---

Description

`plot_distribution_expression` generates plots that help to compare the distribution of expression levels through various developmental stages or cell types, highlighting each stage with distinct colors. By default, a log transformation is applied to the expression values.

Usage

```
plot_distribution_expression(
  phyex_set,
  show_identities = TRUE,
  show_strata = FALSE,
  log_transform = TRUE,
  seed = 123
)
```

Arguments

`phyex_set`	A PhyloExpressionSet object (BulkPhyloExpressionSet or ScPhyloExpressionSet).
`show_identities`	Logical, whether to show identity-specific distributions.
`show_strata`	Logical, whether to show stratum-specific distributions.
`log_transform`	Logical, whether to apply log transformation to expression values (default: TRUE).
`seed`	Seed for reproducible color selection.

Value

A ggplot2 object showing expression levels distributions across identities

Recommendation

Apply a square root transformation to enhance the visualization of differences in the distributions:
`plot_distribution_expression(transform_counts(phyex_set, sqrt))`

Author(s)

Filipa Martins Costa

plot_distribution_pTAI

Partial TAI Distribution Plotting Functions

Description

Functions for plotting and comparing partial TAI distributions using PhyloExpressionSet S7 objects.

plot_distribution_pTAI generates 2 plots that help to compare the distribution of the quotient of expression by partial TAI through various developmental stages or cell types, highlighting each stage with distinct colors.

Usage

```
plot_distribution_pTAI(
  phyex_set,
  stages = NULL,
  xlab = "Expression / Partial TAI",
  ylab = "Density",
 main = "Density Distribution of Expression / Partial TAI by Developmental Stage"
)
```

Arguments

`phyex_set`	A PhyloExpressionSet object (BulkPhyloExpressionSet or ScPhyloExpressionSet).
`stages`	A numeric vector specifying the indices of the stages to compare. Each index corresponds to a stage in the PhyloExpressionSet. If NULL, all stages are used.
`xlab`	Label of x-axis.
`ylab`	Label of y-axis.
`main`	Figure title.

Value

A ggplot2 object showing partial TAI distributions

Author(s)

Filipa Martins Costa

plot_distribution_pTAI_qqplot *QQ plot comparing partial TAI distributions across developmental stages against a reference stage*

Description

plot_distribution_partialTAI_qqplot generates a QQ plot to compare the partial TAI distributions of various developmental stages against a reference stage. It visualizes quantile differences between the reference and other stages, highlights each stage with distinct colors, and annotates the plot with the p-values from the nonparametric `ks.test` to indicate the significance of distribution differences.

Usage

```
plot_distribution_pTAI_qqplot(
  phyex_set,
  reference_stage_index = 1,
  xlab = "Quantiles of Reference Stage",
  ylab = "Quantiles of Other Stages",
  main = "QQ Plot: Developmental Stages vs Reference Stage (p-values from KS test)",
  alpha = 0.7,
  size = 1.2
)
```

Arguments

`phyex_set`	A PhyloExpressionSet object (BulkPhyloExpressionSet or ScPhyloExpressionSet).
`reference_stage_index`	An integer specifying the index of the reference developmental stage. The partial TAI distribution of this stage will be used as the reference for comparisons with other stages (default: stage index 1).
`xlab`	Label of x-axis.
`ylab`	Label of y-axis.
`main`	Figure title.
`alpha`	Transparency of the points.
`size`	Size of the points.

Value

A ggplot2 object showing a qqplot of partial TAI distributions

Author(s)

Filipa Martins Costa

plot_distribution_strata — *Plot Distribution of Genes Across Phylostrata*

Description

Create a bar plot showing the distribution of genes across phylostrata, with options for showing observed vs. expected ratios.

Usage

```
plot_distribution_strata(
  strata,
  selected_gene_ids = names(strata),
  as_log_obs_exp = FALSE
)
```

Arguments

`strata`	Named factor vector of phylostratum assignments (names are gene IDs)
`selected_gene_ids`	Character vector of gene IDs to include in the plot (default: all genes in strata)
`as_log_obs_exp`	Logical indicating whether to show log2(observed/expected) ratios instead of raw counts (default: FALSE)

Details

This function visualizes how genes are distributed across different phylostrata. When as_log_obs_exp=FALSE, it shows raw gene counts per stratum. When TRUE, it shows log2 ratios of observed vs. expected gene counts, useful for identifying enrichment or depletion of specific strata in gene sets.

Value

A ggplot2 object showing the phylostratum distribution

Examples

```
# Plot raw gene counts by strata
p1 <- plot_distribution_strata(example_phyex_set@strata)

# Plot observed vs expected ratios for selected genes
p2 <- plot_distribution_strata(example_phyex_set@strata,
                               selected_gene_ids = example_phyex_set@gene_ids[5:20],
                               as_log_obs_exp = TRUE)
```

plot_gatai_results *Plot Comprehensive GATAI Results*

Description

Create a suite of plots summarizing the effects of GATAI gene removal on phylotranscriptomic patterns.

Usage

```
plot_gatai_results(
  phyex_set,
  gatai_result,
  conservation_test = stat_flatline_test,
  runs_threshold = 0.5,
  signature_plot_type = c("separate", "combined")
)
```

Arguments

`phyex_set`	A PhyloExpressionSet object containing the original gene expression data.
`gatai_result`	Result list from `destroy_pattern()`, containing GATAI analysis output.
`conservation_test`	Function for conservation test (default: `stat_flatline_test`).
`runs_threshold`	Threshold for gene removal consistency across runs (default: 0.5).
`signature_plot_type`	Type of signature plot: "separate" for individual plots, "combined" for overlay (default: both options).

Details

This function provides a comprehensive visualization of the impact of GATAI gene removal, including transcriptomic signature plots, gene expression profiles, heatmaps, mean-variance relationships, phylostrata distributions, conservation test comparisons, and convergence diagnostics.

Value

A named list of ggplot/patchwork objects and results:

`signature_plots`	Signature plots before/after GATAI and top variance removal
`heatmap_plot`	Heatmap of GATAI-removed genes
`profiles_plot`	Gene expression profiles of GATAI-removed genes
`profiles_plot_facet`	Faceted gene profiles by strata

`gene_space_plot`	Gene space plot of GATAI-removed genes
`mean_var_plot`	Mean-variance plot highlighting GATAI-removed genes
`strata_plot`	Phylostrata distribution plot (log obs/exp) for GATAI-removed genes
`null_dist_plot`	Null distribution plot with test statistics and p-values
`convergence_plots`	GATAI convergence plots (if available)

Author(s)

Filipa Martins Costa, Stefan Manolache

`plot_gene_heatmap` *Plot Gene Expression Heatmap*

Description

Create a heatmap showing gene expression patterns across conditions with optional dendrograms and gene age annotation.

Usage

```
plot_gene_heatmap(
  phyex_set,
  genes = NULL,
  top_p = NULL,
  top_k = 30,
  std = TRUE,
  show_reps = FALSE,
  cluster_rows = FALSE,
  cluster_cols = FALSE,
  show_gene_age = TRUE,
  show_phylostrata_legend = TRUE,
  show_gene_ids = FALSE,
  gene_annotation = NULL,
  gene_annotation_colors = NULL,
  ...
)
```

Arguments

`phyex_set`	A PhyloExpressionSet object (BulkPhyloExpressionSet or ScPhyloExpressionSet)
`genes`	Character vector of specific gene IDs to include in the heatmap (default: NULL for auto-selection of dynamic genes)

`top_p`	Numeric value specifying the top proportion of genes to include (default: NULL). Ignored if top_k is specified.
`top_k`	Absolute number of top genes to select (default: 30). Takes precedence over top_p.
`std`	Logical indicating whether to standardize expression values (default: TRUE)
`show_reps`	Logical indicating whether to show replicates or collapsed data (default: FALSE)
`cluster_rows`	Logical indicating whether to cluster genes/rows (default: FALSE)
`cluster_cols`	Logical indicating whether to cluster conditions/columns (default: FALSE)
`show_gene_age`	Logical indicating whether to show gene age annotation (default: TRUE)
`show_phylostrata_legend`	Logical indicating whether to show the phylostratum legend (default: TRUE)
`show_gene_ids`	Logical indicating whether to show gene identifiers (default: FALSE)
`gene_annotation`	Data frame with custom gene annotations, rownames should match gene IDs (default: NULL)
`gene_annotation_colors`	Named list of color vectors for custom gene annotations (default: NULL)
`...`	Additional arguments passed to specific methods

Details

This function creates a comprehensive heatmap visualization of gene expression patterns. By default, genes are ordered by their expression angle (developmental trajectory). The function supports clustering of both genes and identities, and can optionally display gene age (phylostratum) as a colored annotation bar.

For bulk data, the heatmap shows expression across developmental conditions. For single-cell data, the heatmap shows expression across cell types.

The gene age annotation uses the PS_colours function to create a consistent color scheme across different myTAI visualizations.

Custom gene annotations can be provided via the `gene_annotation` parameter, which should be a data frame with gene IDs as rownames and annotation categories as columns. Corresponding colors should be provided via `gene_annotation_colors` as a named list where names match the annotation column names.

Value

A ggplot object (converted from pheatmap) showing the gene expression heatmap

Examples

```
# Basic heatmap with gene age annotation
p1 <- plot_gene_heatmap(example_phyex_set, show_gene_age = TRUE)

# Single-cell heatmap with subset of cells
p2 <- plot_gene_heatmap(example_phyex_set_sc, show_reps = TRUE, max_cells_per_type = 3)
```

```
# Custom gene annotation example
gene_ids <- example_phyex_set@gene_ids[1:3]
gene_annot <- data.frame(
  Category = c("High", "Medium", "Low"),
  row.names = gene_ids
)
colors <- list(Category = c("High" = "red", "Medium" = "yellow", "Low" = "blue"))
p3 <- plot_gene_heatmap(example_phyex_set |> select_genes(gene_ids), gene_annotation = gene_annot,
                        gene_annotation_colors = colors, show_gene_age = FALSE)
```

plot_gene_profiles *Plot Individual Gene Expression Profiles*

Description

Create a plot showing expression profiles for individual genes across developmental stages or cell types, with various visualization options.

Usage

```
plot_gene_profiles(
  phyex_set,
  genes = NULL,
  show_set_mean = FALSE,
  show_reps = FALSE,
  transformation = c("log", "std_log", "none"),
  colour_by = c("manual", "strata", "stage"),
  colours = NULL,
  max_genes = 100,
  show_labels = TRUE,
  label_size = 1.75,
  show_legend = TRUE,
  facet_by_strata = FALSE
)
```

Arguments

`phyex_set`	A PhyloExpressionSet object (BulkPhyloExpressionSet or ScPhyloExpression-Set)
`genes`	Character vector of gene IDs to plot. If NULL, top expressing genes are selected
`show_set_mean`	Logical indicating whether to show the mean expression across all genes (default: FALSE)
`show_reps`	Logical indicating whether to show individual replicates (bulk) or cells (single-cell) (default: FALSE)
`transformation`	Character string specifying expression transformation: "log" (log1p), "std_log" (standardized log1p), or "none" (default: "log")

`colour_by`	Character string specifying coloring scheme: "strata" (by phylostratum), "stage" (by developmental stage/cell type), or "manual" (default: "manual")
`colours`	Optional vector of colors for manual coloring (default: NULL)
`max_genes`	Maximum number of genes to plot when genes=NULL (default: 100)
`show_labels`	Logical indicating whether to show gene labels (default: TRUE)
`label_size`	Font size of gene id labels if shown (default: 0.5).
`show_legend`	Logical indicating whether to show legend (default: TRUE)
`facet_by_strata`	Logical indicating whether to facet by phylostratum (default: FALSE)

Details

This function creates detailed visualizations of individual gene expression patterns across development (bulk data) or cell types (single-cell data). Genes can be colored by phylostratum or developmental stage, and various transformations can be applied to highlight different aspects of the data.

Value

A ggplot2 object showing gene expression profiles

Author(s)

Filipa Martins Costa, Stefan Manolache, Hajk-Georg Drost

Examples

```
# Plot specific genes for bulk data
p1 <- plot_gene_profiles(example_phyex_set, genes = example_phyex_set@gene_ids[1:5])

# Plot for single-cell data with faceting by strata
p2 <- plot_gene_profiles(example_phyex_set_sc, facet_by_strata = TRUE)
```

`plot_gene_space` *Plot Gene Space Using PCA*

Description

Create a PCA plot showing genes in expression space with ideal expression patterns overlaid as reference points.

Usage

```
plot_gene_space(
  phyex_set,
  top_p = 0.2,
  genes = NULL,
  colour_by = c("identity", "strata")
)
```

Arguments

`phyex_set`	A PhyloExpressionSet object (BulkPhyloExpressionSet or ScPhyloExpressionSet)
`top_p`	Proportion of most dynamic genes to include when genes=NULL (default: 0.2)
`genes`	Character vector of specific genes to plot. If NULL, uses top dynamic genes
`colour_by`	Character string specifying coloring scheme: "identity" (by peak expression stage/identity) or "strata" (by phylostratum) (default: "identity")

Details

This function creates a PCA visualization of genes in expression space, with ideal expression patterns (early, mid, late, reverse mid) overlaid as reference points. The analysis uses log-transformed and standardized expression values. Genes are colored either by their phylostratum or by their peak expression stage.

Value

A ggplot2 object showing the gene space PCA plot

Examples

```
# Plot gene space colored by identity
p1 <- plot_gene_space(example_phyex_set, colour_by = "identity")

# Plot specific genes colored by strata
p2 <- plot_gene_space(example_phyex_set,
                      genes = example_phyex_set@gene_ids[1:5],
                      colour_by = "strata")
```

`plot_mean_var` *Plot Mean-Variance Relationship*

Description

Create a scatter plot showing the relationship between mean expression and variance for genes, colored by phylostratum, with optional highlighting and labeling of specific genes.

Usage

```
plot_mean_var(
  phyex_set,
  highlight_genes = NULL,
  colour_by = c("none", "strata")
)
```

Arguments

`phyex_set`	A PhyloExpressionSet object (BulkPhyloExpressionSet or ScPhyloExpressionSet) containing gene expression data.
`highlight_genes`	Optional character vector of gene IDs to highlight and label on the plot.
`colour_by`	Character string specifying coloring scheme: "none" (default), "strata" colors by phylostratum

Details

This function plots the mean expression versus variance for each gene, with points colored by phylostratum. Optionally, specific genes can be highlighted and labeled. This visualization helps identify expression patterns and heteroscedasticity in the data.

The function uses collapsed expression data (averaged across replicates for bulk data, or averaged across cells per cell type for single-cell data).

Value

A ggplot2 object showing the mean-variance relationship.

Examples

```
# Create mean-variance plot for bulk data
mv_plot <- plot_mean_var(example_phyex_set)

# Highlight and label specific genes in single-cell data
mv_plot_sc <- plot_mean_var(example_phyex_set_sc,
  highlight_genes = example_phyex_set_sc@gene_ids[1:3])

# Color by phylostratum
mv_plot_colored <- plot_mean_var(example_phyex_set, colour_by = "strata")
```

`plot_null_txi_sample` *Plot Null TXI Sample Distribution*

Description

Create a plot showing the null TXI distribution sample compared to the observed test TXI values across developmental stages.

Usage

```
plot_null_txi_sample(test_result)
```

Arguments

`test_result`	A ConservationTestResult object containing null TXI distributions

Details

This function creates a visualization of the null hypothesis testing by plotting: - Gray lines representing individual null TXI samples from permutations - A horizontal line showing the mean of null distributions - A colored line showing the observed test TXI values

The plot helps visualize how the observed TXI pattern compares to what would be expected under the null hypothesis of no conservation signal.

Value

A ggplot2 object showing null samples as gray lines and test TXI as colored line

`plot_relative_expression_bar` *Plot Mean Relative Expression Levels as Barplot*

Description

Plots mean relative expression levels for age category groups using a PhyloExpressionSet S7 object, with statistical testing.

Usage

```
plot_relative_expression_bar(phyex_set, groups, p_adjust_method = NULL, ...)
```

Arguments

`phyex_set`	A PhyloExpressionSet object (BulkPhyloExpressionSet or ScPhyloExpression-Set).
`groups`	A list of integer vectors specifying age categories (e.g., phylostrata) for each group (2+ groups).
`p_adjust_method`	P-value adjustment for multiple testing.
`...`	Further arguments passed to ggplot2 geoms.

Value

ggplot2 object.

`plot_relative_expression_line` *Plot Relative Expression Profiles (Line Plot)*

Description

Plots relative expression profiles for age categories using a PhyloExpressionSet S7 object.

Usage

```
plot_relative_expression_line(
  phyex_set,
  groups,
  modules = NULL,
  adjust_range = TRUE,
  alpha = 0.1,
  ...
)
```

Arguments

`phyex_set`	A PhyloExpressionSet object (BulkPhyloExpressionSet or ScPhyloExpression-Set).
`groups`	A list of integer vectors specifying age categories (e.g., phylostrata) for each group (1 or 2 groups).
`modules`	Optional list for shading modules: list(early=..., mid=..., late=...).
`adjust_range`	Logical, adjust y-axis range for both panels (if 2 groups).
`alpha`	Transparency for shaded module area.
`...`	Further arguments passed to ggplot2 geoms.

Value

ggplot2 object or list of ggplot2 objects.

plot_sample_space *Plot Sample Space Visualization*

Description

Create a dimensional reduction plot to visualize sample relationships in gene expression space using PCA or UMAP.

Usage

```
plot_sample_space(
  phyex_set,
  method = c("PCA", "UMAP"),
  colour_by = c("identity", "TXI"),
  seed = 42,
  ...
)
```

Arguments

`phyex_set`	A PhyloExpressionSet object (BulkPhyloExpressionSet or ScPhyloExpressionSet)
`method`	Character string specifying the dimensionality reduction method: "PCA" or "UMAP" (default: "PCA")
`colour_by`	Character string specifying what to colour by: "identity" (default), "TXI"
`seed`	Integer seed for reproducible UMAP results (default: 42)
`...`	Additional arguments passed to specific methods

Details

This function performs log1p transformation on expression data, removes genes with zero variance, and applies the specified dimensionality reduction method. Samples are coloured by their group assignments or TAI values.

Value

A ggplot2 object showing the sample space visualisation

Examples

```
# Create PCA plot coloured by identity
pca_plot <- plot_sample_space(example_phyex_set, method = "PCA", colour_by = "identity")

# Create UMAP plot coloured by TXI
if (requireNamespace("uwot", quietly = TRUE)) {
    umap_plot <- plot_sample_space(example_phyex_set, method = "UMAP", colour_by = "TXI")
}
```

plot_signature *Plot Transcriptomic Signature*

Description

Create a plot of the transcriptomic index signature across developmental stages or cell types, with options for showing individual samples/cells and statistical testing.

Usage

```
plot_signature(
  phyex_set,
  show_reps = TRUE,
  show_p_val = TRUE,
  conservation_test = stat_flatline_test,
  colour = NULL,
  ...
)
```

Arguments

`phyex_set`	A PhyloExpressionSet object (BulkPhyloExpressionSet or ScPhyloExpressionSet)
`show_reps`	Logical, whether to show individual replicates
`show_p_val`	Logical, whether to show conservation test p-value
`conservation_test`	Function, conservation test to use for p-value calculation
`colour`	Character, custom color for the plot elements
`...`	Additional arguments passed to specific methods

Details

This function creates visualizations appropriate for the data type:

Bulk data (BulkPhyloExpressionSet): - Line plots showing TXI trends across developmental stages - Optional individual biological replicates as jittered points - Optional conservation test p-values

Single-cell data (ScPhyloExpressionSet): - Sina plots showing TXI distributions across cell types or other identities - Mean TXI values overlaid as line - Optional individual cells using geom_sina for better visualization - Flexible identity selection from metadata via additional parameters: - ‘primary_identity‘: Character, name of metadata column for x-axis (default: current selected identities) - ‘secondary_identity‘: Character, name of metadata column for coloring/faceting - ‘facet_by_secondary‘: Logical, whether to facet by secondary identity (default: FALSE uses colouring)

Value

A ggplot2 object showing the transcriptomic signature

Examples

```
# Basic signature plot for bulk data
p <- plot_signature(example_phyex_set)
```

`plot_signature_gene_quantiles`
Plot Signature Across Gene Expression Quantiles

Description

Create a plot showing how the transcriptomic signature changes when genes are progressively removed based on expression quantiles.

Usage

```
plot_signature_gene_quantiles(
  phyex_set,
  quantiles = c(1, 0.99, 0.95, 0.9, 0.8),
  selection_FUN = genes_top_mean,
  ...
)
```

Arguments

`phyex_set`	A PhyloExpressionSet object (BulkPhyloExpressionSet or ScPhyloExpressionSet)
`quantiles`	Numeric vector of quantiles to test (default: c(1.0, 0.99, 0.95, 0.90, 0.80))
`selection_FUN`	Function to select genes for removal (default: genes_top_mean)
`...`	Additional arguments passed to plot_signature_multiple

Details

This function systematically removes genes based on expression quantiles and shows how the transcriptomic signature changes. This is useful for understanding the contribution of highly expressed genes to the overall pattern and for assessing the robustness of phylotranscriptomic patterns.

The analysis works with both bulk and single-cell data, helping to determine whether phylotranscriptomic patterns are driven by a few highly expressed genes or represent broad transcriptomic trends.

Value

A ggplot2 object showing signatures across different quantiles

Examples

```
# Plot signature across expression quantiles for bulk data
phyex_set <- example_phyex_set |>
    select_genes(example_phyex_set@gene_ids[1:100])
phyex_set@null_conservation_sample_size <- 500
p <- plot_signature_gene_quantiles(phyex_set, quantiles = c(0.95, 0.90))
```

---

plot_signature_multiple

Plot Multiple Transcriptomic Signatures

Description

Create a plot comparing multiple transcriptomic signatures on the same axes, with options for statistical testing and transformations.

Usage

```
plot_signature_multiple(
  phyex_sets,
  legend_title = "Phylo Expression Set",
  show_p_val = TRUE,
  conservation_test = stat_flatline_test,
  transformation = NULL,
  colours = NULL,
  ...
)
```

Arguments

`phyex_sets`	A vector of PhyloExpressionSet objects to compare (BulkPhyloExpressionSet or ScPhyloExpressionSet)
`legend_title`	Title for the legend (default: "Phylo Expression Set")
`show_p_val`	Logical indicating whether to show p-values (default: TRUE)
`conservation_test`	Function to use for conservation testing (default: stat_flatline_test)
`transformation`	Optional transformation function to apply to all datasets (default: NULL)
`colours`	Optional vector of colors for each dataset (default: NULL)
`...`	Additional arguments passed to plot_signature

Details

This function allows comparison of multiple transcriptomic signatures by overlaying them on the same plot. Each signature is colored differently and can be tested for conservation patterns (bulk data only). If a transformation is provided, it's applied to all datasets before plotting.

The function automatically adapts to the data type: - **Bulk data**: Line plots with optional statistical testing - **Single-cell data**: Violin plots showing distributions

All datasets must use the same axis labels (developmental stages or cell types).

Value

A ggplot2 object showing multiple transcriptomic signatures

Examples

```
# Compare multiple bulk datasets
phyex_set <- example_phyex_set

bulk_list <- c(phyex_set,
  phyex_set |> remove_genes(phyex_set@gene_ids[1:5]))
p <- plot_signature_multiple(bulk_list, legend_title = "Dataset")
```

`plot_signature_transformed` *Plot Signature Under Different Transformations*

Description

Compare transcriptomic signatures under various data transformations to assess the robustness of phylotranscriptomic patterns.

Usage

```
plot_signature_transformed(phyex_set, transformations = COUNT_TRANSFORMS, ...)
```

Arguments

`phyex_set`	A PhyloExpressionSet object (BulkPhyloExpressionSet or ScPhyloExpressionSet)
`transformations`	Named list of transformation functions (default: COUNT_TRANSFORMS)
`...`	Additional arguments passed to plot_signature_multiple

Details

This function applies different transformations to the same dataset and compares the resulting transcriptomic signatures. This is useful for assessing whether phylotranscriptomic patterns are robust to different data processing approaches or are artifacts of specific transformations.

The analysis works with both bulk and single-cell data, helping to determine whether phylotranscriptomic patterns are consistent across different normalization and transformation methods.

Value

A ggplot2 object showing signatures under different transformations

Examples

```
# Single-cell data with custom transformations

phyex_set <- example_phyex_set

custom_transforms <- list(raw = identity, log = log1p)
p <- plot_signature_transformed(phyex_set, transformations = custom_transforms)
```

---

plot_strata_expression

Plot Expression Levels by Phylostratum

Description

Create a boxplot showing the distribution of expression levels for each phylostratum.

Usage

```
plot_strata_expression(phyex_set, aggregate_FUN = mean)
```

Arguments

`phyex_set`	A PhyloExpressionSet object (BulkPhyloExpressionSet or ScPhyloExpressionSet)
`aggregate_FUN`	Function to aggregate expression across identities (default: mean)

Details

This function creates a boxplot visualization showing how expression levels vary across different phylostrata. Each point represents a gene, and the boxes show the distribution of expression levels within each phylostratum.

For bulk data, expression is aggregated across developmental stages. For single-cell data, expression is aggregated across cell types.

Value

A ggplot2 object showing expression distributions by phylostratum

Examples

```
# Plot expression by strata using mean aggregation for bulk data
p1 <- plot_strata_expression(example_phyex_set, aggregate_FUN = mean)

# Plot using median aggregation for single-cell data
p2 <- plot_strata_expression(example_phyex_set_sc, aggregate_FUN = median)
```

`PS_colours` *Generate Phylostratum Colors*

Description

Generate a color palette for phylostrata visualization using a log-scaled transformation.

Usage

```
PS_colours(n)
```

Arguments

`n`	number of colors to generate

Value

A character vector of color codes

Examples

```
# Generate colors for 5 phylostrata
colors <- PS_colours(5)
```

pTXI *Calculate Phylostratum-Specific Transcriptomic Index*

Description

Calculate pTXI values for expression data. This is a generic function that dispatches based on the input type.

Usage

```
pTXI(phyex_set, reps = FALSE)
```

Arguments

`phyex_set`	A PhyloExpressionSet object
`reps`	Whether to return pTXI for each sample instead for each group

Value

Matrix of pTXI values

quantile_rank *Calculate Quantile Ranks*

Description

Calculate quantile ranks for a numeric vector, handling ties using average method.

Usage

```
quantile_rank(x)
```

Arguments

`x`	numeric vector for which to calculate quantile ranks

Value

A numeric vector of quantile ranks between 0 and 1

Examples

```
# Calculate quantile ranks for a vector
ranks <- quantile_rank(c(1, 2, 3, 4, 5))
```

`relative_expression` *Relative Expression Functions*

Description

Functions for computing and plotting relative expression profiles using PhyloExpressionSet S7 objects.

Computes the relative expression profile for a given gene expression matrix. The relative expression is calculated by normalizing the column means of the matrix to a [0, 1] scale.

Usage

```
relative_expression(count_matrix)
```

Arguments

`count_matrix`	A numeric matrix where columns represent developmental stages/cell types and rows represent genes.

Value

A numeric vector of relative expression values for each stage (column) in the input matrix.

`rel_exp_matrix` *Compute Relative Expression Matrix for PhyloExpressionSet*

Description

Computes relative expression profiles for all age categories in a PhyloExpressionSet.

Usage

```
rel_exp_matrix(phyex_set)
```

Arguments

`phyex_set`	A PhyloExpressionSet object (BulkPhyloExpressionSet or ScPhyloExpressionSet).

Value

A matrix with age categories as rows and identities as columns, containing relative expression values.

`remove_genes` *Remove Genes from PhyloExpressionSet*

Description

Remove specified genes from a PhyloExpressionSet object.

Usage

```
remove_genes(
  phyex_set,
  genes,
  new_name = paste(phyex_set@name, "perturbed"),
  reuse_null_txis = TRUE
)
```

Arguments

`phyex_set`	A PhyloExpressionSet object
`genes`	Character vector of gene IDs to remove
`new_name`	Character string for the new dataset name (default: auto-generated)
`reuse_null_txis`	Logical indicating whether to reuse precomputed null conservation TXIs (default: TRUE)

Value

A PhyloExpressionSet object with the specified genes removed

Examples

```
# Remove specific genes
filtered_set <- remove_genes(example_phyex_set, example_phyex_set@gene_ids[1:5],
                             new_name = "Filtered Dataset")
```

`rename_phyex_set` *Rename a PhyloExpressionSet*

Description

Returns a copy of the PhyloExpressionSet with a new name.

Usage

```
rename_phyex_set(phyex_set, new_name)
```

Arguments

`phyex_set`	A PhyloExpressionSet object
`new_name`	Character string for the new dataset name

Value

The PhyloExpressionSet object with the updated name

`save_gatai_results_pdf` *Save GATAI Analysis Results to PDF*

Description

Save removed gene IDs and all GATAI analysis plots to a PDF file.

Usage

```
save_gatai_results_pdf(
  phyex_set,
  gatai_result,
  analysis_dir = "gatai_analysis",
  prefix = "report",
  ...
)
```

Arguments

`phyex_set`	A PhyloExpressionSet object containing the original gene expression data.
`gatai_result`	Result list from `destroy_pattern()`, containing GATAI analysis output.
`analysis_dir`	Directory to save the PDF file.
`prefix`	Optional prefix for the PDF filename (default: "report").
`...`	Additional arguments passed to `plot_gatai_results()`.

Value

Invisibly returns the path to the saved PDF.

ScPhyloExpressionSet *Single-Cell PhyloExpressionSet Class*

Description

S7 class for single-cell phylotranscriptomic expression data. This class stores expression matrices and metadata, with support for dimensional reductions and pseudobulking functionality.

Usage

```
ScPhyloExpressionSet(
  strata = stop("@strata is required"),
  strata_values = stop("@strata_values is required"),
  expression = stop("@expression is required"),
  groups = stop("@groups is required"),
  name = "Phylo Expression Set",
  species = character(0),
  index_type = "TXI",
  identities_label = "Identities",
  gene_ids = character(0),
  null_conservation_sample_size = 5000L,
  .null_conservation_txis = NULL,
  .pseudobulk_cache = list(),
  .TXI_sample = numeric(0),
  metadata = NULL,
  selected_idents = character(0),
  idents_colours = list(),
  reductions = list()
)
```

Arguments

`strata`	Factor vector of phylostratum assignments for each gene
`strata_values`	Numeric vector of phylostratum values used in TXI calculations
`expression`	Sparse or dense matrix of expression counts with genes as rows and cells as columns
`groups`	Factor vector indicating which identity each cell belongs to (derived from selected_idents column in metadata)
`name`	Character string naming the dataset (default: "Phylo Expression Set")
`species`	Character string specifying the species (default: NULL)
`index_type`	Character string specifying the transcriptomic index type (default: "TXI")
`identities_label`	Character string labeling the identities (default: "Cell Type")
`gene_ids`	Character vector of gene identifiers (default: character(0), auto-generated from expression rownames if not provided)

`null_conservation_sample_size`	Numeric value for null conservation sample size (default: 5000)
`.null_conservation_txis`	Precomputed null conservation TXI values (default: NULL)
`.pseudobulk_cache`	Internal cache for pseudobulked expression matrices by different groupings
`.TXI_sample`	Internal storage for computed TXI values
`metadata`	Data frame with cell metadata, where rownames correspond to cell IDs and columns contain cell attributes
`selected_idents`	Character string specifying which metadata column is currently used for grouping cells
`idents_colours`	List of named character vectors specifying colors for each identity level, organized by metadata column name
`reductions`	List of dimensional reduction matrices (PCA, UMAP, etc.) with cells as rows and dimensions as columns

Details

The ScPhyloExpressionSet class provides a comprehensive framework for single-cell phylotranscriptomic analysis. Key features include:

Identity Management: The `selected_idents` property determines which metadata column is used for grouping cells. When changed, it automatically updates the `groups` property and invalidates cached pseudobulk data to ensure consistency.

Dimensional Reductions: The `reductions` property stores pre-computed dimensional reductions (PCA, UMAP, etc.). If not provided during construction from Seurat objects, basic PCA and UMAP are computed automatically.

Color Management: `idents_colours` allows custom color schemes for different metadata columns, ensuring consistent visualization across plots.

Computed Properties: Several properties are computed automatically when accessed:

- `available_idents` - Character vector of factor columns in metadata that can be used for grouping (automatically detected from metadata)
- `expression_collapsed` - Matrix of pseudobulked expression data (genes x identities), created by summing expression within each identity group
- `TXI_sample` - Named numeric vector of TXI (Transcriptomic Age Index) values for each cell, computed using efficient C++ implementation

Inherited computed properties from PhyloExpressionSetBase include:

- `gene_ids` - Character vector of gene identifiers
- `identities` - Character vector of identity labels
- `sample_names` - Character vector of sample names (cell IDs)
- `num_identities` - Integer count of unique cell types/identities
- `num_samples` - Integer count of total cells

- `num_genes` - Integer count of genes
- `num_strata` - Integer count of phylostrata
- `index_full_name` - Full name of the transcriptomic index type
- `group_map` - List mapping identity names to cell IDs
- `TXI` - Numeric vector of TXI values for each identity (computed from pseudobulked expression)
- `null_conservation_txis` - Matrix of null conservation TXI values for statistical testing

These properties use lazy evaluation and caching for optimal performance.

Value

A ScPhyloExpressionSet object

Examples

```
# Create from Seurat object
data(example_phyex_set_sc)
sc_set <- example_phyex_set_sc

# Switch to different cell grouping
sc_set@selected_idents <- "day"

# Access pseudobulked data (computed au  tomatically)
pseudobulk <- sc_set@expression_collapsed

# Access TXI values for each cell
txi_values <- sc_set@TXI_sample
```

`ScPhyloExpressionSet_from_matrix`
Create Single-Cell PhyloExpressionSet from Expression Matrix

Description

Create a ScPhyloExpressionSet object from an expression matrix and metadata.

Usage

```
ScPhyloExpressionSet_from_matrix(
  expression_matrix,
  strata,
  metadata,
  groups_column = NULL,
  name = "Single-Cell Phylo Expression Set",
  ...
)
```

Arguments

`expression_matrix`	Sparse or dense expression matrix with genes as rows and cells as columns
`strata`	Factor vector of phylostratum assignments for each gene
`metadata`	Data frame with cell metadata, rownames should match colnames of expression_matrix
`groups_column`	Character string specifying which metadata column to use for initial grouping (default: first factor column found)
`name`	A character string naming the dataset (default: "Single-Cell Phylo Expression Set")
`...`	Additional arguments passed to ScPhyloExpressionSet constructor

Details

This function creates a ScPhyloExpressionSet from basic components. The `groups_column` parameter determines the initial `selected_idents` value, which can be changed later using the setter. All discrete columns in metadata are automatically converted to factors for consistent handling.

Value

A ScPhyloExpressionSet object

`ScPhyloExpressionSet_from_seurat`
Convert Seurat Object to Single-Cell PhyloExpressionSet

Description

Convert a Seurat object with phylostratum information into a ScPhyloExpressionSet object for single-cell phylotranscriptomic analysis. Automatically extracts dimensional reductions if present, or computes basic PCA and UMAP if none are available.

Usage

```
ScPhyloExpressionSet_from_seurat(
  seurat,
  strata,
  layer = "counts",
  selected_idents = NULL,
  name = "Single-Cell PhyloExpressionSet",
  seed = 42,
  ...
)
```

Arguments

`seurat`	A Seurat object containing single-cell expression data
`strata`	Factor vector of phylostratum assignments for each gene
`layer`	Character string specifying which layer to use from the Seurat object (default: "counts")
`selected_idents`	Character string specifying which metadata column to use for grouping (default: NULL, uses active idents)
`name`	A character string naming the dataset (default: "Single-Cell Phylo Expression Set")
`seed`	Integer seed for reproducible UMAP computation (default: 42)
`...`	Additional arguments passed to ScPhyloExpressionSet constructor

Value

A ScPhyloExpressionSet object

`select_genes` *Select Genes from PhyloExpressionSet*

Description

Extract a subset of genes from a PhyloExpressionSet object.

Usage

```
select_genes(phyex_set, ...)
```

Arguments

`phyex_set`	A PhyloExpressionSet object
`...`	Additional arguments passed to methods (typically includes 'genes' parameter)

Value

A PhyloExpressionSet object containing only the selected genes

Examples

```
# Select specific genes
selected_set <- select_genes(example_phyex_set, example_phyex_set@gene_ids[1:10])
```

`set_expression` *Gene Expression Transformation Functions*

Description

Collection of functions for transforming gene expression data in PhyloExpressionSet objects.

Generic function to set the expression matrix in a PhyloExpressionSet object.

Usage

```
set_expression(phyex_set, new_expression, new_name = NULL, ...)
```

Arguments

`phyex_set`	A PhyloExpressionSet object
`new_expression`	Matrix to set as the new expression
`new_name`	Optional new name for the dataset
`...`	Additional arguments

Value

A PhyloExpressionSet object with updated expression

`stat_early_conservation_test` *Early Conservation Test*

Description

Test for early conservation patterns in transcriptomic data by comparing early developmental stages to mid and late stages.

Usage

```
stat_early_conservation_test(phyex_set, modules, ...)
```

Arguments

`phyex_set`	A PhyloExpressionSet object
`modules`	A named list with elements 'early', 'mid', and 'late' containing stage indices for each developmental module
`...`	Additional arguments passed to stat_generic_conservation_test

Details

The early conservation test evaluates whether early developmental stages show lower transcriptomic index values (indicating older genes) compared to later stages. The test computes a score based on the minimum difference between mid vs. early and late vs. early TXI values.

Value

A ConservationTestResult object with early conservation test results

See Also

`stat_generic_conservation_test`, `stat_late_conservation_test`

Examples

```
# Define developmental modules
p <- example_phyex_set_old |>
     select_genes(example_phyex_set_old@gene_ids[1:100])
modules <- list(early = 1:2, mid = 3:5, late = 6:7)
result <- stat_early_conservation_test(p, modules=modules)
```

`stat_flatline_test` *Flat Line Test for Conservation Pattern*

Description

Perform a flat line test to assess whether the transcriptomic index profile shows a flat (non-varying) pattern across developmental stages.

Usage

```
stat_flatline_test(phyex_set, ...)
```

Arguments

`phyex_set`	A PhyloExpressionSet object
`...`	Additional arguments passed to stat_generic_conservation_test

Details

The flat line test evaluates whether the TXI profile remains constant across developmental stages by testing the variance of the profile against a null distribution. A significant result indicates rejection of the flat line pattern.

Value

A test result object containing p-value and test statistics

See Also

`stat_generic_conservation_test`

Examples

```
# Perform flat line test
result <- stat_flatline_test(example_phyex_set)
```

`stat_generic_conservation_test`
Generic Conservation Test Framework

Description

Perform a generic conservation test by comparing observed TXI patterns against a null distribution generated by permutation.

Usage

```
stat_generic_conservation_test(
  phyex_set,
  test_name,
  scoring_function,
  fitting_dist,
  alternative = c("two-sided", "greater", "less"),
  p_label = p_label,
  custom_null_txis = NULL,
  plot_result = TRUE
)
```

Arguments

`phyex_set`	A PhyloExpressionSet object
`test_name`	Character string naming the test
`scoring_function`	Function to compute test statistic from TXI profile
`fitting_dist`	Distribution object for fitting the null distribution
`alternative`	Character string specifying alternative hypothesis: "two-sided", "greater", or "less" (default: "two-sided")
`p_label`	Label for p-value in results (default: p_label)
`custom_null_txis`	Optional custom null TXI distribution (default: NULL)
`plot_result`	Logical indicating whether to plot results (default: TRUE)

Details

This function provides a generic framework for conservation testing by: 1. Generating null TXI distributions via permutation 2. Computing test statistics using the provided scoring function 3. Fitting the specified distribution to the null sample 4. Computing p-values based on the alternative hypothesis

Value

A ConservationTestResult object containing test statistics and p-values

See Also

`stat_flatline_test`, `stat_early_conservation_test`

`stat_late_conservation_test`

Late Conservation Test

Description

Test for late conservation patterns in transcriptomic data by comparing late developmental stages to early and mid stages.

Usage

```
stat_late_conservation_test(phyex_set, modules, ...)
```

Arguments

`phyex_set`	A PhyloExpressionSet object
`modules`	A named list with elements 'early', 'mid', and 'late' containing stage indices for each developmental module
`...`	Additional arguments passed to stat_generic_conservation_test

Details

The late conservation test evaluates whether later developmental stages show lower transcriptomic index values (indicating older genes) compared to earlier stages. The test computes a score based on the minimum difference between early vs. late and mid vs. late TXI values.

Value

A ConservationTestResult object with late conservation test results

See Also

`stat_generic_conservation_test`, `stat_early_conservation_test`

Examples

```
# Define developmental modules
modules <- list(early = 1:2, mid = 3:5, late = 6:7)
result <- stat_late_conservation_test(example_phyex_set_old, modules)
```

`stat_pairwise_test` *Pairwise Conservation Test*

Description

Test for significant differences in transcriptomic index values between two groups of developmental stages.

Usage

```
stat_pairwise_test(phyex_set, modules, alternative = c("greater", "less"), ...)
```

Arguments

`phyex_set`	A PhyloExpressionSet object
`modules`	A named list with elements 'contrast1' and 'contrast2' containing stage indices for each contrast group
`alternative`	Character string specifying the alternative hypothesis: "greater" (contrast1 > contrast2) or "less" (contrast1 < contrast2)
`...`	Additional arguments passed to stat_generic_conservation_test

Details

The pairwise test compares the mean transcriptomic index values between two groups of developmental stages. This is useful for testing specific hypotheses about differences in gene age composition between developmental periods.

Value

A ConservationTestResult object with pairwise test results

Author(s)

Jaruwatana Sodai Lotharukpong

See Also

`stat_generic_conservation_test`

Examples

```
# Define contrast groups
modules <- list(contrast1 = 1:3, contrast2 = 4:7)
result <- stat_pairwise_test(example_phyex_set, modules, alternative = "greater")
```

---

stat_reductive_hourglass_test

Reductive Hourglass Test

Description

Test for reductive hourglass patterns in transcriptomic data by comparing early and late developmental stages to mid developmental stages.

Usage

```
stat_reductive_hourglass_test(phyex_set, modules, ...)
```

Arguments

`phyex_set`	A PhyloExpressionSet object
`modules`	A named list with elements 'early', 'mid', and 'late' containing stage indices for each developmental module
`...`	Additional arguments passed to stat_generic_conservation_test

Details

The reductive hourglass test evaluates whether mid developmental stages show lower transcriptomic index values (indicating older genes) compared to both early and late stages. This creates an hourglass-shaped pattern where ancient genes dominate during mid-development. The test computes a score based on the minimum difference between early vs. mid and late vs. mid TXI values.

Value

A ConservationTestResult object with reductive hourglass test results

See Also

`stat_generic_conservation_test`, `stat_reverse_hourglass_test`

Examples

```
# Define developmental modules
modules <- list(early = 1:2, mid = 3:5, late = 6:7)
result <- stat_reductive_hourglass_test(example_phyex_set_old, modules=modules)
```

stat_reverse_hourglass_test

Reverse Hourglass Test

Description

Test for reverse hourglass patterns in transcriptomic data by comparing mid developmental stages to early and late developmental stages.

Usage

```
stat_reverse_hourglass_test(phyex_set, modules, ...)
```

Arguments

`phyex_set`	A PhyloExpressionSet object
`modules`	A named list with elements 'early', 'mid', and 'late' containing stage indices for each developmental module
`...`	Additional arguments passed to stat_generic_conservation_test

Details

The reverse hourglass test evaluates whether mid developmental stages show higher transcriptomic index values (indicating younger genes) compared to both early and late stages. This creates a reverse hourglass pattern where recently evolved genes dominate during mid-development. The test computes a score based on the minimum difference between mid vs. early and mid vs. late TXI values.

Value

A ConservationTestResult object with reverse hourglass test results

See Also

`stat_generic_conservation_test`, `stat_reductive_hourglass_test`

Examples

```
# Define developmental modules
modules <- list(early = 1:2, mid = 3:5, late = 6:7)
result <- stat_reverse_hourglass_test(example_phyex_set_old, modules=modules)
```

`strata_enrichment` *Calculate Phylostratum Enrichment*

Description

Calculate log2(observed/expected) enrichment ratios for phylostrata in a selected gene set compared to the background distribution.

Usage

```
strata_enrichment(strata, selected_gene_ids)
```

Arguments

`strata`	Named factor vector of phylostratum assignments (names are gene IDs)
`selected_gene_ids`	Character vector of gene IDs to test for enrichment

Details

This function calculates enrichment or depletion of phylostrata in a gene set by comparing the observed proportion of each stratum in the selected genes to the expected proportion based on the background distribution in all genes.

Positive values indicate enrichment (more genes than expected), while negative values indicate depletion (fewer genes than expected).

Value

A data frame with columns:

Stratum Phylostratum factor levels

log_obs_exp Log2 ratio of observed vs expected proportions

Examples

```
# Calculate enrichment for a gene set
enrichment <- strata_enrichment(example_phyex_set@strata, example_phyex_set@gene_ids[1:30])
print(enrichment)
```

sTXI *Calculate Stratum-Specific Transcriptomic Index*

Description

Calculate the stratum-specific transcriptomic index (sTXI) by summing pTXI values within each phylostratum.

Usage

```
sTXI(phyex_set, option = "identity")
```

Arguments

`phyex_set`	A PhyloExpressionSet object
`option`	Character string specifying calculation method: - "identity": Sum pTXI values within each stratum - "add": Cumulative sum across strata

Value

Matrix of sTXI values with strata as rows and identities as columns

Examples

```
# Calculate sTXI values
stxi_values <- sTXI(example_phyex_set, option = "identity")
stxi_cumsum <- sTXI(example_phyex_set, option = "add")
```

TAI *Calculate Transcriptomic Age Index (TAI)*

Description

Calculate the transcriptomic age index values for a PhyloExpressionSet. This function provides backward compatibility with the old TAI() function.

Usage

```
TAI(phyex_set)
```

Arguments

`phyex_set`	A PhyloExpressionSet object

Value

Numeric vector of TAI values for each identity

Examples

```
# Calculate TAI values
tai_values <- TAI(example_phyex_set)
```

---

taxid *Retrieve taxonomy categories from NCBI Taxonomy*

Description

This function retrieves category information from NCBI Taxonomy and is able to filter kingdom specific taxids.

Usage

```
taxid(db.path, download = FALSE, update = FALSE, filter = NULL)
```

Arguments

`db.path`	path to download and store the NCBI Taxonomy `categories.dmp` file. Default is the `tempdir()` directory.
`download`	a logical value specifying whether or not the `categories.dmp` shall be downloaded (`download = TRUE`) or whether a local version already exists on the users machine (`download = TRUE` - in this case please specify the `db.path` argument to target the local `categories.dmp` file).
`update`	should the local file be updated? Please specify the `db.path` argument to target the local `categories.dmp` file.
`filter`	a character string specifying the kingdom of life for which taxids shall be returned. Options are `"Archea"`, `"Bacteria"`, `"Eukaryota"`, `"Viruses"`, `"Unclassified"`.

Value

A tibble object containing taxonomy category information

Author(s)

Hajk-Georg Drost

Examples

```
# download categories.dmp file to current working directory
# and filter for 'Archea' taxids
Archea.taxids <- taxid(db.path = getwd(), filter = "Archea", download = TRUE)

# Once the NCBI Taxonomy 'categories.dmp' file is downloaded to your machine ('download = TRUE')
# the 'taxid()' function can be proceed on the local 'categories.dmp' file
# e.g. filter for Virus taxids
Virus.taxids <- taxid(db.path = getwd(), filter = "Viruses")
```

`TDI` *Calculate Transcriptomic Divergence Index (TDI)*

Description

Calculate the transcriptomic divergence index values for a PhyloExpressionSet. This function provides backward compatibility with the old TDI() function.

Usage

```
TDI(phyex_set)
```

Arguments

`phyex_set`	A PhyloExpressionSet object

Value

Numeric vector of TDI values for each identity

`TEI` *Calculate Transcriptomic Evolutionary Index (TEI)*

Description

Calculate the transcriptomic evolutionary index values for a PhyloExpressionSet. This function provides backward compatibility with the old TEI() function.

Usage

```
TEI(phyex_set)
```

Arguments

`phyex_set`	A PhyloExpressionSet object

Value

Numeric vector of TEI values for each identity

TestResult *Test Result S7 Class*

Description

S7 class for storing and manipulating statistical test results from phylotranscriptomic conservation tests.

Usage

```
TestResult(
  method_name = stop("@method_name is required"),
  test_stat = stop("@test_stat is required"),
  fitting_dist = stop("@fitting_dist is required"),
  params = stop("@params is required"),
  alternative = "two-sided",
  null_sample = stop("@null_sample is required"),
  data_name = character(0),
  p_label = "p_val"
)
```

Arguments

`method_name`	Character string identifying the test method
`test_stat`	Numeric test statistic value
`fitting_dist`	Distribution object used for null hypothesis testing
`params`	List of fitted distribution parameters
`alternative`	Character string specifying alternative hypothesis ("two-sided", "less", "greater")
`null_sample`	Numeric vector of null distribution samples
`data_name`	Character string naming the dataset (optional)
`p_label`	Character string for p-value label (default: "p_val")

Details

The TestResult class provides computed properties including: - ‘p_value‘: Computed p-value based on test statistic and fitted distribution

Value

A TestResult object

tf — *Short Alias for Transform Counts*

Description

Convenience alias for transform_counts function.

Usage

```
tf(phyex_set, FUN, FUN_name = deparse(substitute(FUN)), new_name = NULL, ...)
```

Arguments

`phyex_set`	A PhyloExpressionSet object
`FUN`	Function to apply
`FUN_name`	Name of the transformation function (optional)
`new_name`	Name for the new dataset (optional)
`...`	Additional arguments passed to FUN

Value

A PhyloExpressionSet object with transformed expression data

See Also

transform_counts

tf_PS — *Transform Phylostratum Values*

Description

This function performs transformation of phylostratum values.

Usage

```
tf_PS(phyex_set, transform = "qr")
```

Arguments

`phyex_set`	a `PhyloExpressionSet` S7 object.
`transform`	a character vector of any valid function that transforms PS values. Possible values can be: • `transform = "qr"` (or `"quantilerank"`) : quantile rank transformation analogous to Julia function `StatsBase.quantilerank` using `method = :tied`.

Details

This function transforms the phylostratum assignment. The return value of this function is a PhyloExpressionSet object with transformed phylostratum `tfPhylostratum` as the first column, satisfying `PhyloExpressionSetBase`. Note that the input `transform` must be an available function, currently limited to only `"qr"` (or `"quantilerank"`).

Value

a `PhyloExpressionSet` object storing transformed Phylostratum levels.

Author(s)

Jaruwatana Sodai Lotharukpong and Lukas Maischak

See Also

`tf`

Examples

```
# get the relative expression profiles for each phylostratum
tfPES <- tf_PS(example_phyex_set, transform = "qr")
```

`tf_stability` *Perform Permutation Tests Under Different Transformations*

Description

tf_stability statistically evaluates the stability of phylotranscriptomics permutation tests (e.g., `stat_flatline_test`, `stat_reductive_hourglass_test`, etc.) under different data transformations using a `PhyloExpressionSet`.

Usage

```
tf_stability(
  phyex_set,
  conservation_test = stat_flatline_test,
  transforms = COUNT_TRANSFORMS
)
```

Arguments

`phyex_set`	a `PhyloExpressionSet`.
`conservation_test`	a conservation test function (e.g. `stat_flatline_test`, `stat_reductive_hourglass_test`, etc.)
`transforms`	named list of transformation functions (default: `COUNT_TRANSFORMS`)

Details

Assesses the stability of data transforms on the permutation test of choice. See `tf`, `stat_flatline_test`, `stat_reductive_hourglass_test`, etc.

Value

Named numeric vector of p-values for each transformation.

Author(s)

Jaruwatana Sodai Lotharukpong

References

Lotharukpong JS et al. (2023) (unpublished)

`TI_map` *Transcriptomic Index Name Mapping*

Description

Named list mapping transcriptomic index abbreviations to full names.

Usage

```
TI_map
```

Format

A named list with 5 elements:

TXI Transcriptomic Index

TAI Transcriptomic Age Index

TDI Transcriptomic Divergence Index

TPI Transcriptomic Polymorphism Index

TEI Transcriptomic Evolutionary Index

TPI — *Calculate Transcriptomic Polymorphism Index (TPI)*

Description

Calculate the transcriptomic polymorphism index values for a PhyloExpressionSet. This function provides backward compatibility with the old TPI() function.

Usage

```
TPI(phyex_set)
```

Arguments

`phyex_set`	A PhyloExpressionSet object

Value

Numeric vector of TPI values for each identity

transform_counts — *Transform Expression Counts in PhyloExpressionSet*

Description

Apply a transformation function to the expression counts in a PhyloExpressionSet.

Usage

```
transform_counts(
  phyex_set,
  FUN,
  FUN_name = deparse(substitute(FUN)),
  new_name = NULL,
  ...
)
```

Arguments

`phyex_set`	A PhyloExpressionSet object
`FUN`	Function to apply
`FUN_name`	Name of the transformation function (optional)
`new_name`	Name for the new dataset (optional)
`...`	Additional arguments passed to FUN

Value

A PhyloExpressionSet object with transformed expression data

`TXI` *Calculate Transcriptomic Index (TXI)*

Description

Calculate the transcriptomic index values for a PhyloExpressionSet. This function provides backward compatibility with the old TXI() function.

Usage

```
TXI(phyex_set)
```

Arguments

`phyex_set`	A PhyloExpressionSet object

Value

Numeric vector of TXI values for each identity

Examples

```
# Calculate TXI values
txi_values <- TXI(example_phyex_set)
```

`TXI_conf_int` *Confidence Intervals for Transcriptomic Index (TXI)*

Description

Compute confidence intervals for the TXI using bootstrapped TXI values.

Usage

```
TXI_conf_int(phyex_set, probs = c(0.025, 0.975))
```

Arguments

`phyex_set`	A BulkPhyloExpressionSet object
`probs`	Numeric vector of probabilities for the confidence interval (default: c(0.025, 0.975))

Details

This function returns confidence intervals for the TXI for each identity (sample or group), based on the bootstrapped TXI values stored in the PhyloExpressionSet object.

Value

A tibble with first column Identity names, second column lower bound, third column upper bound

---

`TXI_std_dev` *Standard Deviation for TXI*

Description

Return a named vector of standard deviations for the TXI for each identity.

Usage

```
TXI_std_dev(phyex_set)
```

Arguments

`phyex_set`	A BulkPhyloExpressionSet object

Value

Named numeric vector of standard deviations, names are identities

Index