Package: myTAI (via r-universe)

October 8, 2024

Type Package

Title Evolutionary Transcriptomics

Version 2.0.0

Date 2024-10-02

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.

VignetteBuilder knitr

NeedsCompilation yes

License GPL-2

Depends R (>= 3.1.1)

Imports Rcpp, nortest (>= 1.0-2), fitdistrplus (>= 1.1-5), parallel (>= 3.1.1), foreach (>= 1.4.2), doParallel (>= 1.0.8), dplyr (>= 0.3.0), RColorBrewer (>= 1.1-2), methods (>= 3.1.1), graphics (>= 3.1.1), stats (>= 3.1.1), grDevices (>= 3.1.1), utils (>= 3.1.1), reshape2 (>= 1.4.1), ggplot2 (>= 1.0.1), readr (>= 0.2.2), tibble, scales, ggpubr, Matrix, DESeq2 (>= 1.29.15)

Suggests knitr (>= 1.6), rmarkdown (>= 0.3.3), devtools (>= 1.6.1), testthat (>= 0.9.1), mgcv, Seurat, decor, RcppThread

LinkingTo RcppArmadillo, Rcpp, cpp11, RcppThread, RcppEigen

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

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

RoxygenNote 7.3.2

Encoding UTF-8

SystemRequirements C++

Repository https://drostlab.r-universe.dev

RemoteUrl https://github.com/drostlab/mytai

RemoteRef HEAD

RemoteSha 04f2c7cde03f6edbc363816ed0d7de7713c020ea

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 ExpressionSet.

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

Usage

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

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet 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 or RE). 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, RE, REMatrix

Examples

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

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

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

# Example 3
# compute the variance profiles for each phylostratum
age.apply(PhyloExpressionSetExample, 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(PhyloExpressionSetExample, function(x) apply(x , 2 , range), as.list = TRUE)
```

`bootMatrix` *Compute a Permutation Matrix for Test Statistics*

Description

This function computes the TAI for a row permutated PhyloExpressionSet or DivergenceExpressionSet.

One can specify the number of permutations which corresponds to the number of `TAI` or `TDI` profiles that are being returned as data matrix. The function then returns a `TAI` or `TDI` matrix holding the `TAI` or `TDI` profiles of the permutated PhyloExpressionSets or DivergenceExpressionSets. This procedure can be used for building test statistics based on the `TAI` or `TDI` profiles.

Usage

```
bootMatrix(ExpressionSet, permutations = 1000)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.
`permutations`	a numeric value specifying the number of permutations to be performed.

Details

The sampled `TAI` or `TDI` matrix samples the phylostratum or divergence-stratum vector of a given PhyloExpressionSet or DivergenceExpressionSet and computes the corresponding `TAI` or `TDI` profiles of the randomly assigned phylostrata or divergence-strata. This sampling is then performed N times, yielding N randomly sampled `TAI` or `TDI` profiles. This random `TAI` or `TDI` profile matrix can then be used to perform statistical tests (such as the `FlatLineTest`, `ReductiveHourglassTest`, or `EarlyConservationTest`) based on the significance of `TAI` or `TDI` patterns.

Value

a numeric matrix representing N randomly permuted `TAI` or `TDI` profiles.

Author(s)

Hajk-Georg Drost

References

Quint M et al. (2012). *A transcriptomic hourglass in plant embryogenesis*. Nature (490): 98-101.

Drost HG et al. (2015) Mol Biol Evol. 32 (5): 1221-1231 doi:10.1093/molbev/msv012

See Also

`FlatLineTest`, `ReductiveHourglassTest`, `EarlyConservationTest`

Examples

```
# read standard phylotranscriptomics data
data(PhyloExpressionSetExample)
data(DivergenceExpressionSetExample)

# example PhyloExpressionSet using 100 permutations
randomTAI.Matrix <- bootMatrix(PhyloExpressionSetExample, permutations = 100)

# example DivergenceExpressionSet using 100 permutations
randomTDI.Matrix <- bootMatrix(DivergenceExpressionSetExample, permutations = 100)
```

`bootTEI` *Compute a Permutation Matrix of Transcriptome Evolutionary Index (TEI)*

Description

This function computes the transcriptome evolutionary index (TEI) using permuted strata values.

Usage

```
bootTEI(
  ExpressionSet,
  Phylostratum = NULL,
  permutations = 100,
  split = 1e+05,
  showprogress = TRUE,
  threads = 1
)
```

Arguments

`ExpressionSet`	expression object with rownames as GeneID (dgCMatrix) or standard PhyloExpressionSet object.
`Phylostratum`	a named vector representing phylostratum per GeneID with names as GeneID (not used if Expression is PhyloExpressionSet).
`permutations`	a numeric value specifying the number of permutations to be performed.
`split`	specify number of columns to split
`showprogress`	boolean if progressbar should be shown
`threads`	specify number of threads

Details

The strata values are sampled and the global `TEI` is calculated N times.

Value

a numeric matrix storing the TEI values based on permuted strata.

Author(s)

Kristian K Ullrich

References

Domazet-Loso T. and Tautz D. (2010). *A phylogenetically based transcriptome age index mirrors ontogenetic divergence patterns*. Nature (468): 815-818.

Quint M et al. (2012). *A transcriptomic hourglass in plant embryogenesis*. Nature (490): 98-101.

Drost HG et al. (2015) Mol Biol Evol. 32 (5): 1221-1231 doi:10.1093/molbev/msv012

Examples

```
# reading a standard PhyloExpressionSet
data(PhyloExpressionSetExample, package = "myTAI")

# computing partial TEI contribution per gene
bM <- bootTEI(PhyloExpressionSetExample)
```

`CollapseReplicates` *Combine Replicates in an ExpressionSet*

Description

This function takes an `ExpressionSet` object storing either a constant or variable number of biological or technical replicates per stage and collapses replicate expression levels using a defined `FUN` (window function).

Usage

```
CollapseReplicates(ExpressionSet, nrep, FUN, stage.names = NULL)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.
`nrep`	either a numeric value specifying the constant number of replicates per stage or a numeric vector specifying the variable number of replicates for each stage position.
`FUN`	a window function (e.g., `mean`, `median`, `max`, `min`, etc.) specifying how replicate expression levels should be collapsed.
`stage.names`	a character vector specifying the new names of collapsed stages.

Author(s)

Hajk-Georg Drost

Examples

```
data(PhyloExpressionSetExample)

# combine the expression levels of the 2 replicates (const) per stage
# using mean as window function and rename new stages: "S1","S2","S3"
CollapseReplicates(ExpressionSet = PhyloExpressionSetExample[1:5,1:8],
                   nrep          = 2,
                   FUN           = mean,
                   stage.names   = c("S1","S2","S3"))

# combine the expression levels of the 2 replicates (stage one), 2 replicates (stage two),
# and 3 replicates (stage three) using mean as window function
# and rename new stages: "S1","S2","S3"
CollapseReplicates(ExpressionSet = PhyloExpressionSetExample[1:5,1:9],
                   nrep          = c(2,2,3),
                   FUN           = mean,
                   stage.names   = c("S1","S2","S3"))
```

CombinatorialSignificance

Compute the Statistical Significance of Each Replicate Combination

---

Description

In case a PhyloExpressionSet or DivergenceExpressionSet stores replicates for each developmental stage or experiment, this function allows to compute the p-values quantifying the statistical significance of the underlying pattern for all combinations of replicates.

Usage

```
CombinatorialSignificance(
  ExpressionSet,
  replicates,
  TestStatistic = "FlatLineTest",
  permutations = 1000,
  parallel = FALSE
)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.

`replicates`	a numeric vector storing the number of replicates within each developmental stage or experiment. In case replicate stores only one value, then the function assumes that each developmental stage or experiment stores the same number of replicates.
`TestStatistic`	a string defining the type of test statistics to be used to quantify the statistical significance the present phylotranscriptomics pattern. Default is `TestStatistic = "FlatLineTest"`.
`permutations`	a numeric value specifying the number of permutations to be performed for the `FlatLineTest`.
`parallel`	a boolean value specifying whether parallel processing (multicore processing) shall be performed.

Details

The intention of this analysis is to validate that there exists no sequence of replicates (for all possible combination of replicates) that results in a non-significant pattern, when the initial pattern with combined replicates was shown to be significant.

A small Example:

Assume PhyloExpressionSet stores 3 developmental stages with 3 replicates measured for each stage. The 9 replicates in total are denoted as: 1.1, 1.2, 1.3, 2.1, 2.2, 2.3, 3.1, 3.2, 3.3. Now the function computes the statistical significance of each pattern derived by the corresponding combination of replicates, e.g.

- 1.1, 2.1, 3.1 -> p-value for combination 1
- 1.1, 2.2, 3.1 -> p-value for combination 2
- 1.1, 2.3, 3.1 -> p-value for combination 3
- 1.2, 2.1, 3.1 -> p-value for combination 4
- 1.2, 2.1, 3.1 -> p-value for combination 5
- 1.2, 2.1, 3.1 -> p-value for combination 6
- 1.3, 2.1, 3.1 -> p-value for combination 7
- 1.3, 2.2, 3.1 -> p-value for combination 8
- 1.3, 2.3, 3.1 -> p-value for combination 9

This procedure yields 27 p-values for the 3^3 $(n_stages^n_replicates)$ replicate combinations.

Note, that in case you have a large amount of stages/experiments and a large amount of replicates the computation time will increase by $n_stages^n_replicates$. For 11 stages and 4 replicates, 4^11 = 4194304 p-values have to be computed. Each p-value computation itself is based on a permutation test running with 1000 or more permutations. Be aware that this might take some time.

The p-value vector returned by this function can then be used to plot the p-values to see whether an critical value α is exeeded or not (e.g. $\alpha = 0.05$).

The function receives a standard PhyloExpressionSet or DivergenceExpressionSet object and a vector storing the number of replicates present in each stage or experiment. Based on these arguments the function computes all possible replicate combinations using the `expand.grid` function and performs a permutation test (either a `FlatLineTest` for each replicate combination. The *permutation* parameter of this function specifies the number of permutations that shall be performed for each permutation test. When all p-values are computed, a numeric vector storing the corresponding p-values for each replicate combination is returned.

In other words, for each replicate combination present in the PhyloExpressionSet or DivergenceExpressionSet object, the TAI or TDI pattern of the corresponding replicate combination is tested for its statistical significance based on the underlying test statistic.

This function is also able to perform all computations in parallel using multicore processing. The underlying statistical tests are written in C++ and optimized for fast computations.

Value

a numeric vector storing the p-values returned by the underlying test statistic for all possible replicate combinations.

Author(s)

Hajk-Georg Drost

References

Drost HG et al. (2015) Mol Biol Evol. 32 (5): 1221-1231 doi:10.1093/molbev/msv012

See Also

`expand.grid`, `FlatLineTest`

Examples

```
## Not run:
# load a standard PhyloExpressionSet
data(PhyloExpressionSetExample)

# we assume that the PhyloExpressionSetExample
# consists of 3 developmental stages
# and 2 replicates for stage 1, 3 replicates for stage 2,
# and 2 replicates for stage 3
# FOR REAL ANALYSES PLEASE USE: permutations = 1000 or 10000
# BUT NOTE THAT THIS TAKES MUCH MORE COMPUTATION TIME
p.vector <- CombinatorialSignificance(ExpressionSet = PhyloExpressionSetExample,
                                      replicates    = c(2,3,2),
                                      TestStatistic = "FlatLineTest",
                                      permutations  = 1000,
                                      parallel      = FALSE)

## End(Not run)
```

`DiffGenes` *Differential Gene Expression Analysis*

Description

Detect differentially expressed genes (DEGs) in a standard `ExpressionSet` object.

Usage

```
DiffGenes(
  ExpressionSet,
  nrep,
  method = "foldchange",
  p.adjust.method = NULL,
  comparison = NULL,
  alpha = NULL,
  filter.method = NULL,
  n = NULL,
  stage.names = NULL
)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.
`nrep`	either a numeric value specifying the constant number of replicates per stage or a numeric vector specifying the variable number of replicates for each stage position.
`method`	method to detect differentially expressed genes.
`p.adjust.method`	p value correction method that is passed to `p.adjust`. Available options are: • `p.adjust.method = "BH"` (Benjamini-Hochberg correction) • `p.adjust.method = "bonferroni"` (Bonferroni correction) • `p.adjust.method = "holm"` • `p.adjust.method = "hochberg"` • `p.adjust.method = "hommel"` • `p.adjust.method = "BY"` • `p.adjust.method = "fdr"` If `p.adjust.method = NULL` (Default) then no p-value correction is performed.
`comparison`	a character string specifying whether genes having fold-change or p-values below, above, or below AND above (both) the `alpha` value should be excluded from the dataset. In case `comparison = "both"` is chosen, the `cut.off` argument must be a two dimensional vector defining the lower `alpha` value at the first position and the upper `alpha` value at the second position.

`alpha`	a numeric value specifying the cut-off value above which Genes fulfilling the corresponding fold-change, log-fold-change, or p-value should be retained and returned by `DiffGenes`.
`filter.method`	a method how to `alpha` values in multiple stages. Options are `"const"`, `"min-set"`, and `"n-set"`.
`n`	a numeric value for `method = "n-set"`.
`stage.names`	a character vector specifying the new names of collapsed stages.

Details

All methods to perform detection of differentially expressed genes assume that your input dataset has been normalized before passing it to *DiffGenes*. For RNA-Seq data *DiffGenes* assumes that the libraries have been normalized to have the same size, i.e., to have the same expected column sum under the null hypothesis.

Available methods for the detection of differentially expressed genes:

- `method = "foldchange"`: ratio of replicate geometric means between developmental stages. Here, the *DiffGenes* functions assumes that absolute expression levels are stored in your input `ExpresisonSet`.
- `method = "log-foldchange"`: difference of replicate arithmetic means between developmental stages. Here, the *DiffGenes* functions assumes that *log2a* transformed expression levels are stored in your input `ExpresisonSet`.
- `method = "t.test"`: Welch t.test between replicate expression levels of two samples.
- `method = "wilcox.test"`: Wilcoxon Rank Sum Test between replicate expression levels of two samples.

Exclude non differentially expressed genes from the result dataset:

When specifying the `alpha` argument you furthermore, need to specify the `filter.method` to decide how non differentially expressed genes should be classified in multiple sample comparisons and which genes should be retained in the final dataset returned by `DiffGenes`. In other words, all genes < `alpha` based on the following `filter.method` are removed from the result dataset.

Following extraction criteria are implemented in this function:

- `const`: all genes that have at least one sample comparison that undercuts or exceeds the `alpha` value `cut.off` will be excluded from the `ExpressionSet`. Hence, for a 7 stage `ExpressionSet` genes passing the `alpha` threshold in 6 stages will be retained in the `ExpressionSet`.
- `min-set`: genes passing the `alpha` value in `ceiling(n/2)` stages will be retained in the `ExpressionSet`, where *n* is the number of stages in the `ExpressionSet`.
- `n-set`: genes passing the `alpha` value in `n` stages will be retained in the `ExpressionSet`. Here, the argument `n` needs to be specified.

Note

In case input `ExpressionSet` objects store 0 values, internally all expression levels are shifted by `+1` to allow sufficient fold-change and p-value computations. Additionally, a warning is printed to the console in case expression levels have been automatically shifted.

Author(s)

Hajk-Georg Drost

See Also

Expressed

Examples

```
data(PhyloExpressionSetExample)

# Detection of DEGs using the fold-change measure
DEGs <- DiffGenes(ExpressionSet = PhyloExpressionSetExample[ ,1:8],
                  nrep          = 2,
                  comparison    = "below",
                  method        = "foldchange",
                  stage.names   = c("S1","S2","S3"))


head(DEGs)


# Detection of DEGs using the log-fold-change measure
# when choosing method = "log-foldchange" it is assumed that
# your input expression matrix stores log2 expression levels
log.DEGs <- DiffGenes(ExpressionSet = tf(PhyloExpressionSetExample[1:5,1:8],log2),
                      nrep          = 2,
                      comparison    = "below",
                      method        = "log-foldchange",
                      stage.names   = c("S1","S2","S3"))


head(log.DEGs)


# Remove fold-change values < 2 from the dataset:

## first have a look at the range of fold-change values of all genes
apply(DEGs[ , 3:8],2,range)

# now remove genes undercutting the alpha = 2 threshold
# hence, remove genes having p-values <= 0.05 in at
# least one sample comparison
DEGs.alpha <- DiffGenes(ExpressionSet = PhyloExpressionSetExample[1:250 ,1:8],
                        nrep          = 2,
                        method        = "t.test",
                        alpha         = 0.05,
                        comparison    = "above",
                        filter.method = "n-set",
                        n             = 1,
                        stage.names   = c("S1","S2","S3"))
```

```
# now again have a look at the range and find
# that fold-change values of 2 are the min value
apply(DEGs.alpha[ , 3:5],2,range)

# now check whether each example has at least one stage with a p-value <= 0.05
head(DEGs.alpha)
```

`DivergenceExpressionSetExample`

An Example DivergenceExpressionSet Data Set

Description

A standard DivergenceExpressionSet is a `data.frame` consisting of a standardized sequence of columns to store the age information for each gene and its corresponding gene expression profile.

The standard is defined as follows:

Divergencestratum | GeneID | Expression-level 1 | ... | Expression-level N

Details

This example DivergenceExpressionSet dataset covers 7 developmental stages of Arabidopsis thaliana embryo development. The initial gene expression dataset was published by Xiang et al., 2011 (see references section) and was then used by Quint et al., 2012 (see references section) to assign sequence divergence values (divergence strata) to each gene expression profile.

Value

a standard DivergenceExpressionSet object.

Author(s)

Hajk-Georg Drost

Source

http://www.plantphysiol.org/content/156/1/346/suppl/DC1

References

Quint M et al. 2012. "A transcriptomic hourglass in plant embryogenesis". Nature (490): 98-101. Supplementary Table 2 : http://www.nature.com/nature/journal/v490/n7418/full/nature11394.html

Xiang D et al. 2011. "Genome-Wide Analysis Reveals Gene Expression and Metabolic Network Dynamics during Embryo Development in Arabidopsis". Plant Physiology (156): 346-356. Supplemental Table 1 : http://www.plantphysiol.org/content/156/1/346/suppl/DC1

See Also

`PhyloExpressionSetExample`

`EarlyConservationTest` *Perform Reductive Early Conservation Test*

Description

The *Reductive Early Conservation Test* aims to statistically evaluate the existence of a monotonically increasing phylotranscriptomic pattern based on `TAI` or `TDI` computations. The corresponding p-value quantifies the probability that a given TAI or TDI pattern (or any phylotranscriptomics pattern) does not follow an early conservation like pattern. A p-value < 0.05 indicates that the corresponding phylotranscriptomics pattern does indeed follow an early conservation (low-high-high) shape.

Usage

```
EarlyConservationTest(
  ExpressionSet,
  modules = NULL,
  permutations = 1000,
  lillie.test = FALSE,
  plotHistogram = FALSE,
  runs = 10,
  parallel = FALSE,
  gof.warning = FALSE,
  custom.perm.matrix = NULL
)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.
`modules`	a list storing three elements: early, mid, and late. Each element expects a numeric vector specifying the developmental stages or experiments that correspond to each module. For example, `module` = list(early = 1:2, mid = 3:5, late = 6:7) devides a dataset storing seven developmental stages into 3 modules.
`permutations`	a numeric value specifying the number of permutations to be performed for the `ReductiveHourglassTest`.
`lillie.test`	a boolean value specifying whether the Lilliefors Kolmogorov-Smirnov Test shall be performed to quantify the goodness of fit.
`plotHistogram`	a boolean value specifying whether a *Lillifor's Kolmogorov-Smirnov-Test* shall be performed to test the goodness of fit of the approximated distribution, as well as additional plots quantifying the significance of the observed phylotranscriptomic pattern.
`runs`	specify the number of runs to be performed for goodness of fit computations, in case `plotHistogram` = `TRUE`. In most cases `runs` = 100 is a reasonable choice. Default is `runs` = 10 (because it takes less computation time for demonstration purposes).

`parallel`	performing `runs` in parallel (takes all cores of your multicore machine).
`gof.warning`	a logical value indicating whether non significant goodness of fit results should be printed as warning. Default is `gof.warning = FALSE`.
`custom.perm.matrix`	a custom `bootMatrix` (permutation matrix) to perform the underlying test statistic. Default is `custom.perm.matrix = NULL`.

Details

The *reductive early conservation test* is a permutation test based on the following test statistic.

(1) A set of developmental stages is partitioned into three modules - early, mid, and late - based on prior biological knowledge.

(2) The mean `TAI` or `TDI` value for each of the three modules T_early, T_mid, and T_late are computed.

(3) The two differences D1 = T_mid - T_early and D2 = T_late - T_early are calculated.

(4) The minimum D_min of D1 and D2 is computed as final test statistic of the reductive hourglass test.

In order to determine the statistical significance of an observed minimum difference D_min the following permutation test was performed. Based on the `bootMatrix` D_min is calculated from each of the permuted `TAI` or `TDI` profiles, approximated by a Gaussian distribution with method of moments estimated parameters returned by `fitdist`, and the corresponding p-value is computed by `pnorm` given the estimated parameters of the Gaussian distribution. The *goodness of fit* for the random vector *D_min* is statistically quantified by an Lilliefors (Kolmogorov-Smirnov) test for normality.

In case the parameter *plotHistogram = TRUE*, a multi-plot is generated showing:

(1) A Cullen and Frey skewness-kurtosis plot generated by `descdist`. This plot illustrates which distributions seem plausible to fit the resulting permutation vector D_min. In the case of the *reductive early conservation test* a normal distribution seemed plausible.

(2) A histogram of D_min combined with the density plot is plotted. D_min is then fitted by a normal distribution. The corresponding parameters are estimated by *moment matching estimation* using the `fitdist` function.

(3) A plot showing the p-values for N independent runs to verify that a specific p-value is biased by a specific permutation order.

(4) A barplot showing the number of cases in which the underlying goodness of fit (returned by Lilliefors (Kolmogorov-Smirnov) test for normality) has shown to be significant (`TRUE`) or not significant (`FALSE`). This allows to quantify the permutation bias and their implications on the goodness of fit.

Value

a list object containing the list elements:

`p.value` : the p-value quantifying the statistical significance (low-high-high pattern) of the given phylotranscriptomics pattern.

`std.dev` : the standard deviation of the N sampled phylotranscriptomics patterns for each developmental stage S.

`lillie.test` : a boolean value specifying whether the *Lillifors KS-Test* returned a p-value > 0.05, which indicates that fitting the permuted scores with a normal distribution seems plausible.

Author(s)

Hajk-Georg Drost

References

Drost HG et al. (2015) Mol Biol Evol. 32 (5): 1221-1231 doi:10.1093/molbev/msv012

Quint M et al. (2012). *A transcriptomic hourglass in plant embryogenesis*. Nature (490): 98-101.

Piasecka B, Lichocki P, Moretti S, et al. (2013) *The hourglass and the early conservation models co-existing patterns of developmental constraints in vertebrates*. PLoS Genet. 9(4): e1003476.

See Also

`ecScore`, `bootMatrix`, `FlatLineTest`,`ReductiveHourglassTest`, `ReverseHourglassTest`, `PlotSignature`, `LateConservationTest`

Examples

```
data(PhyloExpressionSetExample)

# perform the early conservation test for a PhyloExpressionSet
# here the prior biological knowledge is that stages 1-2 correspond to module 1 = early,
# stages 3-5 to module 2 = mid (phylotypic module), and stages 6-7 correspond to
# module 3 = late
EarlyConservationTest(PhyloExpressionSetExample,
                       modules = list(early = 1:2, mid = 3:5, late = 6:7),
                       permutations = 1000)


# use your own permutation matrix based on which p-values (EarlyConservationTest)
# shall be computed
custom_perm_matrix <- bootMatrix(PhyloExpressionSetExample,100)

EarlyConservationTest(PhyloExpressionSetExample,
                       modules = list(early = 1:2, mid = 3:5, late = 6:7),
                       custom.perm.matrix = custom_perm_matrix)
```

`ecScore` *Compute the Hourglass Score for the EarlyConservationTest*

Description

This function computes the EarlyConservationTest score for a given TAI or TDI pattern.

The reductive early conservation test is a permutation test based on the following test statistic.

- A set of developmental stages is partitioned into three modules - early, mid, and late - based on prior biological knowledge.

- The mean TAI or TDI value for each of the three modules T_early, T_mid, and T_late are computed.

- The two differences D1 = T_mid - T_early and D2 = T_late - T_early are calculated.

- The minimum D_min of D1 and D2 is computed as final test statistic of the reductive early conservation test.

This function *ecScore* computes the *D_min* value for a given TAI or TDI stored in the `age_vals` argument.

Usage

```
ecScore(age_vals, early, mid, late, profile.warn = FALSE)
```

Arguments

`age_vals`	a numeric vector containing TAI or TDI values for each developmental stage s.
`early`	a numeric vector storing the numeric stage values that correspond to the early phase of development.
`mid`	a numeric vector storing the numeric stage values that correspond to the middle phase of development.
`late`	a numeric vector storing the numeric stage values that correspond to the late phase of development.
`profile.warn`	a boolean value indicating whether a warning is printed when a low-mid-high pattern isn't followed.

Value

a numeric value representing the early conservation score.

Author(s)

Hajk-Georg Drost

See Also

EarlyConservationTest, TAI, TDI

Examples

```
# read standard phylotranscriptomics data
data(PhyloExpressionSetExample)
data(DivergenceExpressionSetExample)

# Example PhyloExpressionSet:

# compute the TAI profile
TAIs <- TAI(PhyloExpressionSetExample)

# compute the early conservation score for the TAI profile
ec_score <- ecScore(age_vals = TAIs,early = 1:2,mid = 3:5,late = 6:7)


# Example DivergenceExpressionSet:

# compute the TDI profile
TDIs <- TDI(DivergenceExpressionSetExample)

# compute the early conservation score for the TDI profile
ec_score <- ecScore(age_vals = TDIs,early = 1:2,mid = 3:5,late = 6:7)

# compute ecScore() vector from bootMatrix()
apply(bootMatrix(PhyloExpressionSetExample,10),1,ecScore,early = 1:2,mid = 3:5,late = 6:7)

# get warning if the expected pattern isn't followed
ec_score <- ecScore(age_vals = TAIs,early = 1:2,mid = 3:5,late = 6:7,profile.warn=TRUE)
```

`EnrichmentTest` *Phylostratum or Divergence Stratum Enrichment of a given Gene Set based on Fisher's Test*

Description

This function computes the significance of enriched (over or underrepresented) Phylostrata or Divergence Strata within an input `test.set` based on the `fisher.test`. Please concult `PlotEnrichment` for details.

Usage

```
EnrichmentTest(
  ExpressionSet,
  test.set,
  use.only.map = FALSE,
  measure = "log-foldchange",
  complete.bg = TRUE,
  epsilon = 1e-05
)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.
`test.set`	a character vector storing the gene ids for which PS/DS enrichment analyses should be performed.
`use.only.map`	a logical value indicating whether instead of a standard `ExpressionSet` only a `Phylostratigraphic Map` or `Divergene Map` is passed to this function.
`measure`	a character string specifying the measure that should be used to quantify over and under representation of PS/DS. Measures can either be `measure = "foldchange"` (odds) or `measure = "log-foldchange"` (log-odds).
`complete.bg`	a logical value indicating whether the entire background set of the input ExpressionSet should be considered when performing Fisher's exact test (`complete.bg = TRUE`) or whether genes that are stored in test.set should be excluded from the background set before performing Fisher's exact test (`complete.bg = FALSE`).
`epsilon`	a small value to shift values by epsilon to avoid log(0) computations.

Author(s)

Hajk-Georg Drost

See Also

`PlotEnrichment`, `fisher.test`

Examples

```
data(PhyloExpressionSetExample)

set.seed(123)
test_set <- sample(PhyloExpressionSetExample[ , 2],1000)

E.Result <- EnrichmentTest(ExpressionSet = PhyloExpressionSetExample,
                           test.set      = test_set ,
                           measure       = "log-foldchange")

# get the log-fold change table
E.Result$enrichment.matrix

# get P-values for the enrichment significance for each Phylostratum
E.Result$p.values
```

`Expressed` *Filter for Expressed Genes*

Description

This function takes an ExpressionSet object and removes genes from the gene expression matrix that have an expression level below, above, or below AND above a defined `cut.off` value. Hence, this function allows to remove genes that have been defined as *not expressed* or *outliers* and returns an `ExpressionSet` retaining only expressed genes.

Usage

```
Expressed(
  ExpressionSet,
  cut.off,
  method = "const",
  comparison = "below",
  n = NULL
)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.
`cut.off`	a numeric value specifying the expression cut-off to define genes as *not expressed* (`comparison = "below"`) , *outliers* (`comparison = "above"`), or both (`comparison = "both"`). See `comparison` for details. In case `comparison = "both"`, the `cut.off` argument must be a two dimensional vector defining the lower `cut.off` value at the first position and the upper `cut.off` value at the second position.
`method`	a method defining how to treat gene expression values in multiple stages. The corresponding method that is chosen allows to control the stage-wise fulfillment of the threshold criteria. Options are `"const"`, `"min-set"`, and `"n-set"`.
`comparison`	a character string specifying whether genes having expression levels below, above, or below AND above (both) the `cut.off` value should be excluded from the dataset. In case `comparison = "both"` is chosen, the `cut.off` argument must be a two dimensional vector defining the lower `cut.off` value at the first position and the upper `cut.off` value at the second position.
`n`	a numeric value for `method = "n-set"`.

Details

This filter function allows users to remove genes from the `ExpressionSet` object that undercut or exceed a certain expression level `cut.off`.

Following extraction criteria are implemented in this function:

- `const`: all genes that have at least one stage that undercuts or exceeds the expression `cut.off` will be excluded from the `ExpressionSet`. Hence, for a 7 stage `ExpressionSet` genes passing the expression level `cut.off` in 6 stages will be retained in the `ExpressionSet`.
- `min-set`: genes passing the expression level `cut.off` in `ceiling(n/2)` stages will be retained in the `ExpressionSet`, where *n* is the number of stages in the `ExpressionSet`.
- `n-set`: genes passing the expression level `cut.off` in `n` stages will be retained in the `ExpressionSet`. Here, the argument `n` needs to be specified.

Author(s)

Hajk-Georg Drost

Examples

```
data(PhyloExpressionSetExample)

# remove genes that have an expression level below 8000
# in at least one developmental stage
FilterConst <- Expressed(ExpressionSet = PhyloExpressionSetExample,
                         cut.off       = 8000,
                         method        = "const",
                         comparison    = "below")

dim(FilterConst) # check number of retained genes

# remove genes that have an expression level below 8000
# in at least 3 developmental stages
# (in this case: ceiling(7/2) = 4 stages fulfilling the cut-off criteria)
FilterMinSet <- Expressed(ExpressionSet = PhyloExpressionSetExample,
                          cut.off       = 8000,
                          method        = "min-set",
                          comparison    = "below")

dim(FilterMinSet) # check number of retained genes

# remove genes that have an expression level below 8000
# in at least 5 developmental stages (in this case: n = 2 stages fulfilling the criteria)
FilterNSet <- Expressed(ExpressionSet = PhyloExpressionSetExample,
                        cut.off       = 8000,
                        method        = "n-set",
                        comparison    = "below",
                        n             = 2)

dim(FilterMinSet) # check number of retained genes



# remove expression levels that exceed the cut.off criteria
FilterMinSet <- Expressed(ExpressionSet = PhyloExpressionSetExample,
                          cut.off       = 12000,
                          method        = "min-set",
                          comparison    = "above")
```

```
dim(FilterMinSet) # check number of retained genes


# remove expression levels that undercut AND exceed the cut.off criteria
FilterMinSet <- Expressed(ExpressionSet = PhyloExpressionSetExample,
                          cut.off       = c(8000,12000),
                          method        = "min-set",
                          comparison    = "both")

dim(FilterMinSet) # check number of retained genes
```

`FlatLineTest` *Perform Flat Line Test*

Description

This function quantifies the statistical significance of an observed phylotranscriptomic pattern. In detail, the *Flat Line Test* quantifies any significant deviation of an observed phylotranscriptomic pattern from a flat line.

Usage

```
FlatLineTest(
  ExpressionSet,
  permutations = 10000,
  plotHistogram = FALSE,
  runs = 10,
  parallel = FALSE,
  custom.perm.matrix = NULL
)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.
`permutations`	a numeric value specifying the number of permutations that shall be performed for the *FlatLineTest*.
`plotHistogram`	a logical value indicating whether a detailed statistical analysis concerning the goodness of fit should be performed.
`runs`	specify the number of runs to be performed for goodness of fit computations. In most cases runs = 100 is a reasonable choice.
`parallel`	performing `runs` in parallel (takes all cores of your multicore machine).
`custom.perm.matrix`	a custom `bootMatrix` (permutation matrix) to perform the underlying test statistic. Default is `custom.perm.matrix = NULL`.

Details

Internally the function performs N phylotranscriptomics pattern computations (`TAI` or `TDI`) based on sampled PhyloExpressionSets or DivergenceExpressionSets (see `bootMatrix`). The test statistics is being developed as follows:

The variance *V_pattern* of the S phylotranscriptomics values defines the test statistic for the `FlatLineTest`. The basic assumption is, that the variance of a flat line should be equivalent to zero for a perfect flat line. Any deviation from a flat line can be measured with a variance value > 0.

To determine the null distribution of *V_p*, all PS or DS values within each developmental stage s are randomly permuted, S surrogate phylotranscriptomics values are computed from this permuted dataset, and a surrogate value of *V_p* is computed from these S phylotranscriptomics values. This permutation process is repeated N times, yielding a histogram of *V_p*.

After applying a *Lilliefors Kolmogorov-Smirnov Test for gamma distribution*, *V_p* is approximated by a *gamma distribution*. The two parameters of the *gamma distribution* are estimated by the function `fitdist` from the **fitdistrplus** package by *moment matching estimation*. The fitted *gamma distribution* is considered the null distribution of *V_pattern*, and the p-value of the observed value of *V_p* is computed from this null distribution.

In case the parameter *plotHistogram* = *TRUE*, a multi-plot is generated showing:

(1) A Cullen and Frey skewness-kurtosis plot generated by `descdist`).

(2) A histogram of V_p combined with the density plot using the Method of Moments estimated parameters returned by the `fitdist` function using a gamma distribution.

(3) A plot showing the p-values for N independent runs to verify that a specific p-value is biased by a specific permutation order.

The *goodness of fit* for the random vector *V_p* is quantified statistically by an adapted Lilliefors (Kolmogorov-Smirnov) test for gamma distributions.

Value

a list object containing the list elements:

- `p.value` the p-value quantifying the statistical significance (deviation from a flat line) of the given phylotranscriptomics pattern.
- `std.dev` the standard deviation of the N sampled phylotranscriptomics patterns for each developmental stage S.
- `std.dev` the Kolmogorov-Smirnov test satistics for fitting a gamma distribution to the variances of the dataset with permuted phylostrata.

Note

In case there are extreme outlier expression values stored in the dataset (PhyloExpressionSet or DivergenceExpressionSet), the internal `fitdist` function that is based on the `bootMatrix` output might return a warning: "In densfun(x, parm[1], parm[2], ...) : NaNs were produced" which indicates that permutation results caused by extreme outlier expression values that could not be fitted accordingly. This warning will not be printed out when the corresponding outlier values are extracted from the dataset.

Author(s)

Hajk-Georg Drost

References

Drost HG et al. (2015) Mol Biol Evol. 32 (5): 1221-1231 doi:10.1093/molbev/msv012

Quint M et al. (2012). A transcriptomic hourglass in plant embryogenesis. Nature (490): 98-101.

M. L. Delignette-Muller, R. Pouillot, J.-B. Denis and C. Dutang (2014), fitdistrplus: help to fit of a parametric distribution to non-censored or censored data.

Cullen AC and Frey HC (1999) Probabilistic techniques in exposure assessment. Plenum Press, USA, pp. 81-159.

Evans M, Hastings N and Peacock B (2000) Statistical distributions. John Wiley and Sons Inc.

Sokal RR and Rohlf FJ (1995) Biometry. W.H. Freeman and Company, USA, pp. 111-115.

Juergen Gross and bug fixes by Uwe Ligges (2012). nortest: Tests for Normality. R package version 1.0-2.

http://CRAN.R-project.org/package=nortest

Dallal, G.E. and Wilkinson, L. (1986): An analytic approximation to the distribution of Lilliefors test for normality. The American Statistician, 40, 294-296.

Stephens, M.A. (1974): EDF statistics for goodness of fit and some comparisons. Journal of the American Statistical Association, 69, 730-737.

http://stackoverflow.com/questions/4290081/fitting-data-to-distributions?rq=1

http://stats.stackexchange.com/questions/45033/can-i-use-kolmogorov-smirnov-test-and-estimate-distribution-parameters

http://cran.r-project.org/doc/contrib/Ricci-distributions-en.pdf

http://cran.r-project.org/doc/contrib/Ricci-distributions-en.pdf

See Also

`TAI`, `TDI`, `PlotSignature`, `bootMatrix`

Examples

```
# read standard phylotranscriptomics data
data(PhyloExpressionSetExample)

# example PhyloExpressionSet using 100 permutations
FlatLineTest(PhyloExpressionSetExample,
             permutations  = 100,
             plotHistogram = FALSE)

# use your own permutation matrix based on which p-values (FlatLineTest)
# shall be computed
custom_perm_matrix <- bootMatrix(PhyloExpressionSetExample,100)

FlatLineTest(PhyloExpressionSetExample,
                custom.perm.matrix = custom_perm_matrix)
```

geom.mean *Geometric Mean*

Description

This function computes the geometric mean of a numeric input vector `x`.

Usage

```
geom.mean(x)
```

Arguments

`x`	a numeric vector for which geometric mean computations shall be performed.

Author(s)

Hajk-Georg Drost

Examples

```
x <- 1:10

geom.mean(x)
```

GroupDiffs *Quantify the significant differences between gene expression distributions of age groups*

Description

This function performs a test to quantify the statistical significance between the global expression level distributions of groups of PS or DS. It therefore, allows users to investigate significant groups of PS or DS that significantly differ in their gene expression level distibution within specific developmental stages or experiments.

Usage

```
GroupDiffs(
  ExpressionSet,
  Groups = NULL,
  legendName = NULL,
  stat.test = "wilcox.test",
  gene.set = NULL,
  ...
)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.
`Groups`	a list containing the phylostrata or divergence strata that correspond to the same phylostratum class or divergence class. For ex. evolutionary old phylostrata: PS1-3 (Class 1) and evolutionary young phylostrata: PS4-12 (Class 2). In this case, the list could be assigned as, `Groups` = list(c(1:3), c(4:12)).
`legendName`	a character string specifying whether "PS" or "DS" are used to compute relative expression profiles.
`stat.test`	the statistical test to quantify PS or DS group differences.
`gene.set`	a character vector storing the gene ids for which group specific differences shall be statistically quantified.
`...`	additional plot parameters.

Details

The purpose of this function is to detect groups of PS or DS that significantly differ in their gene expression level distributions on a global (transcriptome) level. Since relative expression levels (`PlotRE`) or PS or DS specific mean expression levels (`PlotMeans`) are biased by highly expressed genes, this function allows users to objectively test the significant difference of transcriptome expression between groups of PS or DS in a specific developmental stage or experiment.

Author(s)

Hajk-Georg Drost

See Also

`PlotGroupDiffs`, `PlotMeans`, `PlotRE`, `PlotBarRE`, `PlotCategoryExpr`

Examples

```
data(PhyloExpressionSetExample)
# perform a Wilcoxon Rank Sum test to statistically quantify the
# difference between PS-Group 1 expression levels versus PS-Group 2
# expression levels
GroupDiffs(ExpressionSet = PhyloExpressionSetExample,
           Groups       = list(group_1 = 1:3,group_2 = 4:12),
            legendName   = "PS")

# quantify the significant difference of a selected set of genes
set.seed(123)
ExampleGeneSet <- sample(PhyloExpressionSetExample[ , 2],5000)

GroupDiffs(ExpressionSet = PhyloExpressionSetExample,
           Groups       = list(group_1 = 1:3,group_2 = 4:12),
           legendName   = "PS",
           gene.set     = ExampleGeneSet)
```

harm.mean *Harmonic Mean*

Description

This function computes the harmonic mean of a numeric input vector `x`.

Usage

```
harm.mean(x)
```

Arguments

x	a numeric vector for which harmonic mean computations shall be performed.

Author(s)

Hajk-Georg Drost

Examples

```
x <- 1:10

harm.mean(x)
```

`LateConservationTest` *Perform Reductive Late Conservation Test*

Description

The *Reductive Late Conservation Test* aims to statistically evaluate the existence of a monotonically decreasing phylotranscriptomic pattern based on `TAI` or `TDI` computations. The corresponding p-value quantifies the probability that a given TAI or TDI pattern (or any phylotranscriptomics pattern) does not follow an late conservation like pattern. A p-value < 0.05 indicates that the corresponding phylotranscriptomics pattern does indeed follow an late conservation (high-high-low) shape.

Usage

```
LateConservationTest(
  ExpressionSet,
  modules = NULL,
  permutations = 1000,
  lillie.test = FALSE,
  plotHistogram = FALSE,
  runs = 10,
  parallel = FALSE,
  gof.warning = FALSE,
  custom.perm.matrix = NULL
)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.
`modules`	a list storing three elements: early, mid, and late. Each element expects a numeric vector specifying the developmental stages or experiments that correspond to each module. For example, `module` = list(early = 1:2, mid = 3:5, late = 6:7) divides a dataset storing seven developmental stages into 3 modules.
`permutations`	a numeric value specifying the number of permutations to be performed for the `ReductiveHourglassTest`.
`lillie.test`	a boolean value specifying whether the Lilliefors Kolmogorov-Smirnov Test shall be performed to quantify the goodness of fit.
`plotHistogram`	a boolean value specifying whether a *Lillifor's Kolmogorov-Smirnov-Test* shall be performed to test the goodness of fit of the approximated distribution, as well as additional plots quantifying the significance of the observed phylotranscriptomic pattern.
`runs`	specify the number of runs to be performed for goodness of fit computations, in case `plotHistogram` = `TRUE`. In most cases `runs` = 100 is a reasonable choice. Default is `runs` = 10 (because it takes less computation time for demonstration purposes).
`parallel`	performing `runs` in parallel (takes all cores of your multicore machine).

`gof.warning`	a logical value indicating whether non significant goodness of fit results should be printed as warning. Default is `gof.warning = FALSE`.
`custom.perm.matrix`	a custom `bootMatrix` (permutation matrix) to perform the underlying test statistic. Default is `custom.perm.matrix = NULL`.

Details

The *reductive late conservation test* is a permutation test based on the following test statistic.

(1) A set of developmental stages is partitioned into three modules - early, mid, and late - based on prior biological knowledge.

(2) The mean `TAI` or `TDI` value for each of the three modules T_early, T_mid, and T_late are computed.

(3) The two differences D1 = T_early - T_late and D2 = T_mid - T_late are calculated.

(4) The minimum D_min of D1 and D2 is computed as final test statistic of the reductive hourglass test.

In order to determine the statistical significance of an observed minimum difference D_min the following permutation test was performed. Based on the `bootMatrix` D_min is calculated from each of the permuted `TAI` or `TDI` profiles, approximated by a Gaussian distribution with method of moments estimated parameters returned by `fitdist`, and the corresponding p-value is computed by `pnorm` given the estimated parameters of the Gaussian distribution. The *goodness of fit* for the random vector *D_min* is statistically quantified by an Lilliefors (Kolmogorov-Smirnov) test for normality.

In case the parameter *plotHistogram = TRUE*, a multi-plot is generated showing:

(1) A Cullen and Frey skewness-kurtosis plot generated by `descdist`. This plot illustrates which distributions seem plausible to fit the resulting permutation vector D_min. In the case of the *reductive late conservation test* a normal distribution seemed plausible.

(2) A histogram of D_min combined with the density plot is plotted. D_min is then fitted by a normal distribution. The corresponding parameters are estimated by *moment matching estimation* using the `fitdist` function.

(3) A plot showing the p-values for N independent runs to verify that a specific p-value is biased by a specific permutation order.

(4) A barplot showing the number of cases in which the underlying goodness of fit (returned by Lilliefors (Kolmogorov-Smirnov) test for normality) has shown to be significant (`TRUE`) or not significant (`FALSE`). This allows to quantify the permutation bias and their implications on the goodness of fit.

Value

a list object containing the list elements:

`p.value` : the p-value quantifying the statistical significance (low-high-high pattern) of the given phylotranscriptomics pattern.

`std.dev` : the standard deviation of the N sampled phylotranscriptomics patterns for each developmental stage S.

`lillie.test` : a boolean value specifying whether the *Lillifors KS-Test* returned a p-value > 0.05, which indicates that fitting the permuted scores with a normal distribution seems plausible.

Author(s)

Hajk-Georg Drost and Jaruwatana Sodai Lotharukpong

References

Drost HG et al. (2015) Mol Biol Evol. 32 (5): 1221-1231 doi:10.1093/molbev/msv012

Quint M et al. (2012). *A transcriptomic hourglass in plant embryogenesis*. Nature (490): 98-101.

Piasecka B, Lichocki P, Moretti S, et al. (2013) *The hourglass and the early conservation models co-existing patterns of developmental constraints in vertebrates*. PLoS Genet. 9(4): e1003476.

See Also

`lcScore`, `bootMatrix`, `FlatLineTest`,`ReductiveHourglassTest`, `ReverseHourglassTest`, `PlotSignature`

Examples

```
data(PhyloExpressionSetExample)

# perform the late conservation test for a PhyloExpressionSet
# here the prior biological knowledge is that stages 1-2 correspond to module 1 = early,
# stages 3-5 to module 2 = mid (phylotypic module), and stages 6-7 correspond to
# module 3 = late
LateConservationTest(PhyloExpressionSetExample,
                       modules = list(early = 1:2, mid = 3:5, late = 6:7),
                       permutations = 1000)


# use your own permutation matrix based on which p-values (LateConservationTest)
# shall be computed
custom_perm_matrix <- bootMatrix(PhyloExpressionSetExample,100)

LateConservationTest(PhyloExpressionSetExample,
                       modules = list(early = 1:2, mid = 3:5, late = 6:7),
                       custom.perm.matrix = custom_perm_matrix)
```

`lcScore` *Compute the Hourglass Score for the LateConservationTest*

Description

This function computes the LateConservationTest score for a given `TAI` or `TDI` pattern.

The reductive late conservation test is a permutation test based on the following test statistic.

- A set of developmental stages is partitioned into three modules - early, mid, and late - based on prior biological knowledge.

- The mean `TAI` or `TDI` value for each of the three modules T_early, T_mid, and T_late are computed.

- The two differences D1 = T_early - T_late and D2 = T_mid - T_late are calculated.

- The minimum D_min of D1 and D2 is computed as final test statistic of the reductive late conservation test.

This function *lcScore* computes the *D_min* value for a given `TAI` or `TDI` stored in the `age_vals` argument.

Usage

```
lcScore(age_vals, early, mid, late, profile.warn = FALSE)
```

Arguments

`age_vals`	a numeric vector containing `TAI` or `TDI` values for each developmental stage s.
`early`	a numeric vector storing the numeric stage values that correspond to the early phase of development.
`mid`	a numeric vector storing the numeric stage values that correspond to the middle phase of development.
`late`	a numeric vector storing the numeric stage values that correspond to the late phase of development.
`profile.warn`	a boolean value indicating whether a warning is printed when a high-mid-low pattern isn't followed.

Value

a numeric value representing the late conservation score.

Author(s)

Hajk-Georg Drost and Jaruwatana Sodai Lotharukpong

See Also

`LateConservationTest`, `TAI`, `TDI`

Examples

```
# read standard phylotranscriptomics data
data(PhyloExpressionSetExample)
data(DivergenceExpressionSetExample)

# Example PhyloExpressionSet:

# compute the TAI profile
TAIs <- TAI(PhyloExpressionSetExample)

# compute the late conservation score for the TAI profile
lc_score <- lcScore(age_vals = TAIs,early = 1:2,mid = 3:5,late = 6:7)
```

```
# Example DivergenceExpressionSet:

# compute the TDI profile
TDIs <- TDI(DivergenceExpressionSetExample)

# compute the late conservation score for the TDI profile
lc_score <- lcScore(age_vals = TDIs,early = 1:2,mid = 3:5,late = 6:7)

# compute lcScore() vector from bootMatrix()
apply(bootMatrix(PhyloExpressionSetExample,10),1,lcScore,early = 1:2,mid = 3:5,late = 6:7)

# get warning if the expected pattern isn't followed
lc_score <- lcScore(age_vals = TAIs,early = 1:2,mid = 3:5,late = 6:7,profile.warn=TRUE)
```

MatchMap — *Match a Phylostratigraphic Map or Divergence Map with a ExpressionMatrix*

Description

This function matches a *Phylostratigraphic Map* or *Divergence Map* only storing unique gene ids with a ExpressionMatrix also storing only unique gene ids.

Usage

```
MatchMap(Map, ExpressionMatrix, remove.duplicates = FALSE, accumulate = NULL)
```

Arguments

`Map`	a standard *Phylostratigraphic Map* or *Divergence Map* object.
`ExpressionMatrix`	a standard ExpressionMatrix object.
`remove.duplicates`	a logical value indicating whether duplicate gene ids should be removed from the data set.
`accumulate`	an accumulation function such as `mean()`, `median()`, or `min()` to accumulate multiple expression levels that map to the same unique gene id present in the `ExpressionMatrix`.

Details

In phylotranscriptomics analyses two major techniques are performed to obtain standard *Phylostratigraphic map* or *Divergence map* objects.

To obtain a *Phylostratigraphic Map*, *Phylostratigraphy* (Domazet-Loso et al., 2007) has to be performed. To obtain a *Divergence Map*, orthologous gene detection, Ka/Ks computations, and decilation (Quint et al., 2012; Drost et al., 2015) have to be performed.

The resulting standard *Phylostratigraphic Map* or *Divergence Map* objects consist of 2 colums storing the phylostratum assignment or divergence stratum assignment of a given gene in column one, and the corresponding gene id of that gene on columns two.

A standard ExpressionMatrix is a common gene expression matrix storing the gene ids or probe ids in the first column, and all experiments/stages/replicates in the following columns.

The *MatchMap* function takes both standard datasets: *Map* and *ExpressionMatrix* to merge both data sets to obtain a standard PhyloExpressionSet or DivergenceExpressionSet object.

This procedure is analogous to `merge`, but is customized to the *Phylostratigraphic Map*, *Divergence Map*, and *ExpressionMatrix* standards to allow a faster and more intuitive usage.

In case you work with an ExpressionMatrix that stores multiple expression levels for a unique gene id, you can specify the `accumulation` argument to accumulate these multiple expression levels to obtain one expression level for one unique gene.

Value

a standard PhyloExpressionSet or DivergenceExpressionSet object.

Author(s)

Hajk-Georg Drost

References

Domazet-Loso T, Brajkovic J, Tautz D (2007) A phylostratigraphy approach to uncover the genomic history of major adaptations in metazoan lineages. Trends Genet. 23: 533-9.

Domazet-Loso T, Tautz D (2010) A phylogenetically based transcriptome age index mirrors ontogenetic divergence patterns. Nature 468: 815-8.

Quint M., Drost H.G., Gabel A., Ullrich K.K., Boenn M., Grosse I. (2012) A transcriptomic hourglass in plant embryogenesis. Nature 490: 98-101.

Drost HG et al. (2015) Mol Biol Evol. 32 (5): 1221-1231 doi:10.1093/molbev/msv012

Examples

```
# load a standard PhyloExpressionSet
data(PhyloExpressionSetExample)

# in a standard PhyloExpressionSet,
# column one and column two denote a standard
# phylostratigraphic map
PhyloMap <- PhyloExpressionSetExample[ , 1:2]

# look at the phylostratigraphic map standard
head(PhyloMap)

# in a standard PhyloExpressionSet, column two combined
# with column 3 - N denote a standard ExpressionMatrix
ExpressionMatrixExample <- PhyloExpressionSetExample[ , c(2,3:9)]
```

```
# these two data sets shall illustrate an example
# phylostratigraphic map that is returned
# by a standard phylostratigraphy run, and a expression set
# that is the result of expression data analysis
# (background correction, normalization, ...)

# now we can use the MatchMap function to merge both data sets
# to obtain a standard PhyloExpressionSet

PES <- MatchMap(PhyloMap, ExpressionMatrixExample)

# note that the function returns a head()
# of the matched gene ids to enable
# the user to find potential mis-matches

# the entire procedure is analogous to merge()
# with two data sets sharing the same gene ids
# as column (primary key)
PES_merge <- merge(PhyloMap, ExpressionMatrixExample)
```

`omitMatrix` *Compute TAI or TDI Profiles Omitting a Given Gene*

Description

For each gene i, exclude the corresponding gene i from the global PhyloExpressionSet or DivergenceExpressionSet and compute the `TAI` or `TDI` profile for the corresponding global PhyloExpressionSet or DivergenceExpressionSet with excluded gene i.

This procedure results in a TAI or TDI profile Matrix storing the TAI or TDI profile for each omitted gene i.

Usage

```
omitMatrix(ExpressionSet)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.

Value

a numeric matrix storing TAI or TDI profile for each omitted gene i.

Author(s)

Hajk-Georg Drost

Examples

```
# read standard phylotranscriptomics data
data(PhyloExpressionSetExample)
data(DivergenceExpressionSetExample)

# example PhyloExpressionSet
omMatrix_ps <- omitMatrix(PhyloExpressionSetExample)

# example DivergenceExpressionSet
omMatrix_ds <- omitMatrix(DivergenceExpressionSetExample)
```

`pairScore` *Compute the Pairwise Difference in TAI Score*

Description

This function computes the PairwiseTest score for a given `TAI` or `TDI` pattern.

The pair test is a permutation test based on the following test statistic.

- A PhyloExpressionSet is partitioned into contrast pairs - contrast1 and contrast2 - based on prior biological knowledge. This prior knowledge could include sexual, ecological and genetic backgrounds.

- The mean `TAI` or `TDI` value for each of the two contrasts contrast1 and contrast2 are computed.

- The pairwise differences D_constrast = contrast1 - contrast2 is calculated as final test statistic of the pair test, when the `altHypothesis` is specified as "greater". When the `altHypothesis` is specified as "less", sign of D_constrast is reversed.

This function *pairScore* computes the *D_contrast* value for a given `TAI` or `TDI` stored in the `age_vals` argument.

Usage

```
pairScore(age_vals, contrast1, contrast2, altHypothesis = NULL)
```

Arguments

`age_vals`	a numeric vector containing `TAI` or `TDI` values for each developmental stage s.
`contrast1`	a numeric vector storing the numeric stage values that correspond to the query contrast.
`contrast2`	a numeric vector storing the numeric stage values that correspond to the subject contrast.
`altHypothesis`	a character string defining the alternative hypothesis used to quantify the statistical significance in the present phylotranscriptomics pattern. Possible values can be: • `altHypothesis` = `"greater"` : contrast1 > contrast2 • `altHypothesis` = `"less"` : contrast1 < contrast2

Value

a numeric value representing the pair score.

Author(s)

Hajk-Georg Drost and Jaruwatana Sodai Lotharukpong

See Also

`PairwiseTest`, `TAI`, `TDI`

Examples

```
# read standard phylotranscriptomics data
data(PhyloExpressionSetExample)
data(DivergenceExpressionSetExample)

# Example PhyloExpressionSet:

# compute the TAI profile
TAIs <- TAI(PhyloExpressionSetExample)

# compute the pair score for the first two stages in the TAI profile
# we test whether TAI in contrast1 is greater than contrast 2.
pair_score <- pairScore(age_vals = TAIs,contrast1 = 1,contrast2 = 2,
                        altHypothesis="greater")


# Example DivergenceExpressionSet:

# compute the TDI profile
TDIs <- TDI(DivergenceExpressionSetExample)

# compute the pair score for the first two stages in the TDI profile
# we test whether TDI in contrast1 is greater than contrast 2.
pair_score <- pairScore(age_vals = TDIs,contrast1 = 1,contrast2 = 2,
                        altHypothesis="greater")

# compute pairScore() vector from bootMatrix()
apply(bootMatrix(PhyloExpressionSetExample,10),1,
      pairScore,contrast1 = 1,contrast2 = 2, altHypothesis="greater")
```

`PairwiseTest` *Perform Pairwise Difference Test*

Description

The *Pairwise Difference Test* aims to statistically evaluate the pairwise difference in the phylotranscriptomic pattern between two contrasts based on `TAI` or `TDI` computations. The corresponding p-value quantifies the probability that a given TAI or TDI pattern (or any phylotranscriptomics pattern) does not differ from the alternative hypothesis (specifying the direction of difference). A p-value < 0.05 indicates that the corresponding phylotranscriptomics pattern does indeed differ.

Usage

```
PairwiseTest(
  ExpressionSet,
  modules = NULL,
  altHypothesis = NULL,
  permutations = 1000,
  lillie.test = FALSE,
  plotHistogram = FALSE,
  runs = 10,
  parallel = FALSE,
  gof.warning = FALSE,
  custom.perm.matrix = NULL
)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.
`modules`	a list storing two elements: contrast1 and contrast2. Each element expects a numeric vector specifying the developmental stages or experiments that correspond to each module. For example, `module` = list(contrast1 = 1:2, contrast2 = 3:7) divides a dataset storing seven developmental stages into 2 modules (contrasts).
`altHypothesis`	a character string defining the alternative hypothesis (i.e. direction of difference) used to quantify the statistical significance in the present phylotranscriptomics pattern. Possible values can be: • `altHypothesis` = `"greater"` : contrast1 > contrast2 • `altHypothesis` = `"less"` : contrast1 < contrast2
`permutations`	a numeric value specifying the number of permutations to be performed for the `ReductiveHourglassTest`.
`lillie.test`	a boolean value specifying whether the Lilliefors Kolmogorov-Smirnov Test shall be performed to quantify the goodness of fit.
`plotHistogram`	a boolean value specifying whether a *Lillifor's Kolmogorov-Smirnov-Test* shall be performed to test the goodness of fit of the approximated distribution, as well as additional plots quantifying the significance of the observed phylotranscriptomic pattern.
`runs`	specify the number of runs to be performed for goodness of fit computations, in case `plotHistogram` = `TRUE`. In most cases `runs` = 100 is a reasonable choice. Default is `runs` = 10 (because it takes less computation time for demonstration purposes).

`parallel`	performing `runs` in parallel (takes all cores of your multicore machine).
`gof.warning`	a logical value indicating whether non significant goodness of fit results should be printed as warning. Default is `gof.warning = FALSE`.
`custom.perm.matrix`	a custom `bootMatrix` (permutation matrix) to perform the underlying test statistic. Default is `custom.perm.matrix = NULL`.

Details

The *reductive pairwise difference test* is a permutation test based on the following test statistic.

(1) A PhyloExpressionSet is partitioned into contrast pairs - contrast1 and contrast2 - based on prior biological knowledge. This prior knowledge could include sexual, ecological and genetic backgrounds.

(2) The mean `TAI` or `TDI` value for each of the two contrasts contrast1 and contrast2 are computed.

(3) The pairwise differences D_contrast = contrast1 - contrast2 is calculated as final test statistic of the pairwise test, when `altHypothesis` is specified as "greater". When `altHypothesis` is specified as "less", sign of D_contrast is reversed.

In order to determine the statistical significance of an observed pairwise difference D_contrast the following permutation test was performed. Based on the `bootMatrix` D_contrast is calculated from each of the permuted `TAI` or `TDI` profiles, approximated by a Gaussian distribution with method of moments estimated parameters returned by `fitdist`, and the corresponding p-value is computed by `pnorm` given the estimated parameters of the Gaussian distribution. The *goodness of fit* for the random vector *D_contrast* is statistically quantified by an Lilliefors (Kolmogorov-Smirnov) test for normality.

In case the parameter *plotHistogram = TRUE*, a multi-plot is generated showing:

(1) A Cullen and Frey skewness-kurtosis plot generated by `descdist`. This plot illustrates which distributions seem plausible to fit the resulting permutation vector D_contrast In the case of the *pairwise difference test* a normal distribution seemed plausible.

(2) A histogram of D_contrast combined with the density plot is plotted. D_contrast is then fitted by a normal distribution. The corresponding parameters are estimated by *moment matching estimation* using the `fitdist` function.

(3) A plot showing the p-values for N independent runs to verify that a specific p-value is biased by a specific permutation order.

(4) A barplot showing the number of cases in which the underlying goodness of fit (returned by Lilliefors (Kolmogorov-Smirnov) test for normality) has shown to be significant (`TRUE`) or not significant (`FALSE`). This allows to quantify the permutation bias and their implications on the goodness of fit.

Value

a list object containing the list elements:

`p.value` : the p-value quantifying the statistical significance of the given phylotranscriptomics pattern.

`std.dev` : the standard deviation of the N sampled phylotranscriptomics patterns for each developmental stage S.

`lillie.test` : a boolean value specifying whether the *Lillifors KS-Test* returned a p-value > 0.05, which indicates that fitting the permuted scores with a normal distribution seems plausible.

Author(s)

Hajk-Georg Drost and Jaruwatana Sodai Lotharukpong

References

Drost HG et al. (2015) Mol Biol Evol. 32 (5): 1221-1231 doi:10.1093/molbev/msv012

Lotharukpong JS et al. (2024) (unpublished)

See Also

`lcScore`, `bootMatrix`, `FlatLineTest`,`ReductiveHourglassTest`, `ReverseHourglassTest`, `PlotSignature`

Examples

```
data(PhyloExpressionSetExample)

# perform the pairwise difference test for a PhyloExpressionSet
# here the prior biological knowledge is that stages 1-2 correspond to contrast1,
# stages 3-7 correspond to contrast 2.
# We test whether TAI in contrast1 is greater than contrast 2.
PairwiseTest(PhyloExpressionSetExample,
         modules = list(contrast1 = 1:2, contrast2 = 3:7),
         altHypothesis = "greater",
         permutations = 1000)

# We can also test whether TAI in contrast1 is less than contrast 2.
PairwiseTest(PhyloExpressionSetExample,
         modules = list(contrast1 = 1:2, contrast2 = 3:7),
         altHypothesis = "less",
         permutations = 1000)

# if we only want to test whether TAI in stage 1 (contrast 1) is greater than stage 3 (contrast 2).
PairwiseTest(PhyloExpressionSetExample,
         modules = list(contrast1 = 1, contrast2 = 2),
         altHypothesis = "greater",
         permutations = 1000)

# use your own permutation matrix based on which p-values (PairwiseTest)
# shall be computed
custom_perm_matrix <- bootMatrix(PhyloExpressionSetExample,100)
# We test whether TAI in contrast1 is greater than contrast 2.
PairwiseTest(PhyloExpressionSetExample,
         modules = list(contrast1 = 1:2, contrast2 = 3:7),
         altHypothesis = "greater",
         custom.perm.matrix = custom_perm_matrix)
```

`PhyloExpressionSetExample`

An Example PhyloExpressionSet Data Set

Description

A standard PhyloExpressionSet is a `data.frame` consisting of a standardized sequence of columns to store the age information for each gene and its corresponding gene expression profile.

The standard is defined as follows:

Phylostratum | GeneID | Expression-level 1 | ... | Expression-level N

Details

This dataset covers 7 developmental stages of Arabidopsis thaliana embryo development. The initial gene expression dataset was published by Xiang et al., 2011 (see references section) and was then used by Quint et al., 2012 (see references section) to assign evolutionary ages to each gene expression profile.

Value

a standard PhyloExpressionSet object.

Author(s)

Hajk-Georg Drost

Source

http://www.plantphysiol.org/content/156/1/346/suppl/DC1

References

Quint M et al. 2012. "A transcriptomic hourglass in plant embryogenesis". Nature (490): 98-101. Supplementary Table 2 : http://www.nature.com/nature/journal/v490/n7418/full/nature11394.html

Xiang D et al. 2011. "Genome-Wide Analysis Reveals Gene Expression and Metabolic Network Dynamics during Embryo Development in Arabidopsis". Plant Physiology (156): 346-356. Supplemental Table 1 : http://www.plantphysiol.org/content/156/1/346/suppl/DC1

See Also

`DivergenceExpressionSetExample`

PlotBarRE *Plot Mean Relative Expression Levels as Barplot*

Description

This function takes a PhyloExpressionSet or DivergenceExpressionSet object as input and computes for two or more defined phylostratum (divergence stratum) classes the statistical significance of the differences of mean relative expression of these two (or more) corresponding phylostratum (divergence stratum) classes. As test-statistic, the function performs a nonparametric `kruskal.test` based on relative expression values stored within each defined phylostratum class.

Usage

```
PlotBarRE(
  ExpressionSet,
  Groups = NULL,
  wLength = 0.1,
  ratio = FALSE,
  p.adjust.method = NULL,
  ...
)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.
`Groups`	a list containing the phylostrata or divergence-strata that correspond to the same phylostratum class or divergence class. For ex. evolutionary old phylostrata: PS1-3 (Class 1) and evolutionary young phylostrata: PS4-12 (Class 2). In this case, the `Groups` list could be assigned as, `Groups` = list(c(1:3), c(4:12)). It is also possible to define more than 2 groups of evolutionary ages. For ex. `Groups` = list(c(1:3),c(4:8),c(9:12)) would perform a `kruskal.test` to determine the statistical significance of the evolutionary classes PS1-3, PS4-6, and PS9-12 based on their corresponding mean relative expression levels.
`wLength`	a numeric value defining the whiskers length above the bars. In case there are numerous different phylostratum classes a smaller `wLength` parameter should be used for better visualizations.
`ratio`	a boolean value specifying whether the bars in the barplot represent the mean relative expression level of phylostrata belonging to the same phylostratum class. In case `ratio` = TRUE, the ratio of the mean relative expression level of the two phylostrata classes is plotted as lines within the barplot. This parameter can only be used for 2 class comparisons.
`p.adjust.method`	correction method to adjust p-values for multiple comparisons (see `p.adjust` for possible methods). E.g., `p.adjust.method` = `"BH"` (Benjamini & Hochberg (1995)) or `p.adjust.method` = `"bonferroni"` (Bonferroni correction).
`...`	default graphics parameters.

Details

In case a large number of developmental stages is included in the input `ExpressionSet`, p-values returned by `PlotBarRE` should be adjusted for multiple comparisons which can be done by specifying the `p.adjust.method` argument.

Value

A barplot comparing Phylostratum-Classes by its mean relative expression levels. Significant stages are marked by '*' referring to statistically significant differences:

(1) '*' = P-Value <= 0.05

(2) '**' = P-Value <= 0.005

(3) '***' = P-Value <= 0.0005

Author(s)

Hajk-Georg Drost

References

Quint M et al. 2012. "A transcriptomic hourglass in plant embryogenesis". Nature (490): 98-101.

Domazet-Loso T. and Tautz D. 2010. "A phylogenetically based transcriptome age index mirrors ontogenetic divergence patterns". Nature (468): 815-818.

Myles Hollander and Douglas A. Wolfe (1973), Nonparametric Statistical Methods. New York: John Wiley & Sons. Pages 115-120.

See Also

`RE`, `REMatrix`,`PlotRE`, `kruskal.test`

Examples

```
# read standard phylotranscriptomics data
data(PhyloExpressionSetExample)
data(DivergenceExpressionSetExample)

# example PhyloExpressionSet
PlotBarRE(ExpressionSet = PhyloExpressionSetExample,
          Groups        = list(c(1:3), c(4:12)))


# example DivergenceExpressionSet
PlotBarRE(ExpressionSet = DivergenceExpressionSetExample,
          Groups        = list(c(1:5), c(6:10)))


# Perform PlotBarRE() with p-value adjustment method Benjamini & Hochberg (1995)
PlotBarRE(ExpressionSet   = PhyloExpressionSetExample,
          Groups          = list(c(1:3), c(4:12)),
          p.adjust.method = "BH")
```

```
# Example: plot ratio
# the ratio curve visualizes the ratio between bar 1 / bar 2
# the z - axis shows the corresponding ratio value of bar 1 / bar 2
PlotBarRE(ExpressionSet = PhyloExpressionSetExample,
          Groups        = list(c(1:3), c(4:12)),
          ratio         = TRUE)
```

PlotCategoryExpr — *Plot the Expression Levels of each Age or Divergence Category as Boxplot, Violinplot, or Dotplot*

Description

This function visualizes the expression level distribution of each phylostratum during each time point or experiment as boxplot, dot plot, or violin plot enabling users to quantify the age (PS) or divergence (DS) category specific contribution to the corresponding transcriptome.

Usage

```
PlotCategoryExpr(
  ExpressionSet,
  legendName,
  test.stat = TRUE,
  type = "category-centered",
  distr.type = "boxplot",
  y.ticks = 10,
  log.expr = FALSE,
  gene.set = NULL
)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.
`legendName`	a character string specifying whether "PS" or "DS" are used to compute relative expression profiles.
`test.stat`	a logical value indicating whether a Benjamini-Hochberg adjusted `kruskal.test` should be applied to determine significant differences in age or divergence category specific expression.
`type`	type of age or divergence category comparison. Specifications can be `type = "category-centered"` or `type = "stage-centered"`.
`distr.type`	format of visualizing age or divergence category specific expression distributions. Either `distr.type = "boxplot"`, `distr.type = "dotplot"`, or `distr.type = "violin"`.

`y.ticks`	number of y-axis ticks (default `y.ticks = 10`).
`log.expr`	a logical value specifying whether or not expression levels should internally be log2-transformed before visualization.
`gene.set`	a character vector storing the gene ids for which gene expression levels shall be visualized.

Details

This way of visualizing the gene expression distribution of each age (PS) or divergence (DS) category during all developmental stages or experiments allows users to detect specific age or divergence categories contributing significant levels of gene expression to the underlying biological process (transcriptome).

This quantification allows users to conclude that genes originating in specific PS or DS contribute significantly more to the overall transcriptome than other genes originating from different PS or DS categories. More specialized analyses such as `PlotMeans`, `PlotRE`, `PlotBarRE`, etc. will then allow to study the exact mean expression patterns of these age or divergence categories.

The statistical quantification of differences between expression levels of different age or divergence categories is done by performing a `kruskal.test` with Benjamini & Hochberg p-value adjustment for multiple comparisons.

- `type = "category-centered"` Here, the `kruskal.test` quantifies the differences of gene expression between all combinations of age or divergence categories for each stage or experiment separately. Here, a significant p-value quantifies that there is at least one pairwise comparison for which age or divergence categories significantly differ in their gene expression distribution. This type of analysis allows users to detect stages or experiments that show high diviation between age or divergence category contributions to the overall transcriptome or no significant deviations of age or divergence categories, suggesting equal age or divergence category contributions to the overall transcriptome.
- `type = "stage-centered"` Here, the `kruskal.test` quantifies the differences of gene expression between all stages or experiments for each age or divergence category separately. Hence, the test quantifies whether or not the gene expression distribution of a single age or divergence category significantly changes throughout development or experiments. This type of analysis allows users to detect specific age or divergence categories that significantly change their expression levels throughout development or experiments.

Argument Specifications:

Argument: type

- `type = "category-centered"` This specification allows users to compare the differences between all age or divergence categories during the same stage or experiment.
- `type = "stage-centered"` This specification allows users to compare the differences between all age or divergence categories between stages or experiments.

Argument: distr.type

- `distr.type = "boxplot"` This specification allows users to visualize the expression distribution of all PS or DS as boxplot.

- `distr.type = "violin"` This specification allows users to visualize the expression distribution of all PS or DS as violin plot.
- `distr.type = "dotplot"` This specification allows users to visualize the expression distribution of all PS or DS as dot plot.

Finally, users can specify a `gene.set` (a subset of genes included in the input `ExpressioSet`) for which expression levels should be visualized as boxplot, dotplot, or violinplot.

Value

A boxplot, violin plot, or dot plot visualizing the gene expression levels of different PS or DS categories.

Furthermore, the statistical test results returned from the `kruskal.test` are printed to the console.

(1) '*' = P-Value <= 0.05

(2) '**' = P-Value <= 0.005

(3) '***' = P-Value <= 0.0005

(4) 'n.s.' = not significant = P-Value > 0.05

Author(s)

Hajk-Georg Drost

See Also

`PlotMeans`, `PlotRE`, `PlotBarRE`, `age.apply`, `pTAI`, `pTDI`, `pStrata`, `pMatrix`, `TAI`, `TDI`

Examples

```
data(PhyloExpressionSetExample)
data(DivergenceExpressionSetExample)

## Not run:

# category-centered visualization of PS specific expression level distributions (log-scale)
PlotCategoryExpr(ExpressionSet = PhyloExpressionSetExample,
                     legendName    = "PS",
                     test.stat     = TRUE,
                     type          = "category-centered",
                     distr.type    = "boxplot",
                     log.expr      = TRUE)


# stage-centered visualization of PS specific expression level distributions (log-scale)
PlotCategoryExpr(ExpressionSet = PhyloExpressionSetExample,
                     legendName    = "PS",
                     test.stat     = TRUE,
                     distr.type    = "boxplot",
                     type          = "stage-centered",
                     log.expr      = TRUE)
```

```
# category-centered visualization of PS specific expression level distributions (log-scale)
# as violoin plot
PlotCategoryExpr(ExpressionSet = PhyloExpressionSetExample,
                     legendName    = "PS",
                     test.stat     = TRUE,
                     distr.type    = "violin",
                     type          = "stage-centered",
                     log.expr      = TRUE)




# analogous for DivergenceExpressionSets
PlotCategoryExpr(ExpressionSet = DivergenceExpressionSetExample,
                     legendName    = "DS",
                     test.stat     = TRUE,
                     type          = "category-centered",
                     distr.type    = "boxplot",
                     log.expr      = TRUE)


# visualize the expression levels of 500 example genes
set.seed(234)
example.gene.set <- PhyloExpressionSetExample[sample(1:25260,500) , 2]

PlotCategoryExpr(ExpressionSet = PhyloExpressionSetExample,
                 legendName    = "PS",
                 test.stat     = TRUE,
                 type          = "category-centered",
                 distr.type    = "boxplot",
                 log.expr      = TRUE,
                 gene.set      = example.gene.set)


## End(Not run)
```

`PlotCIRatio` *Plot Transcriptome Index using bootstrapping and confidence intervals*

Description

Function to plot and compare the confidence intervals of Transcriptome Index between transformed and non-transformed expression data by using bootstrapping appraoches instead of permutation tests used in `PlotSignature`.

Usage

```
PlotCIRatio(ExpressionSet, measure, nbootstraps)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet, DivergenceExpressionSet or PolymorphismsExpressionSet object.
`measure`	type of transcriptome index that shall be computed. E.g. measure = "TAI" (Transcriptome Age Index), measure = "TDI" (Transcriptome Divergence Index), measure = "TPI" (Transcriptome Polymorphism Index).
`nbootstraps`	number of independent bootstraps.

Details

This function can be used to check potential outliers (e.g. a few exramly highly expressed genes) in transcriptome. Since Transcriptome Index is weigthed mean, it could be easily affectd by outliers. So, we have to check potential outliers in the transcriptome data. Because log or sqrt trandformation can alleviate the effect of outliers, if there are some outliers, we could see the confidence intervals (genetated by bootstrapping) from non-trandformed expression data are much higher and more variable than from log or sqrt trandformed expression data. In order to compare the range of confidence intervals in the same scale, we plotted the ratio of upper to lower confidence interval boundary across development.

Author(s)

Jialin Liu

See Also

`PlotSignature`

Examples

```
data("PhyloExpressionSetExample")
PlotCIRatio(PhyloExpressionSetExample,"TAI",5)
```

`PlotContribution` *Plot Cumulative Transcriptome Index*

Description

This function computes the cumulative contribution of each Phylostratum or Divergence Stratum to the global `TAI` or `TDI` profile.

Usage

```
PlotContribution(
  ExpressionSet,
  legendName = NULL,
  xlab = "Ontogeny",
  ylab = "Transcriptome Index",
  main = "",
  y.ticks = 10
)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.
`legendName`	a character string specifying whether "PS" or "DS" are used to compute relative expression profiles.
`xlab`	label of x-axis.
`ylab`	label of y-axis.
`main`	main title.
`y.ticks`	a numeric value specifying the number of ticks to be drawn on the y-axis.

Details

Introduced by Domazet-Loso and Tautz (2010), this function allows users to visualize the cumulative contribution of each Phylostratum or Divergence Stratum to the global Transcriptome Age Index or Transcriptome Divergence Index profile to quantify how each Phylostratum or Divergence Stratum influences the profile of the global TAI or TDI pattern.

Author(s)

Hajk-Georg Drost

References

Domazet-Loso T. and Tautz D. (2010). A phylogenetically based transcriptome age index mirrors ontogenetic divergence patterns. Nature (468): 815-818.

See Also

`pTAI`, `pTDI`, `TAI`, `TDI`, `PlotSignature`

Examples

```
data(PhyloExpressionSetExample)
data(DivergenceExpressionSetExample)

# visualize phylostratum contribution to global TAI
PlotContribution(PhyloExpressionSetExample, legendName = "PS")
```

```
# visualize divergence stratum contribution to global TDI
PlotContribution(DivergenceExpressionSetExample, legendName = "DS")
```

PlotCorrelation *Plot the Correlation Between Phylostrata and Divergence Strata*

Description

This function plots the correlation coefficient between phylostratum values and divergence-stratum values of a given PhyloExpressionSet and DivergenceExpressionSet.

This function can be used to test whether a given PS distribution and DS distribution are linear correlated so that the independence of PS and DS can be assumed for subsequent analyses (Quint et al., 2012).

Usage

```
PlotCorrelation(
  PhyloExpressionSet,
  DivergenceExpressionSet,
  method = "pearson",
  linearModel = FALSE,
  xlab = "Phylostratum",
  ylab = "Divergencestratum"
)
```

Arguments

`PhyloExpressionSet`	a standard PhyloExpressionSet object.
`DivergenceExpressionSet`	a standard DivergenceExpressionSet object.
`method`	a character string specifying the correlation method to cbe used, e.g. "pearson", "kendall", "spearman".
`linearModel`	a boolean value specifying whether a linear model should be fitted to the data and furthermore, should be visualized in the corresponding plot.
`xlab`	label of x-axis.
`ylab`	label of y-axis.

Value

a jitter-correlation-plot of PS and DS correlation.

Author(s)

Hajk-Georg Drost

References

Quint M et al. (2012). *A transcriptomic hourglass in plant embryogenesis*. Nature (490): 98-101.
Drost HG et al. (2015) *Evidence for Active Maintenance of Phylotranscriptomic Hourglass Patterns in Animal and Plant Embryogenesis*. Mol Biol Evol. 32 (5): 1221-1231 doi:10.1093/molbev/msv012.

See Also

cor

Examples

```
# read standard phylotranscriptomics data
data(PhyloExpressionSetExample)
data(DivergenceExpressionSetExample)

# plot the PS and DS correlation
PlotCorrelation(PhyloExpressionSetExample,
                DivergenceExpressionSetExample,
                method      = "pearson",
                linearModel = TRUE)
```

PlotDistribution *Plot the Frequency Distribution of Phylostrata or Divergence Strata*

Description

This function plots the frequency distribution of genes within the corresponding *phylostratigraphic map* or *divergence map* and can be used to fastly visualize the PS or DS distribution of a given phylostratum vector or divergence-stratum vector.

Usage

```
PlotDistribution(
  PhyloExpressionSet,
  legendName = "PS",
  as.ratio = FALSE,
  use.only.map = FALSE,
  xlab = NULL,
  ylab = NULL
)
```

Arguments

`PhyloExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.
`legendName`	a character string specifying whether "PS" or "DS" are visualized.
`as.ratio`	a boolean value specifying whether the relative frequencies instead of absolute frequencies shall be plotted.
`use.only.map`	logical value indicating whether or not a Phylostratigraphic Map or Divergence Map should be passed to the `ExpressionSet` argument instead of a standard `ExpressionSet` object.
`xlab`	label of the x-axis.
`ylab`	label of the y-axis.

Details

The frequency distribution of all genes or a subset of genes might be of interest for subsequent analyses.

For Example:

Filtering genes using gene cluster algorithms can result in different groups (classes) of genes. For each gene group the phylostratum or divergence-stratum distribution can be visualized using this function and can be compared between different groups.

This analysis allows to compare different gene expression profiles (or gene groups in general) based on their evolutionary origins or evolutionary relationships.

Value

a barplot showing the phylostratum distribution or divergence-stratum distribution of a given numeric vector containing PS or DS values.

Author(s)

Hajk-Georg Drost

See Also

`PlotSelectedAgeDistr`

Examples

```
# load PhyloExpressionSet
data(PhyloExpressionSetExample)

# plot the phylostratum distribution of a PhyloExpressionSet
PlotDistribution(PhyloExpressionSetExample)

# plot the relative frequency distribution of a PhyloExpressionSet
PlotDistribution(PhyloExpressionSetExample, as.ratio = TRUE)
```

```
# a example for visualizing the PS distribution for a subset of genes
PlotDistribution(PhyloExpressionSetExample[sample(20000,5000) , ], as.ratio = TRUE)
```

`PlotEnrichment` *Plot the Phylostratum or Divergence Stratum Enrichment of a given Gene Set*

Description

This function computes and visualizes the significance of enriched (over or underrepresented) Phylostrata or Divergence Strata within an input `test.set`.

Usage

```
PlotEnrichment(
  ExpressionSet,
  test.set,
  use.only.map = FALSE,
  measure = "log-foldchange",
  complete.bg = TRUE,
  legendName = "",
  over.col = "steelblue",
  under.col = "midnightblue",
  epsilon = 1e-05,
  cex.legend = 1,
  cex.asterisk = 1,
  plot.bars = TRUE,
  p.adjust.method = NULL,
  ...
)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object (in case `only.map = FALSE`).
`test.set`	a character vector storing the gene ids for which PS/DS enrichment analyses should be performed.
`use.only.map`	a logical value indicating whether instead of a standard `ExpressionSet` only a `Phylostratigraphic Map` or `Divergene Map` is passed to this function.
`measure`	a character string specifying the measure that should be used to quantify over and under representation of PS/DS. Measures can either be `measure = "foldchange"` (odds) or `measure = "log-foldchange"` (log-odds).

`complete.bg`	a logical value indicating whether the entire background set of the input `ExpressionSet` should be considered when performing Fisher's exact test (`complete.bg = TRUE`) or whether genes that are stored in `test.set` should be excluded from the background set before performing Fisher's exact test (`complete.bg = FALSE`).
`legendName`	a character string specifying whether "PS" or "DS" are used to compute relative expression profiles.
`over.col`	color of the overrepresentation bars.
`under.col`	color of the underrepresentation bars.
`epsilon`	a small value to shift values by epsilon to avoid log(0) computations.
`cex.legend`	the `cex` value for the legend.
`cex.asterisk`	the `cex` value for the asterisk.
`plot.bars`	a logical value specifying whether or not bars should be visualized or whether only `p.values` and `enrichment.matrix` should be returned.
`p.adjust.method`	correction method to adjust p-values for multiple comparisons (see `p.adjust` for possible methods). E.g., `p.adjust.method = "BH"` (Benjamini & Hochberg (1995)) or `p.adjust.method = "bonferroni"` (Bonferroni correction).
`...`	default graphics parameters.

Details

This *Phylostratum* or *Divergence Stratum* enrichment analysis is motivated by Sestak and Domazet-Loso (2015) who perform *Phylostratum* or *Divergence Stratum* enrichment analyses to correlate organ evolution with the origin of organ specific genes.

In detail this function takes the *Phylostratum* or *Divergence Stratum* distribution of all genes stored in the input `ExpressionSet` as background set and the *Phylostratum* or *Divergence Stratum* distribution of the `test.set` and performes a `fisher.test` for each *Phylostratum* or *Divergence Stratum* to quantify the statistical significance of over- or underrepresentated *Phylostrata* or *Divergence Strata* within the set of selected `test.set` genes.

To visualize the odds or log-odds of over or underrepresented genes within the `test.set` the following procedure is performed:

- N_ij denotes the number of genes in group j and deriving from PS i, with *i = 1, .. , n* and where *j = 1* denotes the background set and *j = 2* denotes the `test.set`
- N_i. denotes the total number of genes within PS i
- N_.j denotes the total number of genes within group j
- N_.. is the total number of genes within all groups j and all PS i
- f_ij = N_ij / N_.. and g_ij = f_ij / f_.j denote relative frequencies between groups
- f_i. denotes the between group sum of f_ij

The result is the fold-change value (odds) denoted as C = g_i2 / f_i. which is visualized above and below zero.

In case a large number of Phylostrata or Divergence Strata is included in the input `ExpressionSet`, p-values returned by `PlotEnrichment` should be adjusted for multiple comparisons which can be done by specifying the `p.adjust.method` argument.

Author(s)

Hajk-Georg Drost

References

Sestak and Domazet-Loso (2015). Phylostratigraphic Profiles in Zebrafish Uncover Chordate Origins of the Vertebrate Brain. Mol. Biol. Evol. 32(2): 299-312.

See Also

`EnrichmentTest`, `fisher.test`

Examples

```
data(PhyloExpressionSetExample)

set.seed(123)
test_set <- sample(PhyloExpressionSetExample[ , 2],10000)

## Examples with complete.bg = TRUE
## Hence: the entire background set of the input ExpressionSet is considered
## when performing Fisher's exact test

# measure: log-foldchange
PlotEnrichment(ExpressionSet = PhyloExpressionSetExample,
               test.set      = test_set ,
               legendName    = "PS",
               measure       = "log-foldchange")


# measure: foldchange
PlotEnrichment(ExpressionSet = PhyloExpressionSetExample,
               test.set      = test_set ,
               legendName    = "PS",
               measure       = "foldchange")


## Examples with complete.bg = FALSE
## Hence: the test.set genes are excluded from the background set before
## Fisher's exact test is performed


# measure: log-foldchange
PlotEnrichment(ExpressionSet = PhyloExpressionSetExample,
               test.set      = test_set ,
                complete.bg  = FALSE,
               legendName    = "PS",
               measure       = "log-foldchange")


# measure: foldchange
PlotEnrichment(ExpressionSet = PhyloExpressionSetExample,
               test.set      = test_set ,
               complete.bg   = FALSE,
               legendName    = "PS",
               measure       = "foldchange")
```

PlotGeneSet *Plot the Expression Profiles of a Gene Set*

Description

This function simply visualizes the gene expression profiles of a defined subset of genes stored in the input `ExpressionSet`.

Usage

```
PlotGeneSet(
  ExpressionSet,
  gene.set,
  get.subset = FALSE,
  use.only.map = FALSE,
  colors = NULL,
  plot.legend = TRUE,
  y.ticks = 6,
  digits.ylab = 4,
  ...
)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.
`gene.set`	a character vector storing the gene ids for which gene expression profiles shall be visualized.
`get.subset`	a logical value indicating whether or not an `ExpressionSet` subset of the selected `gene.set` should be retuned.
`use.only.map`	a logical value indicating whether instead of a standard `ExpressionSet` only a `Phylostratigraphic Map` or `Divergene Map` is passed to the function.
`colors`	colors for gene expression profiles. Default: `colors = NULL`, hence default colours are used.
`plot.legend`	a logical value indicating whether gene ids should be printed as legend next to the plot.
`y.ticks`	a numeric value specifying the number of ticks to be drawn on the y-axis.
`digits.ylab`	a numeric value specifying the number of digits shown for the expression levels on the y-axis.
`...`	additional parameters passed to `matplot`.

Details

This function simply visualizes or subsets the gene expression levels of a set of genes that are stored in the input `ExpressionSet`.

Author(s)

Hajk-Georg Drost

See Also

`SelectGeneSet`, `PlotEnrichment`, `DiffGenes`

Examples

```
data(PhyloExpressionSetExample)

# the best parameter setting to visualize this plot:
# png("test_png.png",700,400)
PlotGeneSet(ExpressionSet = PhyloExpressionSetExample,
            gene.set      = PhyloExpressionSetExample[1:5, 2],
            type          = "l",
            lty           = 1,
            lwd           = 4,
            xlab          = "Ontogeny",
            ylab          = "Expression Level")

# dev.off()

# In case you would like to work with the expression levels
# of selected genes you can specify the 'get.subset' argument:

PlotGeneSet(ExpressionSet = PhyloExpressionSetExample,
            gene.set      = PhyloExpressionSetExample[1:5, 2],
            get.subset    = TRUE)


# get a gene subset using only a phylostratihraphic map
ExamplePSMap <- PhyloExpressionSetExample[ , 1:2]

PlotGeneSet(ExpressionSet = ExamplePSMap,
            gene.set      = PhyloExpressionSetExample[1:5, 2],
            get.subset    = TRUE,
            use.only.map  = TRUE)
```

PlotGroupDiffs — *Plot the significant differences between gene expression distributions of PS or DS groups*

Description

This function performs a test to quantify the statistical significance between the global expression level distributions of groups of PS or DS. It therefore, allows users to investigate significant groups of PS or DS that significantly differ in their gene expression level distibution within specific developmental stages or experiments.

Usage

```
PlotGroupDiffs(
  ExpressionSet,
  Groups = NULL,
  legendName = NULL,
  stat.test = "wilcox.test",
  col = c("turquoise3", "magenta3"),
  plot.type = NULL,
  gene.set = NULL,
  ...
)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.
`Groups`	a list containing the phylostrata or divergence strata that correspond to the same phylostratum class or divergence class. For ex. evolutionary old phylostrata: PS1-3 (Class 1) and evolutionary young phylostrata: PS4-12 (Class 2). In this case, the list could be assigned as, `Groups` = list(c(1:3), c(4:12)).
`legendName`	a character string specifying whether "PS" or "DS" are used to compute relative expression profiles.
`stat.test`	the statistical test to quantify PS or DS group differences.
`col`	colors for the two box plots representing the expression level distributions of selected PS/DS groups.
`plot.type`	the type of plot that shall be drawn to visualized the difference in PS/DS group specific expression .
`gene.set`	a character vector storing the gene ids for which group specific differences shall be statistically quantified.
`...`	additional plot parameters.

Details

The purpose of this function is to detect groups of PS or DS that significantly differ in their gene expression level distributions on a global (transcriptome) level. Since relative expression levels (`PlotRE`) or PS or DS specific mean expression levels (`PlotMeans`) are biased by highly expressed genes, this function allows users to objectively test the significant difference of transcriptome expression between groups of PS or DS in a specific developmental stage or experiment.

In particular, this function divides (for each developmental stage separately) the gene expression levels into two groups: Group1 = genes deriving from selected PS/DS in group 1 and Group2 =

genes deriving from selected PS/DS in group 2. Within each stage the expression level distributions between group 1 and group 2 are statistically quantified using a `wilcox.test`.

Author(s)

Hajk-Georg Drost

See Also

`PlotMeans`, `PlotRE`, `PlotBarRE`, `PlotCategoryExpr`, `GroupDiffs`

Examples

```
data(PhyloExpressionSetExample)

PlotGroupDiffs(ExpressionSet = PhyloExpressionSetExample,
               Groups        = list(group_1 = 1:3,group_2 = 4:12),
               legendName    = "PS",
               type          = "b",
               lwd           = 6,
               xlab          = "Ontogeny")


# only receive the p-values without the corresponding plot
PlotGroupDiffs(ExpressionSet = PhyloExpressionSetExample,
               Groups        = list(group_1 = 1:3,group_2 = 4:12),
               legendName    = "PS",
               plot.p.vals   = FALSE,
               type          = "b",
               lwd           = 6,
               xlab          = "Ontogeny")


# quantify the significant difference of a selected set of genes
# only receive the p-values without the corresponding plot
set.seed(123)
ExampleGeneSet <- sample(PhyloExpressionSetExample[ , 2],5000)

PlotGroupDiffs(ExpressionSet = PhyloExpressionSetExample,
               Groups        = list(group_1 = 1:3,group_2 = 4:12),
               legendName    = "PS",
               plot.p.vals   = FALSE,
               gene.set      = ExampleGeneSet)


# plot differences as boxplot for each developmental stage
PlotGroupDiffs(ExpressionSet = tf(PhyloExpressionSetExample,log2),
               Groups        = list(group_1 = 1:3,group_2 = 4:12),
               legendName    = "PS",
               plot.type     = "boxplot")
```

PlotMeans *Plot Mean Expression Profiles*

Description

This function computes for each age category the corresponding mean expression profile.

Usage

```
PlotMeans(
  ExpressionSet,
  Groups = NULL,
  modules = NULL,
  legendName = "age",
  xlab = "Ontogeny",
  ylab = "Mean Expression Level",
  main = "",
  y.ticks = 10,
  adjust.range = TRUE,
  alpha = 0.008,
  ...
)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.
`Groups`	a list containing the age categories for which mean expression levels shall be drawn. For ex. evolutionary users can compare old phylostrata: PS1-3 (Class 1) and evolutionary young phylostrata: PS4-12 (Class 2). In this example, the list could be assigned as, `Groups = list(c(1:3), c(4:12))`. The group options is limited to 2 Groups.
`modules`	a list storing three elements for specifying the modules: early, mid, and late. Each element expects a numeric vector specifying the developmental stages or experiments that correspond to each module. For example, `module = list(early = 1:2, mid = 3:5, late = 6:7)` devides a dataset storing seven developmental stages into 3 modules. Default is `modules = NULL`. But if specified, a shaded are will be drawn to illustrate stages corresponding to the mid module.
`legendName`	a character string specifying the legend title.
`xlab`	label of x-axis.
`ylab`	label of y-axis.
`main`	main text.
`y.ticks`	number of ticks that shall be drawn on the y-axis.
`adjust.range`	logical indicating whether or not the y-axis scale shall be adjusted to the same range in case two groups are specified. Default is `adjust.range = TRUE`.

`alpha`	transparency of the shaded area (between [0,1]). Default is `alpha = 0.1`.
`...`	place holder for old version of PlotMeans that was based on base graphics instead of ggplot2.

Details

This plot may be useful to compare the absolute mean expression levels of each age category across stages.

In different developmental processes different phylostratum or divergence-stratum classes might be more expressed than others, hence contributing more to the overall phylotranscriptomics pattern (`TAI` or `TDI`). This plot can help to identify the phylostratum or divergence-stratum classes that contributes most to the overall transcriptome of the given developmental process.

Value

a plot showing mean expression profiles of each age category.

Author(s)

Hajk-Georg Drost

See Also

`PlotBarRE`, `RE`, `REMatrix`, `PlotRE`

Examples

```
### Example using a PhyloExpressionSet
### and DivergenceExpressionSet
# load PhyloExpressionSet
data(PhyloExpressionSetExample)

# load PhyloExpressionSet
data(DivergenceExpressionSetExample)

# plot evolutionary old PS (PS1-3) vs evolutionary young PS (PS4-12)
PlotMeans(PhyloExpressionSetExample,
          Groups = list(c(1:3), c(4:12)),
          legendName = "PS",
          adjust.range = TRUE)

# if users wish to not adjust the y-axis scale when
# 2 groups are selected they can specify: adjust.range = FALSE
PlotMeans(PhyloExpressionSetExample,
          Groups = list(c(1:3), c(4:12)),
          legendName = "PS",
          adjust.range = FALSE)


# plot conserved DS (DS1-5) vs divergent DS (PS6-10)
# NOTE: DS are always defined in the range 1, 2, ... , 10.
# Hence, make sure that your groups are within this range!
PlotMeans(DivergenceExpressionSetExample,
          Groups = list(c(1:5), c(6:10)),
          legendName = "DS",
          adjust.range = TRUE)
```

PlotMedians *Plot Median Expression Profiles*

Description

This function computes for each age category the corresponding median expression profile.

Usage

```
PlotMedians(
  ExpressionSet,
  Groups = NULL,
  legendName = "age",
  xlab = "Ontogeny",
  ylab = "Median Expression Level",
  main = "",
  y.ticks = 10,
  adjust.range = TRUE
)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.
`Groups`	a list containing the age categories for which median expression levels shall be drawn. For ex. evolutionary users can compare old phylostrata: PS1-3 (Class 1) and evolutionary young phylostrata: PS4-12 (Class 2). In this example, the list could be assigned as, `Groups = list(c(1:3), c(4:12))`. The group options is limited to 2 Groups.
`legendName`	a character string specifying the legend title.
`xlab`	label of x-axis.
`ylab`	label of y-axis.
`main`	main text.
`y.ticks`	number of ticks that shall be drawn on the y-axis.
`adjust.range`	logical indicating whether or not the y-axis scale shall be adjusted to the same range in case two groups are specified. Default is `adjust.range = TRUE`.

Details

This plot may be useful to compare the absolute median expression levels of each age category across stages.

In different developmental processes different phylostratum or divergence-stratum classes might be more expressed than others, hence contributing more to the overall phylotranscriptomics pattern (`TAI` or `TDI`). This plot can help to identify the phylostratum or divergence-stratum classes that contributes most to the overall transcriptome of the given developmental process.

Value

a plot showing median expression profiles of each age category.

Author(s)

Hajk-Georg Drost

See Also

`PlotBarRE`, `RE`, `REMatrix`, `PlotRE`

Examples

```
### Example using a PhyloExpressionSet
### and DivergenceExpressionSet
# load PhyloExpressionSet
data(PhyloExpressionSetExample)

# load PhyloExpressionSet
data(DivergenceExpressionSetExample)

# plot evolutionary old PS (PS1-3) vs evolutionary young PS (PS4-12)
PlotMedians(PhyloExpressionSetExample,
          Groups = list(c(1:3), c(4:12)),
          legendName = "PS",
          adjust.range = TRUE)

# if users wish to not adjust the y-axis scale when
# 2 groups are selected they can specify: adjust.range = FALSE
PlotMedians(PhyloExpressionSetExample,
          Groups = list(c(1:3), c(4:12)),
          legendName = "PS",
          adjust.range = FALSE)


# plot conserved DS (DS1-5) vs divergent DS (PS6-10)
# NOTE: DS are always defined in the range 1, 2, ... , 10.
# Hence, make sure that your groups are within this range!
PlotMedians(DivergenceExpressionSetExample,
          Groups = list(c(1:5), c(6:10)),
          legendName = "DS",
          adjust.range = TRUE)
```

PlotPattern *Plot the Transcriptome Age Index or Transcriptome Divergence Index*

Description

Function to plot the TAI or TDI of a given PhyloExpressionSet or DivergenceExpressionSet object. This function plot the `TAI` or `TDI` of a given PhyloExpressionSet or DivergenceExpressionSet object.

Usage

```
PlotPattern(
  ExpressionSet,
  TestStatistic = "FlatLineTest",
  modules = NULL,
  permutations = 1000,
  lillie.test = FALSE,
  digits.ylab = 4,
  p.value = TRUE,
  shaded.area = FALSE,
  y.ticks = 4,
  custom.perm.matrix = NULL,
  ...
)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.
`TestStatistic`	a string defining the type of test statistics to be used to quantify the statistical significance the present phylotranscriptomics pattern. Possible values can be: `TestStatistic = "FlatLineTest"` : Statistical test for the deviation from a flat line. `TestStatistic = "ReductiveHourglassTest"` : Statistical test for the existence of a hourglass shape (high-low-high pattern). `TestStatistic = "EarlyConservationTest"` : Statistical test for the existence of a earlyconservation pattern (low-high-high pattern).
`modules`	a list storing three elements for the `ReductiveHourglassTest` or `EarlyConservationTest`: early, mid, and late. Each element expects a numeric vector specifying the developmental stages or experiments that correspond to each module. For example, `module = list(early = 1:2, mid = 3:5, late = 6:7)` devides a dataset storing seven developmental stages into 3 modules.
`permutations`	a numeric value specifying the number of permutations to be performed for the `FlatLineTest`, `EarlyConservationTest` or `ReductiveHourglassTest`.
`lillie.test`	a boolean value specifying whether the Lilliefors Kolmogorov-Smirnov Test shall be performed.

`digits.ylab`	a numeric value specifying the number of digits shown for the TAI or TDI values on the y-axis.
`p.value`	a boolean value specifying whether the p-value of the test statistic shall be printed within the plot area.
`shaded.area`	a boolean value specifying whether a shaded area shall be drawn for the developmental stages defined to be the presumptive phylotypic period.
`y.ticks`	a numeric value specifying the number of ticks to be drawn on the y-axis.
`custom.perm.matrix`	a custom `bootMatrix` (permutation matrix) to perform the underlying test statistic visualized by `PlotPattern`. Default is `custom.perm.matrix = NULL`.
`...`	default plot parameters.

Details

This function computes a permutation test quantifying the statistical significance of the prensent phylotranscriptomics pattern. The user can choose between the `FlatLineTest`, `ReductiveHourglassTest`, or `EarlyConservationTest`. The `FlatLineTest` tests for any significant deviation from a flat line. Each period or stage that significantly deviates from a flat line, might be governed by stronger selective pressure (in terms of natural selection) compared to other stages or periods of development. The `ReductiveHourglassTest` specificly tests for the statistical significance of a molecular hourglass pattern (high-low-high pattern) with prior biological knowlegde. The corresponding p-value that is printed within the plot (by default) specifies the statistical significance of the chosen test statistic.

The `EarlyConservationTest` specificly tests for the statistical significance of a low-high-high pattern (monotonically increasing function) with prior biological knowlegde. The corresponding p-value that is printed within the plot (by default) specifies the statistical significance of the chosen test statistic.

The x-axis denotes the developmental series (time course / experiments / ontogeny) of the input ExpressionSet and the y-axis denotes the corresponding mean transcriptome age value (`TAI` or `TDI`) of the given ExpressionSet.

Furthermore, the grey lines above and below the actual phylotranscriptomics pattern denotes the standard deviations of `TAI` or `TDI` values that have been estimated from the `bootMatrix`. A low mean transcriptome age value denotes an evolutionary older transcriptome being active during the corresponding periods, whereas a high mean transcriptome age value denotes an evolutionary younger transcriptome being active during the corresponding periods. For mean transcriptome divergence, a low mean transcriptome divergence value denotes a more conserved transcriptome being active (between two species), whereas a high mean transcriptome divergence value denotes a more divergent transcriptome being active (between two species) - in terms of protein-sequence substitution rates.

This function is useful to fastly plot the `TAI` or `TDI` profile of a given PhyloExpressionSet or DivergenceExpressionSet object and the statistical significance of the corresponding pattern. Internally the function calls several graphics functions, such as `plot`, `axis`, and `legend`. For the ellipsis argument `...` all graphics specific arguments can be defined. Internally the function specific arguments for e.g. `plot`, `axis`, and `legend` will be detected and are passed to the corresponding function.

Hence, when calling the function `PlotPattern`, one can specify arguments for `plot` and `axis` and `legend` as `...`.

In case prior biological knowledge is present for a specific period of development, the `shaded.area` argument can be set to `TRUE` and the function will use the values stored in the `mid` argument to draw a shaded area within the corresponding period of development.

Value

a plot visualizing the phylotranscriptomic pattern of a given PhyloExpressionSet or DivergenceExpressionSet object.

The corresponding *p-value* of the test statistic is named as follows:

`p_flt` = p-value of the corresponding `FlatLineTest`

`p_rht` = p-value of the corresponding `ReductiveHourglassTest`

`p_ect` = p-value of the corresponding `EarlyConservationTest`

Author(s)

Hajk-Georg Drost

References

Domazet-Loso T and Tautz D. (2010). *A phylogenetically based transcriptome age index mirrors ontogenetic divergence patterns*. Nature (468): 815-818.

Quint M et al. (2012). *A transcriptomic hourglass in plant embryogenesis*. Nature (490): 98-101.

Drost HG et al. (2015) Mol Biol Evol. 32 (5): 1221-1231 doi:10.1093/molbev/msv012.

See Also

`TAI`, `TDI`, `FlatLineTest`, `ReductiveHourglassTest`, `EarlyConservationTest`, `PlotSignature`

Examples

```
# load PhyloExpressionSet
data(PhyloExpressionSetExample)

# only visualize the TAI profile without any test statistics...
# this is equavalent to performing: plot(TAI(PhyloExpressionSetExample), type = "l", lwd = 6)
PlotPattern(ExpressionSet = PhyloExpressionSetExample,
            TestStatistic = NULL,
            type          = "l",
            xlab          = "Ontogeny",
            ylab          = "TAI",
            lwd           = 9)

# the simplest example of plotting the TAI profile of a given PhyloExpressionSet:
# In this case (default) the FlatLineTest will be performed to quantify
# the statistical significance of the present TAI pattern and will be drawn as 'p = ... '
# in the plot

PlotPattern(ExpressionSet = PhyloExpressionSetExample,
            TestStatistic = "FlatLineTest",
            permutations  = 100,
            type          = "l",
            xlab          = "Ontogeny",
            ylab          = "TAI",
            lwd           = 9)

# an example performing the ReductiveHourglassTest and printing the p-value
# and shaded area of the presumptive phylotypic period into the plot
# Here the 'p = ...' denotes the p-value that is returned by the ReductiveHourglassTest

PlotPattern(
            ExpressionSet = PhyloExpressionSetExample,
            TestStatistic = "ReductiveHourglassTest",
            modules       = list(early = 1:2,mid = 3:5,late = 6:7),
            permutations  = 100,
            p.value       = TRUE,
            shaded.area   = TRUE,
            xlab          = "Ontogeny",
            ylab          = "TAI",
            type          = "l",
            lwd           = 9)

# testing for early conservation model
PlotPattern( ExpressionSet = PhyloExpressionSetExample,
             TestStatistic = "EarlyConservationTest",
            modules        = list(early = 1:2,mid = 3:5,late = 6:7),
            permutations   = 100,
            p.value        = TRUE,
            shaded.area    = TRUE,
            xlab           = "Ontogeny",
            ylab           = "TAI",
            type           = "l",
            lwd            = 9)


# use your own permutation matrix
custom_perm_matrix <- bootMatrix(PhyloExpressionSetExample,100)

PlotPattern(ExpressionSet      = PhyloExpressionSetExample,
            TestStatistic      = "FlatLineTest",
            custom.perm.matrix = custom_perm_matrix,
            type               = "l",
            xlab               = "Ontogeny",
            ylab               = "TAI",
            lwd                = 9)
```

PlotRE *Plot Relative Expression Levels*

Description

This function computes for each age category the corresponding relative expression profile.

For each age category the corresponding relative expression profile is being computed as follows:

$$f_js = (e_js - e_jmin)/(e_jmax - e_jmin)$$

where e_jmin and e_jmax denote the minimum/maximum mean expression level of phylostratum j over developmental stages s. This linear transformation corresponds to a shift by e_jmin and a subsequent shrinkage by $e_jmax - e_jmin$. As a result, the relative expression level f_js of developmental stage s with minimum e_js is 0, the relative expression level f_js of the developmental stage s with maximum e_js is 1, and the relative expression levels f_js of all other stages s range between 0 and 1, accordingly.

Usage

```
PlotRE(
  ExpressionSet,
  Groups = NULL,
  modules = NULL,
  legendName = "age",
  xlab = "Ontogeny",
  ylab = "Relative Expression Level",
  main = "",
  y.ticks = 10,
  adjust.range = TRUE,
  alpha = 0.008,
  ...
)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.
`Groups`	a list containing the age categories for which mean expression levels shall be drawn. For ex. evolutionary users can compare old phylostrata: PS1-3 (Class 1) and evolutionary young phylostrata: PS4-12 (Class 2). In this example, the list could be assigned as, `Groups = list(c(1:3), c(4:12))`. The group options is limited to 2 Groups.
`modules`	a list storing three elements for specifying the modules: early, mid, and late. Each element expects a numeric vector specifying the developmental stages or experiments that correspond to each module. For example, `module = list(early = 1:2, mid = 3:5, late = 6:7)` devides a dataset storing seven developmental stages into 3 modules. Default is `modules = NULL`. But if specified, a shaded are will be drawn to illustrate stages corresponding to the mid module.
`legendName`	a character string specifying the legend title.
`xlab`	label of x-axis.
`ylab`	label of y-axis.

`main`	main text.
`y.ticks`	number of ticks that shall be drawn on the y-axis.
`adjust.range`	logical indicating whether or not the y-axis scale shall be adjusted to the same range in case two groups are specified. Default is `adjust.range = TRUE`.
`alpha`	transparency of the shaded area (between [0,1]). Default is `alpha = 0.1`.
`...`	place holder for old version of PlotRE that was based on base graphics instead of ggplot2.

Details

Studying the relative expression profiles of each phylostratum or divergence-stratum enables the detection of common gene expression patterns shared by several phylostrata or divergence-strata.

Finding similar relative expression profiles among phylostrata or divergence-strata suggests that phylostrata or divergence-strata sharing a similar relative expression profile are regulated by similar gene regulatory elements. Hence, these common phylostrata or divergence-strata might govern similar processes in the given developmental time course.

Value

a plot showing the relative expression profiles of phylostrata or divergence-strata belonging to the same group.

Author(s)

Hajk-Georg Drost

References

Domazet-Loso T and Tautz D. 2010. "A phylogenetically based transcriptome age index mirrors ontogenetic divergence patterns". Nature (468): 815-818.

Quint M et al. 2012. "A transcriptomic hourglass in plant embryogenesis". Nature (490): 98-101.

See Also

PlotBarRE, RE, REMatrix

Examples

```
# read standard phylotranscriptomics data
data(PhyloExpressionSetExample)
data(DivergenceExpressionSetExample)

# example PhyloExpressionSet
PlotRE(PhyloExpressionSetExample,
       Groups = list(c(1:3), c(4:12)),
       legendName = "PS")


# or you can choose any combination of groups
```

```
PlotRE(PhyloExpressionSetExample,
       Groups = list(c(1,7,9), c(2:6,8,10:12)),
       legendName = "PS")


# example DivergenceExpressionSet
PlotRE(DivergenceExpressionSetExample,
       Groups = list(c(1:5), c(6:10)),
       legendName = "DS")
```

`PlotReplicateQuality` *Plot the Quality of Biological Replicates*

Description

This function performs several quality checks to validate the biological variation between replicates and stages (experiments).

Usage

```
PlotReplicateQuality(
  ExpressionSet,
  nrep,
  FUN = function(x) log(stats::var(x)),
  legend.pos = "topleft",
  stage.names = NULL,
  ...
)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.
`nrep`	either a numeric value specifying the constant number of replicates per stage or a numeric vector specifying the variable number of replicates for each stage position.
`FUN`	a function that should be applied to quantify the variablity or quality of replicates. The default function is the log(var(x)) quantifying the variance between replicates.
`legend.pos`	the position of the legend, e.g. 'topleft', or 'topright' (see `legend`).
`stage.names`	a character vector specifying the stage names.
`...`	additional graphics parameters.

Details

The following quality checks can be performed:

- Quantification of variability between replicates as density function.

Author(s)

Hajk-Georg Drost

Examples

```
data(PhyloExpressionSetExample)

# visualize log(var(x)) between replicates for each gene and developmental stage
PlotReplicateQuality(ExpressionSet = PhyloExpressionSetExample[1:5000 , 1:8],
                     nrep          = 2,
                     legend.pos   = "topright",
                     ylim          = c(0,0.2),
                     lwd           = 6)
```

`PlotSelectedAgeDistr` *Plot the PS or DS distribution of a selected set of genes*

Description

This function visualizes the PS or DS distribution of a selected set of genes as histogram.

Usage

```
PlotSelectedAgeDistr(
  ExpressionSet,
  gene.set,
  legendName = NULL,
  as.ratio = FALSE,
  use.only.map = FALSE,
  col = "turquoise4",
  xlab = NULL,
  ylab = NULL
)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.
`gene.set`	a character vector storing the gene ids for which gene expression profiles shall be visualized.
`legendName`	a character string specifying whether "PS" or "DS" are are visualized.

`as.ratio`	logical value indicating whether or not relative frequencies shall be visualized.
`use.only.map`	logical value indicating whether or not a Phylostratigraphic Map or Divergence Map should be passed to the `ExpressionSet` argument instead of a standard `ExpressionSet` object.
`col`	colour of the bars.
`xlab`	label of the x-axis.
`ylab`	label of the y-axis.

Author(s)

Hajk-Georg Drost

See Also

`PlotDistribution`

Examples

```
data(PhyloExpressionSetExample)

# generate an example gene set
set.seed(123)
ExGeneSet <- sample(PhyloExpressionSetExample[ , 2], 5000)

# gene count example
PlotSelectedAgeDistr(ExpressionSet = PhyloExpressionSetExample,
                     gene.set      = ExGeneSet,
                     legendName    = "PS",
                     as.ratio      = TRUE)

# relative gene count example
PlotSelectedAgeDistr(ExpressionSet = PhyloExpressionSetExample,
                     gene.set      = ExGeneSet,
                     legendName    = "PS",
                     as.ratio      = FALSE)
```

`PlotSignature` *Plot evolutionary signatures across transcriptomes*

Description

Main function to visualize transcriptome indices.

Usage

```
PlotSignature(
  ExpressionSet,
  measure = "TAI",
  TestStatistic = "FlatLineTest",
  modules = NULL,
  permutations = 1000,
  lillie.test = FALSE,
  p.value = TRUE,
  shaded.area = FALSE,
  custom.perm.matrix = NULL,
  xlab = "Ontogeny",
  ylab = "Transcriptome Index",
  main = "",
  lwd = 4,
  alpha = 0.1,
  y.ticks = 10
)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet, DivergenceExpressionSet or PolymorphismsExpressionSet object.
`measure`	type of transcriptome index that shall be computed. E.g. • `measure = "TAI"` (Transcriptome Age Index) • `measure = "TDI"` (Transcriptome Divergence Index) • `measure = "TPI"` (Transcriptome Polymorphism Index)
`TestStatistic`	a string defining the type of test statistics to be used to quantify the statistical significance the present phylotranscriptomics pattern. Possible values can be: • `TestStatistic = "FlatLineTest"` : Statistical test for the deviation from a flat line • `TestStatistic = "ReductiveHourglassTest"` : Statistical test for the existence of a hourglass shape (high-low-high pattern) • `TestStatistic = "EarlyConservationTest"` : Statistical test for the existence of a early conservation pattern (low-high-high pattern) • `TestStatistic = "LateConservationTest"` : Statistical test for the existence of a late conservation pattern (high-high-low pattern) • `TestStatistic = "ReverseHourglassTest"` : Statistical test for the existence of a reverse hourglass pattern (low-high-low pattern)
`modules`	a list storing three elements for the `ReductiveHourglassTest`, `EarlyConservationTest`, `LateConservationTest`, or `ReverseHourglassTest`: early, mid, and late. Each element expects a numeric vector specifying the developmental stages or experiments that correspond to each module. For example: • `module = list(early = 1:2, mid = 3:5, late = 6:7)` divides a dataset storing seven developmental stages into 3 modules.

`permutations`	a numeric value specifying the number of permutations to be performed for the `FlatLineTest`, `EarlyConservationTest`, `LateConservationTest`, `ReductiveHourglassTest` or `ReverseHourglassTest`.
`lillie.test`	a boolean value specifying whether the Lilliefors Kolmogorov-Smirnov Test shall be performed.
`p.value`	a boolean value specifying whether the p-value of the test statistic shall be printed as a subtitle.
`shaded.area`	a boolean value specifying whether a shaded area shall be drawn for the developmental stages defined to be the presumptive phylotypic period.
`custom.perm.matrix`	a custom `bootMatrix` (permutation matrix) to perform the underlying test statistic visualized by `PlotSignature`. Default is `custom.perm.matrix = NULL`.
`xlab`	label of x-axis.
`ylab`	label of y-axis.
`main`	figure title.
`lwd`	line width.
`alpha`	transparency of the shaded area (between [0,1]). Default is `alpha = 0.1`.
`y.ticks`	number of ticks on the y-axis. Default is `ticks = 10`.

Details

This function substitutes the functionality of the `PlotPattern` function and is based on ggplot2 insead of base R graphics.

The following transcriptome indices can be computed and visualized with this function:

- Transcriptome Age Index (`TAI`)
- Transcriptome Divergence Index (`TDI`)
- Transcriptome Polymorphism Index (`TPI`)

Author(s)

Hajk-Georg Drost

Examples

```
data(PhyloExpressionSetExample)

# plot TAI pattern and perform flat line test
PlotSignature(PhyloExpressionSetExample,
              measure      = "TAI",
              permutations = 100,
              TestStatistic = "FlatLineTest",
              ylab = "Transcriptome Age Index")
```

`PlotSignatureTransformed` *Plot evolutionary signatures across transcriptomes and RNA-seq transformations*

Description

PlotSignatureTransformed aims to statistically evaluate the stability of `ReductiveHourglassTest`, `FlatLineTest`, `ReverseHourglassTest`, `EarlyConservationTest`, or `LateConservationTest` (all based on `TAI` or `TDI` computations) against different data transformations AND plot the resulting TAI profiles using `PlotSignature`. The corresponding p-value quantifies the probability that a given TAI or TDI pattern (or any phylotranscriptomics pattern) does not support the chosen test. A p-value < 0.05 indicates that the corresponding phylotranscriptomics pattern does indeed support the chosen test.

Usage

```
PlotSignatureTransformed(
  ExpressionSet,
  measure = "TAI",
  TestStatistic = "FlatLineTest",
  transforms = c("none", "sqrt", "log2", "rank", "squared"),
  modules = NULL,
  permutations = 1000,
  pseudocount = 1,
  p.value = TRUE,
  shaded.area = FALSE,
  xlab = "Ontogeny",
  ylab = "Transcriptome Index",
  main = "",
  lwd = 4,
  alpha = 0.1,
  y.ticks = 3
)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.
`measure`	type of transcriptome index that shall be computed. E.g. • `measure = "TAI"` (Transcriptome Age Index) • `measure = "TDI"` (Transcriptome Divergence Index) • `measure = "TPI"` (Transcriptome Polymorphism Index)
`TestStatistic`	a string defining the type of test statistics to be used to quantify the statistical significance the present phylotranscriptomics pattern. Possible values can be: • `TestStatistic = "FlatLineTest"` : Statistical test for the deviation from a flat line

- `TestStatistic = "ReductiveHourglassTest"` : Statistical test for the existence of a hourglass shape (high-low-high pattern)
- `TestStatistic = "ReverseHourglassTest"` : Statistical test for the existence of a reverse hourglass pattern (low-high-low pattern)
- `TestStatistic = "EarlyConservationTest"` : Statistical test for the existence of a early conservation pattern (low-high-high pattern)
- `TestStatistic = "LateConservationTest"` : Statistical test for the existence of a late conservation pattern (high-high-low pattern)

`transforms`	a character vector of any valid function that transforms gene expression levels. Available options are: • `transforms = "none"` : No transformation (absolute expression) • `transforms = "log2"` : Computes binary (i.e., base 2) logarithms, $log_2 X$. • `transforms = "log"` : Computes natural logarithms, $log_e X$. • `transforms = "log10"` : Computes common logarithms (i.e., base 10), $log_{10} X$. • `transforms = "sqrt"` : Computes the (principle) square root, $\sqrt{X}$. • `transforms = "vst"` : (Quickly) estimates dispersion trend and applies a variance stabilizing transformation (please make sure that the **DESeq2** package is installed). • `transforms = "rlog"` : (Robustly) estimates dispersion trend and applies a variance stabilizing transformation (please make sure that the **DESeq2** package is installed). • `transforms = "rank"` : Ranks genes from lowest to highest based on their expression levels, at each condition (e.g., developmental stage). The gene's expression value is replaced by its sample rank or average ranks in case of ties. • `transforms = "squared"`: Computes the square, X^2.
`modules`	a list storing three elements for the `ReductiveHourglassTest`, `EarlyConservationTest`, `LateConservationTest`, or `ReverseHourglassTest`: early, mid, and late. Each element expects a numeric vector specifying the developmental stages or experiments that correspond to each module. For example: • `module = list(early = 1:2, mid = 3:5, late = 6:7)` divides a dataset storing seven developmental stages into 3 modules.
`permutations`	a numeric value specifying the number of permutations to be performed for the `FlatLineTest`, `EarlyConservationTest`, `LateConservationTest`, `ReductiveHourglassTest` or `ReverseHourglassTest`.
`pseudocount`	any valid number to add to the expression matrix prior to log transformations.
`p.value`	a boolean value specifying whether the p-value of the test statistic shall be printed within the plot area.
`shaded.area`	a boolean value specifying whether a shaded area shall be drawn for the developmental stages defined to be the presumptive phylotypic period.
`xlab`	label of x-axis.
`ylab`	label of y-axis.
`main`	figure title.

`lwd`	line width.
`alpha`	transparency of the shaded area (between [0,1]). Default is `alpha = 0.1`.
`y.ticks`	number of ticks on the y-axis. Default is `ticks = 3`.

Details

Visualisation and assessment for the stability of data transforms on the permutation test of choice. For details, please consult the main function `PlotSignature`, as well as `tf`, `ReductiveHourglassTest`, `FlatLineTest`, `ReverseHourglassTest`, `LateConservationTest` or `EarlyConservationTest`.

In most cases, users can replace `PlotSignature` simply with `PlotSignatureTransformed` to obtain the multi-panel plot with different transformations to visualise the stability of the pattern observed with `PlotSignature`.

Value

a ggplot object containing the following information for each visualised transcriptome indices: `p.value` : the p-value quantifying the statistical significance (depending on the chosen test) of the given phylotranscriptomics pattern.

Author(s)

Jaruwatana Sodai Lotharukpong

References

Lotharukpong JS et al. (2023) (unpublished)

See Also

`PlotSignature`,`tfStability`, `FlatLineTest`, `ReverseHourglassTest`, `EarlyConservationTest`, `ReductiveHourglassTest`, `LateConservationTest`

Examples

```
## Not run:
data(PhyloExpressionSetExample)

# Flat line test
PlotSignatureTransformed(ExpressionSet = PhyloExpressionSetExample,
                    TestStatistic = "FlatLineTest",
                    transforms = c("none", "log2", "sqrt", "rank", "squared"))

# Reductive hourglass test
PlotSignatureTransformed(ExpressionSet = PhyloExpressionSetExample,
                     TestStatistic = "ReductiveHourglassTest",
                     transforms = c("none", "log2", "sqrt", "rank", "squared"),
                     modules = list(early = 1:2, mid = 3:5, late = 6:7))

library(DESeq2)
PlotSignatureTransformed(ExpressionSet = PhyloExpressionSetExample,
                     TestStatistic = "ReductiveHourglassTest",
                     transforms = c("none", "log2", "sqrt", "vst", "rank", "squared"),
                     modules = list(early = 1:2, mid = 3:5, late = 6:7))

## End(Not run)
```

PlotVars *Plot Variance of Expression Profiles*

Description

This function computes for each age category the corresponding variance expression profile.

Usage

```
PlotVars(
  ExpressionSet,
  Groups = NULL,
  legendName = "age",
  xlab = "Ontogeny",
  ylab = "Variance(Expression Level)",
  main = "",
  y.ticks = 10,
  adjust.range = TRUE
)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.
`Groups`	a list containing the age categories for which variance expression levels shall be drawn. For ex. evolutionary users can compare old phylostrata: PS1-3 (Class 1) and evolutionary young phylostrata: PS4-12 (Class 2). In this example, the list could be assigned as, `Groups = list(c(1:3), c(4:12))`. The group options is limited to 2 Groups.
`legendName`	a character string specifying the legend title.
`xlab`	label of x-axis.
`ylab`	label of y-axis.
`main`	main text.
`y.ticks`	number of ticks that shall be drawn on the y-axis.
`adjust.range`	logical indicating whether or not the y-axis scale shall be adjusted to the same range in case two groups are specified. Default is `adjust.range = TRUE`.

Details

This plot may be useful to compare the absolute variance expression levels of each age category across stages.

In different developmental processes different phylostratum or divergence-stratum classes might be more expressed than others, hence contributing more to the overall phylotranscriptomics pattern (`TAI` or `TDI`). This plot can help to identify the phylostratum or divergence-stratum classes that contributes most to the overall transcriptome of the given developmental process.

Value

a plot showing variance expression profiles of each age category.

Author(s)

Hajk-Georg Drost

See Also

`PlotBarRE`, `RE`, `REMatrix`, `PlotRE`

Examples

```
### Example using a PhyloExpressionSet
### and DivergenceExpressionSet
# load PhyloExpressionSet
data(PhyloExpressionSetExample)

# load PhyloExpressionSet
data(DivergenceExpressionSetExample)

# plot evolutionary old PS (PS1-3) vs evolutionary young PS (PS4-12)
PlotVars(PhyloExpressionSetExample,
          Groups = list(c(1:3), c(4:12)),
          legendName = "PS",
          adjust.range = TRUE)

# if users wish to not adjust the y-axis scale when
# 2 groups are selected they can specify: adjust.range = FALSE
PlotVars(PhyloExpressionSetExample,
          Groups = list(c(1:3), c(4:12)),
          legendName = "PS",
          adjust.range = FALSE)


# plot conserved DS (DS1-5) vs divergent DS (PS6-10)
# NOTE: DS are always defined in the range 1, 2, ... , 10.
# Hence, make sure that your groups are within this range!
PlotVars(DivergenceExpressionSetExample,
          Groups = list(c(1:5), c(6:10)),
          legendName = "DS",
          adjust.range = TRUE)
```

pMatrix *Compute Partial TAI or TDI Values*

Description

This function computes the partial TAI or TDI values for each single gene in a PhyloExpressionSet or DivergenceExpressionSet object.

In detail, each gene gets a *TAI contribution profile* or *TDI contribution profile*.

$$TAI_{is} = f_{is} * ps_i$$

or

$$TDI_{is} = f_{is} * ps_i$$

where TAI_{is} or TDI_{is} is the partial TAI or TDI value of gene i, $f_{is} = e_{is} / \sum e_{is}$ and ps_i is the phylostratum or divergence-stratum of gene i.

Usage

```
pMatrix(ExpressionSet)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.

Details

The partial TAI or TDI matrix can be used to perform different cluster analyses and also gives an overall impression of the contribution of each gene to the global TAI or TDI pattern.

Value

a numeric matrix storing the partial TAI or TDI values for each gene in the corresponding PhyloExpressionSet or DivergenceExpressionSet.

Author(s)

Hajk-Georg Drost

References

Domazet-Loso T and Tautz D. 2010. "A phylogenetically based transcriptome age index mirrors ontogenetic divergence patterns". Nature (468): 815-818.

Examples

```
# read standard phylotranscriptomics data
data(PhyloExpressionSetExample)
data(DivergenceExpressionSetExample)

# example PhyloExpressionSet
PTM_ps <- pMatrix(PhyloExpressionSetExample)

# example DivergenceExpressionSet
PTM_ds <- pMatrix(DivergenceExpressionSetExample)

# boxplot of the pMatrix
boxplot(pMatrix(PhyloExpressionSetExample),outline = FALSE)

# boxplot of the pMatrix using log2 transformed expression levels
boxplot(pMatrix(tf(PhyloExpressionSetExample,log2)))
```

`pMatrixTEI` *Compute Partial Transcriptome Evolutionary Index (TEI) Values*

Description

This function computes the partial transcriptome evolutionary index (TEI) values for each single gene.

In detail, each gene gets a *TEI contribution profile*.

$$TEI_{is} = f_{is} * ps_i$$

where TEI_{is} is the partial TEI value of gene i, $f_{is} = e_{is} / \sum e_{is}$ and ps_i is the phylostratum of gene i.

Usage

```
pMatrixTEI(
  ExpressionSet,
  Phylostratum = NULL,
  split = 1e+05,
  showprogress = TRUE,
  threads = 1
)
```

Arguments

`ExpressionSet`	expression object with rownames as GeneID (dgCMatrix) or standard PhyloExpressionSet object.
`Phylostratum`	a named vector representing phylostratum per GeneID with names as GeneID (not used if Expression is PhyloExpressionSet).
`split`	specify number of columns to split
`showprogress`	boolean if progressbar should be shown
`threads`	specify number of threads

Details

The partial TEI matrix can be used to perform different cluster analyses and also gives an overall impression of the contribution of each gene to the global `TEI` pattern.

Value

a numeric sparse matrix storing the partial TEI values for each gene.

Author(s)

Kristian K Ullrich

References

Domazet-Loso T. and Tautz D. (2010). *A phylogenetically based transcriptome age index mirrors ontogenetic divergence patterns*. Nature (468): 815-818.

Quint M et al. (2012). *A transcriptomic hourglass in plant embryogenesis*. Nature (490): 98-101.

Drost HG et al. (2015) Mol Biol Evol. 32 (5): 1221-1231 doi:10.1093/molbev/msv012

Examples

```
# reading a standard PhyloExpressionSet
data(PhyloExpressionSetExample, package = "myTAI")

# computing partial TEI contribution per gene
pMT <- pMatrixTEI(PhyloExpressionSetExample)
```

pStrata *Compute Partial Strata Values*

Description

This function computes the partial TAI or TDI values for all Phylostrata or Divergence Strata.

Usage

```
pStrata(ExpressionSet)
```

Arguments

ExpressionSet	a standard PhyloExpressionSet or DivergenceExpressionSet object.

Author(s)

Hajk-Georg Drost

Examples

```
data(PhyloExpressionSetExample)

# compute partial TAI values for each Phylostratum
partialStrata <- pStrata(PhyloExpressionSetExample)

# show that colSums of pStrata is equavalent to the TAI values
all.equal(colSums(partialStrata),TAI(PhyloExpressionSetExample))

# show that colSums of pStrata is equavalent to colSums of pMatrix(PhyloExpressionSetExample)
all.equal(colSums(partialStrata),colSums(pMatrix(PhyloExpressionSetExample)))
```

pStrataTEI *Compute Partial Transcriptome Evolutionary Index (TEI) Strata Values*

Description

This function computes the partial transcriptome evolutionary index (TEI) values combined for each strata.

In detail, each gene gets a *TEI contribution profile*.

$$TEI_{is} = f_{is} * ps_i$$

where TEI_{is} is the partial TEI value of gene i, $f_{is} = e_{is}/\sum e_{is}$ and ps_i is the phylostratum of gene i.

TEI_{is} values are combined per ps.

Usage

```
pStrataTEI(
  ExpressionSet,
  Phylostratum = NULL,
  split = 1e+05,
  showprogress = TRUE,
  threads = 1
)
```

Arguments

`ExpressionSet`	expression object with rownames as GeneID (dgCMatrix) or standard PhyloExpressionSet object.
`Phylostratum`	a named vector representing phylostratum per GeneID with names as GeneID (not used if Expression is PhyloExpressionSet).
`split`	specify number of columns to split
`showprogress`	boolean if progressbar should be shown
`threads`	specify number of threads

Details

The partial TEI values combined per strata give an overall impression of the contribution of each strata to the global `TEI` pattern.

Value

a numeric matrix storing the summed partial TEI values for each strata.

Author(s)

Kristian K Ullrich

References

Domazet-Loso T. and Tautz D. (2010). *A phylogenetically based transcriptome age index mirrors ontogenetic divergence patterns*. Nature (468): 815-818.

Quint M et al. (2012). *A transcriptomic hourglass in plant embryogenesis*. Nature (490): 98-101.

Drost HG et al. (2015) Mol Biol Evol. 32 (5): 1221-1231 doi:10.1093/molbev/msv012

Examples

```
# reading a standard PhyloExpressionSet
data(PhyloExpressionSetExample, package = "myTAI")

# computing partial TEI contribution per gene
pS <- pStrataTEI(PhyloExpressionSetExample)
```

`pTAI` *Compute the Phylostratum Contribution to the Global Transcriptome Age Index*

Description

This function takes a standard *ExpressionSet* object and computes the partial contribution of the different phylostrata (ps) to the global Transcriptome Age Index profile.

Usage

```
pTAI(ExpressionSet)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet object.

Details

This way of computing the partial contribution of the different phylostrata (ps) to the global Transcriptome Age Index profile was introduced by Domazet-Loso and Tautz, 2010. This function (`pTAI`) computes the partial TAI contribution for each phylostratum and each developmental stage and returns a data matrix storing the partial TAI contribution value for each phylostratum and each developmental stage.

Author(s)

Hajk-Georg Drost

References

Domazet-Loso T. and Tautz D. (2010). *A phylogenetically based transcriptome age index mirrors ontogenetic divergence patterns*. Nature (468): 815-818.

See Also

`pTDI`, `TAI`, `TDI`, `PlotPattern`

Examples

```
data(PhyloExpressionSetExample)

# get the partial contribution of phylostrata to the global
# TAI pattern
pTAI(PhyloExpressionSetExample)
```

pTDI — *Compute the Divergence Stratum Contribution to the Global Transcriptome Divergence Index*

Description

This function takes a standard *ExpressionSet* object and computes the partial contribution of the different divergence strata (ds) to the global Transcriptome Divergence Index profile.

Usage

```
pTDI(ExpressionSet)
```

Arguments

`ExpressionSet`	a standard DivergenceExpressionSet object.

Details

This function (`pTDI`) computes the partial TDI contribution for each phylostratum and each developmental stage and returns a data matrix storing the partial TDI contribution value for each divergence and each developmental stage.

Author(s)

Hajk-Georg Drost

References

Domazet-Loso T. and Tautz D. (2010). *A phylogenetically based transcriptome age index mirrors ontogenetic divergence patterns*. Nature (468): 815-818.

Quint M et al. (2012). A transcriptomic hourglass in plant embryogenesis. Nature (490): 98-101.

Drost HG et al. (2015). Evidence for Active Maintenance of Phylotranscriptomic Hourglass Patterns in Animal and Plant Embryogenesis. Mol Biol Evol. 32 (5): 1221-1231 doi:10.1093/molbev/msv012.

See Also

`pTAI`, `TAI`, `TDI`, `PlotPattern`

Examples

```
data(DivergenceExpressionSetExample)

# get the partial contribution of divergence strata to the global
# TDI pattern
pTAI(DivergenceExpressionSetExample)
```

`rcpp_boottei_parallel` *rcpp_boottei_parallel*

Description

computes the phylogenetically based transcriptome evolutionary index (TEI) shuffling the strata for permutation statistic

Usage

```
rcpp_boottei_parallel(expression, ps, permutations, ncores = 1L)
```

Arguments

`expression`	ExpressionSet as sparseMatrix
`ps`	named Phylostratum
`permutations`	number of permutations
`ncores`	number of cores

Value

sparseMatrix

Author(s)

Kristian K Ullrich

Examples

```
## load example PhyloExpressionSetExample

data("PhyloExpressionSetExample", package="myTAI")

## convert into sparseMatrix - rownames GeneID

spmat <- as(data.matrix(PhyloExpressionSetExample[,-c(1,2)]),
    "sparseMatrix")
rownames(spmat) <- PhyloExpressionSetExample$GeneID

## create named Phylostratum vector

ps <- setNames(PhyloExpressionSetExample$Phylostratum,
    PhyloExpressionSetExample$GeneID)

## get permutations
rcpp_boottei_parallel(spmat, ps, 100, 1)
```

rcpp_pMatrix_parallel *rcpp_pMatrix_parallel*

Description

computes the partial transcriptome evolutionary index (TEI) values for each single gene

Usage

```
rcpp_pMatrix_parallel(expression, ps, ncores = 1L)
```

Arguments

`expression`	ExpressionSet as sparseMatrix
`ps`	named Phylostratum
`ncores`	number of cores

Value

sparseMatrix

Author(s)

Kristian K Ullrich

Examples

```
## load example PhyloExpressionSetExample

data("PhyloExpressionSetExample", package="myTAI")

## convert into sparseMatrix - rownames GeneID

spmat <- as(data.matrix(PhyloExpressionSetExample[,-c(1,2)]),
    "sparseMatrix")
rownames(spmat) <- PhyloExpressionSetExample$GeneID

## create named Phylostratum vector

ps <- setNames(PhyloExpressionSetExample$Phylostratum,
    PhyloExpressionSetExample$GeneID)

## get pMatrix
rcpp_pMatrix_parallel(spmat, ps)
```

`rcpp_pStrata_parallel` *rcpp_pStrata_parallel*

Description

computes the partial transcriptome evolutionary index (TEI) values combined into strata

Usage

```
rcpp_pStrata_parallel(expression, ps, psgroup, ncores = 1L)
```

Arguments

`expression`	ExpressionSet as sparseMatrix
`ps`	named Phylostratum
`psgroup`	ordered unique Phylostratum
`ncores`	number of cores

Value

sparseMatrix

Author(s)

Kristian K Ullrich

Examples

```
## load example PhyloExpressionSetExample

data("PhyloExpressionSetExample", package="myTAI")

## convert into sparseMatrix - rownames GeneID

spmat <- as(data.matrix(PhyloExpressionSetExample[,-c(1,2)]),
    "sparseMatrix")
rownames(spmat) <- PhyloExpressionSetExample$GeneID

## create named Phylostratum vector

ps <- setNames(PhyloExpressionSetExample$Phylostratum,
    PhyloExpressionSetExample$GeneID)
psgroup <- sort(unique(ps))

## get pStrata
rcpp_pStrata_parallel(spmat, ps, psgroup)
```

rcpp_tei_parallel *rcpp_tei_parallel*

Description

computes the phylogenetically based transcriptome evolutionary index (TEI)

Usage

```
rcpp_tei_parallel(expression, ps, ncores = 1L)
```

Arguments

expression	ExpressionSet as sparseMatrix
ps	named Phylostratum
ncores	number of cores

Value

list

Author(s)

Kristian K Ullrich

Examples

```
## load example sequence data
data("PhyloExpressionSetExample", package="myTAI")
spmat <- as(data.matrix(PhyloExpressionSetExample[,-c(1,2)]), "sparseMatrix")
rownames(spmat) <- PhyloExpressionSetExample$GeneID
ps <- setNames(PhyloExpressionSetExample$Phylostratum, PhyloExpressionSetExample$GeneID)
rcpp_tei_parallel(spmat, ps)
```

RE *Transform to Relative Expression Levels*

Description

This function computes the relative expression profiles of any given gene expression set. The relative expression profile is being computed as follows:

$$f_s = (e_s - e_min)/(e_max - e_min)$$

where e_min and e_max denote the minimum/maximum mean expression level over the developmental stages s. This linear transformation corresponds to a shift by e_min and a subsequent shrinkage by $e_max - e_min$. As a result, the relative expression level f_s of developmental stage s with minimum e_s is 0, the relative expression level f_s of the developmental stage s with maximum e_s is 1, and the relative expression levels f_s of all other stages s range between 0 and 1, accordingly.

Usage

```
RE(ExpressionMatrix)
```

Arguments

`ExpressionMatrix`	a numeric matrix representing a gene expression matrix for which the relative expression profile shall be computed.

Value

a vector containing the relative expression profile of the correspnding data matrix.

Author(s)

Hajk-Georg Drost

References

Domazet-Loso T and Tautz D. (2010). *A phylogenetically based transcriptome age index mirrors ontogenetic divergence patterns*. Nature (468): 815-818.

Quint M et al. (2012). *A transcriptomic hourglass in plant embryogenesis*. Nature (490): 98-101.

See Also

`REMatrix`, `PlotRE`

Examples

```
# read standard phylotranscriptomics data
data(PhyloExpressionSetExample)

# relative expression profile of PS1 genes
RE(PhyloExpressionSetExample[ which(PhyloExpressionSetExample[ , 1] == 1), 3:9 ])
```

ReductiveHourglassTest *Perform the Reductive Hourglass Test*

Description

The *Reductive Hourglass Test* aims to statistically evaluate the existence of a phylotranscriptomic hourglass pattern based on `TAI` or `TDI` computations. The corresponding p-value quantifies the probability that a given TAI or TDI pattern (or any phylotranscriptomics pattern) does not follow an hourglass like shape. A p-value < 0.05 indicates that the corresponding phylotranscriptomics pattern does indeed follow an hourglass (high-low-high) shape.

Usage

```
ReductiveHourglassTest(
  ExpressionSet,
  modules = NULL,
  permutations = 1000,
  lillie.test = FALSE,
  plotHistogram = FALSE,
  runs = 10,
  parallel = FALSE,
  gof.warning = FALSE,
  custom.perm.matrix = NULL
)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.
`modules`	a list storing three elements: early, mid, and late. Each element expects a numeric vector specifying the developmental stages or experiments that correspond to each module. For example, `module` = list(early = 1:2, mid = 3:5, late = 6:7) devides a dataset storing seven developmental stages into 3 modules.
`permutations`	a numeric value specifying the number of permutations to be performed for the `ReductiveHourglassTest`.
`lillie.test`	a boolean value specifying whether the Lilliefors Kolmogorov-Smirnov Test shall be performed to quantify the goodness of fit.
`plotHistogram`	a boolean value specifying whether a *Lillifor's Kolmogorov-Smirnov-Test* shall be performed to test the goodness of fit of the approximated distribution, as well as additional plots quantifying the significance of the observed phylotranscriptomic pattern.
`runs`	specify the number of runs to be performed for goodness of fit computations, in case `plotHistogram` = `TRUE`. In most cases `runs` = 100 is a reasonable choice. Default is `runs` = 10 (because it takes less computation time for demonstration purposes).

`parallel`	performing `runs` in parallel (takes all cores of your multicore machine).
`gof.warning`	a logical value indicating whether non significant goodness of fit results should be printed as warning. Default is `gof.warning = FALSE`.
`custom.perm.matrix`	a custom `bootMatrix` (permutation matrix) to perform the underlying test statistic. Default is `custom.perm.matrix = NULL`.

Details

The reductive hourglass test is a permutation test based on the following test statistic.

(1) A set of developmental stages is partitioned into three modules - early, mid, and late - based on prior biological knowledge.

(2) The mean `TAI` or `TDI` value for each of the three modules T_early, T_mid, and T_late are computed.

(3) The two differences D1 = T_early - T_mid and D2 = T_late - T_mid are calculated.

(4) The minimum D_min of D1 and D2 is computed as final test statistic of the reductive hourglass test.

In order to determine the statistical significance of an observed minimum difference D_min the following permutation test was performed. Based on the `bootMatrix` D_min is calculated from each of the permuted `TAI` or `TDI` profiles, approximated by a Gaussian distribution with method of moments estimated parameters returned by `fitdist`, and the corresponding p-value is computed by `pnorm` given the estimated parameters of the Gaussian distribution. The *goodness of fit* for the random vector *D_min* is statistically quantified by an Lilliefors (Kolmogorov-Smirnov) test for normality.

In case the parameter *plotHistogram* = *TRUE*, a multi-plot is generated showing:

(1) A Cullen and Frey skewness-kurtosis plot generated by `descdist`. This plot illustrates which distributions seem plausible to fit the resulting permutation vector D_min. In the case of the *Reductive Hourglass Test* a normal distribution seemed plausible.

(2) A histogram of D_min combined with the density plot is plotted. D_min is then fitted by a normal distribution. The corresponding parameters are estimated by *moment matching estimation* using the `fitdist` function.

(3) A plot showing the p-values for N independent runs to verify that a specific p-value is biased by a specific permutation order.

(4) A barplot showing the number of cases in which the underlying goodness of fit (returned by Lilliefors (Kolmogorov-Smirnov) test for normality) has shown to be significant (`TRUE`) or not significant (`FALSE`). This allows to quantify the permutation bias and their implications on the goodness of fit.

Value

a list object containing the list elements:

`p.value` : the p-value quantifying the statistical significance (high-low-high pattern) of the given phylotranscriptomics pattern.

`std.dev` : the standard deviation of the N sampled phylotranscriptomics patterns for each developmental stage S.

`lillie.test` : a boolean value specifying whether the *Lillifors KS-Test* returned a p-value > 0.05, which indicates that fitting the permuted scores with a normal distribution seems plausible.

Author(s)

Hajk-Georg Drost

References

Drost HG et al. (2015) Mol Biol Evol. 32 (5): 1221-1231 doi:10.1093/molbev/msv012

Quint M et al. (2012). A transcriptomic hourglass in plant embryogenesis. Nature (490): 98-101.

M. L. Delignette-Muller, R. Pouillot, J.-B. Denis and C. Dutang (2014), fitdistrplus: help to fit of a parametric distribution to non-censored or censored data.

Cullen AC and Frey HC (1999) Probabilistic techniques in exposure assessment. Plenum Press, USA, pp. 81-159.

Evans M, Hastings N and Peacock B (2000) Statistical distributions. John Wiley and Sons Inc.

Sokal RR and Rohlf FJ (1995) Biometry. W.H. Freeman and Company, USA, pp. 111-115.

Juergen Gross and bug fixes by Uwe Ligges (2012). nortest: Tests for Normality. R package version 1.0-2.

http://CRAN.R-project.org/package=nortest

Dallal, G.E. and Wilkinson, L. (1986): An analytic approximation to the distribution of Lilliefors' test for normality. The American Statistician, 40, 294-296.

Stephens, M.A. (1974): EDF statistics for goodness of fit and some comparisons. Journal of the American Statistical Association, 69, 730-737.

http://stackoverflow.com/questions/4290081/fitting-data-to-distributions?rq=1

http://stats.stackexchange.com/questions/45033/can-i-use-kolmogorov-smirnov-test-and-estimate-distribution-parameters

http://cran.r-project.org/doc/contrib/Ricci-distributions-en.pdf

http://cran.r-project.org/doc/contrib/Ricci-distributions-en.pdf

See Also

`rhScore`, `bootMatrix`, `FlatLineTest`, `ReverseHourglassTest`, `EarlyConservationTest`, `PlotSignature`, `LateConservationTest`

Examples

```
data(PhyloExpressionSetExample)

# perform the reductive hourglass test for a PhyloExpressionSet
# here the prior biological knowledge is that stages 1-2 correspond to module 1 = early,
# stages 3-5 to module 2 = mid (phylotypic module), and stages 6-7 correspond to
# module 3 = late
ReductiveHourglassTest(PhyloExpressionSetExample,
                       modules = list(early = 1:2, mid = 3:5, late = 6:7),
                       permutations = 1000)
```

```
# use your own permutation matrix based on which p-values (ReductiveHourglassTest)
# shall be computed
custom_perm_matrix <- bootMatrix(PhyloExpressionSetExample,100)

ReductiveHourglassTest(PhyloExpressionSetExample,
                       modules = list(early = 1:2, mid = 3:5, late = 6:7),
                       custom.perm.matrix = custom_perm_matrix)
```

REMatrix *Compute a Relative Expression Matrix*

Description

This function computes the relative expression profiles of all given phylostrata or divergence-strata within a given PhyloExpressionSet or DivergenceExpressionSet.

Usage

```
REMatrix(ExpressionSet)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.

Details

For each phylostratum or divergence-stratum the corresponding relative expression profile is being computed as follows:

$$f_js = (e_js - e_jmin)/(e_jmax - e_jmin)$$

where e_jmin and e_jmax denote the minimum/maximum mean expression level of phylostratum j over the developmental stages s. This linear transformation corresponds to a shift by e_jmin and a subsequent shrinkage by $e_jmax - e_jmin$. As a result, the relative expression level f_js of developmental stage s with minimum e_js is 0, the relative expression level f_js of the developmental stage s with maximum e_js is 1, and the relative expression levels f_js of all other stages s range between 0 and 1, accordingly.

Author(s)

Hajk-Georg Drost

References

Domazet-Loso T and Tautz D. (2010). *A phylogenetically based transcriptome age index mirrors ontogenetic divergence patterns*. Nature (468): 815-818.

Quint M et al. (2012). *A transcriptomic hourglass in plant embryogenesis*. Nature (490): 98-101.

See Also

RE, PlotRE, PlotBarRE

Examples

```
# read standard phylotranscriptomics data
data(PhyloExpressionSetExample)
data(DivergenceExpressionSetExample)

# example PhyloExpressionSet
REMatrix(PhyloExpressionSetExample)

# example DivergenceExpressionSet
REMatrix(DivergenceExpressionSetExample)
```

---

reversehourglassScore *Compute the Reverse Hourglass Score for the Reverse Hourglass Test*

Description

This function reduces the destruction of an hourglass shaped pattern to a single score value.

Based on a given TAI or TDI pattern the given vector is being divided into three developmental modules: early, mid, and late. The corrisponding TAI or TDI values in each developmental module are accumulated using the *scoringMethod* argument ("max-min" or "mean-mean").

In more detail:

(1) for a given TAI or TDI vector *tai_profile* or *tdi_profile*, we classify each value of *tai_profile* or *tdi_profile* into its corresponding developmental module early, mid, or late.

(2) accumulate the *tai_profile* or *tdi_profile* values in each developmental module using the arithmetic mean (mean) in case scoringMethod = "mean-mean", or accumulate the *tai_profile* or *tdi_profile* values in each developmental module using max for the early and late module and min for the mid module in case scoringMethod = "max-min".

(3) then reduce the three values for each developmental module by computing the difference between: mid - early, and mid - late.

(4) the two difference values are referred to as a_early and a_late.

Each developmental module now has an accumulated representation value which is being reduced to one value using the *method* argument ("max", "min", or "mean").

Given the two accumulated values for each hourglass module: a_early and a_late, we reduce the two given values by:

- *"max"*: $S = maxa_early, a_late$
- *"min"*: $S = mina_early, a_late$
- *"mean"*: $S = meana_early, a_late$

All together this results in a global score *S*. This global score *S* is being returned by this function.

Usage

```
reversehourglassScore(
  age_vals,
  early,
  mid,
  late,
  method,
  scoringMethod,
  profile.warn = FALSE
)
```

Arguments

`age_vals`	a numeric vector containing `TAI` or `TDI` values for each developmental stage s.
`early`	a numeric vector including the numeric stage values that correspond to the early phase of development.
`mid`	a numeric vector including the numeric stage values that correspond to the middle phase of development.
`late`	a numeric vector including the numeric stage values that correspond to the late phase of development.
`method`	to determine the two value reduction value, resulting in the global score S: "max", or "min", or "mean".
`scoringMethod`	method to determine the module accumulation value: "max-min" or "mean-mean".
`profile.warn`	a boolean value indicating whether a warning is printed when the low-high-low pattern isn't followed.

Details

The gpScore is a heuristic score enabling to construct a test statistic to determine the significance of a present (phylotranscriptomic) hourglass pattern.

Value

a numeric value representing the hourglass destruction score.

Author(s)

Hajk-Georg Drost

References

Drost et al. (2015), Evidence for active maintenance of phylotranscriptomic hourglass patterns in animal and plant embryogenesis. Mol Bio Evol.

See Also

ReverseHourglassTest, TAI, TDI

Examples

```
# read standard phylotranscriptomics data
data(PhyloExpressionSetExample)
data(DivergenceExpressionSetExample)

# example PhyloExpressionSet:

# compute the TAI profile
TAIs <- TAI(PhyloExpressionSetExample)

# compute the global reverse hourglass destruction score
# for the TAIs profile using reduction method: mean(mean-mean)
reversehourglass_score <- reversehourglassScore(age_vals = TAIs,early = 1:2,mid = 3:5,late = 6:7,
                    method = "mean",scoringMethod = "mean-mean")


# example DivergenceExpressionSet:

# compute the TDI profile
TDIs <- TDI(DivergenceExpressionSetExample)

# compute the global reverse hourglass destruction score for the TDIs profile
# using reduction method: mean(mean-mean)
reversehourglass_score <- reversehourglassScore(age_vals = TDIs,early = 1:2,mid = 3:5,late = 6:7,
                    method = "mean",scoringMethod = "mean-mean")

# get warning if the expected pattern isn't followed
reversehourglass_score <- reversehourglassScore(age_vals = TAIs,early = 1:2,mid = 3:5,late = 6:7,
                    method = "mean",scoringMethod = "mean-mean",profile.warn=TRUE)
```

ReverseHourglassTest *Perform the Reverse Hourglass Test*

Description

The *Reverse Hourglass Test* aims to statistically evaluate the existence of a reverse hourglass pattern based on TAI or TDI computations. The corresponding p-value quantifies the probability that a given TAI or TDI pattern (or any phylotranscriptomics pattern) does follow an hourglass like shape. A p-value < 0.05 indicates that the corresponding phylotranscriptomics pattern does rather follow a reverse hourglass (low-high-low) shape.

Usage

```
ReverseHourglassTest(
  ExpressionSet,
  modules = NULL,
  permutations = 1000,
  lillie.test = FALSE,
  plotHistogram = FALSE,
  runs = 10,
  parallel = FALSE,
  gof.warning = FALSE,
  custom.perm.matrix = NULL
)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.
`modules`	a list storing three elements: early, mid, and late. Each element expects a numeric vector specifying the developmental stages or experiments that correspond to each module. For example, `module` = list(early = 1:2, mid = 3:5, late = 6:7) devides a dataset storing seven developmental stages into 3 modules.
`permutations`	a numeric value specifying the number of permutations to be performed for the `ReverseHourglassTest`.
`lillie.test`	a boolean value specifying whether the Lilliefors Kolmogorov-Smirnov Test shall be performed to quantify the goodness of fit.
`plotHistogram`	a boolean value specifying whether a *Lillifor's Kolmogorov-Smirnov-Test* shall be performed to test the goodness of fit of the approximated distribution, as well as additional plots quantifying the significance of the observed phylotranscriptomic pattern.
`runs`	specify the number of runs to be performed for goodness of fit computations, in case `plotHistogram` = `TRUE`. In most cases `runs` = 100 is a reasonable choice. Default is `runs` = 10 (because it takes less computation time for demonstration purposes).
`parallel`	performing `runs` in parallel (takes all cores of your multicore machine).
`gof.warning`	a logical value indicating whether non significant goodness of fit results should be printed as warning. Default is `gof.warning` = `FALSE`.
`custom.perm.matrix`	a custom `bootMatrix` (permutation matrix) to perform the underlying test statistic. Default is `custom.perm.matrix` = `NULL`.

Details

The reverse hourglass test is a permutation test based on the following test statistic.

(1) A set of developmental stages is partitioned into three modules - early, mid, and late - based on prior biological knowledge.

(2) The mean `TAI` or `TDI` value for each of the three modules T_early, T_mid, and T_late are computed.

(3) The two differences `D1 = T_mid - T_early` and `D2 = T_mid - T_late` are calculated.

(4) The minimum `D_min` of `D1` and `D2` is computed as final test statistic of the reductive hourglass test.

In order to determine the statistical significance of an observed minimum difference D_min the following permutation test was performed. Based on the `bootMatrix` D_min is calculated from each of the permuted `TAI` or `TDI` profiles, approximated by a Gaussian distribution with method of moments estimated parameters returned by `fitdist`, and the corresponding p-value is computed by `pnorm` given the estimated parameters of the Gaussian distribution. The *goodness of fit* for the random vector *D_min* is statistically quantified by an Lilliefors (Kolmogorov-Smirnov) test for normality.

In case the parameter *plotHistogram* = *TRUE*, a multi-plot is generated showing:

(1) A Cullen and Frey skewness-kurtosis plot generated by `descdist`. This plot illustrates which distributions seem plausible to fit the resulting permutation vector D_min. In the case of the *Reverse Hourglass Test* a normal distribution seemed plausible.

(2) A histogram of `D_min` combined with the density plot is plotted. D_min is then fitted by a normal distribution. The corresponding parameters are estimated by *moment matching estimation* using the `fitdist` function.

(3) A plot showing the p-values for N independent runs to verify that a specific p-value is biased by a specific permutation order.

(4) A barplot showing the number of cases in which the underlying goodness of fit (returned by Lilliefors (Kolmogorov-Smirnov) test for normality) has shown to be significant (`TRUE`) or not significant (`FALSE`). This allows to quantify the permutation bias and their implications on the goodness of fit.

Value

a list object containing the list elements:

`p.value` : the p-value quantifying the statistical significance (low-high-low pattern) of the given phylotranscriptomics pattern.

`std.dev` : the standard deviation of the N sampled phylotranscriptomics patterns for each developmental stage S.

`lillie.test` : a boolean value specifying whether the *Lillifors KS-Test* returned a p-value > 0.05, which indicates that fitting the permuted scores with a normal distribution seems plausible.

Author(s)

Hajk-Georg Drost

References

Drost HG et al. (2015) Mol Biol Evol. 32 (5): 1221-1231 doi:10.1093/molbev/msv012

Quint M et al. (2012). A transcriptomic hourglass in plant embryogenesis. Nature (490): 98-101.

M. L. Delignette-Muller, R. Pouillot, J.-B. Denis and C. Dutang (2014), fitdistrplus: help to fit of a parametric distribution to non-censored or censored data.

Cullen AC and Frey HC (1999) Probabilistic techniques in exposure assessment. Plenum Press, USA, pp. 81-159.

Evans M, Hastings N and Peacock B (2000) Statistical distributions. John Wiley and Sons Inc.

Sokal RR and Rohlf FJ (1995) Biometry. W.H. Freeman and Company, USA, pp. 111-115.

Juergen Gross and bug fixes by Uwe Ligges (2012). nortest: Tests for Normality. R package version 1.0-2.

http://CRAN.R-project.org/package=nortest

Dallal, G.E. and Wilkinson, L. (1986): An analytic approximation to the distribution of Lilliefors' test for normality. The American Statistician, 40, 294-296.

Stephens, M.A. (1974): EDF statistics for goodness of fit and some comparisons. Journal of the American Statistical Association, 69, 730-737.

http://stackoverflow.com/questions/4290081/fitting-data-to-distributions?rq=1

http://stats.stackexchange.com/questions/45033/can-i-use-kolmogorov-smirnov-test-and-estimate-distribution-parameters

http://cran.r-project.org/doc/contrib/Ricci-distributions-en.pdf

http://cran.r-project.org/doc/contrib/Ricci-distributions-en.pdf

See Also

`reversehourglassScore`, `bootMatrix`, `FlatLineTest`, `EarlyConservationTest`, `PlotSignature`

Examples

```
data(PhyloExpressionSetExample)

# perform the reductive hourglass test for a PhyloExpressionSet
# here the prior biological knowledge is that stages 1-2 correspond to module 1 = early,
# stages 3-5 to module 2 = mid (phylotypic module), and stages 6-7 correspond to
# module 3 = late
ReverseHourglassTest(PhyloExpressionSetExample,
                       modules = list(early = 1:2, mid = 3:5, late = 6:7),
                       permutations = 1000)


# use your own permutation matrix based on which p-values (ReverseHourglassTest)
# shall be computed
custom_perm_matrix <- bootMatrix(PhyloExpressionSetExample,100)

ReverseHourglassTest(PhyloExpressionSetExample,
                     modules = list(early = 1:2, mid = 3:5, late = 6:7),
                     custom.perm.matrix = custom_perm_matrix)
```

rhScore *Compute the Hourglass Score for the Reductive Hourglass Test*

Description

This function reduces the destruction of an hourglass shaped pattern to a single score value.

Based on a given TAI or TDI pattern the given vector is being divided into three developmental modules: early, mid, and late. The corrisponding TAI or TDI values in each developmental module are accumulated using the *scoringMethod* argument ("max-min" or "mean-mean").

In more detail:

(1) for a given TAI or TDI vector *tai_profile* or *tdi_profile*, we classify each value of *tai_profile* or *tdi_profile* into its corresponding developmental module early, mid, or late.

(2) accumulate the *tai_profile* or *tdi_profile* values in each developmental module using the arithmetic mean (mean) in case scoringMethod = "mean-mean", or accumulate the *tai_profile* or *tdi_profile* values in each developmental module using max for the early and late module and min for the mid module in case scoringMethod = "max-min".

(3) then reduce the three values for each developmental module by computing the difference between: early - mid, and late - mid.

(4) the two difference values are referred to as a_early and a_late.

Each developmental module now has an accumulated representation value which is being reduced to one value using the *method* argument ("max", "min", or "mean").

Given the two accumulated values for each hourglass module: a_early and a_late, we reduce the two given values by:

- *"max"*: $S = maxa_early, a_late$
- *"min"*: $S = mina_early, a_late$
- *"mean"*: $S = meana_early, a_late$

All together this results in a global score *S*. This global score *S* is being returned by this function rhScore.

Usage

```
rhScore(
  age_vals,
  early,
  mid,
  late,
  method,
  scoringMethod,
  profile.warn = FALSE
)
```

Arguments

`age_vals`	a numeric vector containing `TAI` or `TDI` values for each developmental stage s.
`early`	a numeric vector including the numeric stage values that correspond to the early phase of development.
`mid`	a numeric vector including the numeric stage values that correspond to the middle phase of development.
`late`	a numeric vector including the numeric stage values that correspond to the late phase of development.
`method`	to determine the two value reduction value, resulting in the global score S: "max", or "min", or "mean".
`scoringMethod`	method to determine the module accumulation value: "max-min" or "mean-mean".
`profile.warn`	a bolean value indicating whether a warning is printed when the high-low-high pattern isn't followed.

Details

The gpScore is a heuristic score enabling to construct a test statistic to determine the significance of a present (phylotranscriptomic) hourglass pattern.

Value

a numeric value representing the hourglass destruction score.

Author(s)

Hajk-Georg Drost

References

Drost HG et al. (2015) Mol Biol Evol. 32 (5): 1221-1231 doi:10.1093/molbev/msv012

See Also

`ReductiveHourglassTest`, `TAI`, `TDI`

Examples

```
# read standard phylotranscriptomics data
data(PhyloExpressionSetExample)
data(DivergenceExpressionSetExample)

# example PhyloExpressionSet:

# compute the TAI profile
TAIs <- TAI(PhyloExpressionSetExample)

# compute the global hourglass destruction score
# for the TAIs profile using reduction method: mean(mean-mean)
rh_score <- rhScore(age_vals = TAIs,early = 1:2,mid = 3:5,late = 6:7,
                    method = "mean",scoringMethod = "mean-mean")


# example DivergenceExpressionSet:

# compute the TDI profile
TDIs <- TDI(DivergenceExpressionSetExample)

# compute the global hourglass destruction score for the TDIs profile
# using reduction method: mean(mean-mean)
rh_score <- rhScore(age_vals = TDIs,early = 1:2,mid = 3:5,late = 6:7,
                    method = "mean",scoringMethod = "mean-mean")

# get warning if the expected pattern isn't followed
rh_score <- rhScore(age_vals = TAIs,early = 1:2,mid = 3:5,late = 6:7,
                    method = "mean",scoringMethod = "mean-mean",profile.warn=TRUE)
```

SelectGeneSet *Select a Subset of Genes in an ExpressionSet*

Description

Select a subset of genes stored in the input `ExpressionSet`.

Usage

```
SelectGeneSet(ExpressionSet, gene.set, use.only.map = FALSE)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.
`gene.set`	a character vector storing the gene ids for which gene expression profiles shall be visualized.
`use.only.map`	a logical value indicating whether instead of a standard `ExpressionSet` only a `Phylostratigraphic Map` or `Divergene Map` is passed to the function.

Details

This function selects a subset of genes specified in `gene.set` stored in the input `ExpressionSet` and returns a subset `ExpressionSet`.

This function is useful for studying the evolutionary *properties* of a subset of genes stored in the `ExpressionSet`.

Author(s)

Hajk-Georg Drost

See Also

PlotGeneSet, PlotEnrichment, DiffGenes

Examples

```
data(PhyloExpressionSetExample)

# receive a subset ExpressionSet for the fist 5 genes stored in
# the PhyloExpressionSetExample
SelectGeneSet(ExpressionSet = PhyloExpressionSetExample,
            gene.set      = PhyloExpressionSetExample[1:5, 2])


# get a gene subset using only a phylostratihraphic map
ExamplePSMap <- PhyloExpressionSetExample[ , 1:2]

SelectGeneSet(ExpressionSet = ExamplePSMap,
              gene.set      = PhyloExpressionSetExample[1:5, 2],
              use.only.map  = TRUE)
```

TAI *Compute the Transcriptome Age Index (TAI)*

Description

This function computes the phylogenetically based transcriptome age index (TAI) introduced by Domazet-Loso & Tautz, 2010.

Usage

```
TAI(PhyloExpressionSet)
```

Arguments

PhyloExpressionSet	a standard PhyloExpressionSet object.

Details

The TAI measure represents the weighted arithmetic mean (expression levels as weights for the phylostratum value) over all evolutionary age categories denoted as *phylostra*.

$$TAI_s = \sum (e_i s * ps_i) / \sum e_i s$$

where TAI_s denotes the TAI value in developmental stage s, e_is denotes the gene expression level of gene i in stage s, and ps_i denotes the corresponding phylostratum of gene i, $i = 1, ..., N$ and N = total number of genes.

Internally the function calls the C++ function `cpp_TAI` to speed up TAI computations.

Value

a numeric vector containing the TAI values for all given developmental stages.

Author(s)

Hajk-Georg Drost

References

Domazet-Loso T. and Tautz D. (2010). *A phylogenetically based transcriptome age index mirrors ontogenetic divergence patterns*. Nature (468): 815-818.

Quint M et al. (2012). *A transcriptomic hourglass in plant embryogenesis*. Nature (490): 98-101.

Drost HG et al. (2015) Mol Biol Evol. 32 (5): 1221-1231 doi:10.1093/molbev/msv012

See Also

`TDI`, `PlotPattern`, `FlatLineTest`, `ReductiveHourglassTest`

Examples

```
# reading a standard PhyloExpressionSet
data(PhyloExpressionSetExample)

# computing the TAI profile of a given PhyloExpressionSet object
TAIs <- TAI(PhyloExpressionSetExample)
```

`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".

Author(s)

Hajk-Georg Drost

Examples

```
## Not run:
# 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")

## End(Not run)
```

TDI — *Compute the Transcriptome Divergence Index (TDI)*

Description

This function computes the sequence distance based transcriptome divergence index (TDI) introduced by Quint et al., 2012.

Usage

```
TDI(DivergenceExpressionSet)
```

Arguments

DivergenceExpressionSet	a standard PhyloExpressionSet or DivergenceExpressionSet object.

Details

The TDI measure represents the weighted arithmetic mean (expression levels as weights for the divergence-stratum value) over all gene divergence categories denoted as *divergence-strata*.

$$TDI_s = \sum (e_i s * ds_i) / \sum e_i s$$

where TDI_s denotes the TDI value in developmental stage s, e_is denotes the gene expression level of gene i in stage s, and ds_i denotes the corresponding divergence-stratum of gene i, $i = 1, ..., N$ and N = total number of genes.

Internally the function is written in C++ to speed up TDI computations.

Value

a numeric vector containing the TDI values for all given developmental stages.

Author(s)

Hajk-Georg Drost

References

Quint M et al. (2012). *A transcriptomic hourglass in plant embryogenesis*. Nature (490): 98-101.

Drost HG et al. (2015) Mol Biol Evol. 32 (5): 1221-1231 doi:10.1093/molbev/msv012

See Also

TAI, PlotPattern, FlatLineTest, ReductiveHourglassTest

Examples

```
# reading a standard DivergenceExpressionSet
data(DivergenceExpressionSetExample)

# computing the TDI profile of a given DivergenceExpressionSet object
TDIs <- TDI(DivergenceExpressionSetExample)
```

TEI *Compute the Transcriptome Evolutionary Index (TEI)*

Description

This function computes the phylogenetically based transcriptome evolutionary index (TEI) similar to Domazet-Loso & Tautz, 2010.

Usage

```
TEI(
  ExpressionSet,
  Phylostratum = NULL,
  split = 1e+05,
  showprogress = TRUE,
  threads = 1
)
```

Arguments

`ExpressionSet`	expression object with rownames as GeneID (dgCMatrix) or standard PhyloExpressionSet object.
`Phylostratum`	a named vector representing phylostratum per GeneID with names as GeneID (not used if Expression is PhyloExpressionSet).
`split`	specify number of columns to split
`showprogress`	boolean if progressbar should be shown
`threads`	specify number of threads

Details

The TEI measure represents the weighted arithmetic mean (expression levels as weights for the phylostratum value) over all evolutionary age categories denoted as *phylostra*.

$$TEI_s = \sum (e_is * ps_i) / \sum e_is$$

where TEI_s denotes the TEI value in developmental stage s, e_is denotes the gene expression level of gene i in stage s, and ps_i denotes the corresponding phylostratum of gene i, $i = 1, ..., N$ and N = total number of genes.

Value

a numeric vector containing the TEI values for all given cells or developmental stages.

Author(s)

Kristian K Ullrich

References

Domazet-Loso T. and Tautz D. (2010). *A phylogenetically based transcriptome age index mirrors ontogenetic divergence patterns*. Nature (468): 815-818.

Quint M et al. (2012). *A transcriptomic hourglass in plant embryogenesis*. Nature (490): 98-101.

Drost HG et al. (2015) Mol Biol Evol. 32 (5): 1221-1231 doi:10.1093/molbev/msv012

Examples

```
# reading a standard PhyloExpressionSet
data(PhyloExpressionSetExample, package = "myTAI")

# computing the TEI profile of a given PhyloExpressionSet object
TEI <- TEI(PhyloExpressionSetExample)
```

`tf` *Transform Gene Expression Levels*

Description

This function transforms the gene expression set stored in an input PhloExpressionSet or DivergenceExpressionSet object and returns a PhloExpressionSet or DivergenceExpressionSet object with transformed expression levels. The resulting transformed PhloExpressionSet or DivergenceExpressionSet object can then be used for subsequent analyses based on transformed expression levels.

Usage

```
tf(ExpressionSet, FUN, pseudocount = 0, integerise = FALSE)
```

Arguments

`ExpressionSet`	a standard PhloExpressionSet or DivergenceExpressionSet object.
`FUN`	any valid function that transforms gene expression levels.
`pseudocount`	a numeric value to be added to the expression matrix prior to transformation.
`integerise`	a boolean specifying whether the expression data should be rounded to the nearest integer.

Details

Motivated by the dicussion raised by Piasecka et al., 2013, the influence of gene expression transformation on the global phylotranscriptomics pattern does not seem negligible. Hence, different transformations can result in qualitatively different `TAI` or `TDI` patterns.

Initially, the `TAI` and `TDI` formulas were defined for absolute expression levels. So using the initial `TAI` and `TDI` formulas with transformed expression levels might turn out in qualitatively different patterns when compared with non-transformed expression levels, but might also belong to a different class of models, since different valid expression level transformation functions result in different patterns.

The purpose of this function is to allow the user to study the qualitative impact of different transformation functions on the global `TAI` and `TDI` pattern, or on any subsequent phylotranscriptomics analysis.

The examples using the *PhyloExpressionSetExample* data set show that using common gene expression transformation functions: `log2` (Quackenbush, 2001 and 2002), `sqrt` (Yeung et al., 2001), boxcox, or *inverse hyperbolic sine transformation*, each transformation results in qualitatively different patterns. Nevertheless, for each resulting pattern the statistical significance can be tested using either the FlatLineTest or ReductiveHourglassTest (Drost et al., 2014) to quantify the significance of interest.

Value

a standard PhloExpressionSet or DivergenceExpressionSet object storing transformed gene expression levels.

Author(s)

Hajk-Georg Drost

References

Piasecka B, Lichocki P, Moretti S, et al. (2013) The hourglass and the early conservation models–co-existing patterns of developmental constraints in vertebrates. PLoS Genet. 9(4): e1003476.

Quint M., Drost H.G., Gabel A., Ullrich K.K., Boenn M., Grosse I. (2012) A transcriptomic hourglass in plant embryogenesis. Nature 490: 98-101.

Domazet-Loso T., Tautz D. (2010) A phylogenetically based transcriptome age index mirrors ontogenetic divergence patterns. Nature 468: 815-8.

Drost HG et al. (2015) Mol Biol Evol. 32 (5): 1221-1231 doi:10.1093/molbev/msv012

K.Y. Yeung et al.: Model-based clustering and data transformations for gene expression data. Bioinformatics 2001, 17:977-987

K.Y. Yeung et al.: Supplement to Model-based clustering and data transformations for gene expression data - Data Transformations and the Gaussian mixture assumption. Bioinformatics 2001, 17:977-987

P.A.C. Hoen et al.: Deep sequencing-based expression analysis shows major advances in robustness, resolution and inter-lab portability over five microarray platforms. Nucleic Acids Research 2008, Vol. 36, No. 21

H.H. Thygesen et al.: Comparing transformation methods for DNA microarray data. BMC Bioinformatics 2004, 5:77

John Quackenbush: Microarray data normalization and transformation. Nature Genetics 2002, 32:496-501

John Quackenbush: Computational Analysis of Microarray Data. Nature Reviews 2001, 2:418-427

R. Nadon and J. Shoemaker: Statistical issues with microarrays: processing and analysis. TRENDS in Genetics 2002, Vol. 18 No. 5:265-271

B.P. Durbin et al.: A variance-stabilizing transformation for gene-expression microarray data. Bioinformatics 2002, 18:S105-S110

J. M. Bland et al.: Transforming data. BMJ 1996, 312:770

John B. Burbidge, Lonnie Magee and A. Leslie Robb (1988) Alternative Transformations to Handle Extreme Values of the Dependent Variable. Journal of the American Statistical Association, 83(401): 123-127.

G. E. P. Box and D. R. Cox (1964) An Analysis of Transformations. Journal of the Royal Statistical Society. Series B (Methodological), 26(2): 211-252.

See Also

TAI, TDI, FlatLineTest, ReductiveHourglassTest, tfStability

Examples

```
## Not run:
data(PhyloExpressionSetExample)

# a simple example is to transform the gene expression levels
# of a given PhyloExpressionSet using a sqrt or log2 transformation

PES.sqrt <- tf(PhyloExpressionSetExample, sqrt)

PES.log2 <- tf(PhyloExpressionSetExample, log2)

# plotting the TAI using log2 transformed expression levels
# and performing the Flat Line Test to obtain the p-value
PlotSignature(ExpressionSet = tf(PhyloExpressionSetExample, log2),
            permutations = 1000)



# in case the expression matrix contains 0s, a pseudocount can be added prior
# to certain transformations, e.g. log2(x+1) where 1 is the pseudocount.

PhyloExpressionSetExample[4,3] = 0
PES.log2 <- tf(PhyloExpressionSetExample, log2, pseudocount = 0)

# this should return -Inf at PES.log2[4,3] the issue here is that
# -Inf cannot be used to compute the phylotranscriptomic profile.

PES.log2 <- tf(PhyloExpressionSetExample, log2, pseudocount = 1)
# log2 transformed expression levels can now be used in downstream analyses.


# to perform rank transformation

PES.rank <- tf(PhyloExpressionSetExample, FUN = function(x) apply(x, 2, base::rank))


# rlog and vst transformations are now also possible by loading the DESeq2 package
# and transforming the data with the parameter integerise = TRUE.
library(DESeq2) # make sure the DESeq2 version >= 1.29.15 for rlog
PES.vst <- tf(PhyloExpressionSetExample, vst, integerise = TRUE)

## End(Not run)
```

tfPS *Transform Phylostratum Values*

Description

This function performs transformation of phylostratum values.

Usage

```
tfPS(ExpressionSet, transform)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet 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 `is.ExpressionSet`. Note that the input `transform` must be an available function, currently limited to only `"qr"` (or `"quantilerank"`).

Value

a standard PhloExpressionSet object storing transformed Phylostratum levels.

Author(s)

Jaruwatana Sodai Lotharukpong and Lukas Maischak

See Also

`tf`

Examples

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

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

tfStability *Perform Permutation Tests Under Different Transformations*

Description

tfStability aims to statistically evaluate the stability of ReductiveHourglassTest, FlatLineTest, ReverseHourglassTest, EarlyConservationTest, or LateConservationTest (all based on TAI or TDI computations) against different data transformations. The corresponding p-value quantifies the probability that a given TAI or TDI pattern (or any phylotranscriptomics pattern) does not support the chosen test. A p-value < 0.05 indicates that the corresponding phylotranscriptomics pattern does indeed support the chosen test.

Usage

```
tfStability(
  ExpressionSet,
  TestStatistic = "FlatLineTest",
  transforms = c("none", "sqrt", "log2", "rank", "squared"),
  modules = NULL,
  permutations = 1000,
  pseudocount = 1
)
```

Arguments

`ExpressionSet`	a standard PhyloExpressionSet or DivergenceExpressionSet object.
`TestStatistic`	a string defining the type of test statistics to be used to quantify the statistical significance the present phylotranscriptomics pattern. Possible values can be: • `TestStatistic = "FlatLineTest"` : Statistical test for the deviation from a flat line • `TestStatistic = "ReductiveHourglassTest"` : Statistical test for the existence of a hourglass shape (high-low-high pattern) • `TestStatistic = "ReverseHourglassTest"` : Statistical test for the existence of a reverse hourglass pattern (low-high-low pattern) • `TestStatistic = "EarlyConservationTest"` : Statistical test for the existence of a early conservation pattern (low-high-high pattern) • `TestStatistic = "LateConservationTest"` : Statistical test for the existence of a late conservation pattern (high-high-low pattern)
`transforms`	a character vector of any valid function that transforms gene expression levels.
`modules`	a list storing three elements: early, mid, and late. Each element expects a numeric vector specifying the developmental stages or experiments that correspond to each module. For example, `module` = list(early = 1:2, mid = 3:5, late = 6:7) divides a dataset storing seven developmental stages into 3 modules.
`permutations`	a numeric value specifying the number of permutations to be performed for the FlatLineTest, EarlyConservationTest, LateConservationTest, ReductiveHourglassTest or ReverseHourglassTest.

`pseudocount`	any valid number to add to the expression matrix prior to log transformations.

Details

An assessment for the stability of data transforms on the permutation test of choice. For details, please consult `tf`, `ReductiveHourglassTest`, `FlatLineTest`, `ReverseHourglassTest`, `LateConservationTest` or `EarlyConservationTest`

Value

a vector object containing the vector elements:

`p.value` : the p-value quantifying the statistical significance (depending on the chosen test) of the given phylotranscriptomics pattern under the given data transformation(s).

Author(s)

Jaruwatana Sodai Lotharukpong

References

Lotharukpong JS et al. (2023) (unpublished)

See Also

`rhScore`, `bootMatrix`, `FlatLineTest`, `ReverseHourglassTest`, `EarlyConservationTest`, `ReductiveHourglassTest`, `PlotSignature`, `LateConservationTest`

Examples

```
data(PhyloExpressionSetExample)

# perform the reductive hourglass test for a PhyloExpressionSet
# here the prior biological knowledge is that stages 1-2 correspond to module 1 = early,
# stages 3-5 to module 2 = mid (phylotypic module), and stages 6-7 correspond to
# module 3 = late
tfStability(ExpressionSet = PhyloExpressionSetExample,
                     TestStatistic = "ReductiveHourglassTest",
                     permutations      = 100,
                     transforms = c("log2", "sqrt", "none"),
                     modules = list(early = 1:2, mid = 3:5, late = 6:7))


# it is also possible to test the phylotranscriptomic pattern using rlog
# and vst transforms from DESeq2

library(DESeq2)
tfStability(ExpressionSet = PhyloExpressionSetExample,
                     TestStatistic = "ReductiveHourglassTest",
                     permutations      = 100,
                     transforms = c("log2", "sqrt", "none", "vst"),
                     modules = list(early = 1:2, mid = 3:5, late = 6:7))
```

TPI *Compute the Transcriptome Polymorphism Index (TPI)*

Description

This function computes the Transcriptome Polymorphism Index (TPI) introduced by Gossmann et al., 2015.

Usage

```
TPI(PolymorphismExpressionSet)
```

Arguments

`PolymorphismExpressionSet`	a standard PolymorphismExpressionSet object.

Details

The TPI measure represents the weighted arithmetic mean (expression levels as weights) for the synonymous vs non-synonymous polymorphism ratios.

$$TPI_s = \sum(e_i s * P_N / N / ((P_S + 1)/S)) / \sum e_i s$$

where TPI_s denotes the TPI value in developmental stage s, e_is denotes the gene expression level of gene i in stage s, n denotes the number of genes, PN and PS denote the numbers of non-synonymous and synonymous polymorphisms, and N and S are the numbers of nonsynonymous and synonymous sites, respectively.

Internally the function is written in C++ to speed up TPI computations.

Value

a numeric vector containing the TPI values for all given developmental stages.

Author(s)

Hajk-Georg Drost

References

Gossmann et al. (2015). *Transcriptomes of Plant Gametophytes Have a Higher Proportion of Rapidly Evolving and Young Genes than Sporophytes*. Mol Biol Evol. 33 (7): 1669-1678.

See Also

`TAI`, `TDI`, `PlotSignature`, `PlotPattern`, `FlatLineTest`, `ReductiveHourglassTest`

Examples

```
## Not run:
# reading a standard PolymorphismExpressionSet
data(PolymorphismExpressionSetExample)

# computing the TPI profile of a given PolymorphismExpressionSet object
TPIs <- TPI(PolymorphismExpressionSet)

## End(Not run)
```

Index