Package 'LimROTS'

May 2, 2026

Title LimROTS: A Hybrid Method Integrating Empirical Bayes and Reproducibility-Optimized Statistics for Robust Differential Expression Analysis

Version 1.5.0

Description Differential expression analysis is commonly used to study diverse biological datasets. The reproducibility-optimized test statistic (ROTS) (Elo et al., 2008, <doi:10.1109/tcbb.2007.1078>) uses a modified t-statistic to prioritise features that differ between two or more groups. However, the ROTS Bioconductor implementation (Suomi et al., 2017, <doi:10.1371/journal.pcbi.1005562>) did not accommodate technical or biological covariates. LimROTS (Anwar et al., 2025, <doi:10.1093/bioinformatics/btaf570>) addressed this limitation by combining a reproducibility-optimized test statistic with the limma empirical Bayes approach (Ritchie et al., 2015, <doi:10.1093/nar/gkv007>). This enables the analysis of more complex experimental designs and the incorporation of covariates.

License GPL (>= 2)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.5.0), SummarizedExperiment

biocViews Software, GeneExpression, DifferentialExpression, Microarray, RNASeq, Proteomics, ImmunoOncology, Metabolomics, mRNAMicroarray

URL https://github.com/AliYoussef96/LimROTS, https://aliyoussef96.github.io/LimROTS/

BugReports https://github.com/AliYoussef96/LimROTS/issues

VignetteBuilder knitr

Imports limma, stringr, qvalue, utils, stats, BiocParallel, S4Vectors, dplyr, survival, cmprsk, variancePartition

Suggests BiocStyle, ggplot2, testthat (>= 3.0.0), knitr, rmarkdown, caret, ROTS, mia, miaTime, TreeSummarizedExperiment

Config/testthat/edition 3

git_url https://git.bioconductor.org/packages/LimROTS

git_branch devel

git_last_commit 7309bb5

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-05-01

Author Ali Mostafa Anwar [aut, cre] (ORCID: <https://orcid.org/0000-0002-5201-387X>), Leo Lahti [aut, ths] (ORCID: <https://orcid.org/0000-0001-5537-637X>), Akewak Jeba [aut, ctb] (ORCID: <https://orcid.org/0009-0007-1347-7552>), Eleanor Coffey [aut, ths] (ORCID: <https://orcid.org/0000-0002-9717-5610>), Rasmus Hindström [ctb] (ORCID: <https://orcid.org/0009-0004-5731-178X>)

Maintainer Ali Mostafa Anwar <aliali.mostafa99@gmail.com>

Contents

bootstrapS *Generate Bootstrap Samples*

Description

This function generates bootstrap samples from the input metadata. It samples with replacement within each group defined in the metadata, and optionally adjusts for paired groups.

Usage

```
bootstrapS(niter, meta.info, group)
```

Arguments

niter	Integer. The number of bootstrap samples to generate.
meta.info	Data frame. Metadata containing sample information, where each row corresponds to a sample.
group	Character. The name of the column in meta.info that defines the grouping variable for the samples.

Details

The function works by resampling the row names of the metadata for each group separately.

Value

A matrix of dimension niter x n, where n is the number of samples. Each row corresponds to a bootstrap sample, and each entry is a resampled row name from the metadata.

bootstrapSamples_limRots *Generate Stratified Bootstrap Samples for limRots*

Description

This function generates stratified bootstrap samples based on the groupings and additional factors in the metadata. The function ensures that samples are drawn proportionally based on strata defined by the interaction of factor columns in the metadata.

Usage

```
bootstrapSamples_limRots(niter, meta.info, group)
```

Arguments

niter	Integer. The number of bootstrap samples to generate.
meta.info	Data frame. Metadata containing sample information, where each row corresponds to a sample. Factor columns in meta.info are used to define strata for sampling.
group	Character. The name of the column in meta.info that defines the grouping variable for the samples.

Details

The function works by first identifying the factors in the `meta.info` data frame that are used to create strata for sampling. Within each group defined by `group`, the function samples according to the strata proportions, ensuring that samples are drawn from the correct groups and strata in a proportional manner.

Value

A matrix of dimension `niter` x `n`, where `n` is the number of samples. Each row corresponds to a bootstrap sample, and each entry is a resampled row name from the metadata, stratified by group and additional factors.

`bootstrapSamples_limRots_block` *Generate Stratified Bootstrap Samples with Correlation Blocks*

Description

This function generates stratified bootstrap samples identical to `bootstrapSamples_limRots`, but additionally supports correlation blocks. When `correlation_block` is specified, all samples sharing the same block ID are always selected together during resampling. When `correlation_block` is NULL, the function delegates entirely to `bootstrapSamples_limRots`.

Usage

```
bootstrapSamples_limRots_block(
  niter,
  meta.info,
  group,
  correlation_block = NULL
)
```

Arguments

`niter`	Integer. The number of bootstrap samples to generate.
`meta.info`	Data frame. Metadata containing sample information, where each row corresponds to a sample. Factor columns in `meta.info` are used to define strata for sampling.
`group`	Character. The name of the column in `meta.info` that defines the grouping variable for the samples.
`correlation_block`	Character or NULL. The name of a column in `meta.info` that defines correlation blocks. Samples sharing the same value in this column are always resampled together as a unit. If NULL, the function behaves identically to `bootstrapSamples_limRots`.

Details

The function follows the same logic as `bootstrapSamples_limRots`: within each group defined by `group`, it identifies factor columns to create strata, then samples proportionally within each stratum. When `correlation_block` is not NULL, entire blocks (e.g., repeated measures from the same subject) are resampled together as a unit instead of individual samples.

Value

A matrix of dimension `niter` x `n`, where `n` is the number of samples. Each row corresponds to a bootstrap sample, and each entry is a resampled row name from the metadata, stratified by group and additional factors.

`bootstrapSamples_limRots_cox`

Generate Stratified Bootstrap Samples for Cox limRots with Correlation Blocks

---

Description

This function generates stratified bootstrap samples similar to `bootstrapSamples_limRots`, but additionally supports correlation blocks. When `correlation_block` is specified, all samples sharing the same block ID are always selected together during resampling.

Usage

```
bootstrapSamples_limRots_cox(niter, meta.info, correlation_block = NULL)
```

Arguments

`niter`	Integer. The number of bootstrap samples to generate.
`meta.info`	Data frame. Metadata containing sample information, where each row corresponds to a sample. Factor columns in `meta.info` are used to define strata for sampling.
`correlation_block`	Character or NULL. The name of a column in `meta.info` that defines correlation blocks. Samples sharing the same value in this column are always resampled together as a unit.

Details

When `correlation_block` is not NULL, the function groups samples by their block ID within each group/stratum and resamples entire blocks with replacement, so that correlated samples (e.g., repeated measures from the same subject) are always kept together.

Value

A matrix of dimension `niter` x `n`, where `n` is the number of samples. Each row corresponds to a bootstrap sample, and each entry is a resampled row name from the metadata.

`bootstrap_survival` *Perform Per-Feature Survival Modeling on Bootstrap Resamples*

Description

Perform Per-Feature Survival Modeling on Bootstrap Resamples

Usage

```
bootstrap_survival(x, meta.info, formula.str, competing_risks)
```

Arguments

`x`	A list of data matrices (bootstrap resample), where each element corresponds to a resampled group. Rows represent features (e.g., proteins, metabolites) and columns represent samples.
`meta.info`	A data frame containing the metadata for the samples, including `time`, `event`, and any additional covariates used in `formula.str`.
`formula.str`	A string specifying the formula to be used in model fitting. Must include a `Surv(time, event)` term. The per-feature coefficient term (`y`) is prepended automatically.
`competing_risks`	Logical. If `FALSE` (default), a Cox proportional hazards model is fitted per feature using `coxph`. If `TRUE`, a competing risks model is fitted per feature using `crr` from the `cmprsk` package.

Details

For each feature (row), the function constructs a temporary data frame combining the bootstrap-resampled expression values (`y`) with the sample metadata. It then fits either a Cox proportional hazards model (`competing_risks = FALSE`) or a Fine subdistribution hazard model (`competing_risks = TRUE`) per feature, extracting the coefficient and its standard error for the feature term `y`.

Value

A list containing the following elements:

`d`	A numeric vector of absolute coefficients ($\|\beta\|$) for each feature.
`s`	A numeric vector of standard errors of the coefficients for each feature.

See Also

`coxph`, `crr`

`Boot_parallel` *Parallel processing handling function*

Description

Parallel processing handling function

Usage

```
Boot_parallel(
  BPPARAM = NULL,
  samples,
  data,
  formula.str,
  group,
  groups,
  meta.info,
  a1,
  a2,
  pSamples,
  correlation_block = NULL
)
```

Arguments

`BPPARAM`	A parallel BPPARAM object for distributed computation.
`samples`	bootstrapped samples matrix
`data`	A `SummarizedExperiment` object or a matrix where rows represent features (e.g., genes, proteins) and columns represent samples. The values should be log-transformed.
`formula.str`	A formula string used when covariates are present in meta. info for modeling. It should include "~ 0 + ..." to exclude the intercept from the model.
`group`	A string specifying the column in `meta.info` that represents the groups or conditions for comparison.
`groups`	groups information from `meta.info`
`meta.info`	A data frame containing sample-level metadata, where each row corresponds to a sample. It should include the grouping variable specified in `group`. If `x` is a `SummarizedExperiment` object, `meta.info` must be a vector of the metadata needed for the model to run and can be retrieved using `colData()`.
`a1`	Optional numeric value used in the optimization process. If defined by the user, no optimization occurs.
`a2`	Optional numeric value used in the optimization process. If defined by the user, no optimization occurs.
`pSamples`	a permutated list of samples
`correlation_block`	Character or NULL. The name of a column in `meta.info` that defines correlation blocks. Passed to `Limma_bootstrap` and `Limma_permutating` to account for within-block correlation during model fitting via `duplicateCorrelation`. If NULL, standard independent fitting is used.

Value

A list containing: D, S, pD, pS for bootstrapped data and for permuted data.

`Boot_parallel_survival` *Parallel processing handling function for LimROTS survival*

Description

Parallel processing handling function for LimROTS survival

Usage

```
Boot_parallel_survival(
  BPPARAM = NULL,
  samples,
  data,
  formula.str,
  meta.info,
  a1,
  a2,
  pSamples,
  competing_risks
)
```

Arguments

BPPARAM	A parallel BPPARAM object for distributed computation.
samples	bootstrapped samples matrix
data	A SummarizedExperiment object or a matrix where rows represent features (e.g., genes, proteins) and columns represent samples. The values should be log-transformed.
formula.str	A formula string used when covariates are present in meta. info for modeling. It should include "~ 0 + ..." to exclude the intercept from the model.
meta.info	A data frame containing sample-level metadata, where each row corresponds to a sample. It should include the grouping variable specified in group. If x is a SummarizedExperiment object, meta.info must be a vector of the metadata needed for the model to run and can be retrieved using colData().
a1	Optional numeric value used in the optimization process. If defined by the user, no optimization occurs.
a2	Optional numeric value used in the optimization process. If defined by the user, no optimization occurs.
pSamples	a permutated list of samples
competing_risks	Logical. If TRUE, the competing risks model via crr from cmprsk is used instead of the standard Cox proportional hazards model.

Value

A list containing: D, S, pD, pS for bootstrapped data and for permuted data.

calculateFalseDiscoveryRate *Calculate False Discovery Rate (FDR) Using Permuted Values (Adjusted)*

Description

This function calculates the false discovery rate (FDR) by comparing observed values to permuted values.The function sorts observed values, compares them against permuted data, and computes FDR using the median of permutation results.

Usage

```
calculateFalseDiscoveryRate(observedValues, permutedValues)
```

Arguments

observedValues	Numeric vector. The observed test statistics or values to be evaluated for significance.
permutedValues	Numeric matrix. The permuted test statistics or values, with rows corresponding to the same values as in observedValues and columns representing different permutations.

Value

A numeric vector of the same length as observedValues, containing the estimated FDR for each observed value.

calOverlaps *Calculate Overlaps Between Observed and Permuted Data*

Description

This function calculates the overlap between observed and permuted data for two sets of comparisons. It computes the ratio of overlap between pairs of vectors (res1/res2 and pres1/pres2) after sorting the values.

Usage

```
calOverlaps(D, S, pD, pS, nrow, N, N_len, ssq, niter, overlaps, overlaps_P)
```

Arguments

D	Numeric vector. Observed data values (e.g., differences).
S	Numeric vector. Standard errors or related values associated with the observed data.
pD	Numeric vector. Permuted data values (e.g., differences).
pS	Numeric vector. Standard errors or related values associated with the permuted data.

nrow	Integer. Number of rows in each block of data.
N	Integer vector. Number of top values to consider for overlap calculation.
N_len	Integer. Length of the N vector.
ssq	Numeric. A small constant added to standard errors for stability.
niter	Integer. Number of bootstrap samples or resampling iterations.
overlaps	Numeric matrix. Matrix to store overlap results for observed data.
overlaps_P	Numeric matrix. Matrix to store overlap results for permuted data.

Details

The function calculates overlaps for two sets of comparisons: one for observed data (res1/res2) and one for permuted data (pres1/pres2).For each bootstrap sample, the function orders the two vectors being compared, then calculates the proportion of overlap for the top N values.

Value

A list containing two matrices: overlaps for observed data and overlaps_P for permuted data.

---

calOverlaps_slr *Calculate Overlaps for Single-Label Replicates (SLR)*

Description

This function computes the overlap between two sets of observed and permuted 'values for single-label replicates (SLR). It calculates the proportion of overlap between pairs of vectors (res1/res2 and pres1/pres2) after sorting them.

Usage

```
calOverlaps_slr(D, pD, nrow, N, N_len, niter, overlaps, overlaps_P)
```

Arguments

D	Numeric vector. Observed data values (e.g., differences).
pD	Numeric vector. Permuted data values.
nrow	Integer. Number of rows in each block of data.
N	Integer vector. Number of top values to consider for overlap calculation.
N_len	Integer. Length of the N vector.
niter	Integer. Number of bootstrap samples or resampling iterations.
overlaps	Numeric matrix. Matrix to store overlap results for observed data.
overlaps_P	Numeric matrix. Matrix to store overlap results for permuted data.

Details

The function calculates the overlap for two sets of comparisons: one for observed data (res1/res2) and one for permuted data (pres1/pres2). For each bootstrap sample, the function orders the two vectors being compared, then computes the proportion of overlap for the top N values.

Value

A list containing two matrices: overlaps for observed data and overlaps_P for permuted data.

Check_meta_info *Check if meta info is correct*

Description

Check if meta info is correct

Usage

```
Check_meta_info(meta.info, data, log)
```

Arguments

meta.info	Data frame. Metadata associated with the samples (columns of x). If x is a SummarizedExperiment,
data	A matrix-like object or a SummarizedExperiment containing the data to be analyzed.
log	Logical, indicating whether the data is already log-transformed. Default is TRUE.

Value

Logical

Check_SummarizedExperiment *Check if SummarizedExperiment or data is correct*

Description

Check if SummarizedExperiment or data is correct

Usage

```
Check_SummarizedExperiment(
  x,
  assay.type = NULL,
  meta.info,
  group,
  survival = FALSE
)
```

Arguments

x	A matrix-like object or a SummarizedExperiment containing the data to be analyzed.
assay.type	A character string or numeric index specifying the assay to use if x is a SummarizedExperiment. Default is NULL
meta.info	Data frame. Metadata associated with the samples (columns of x). If x is a SummarizedExperiment,
group	Character. Column name in meta.info that defines the groups or conditions for comparison.
survival	Logical, indicating whether the analysis is survival analysis. Default is FALSE.

Value

a list of data , groups and meta.info

---

countLargerThan *Count Larger Permuted Values (Modified)*

Description

This helper function compares observed values against permuted values and counts the number of permuted values that are greater than or equal to each observed value.

Usage

```
countLargerThan(observedVec, permutedVec)
```

Arguments

observedVec	Numeric vector. The observed values.
permutedVec	Numeric vector. The permuted values to compare against the observed values.

Value

A numeric vector containing the counts of permuted values greater than or equal to the corresponding observed values.

`fit_survival` *Final Per-Feature Survival Model Fit on the Full Dataset*

Description

This function fits a per-feature survival model to the full (non-resampled) data matrix using the observed sample metadata, producing the final statistics used for ranking features.

Usage

```
fit_survival(x, meta.info, formula.str, competing_risks)
```

Arguments

`x`	A data matrix where rows represent features (e.g., proteins, metabolites) and columns represent samples.
`meta.info`	A data frame containing the metadata for the samples. Must include `time`, `event`, and any additional covariates used in `formula.str`.
`formula.str`	A string specifying the formula to be used in model fitting. Must include a `Surv(time, event)` term. The per-feature coefficient term (`y`) is prepended automatically.
`competing_risks`	Logical. If `FALSE` (default), a Cox proportional hazards model is fitted per feature using `coxph`. If `TRUE`, a competing risks model is fitted per feature using `crr` from the `cmprsk` package.

Details

For each feature (row), the function appends the feature expression values as `y` to the sample metadata and fits either a Cox proportional hazards model (`competing_risks = FALSE`) or a subdistribution hazard model (`competing_risks = TRUE`). The coefficient, its standard error, and the exponentiated coefficient (hazard ratio) for the feature term `y` are extracted.

Unlike `permutating_survival` and `bootstrap_survival`, this function operates on the observed (non-permuted, non-resampled) data to produce the final statistics used for feature ranking.

Value

A list containing the following elements:

`d`	A numeric vector of absolute coefficients ($\|\beta\|$) for each feature.
`s`	A numeric vector of standard errors of the coefficients for each feature.
`exp_coef`	A numeric vector of exponentiated coefficients (hazard ratios, e^{β}) for each feature.

See Also

`coxph`, `crr`

`Limma_bootstrap` *Perform Linear Modeling with Covariates using Limma*

Description

This function performs linear modeling using the Limma package while accounting for covariates specified in the `meta.info`. It supports two-group comparisons and multi-group analysis, incorporating covariates through a design matrix.

Usage

```
Limma_bootstrap(x, group, meta.info, formula.str, correlation_block = NULL)
```

Arguments

`x`	A list containing two or more data matrices where rows represent features (e.g., genes, proteins) and columns represent samples. The list should contain at least two matrices for pairwise group comparison.
`group`	A character string indicating the name of the group variable in `meta.info` to be used in the analysis.
`meta.info`	A data frame containing the metadata for the samples. This includes sample grouping and any covariates to be included in the model.
`formula.str`	A string specifying the formula to be used in model fitting. It should follow the standard R formula syntax (e.g., ~ `covariate1` + `covariate2`).
`correlation_block`	Character or NULL. The name of a column in `meta.info` that defines correlation blocks. When not NULL, within-block correlation is estimated via `duplicateCorrelation` and accounted for in `lmFit`. If NULL, standard independent fitting is used.

Details

This function first combines the data matrices from different groups and prepares a design matrix based on the covariates specified in `meta.info` using the provided formula. It fits a linear model using Limma, computes contrasts between groups, and applies empirical Bayes moderation. For two-group comparisons, the function returns log-fold changes and associated statistics. In multi-group settings with a single covariate, it calculates pairwise contrasts and moderated F-statistics.

Value

A list containing the following elements:

`d`	A vector of the test statistics (log-fold changes or F-statistics) for each feature.
`s`	A vector of the standard deviations for each feature, adjusted by the empirical Bayes procedure.

See Also

`lmFit`, `eBayes`, `topTable`, `makeContrasts`

`Limma_fit` *Perform Linear Modeling with Covariates using Limma*

Description

This function performs linear modeling using the Limma package, incorporating covariates in the model fitting process. It is designed to handle both two-group comparisons and multi-group settings with covariates.

Usage

```
Limma_fit(
  x,
  group,
  meta.info,
  formula.str,
  trend,
  robust,
  correlation_block = NULL
)
```

Arguments

`x`	A list containing two or more data matrices where rows represent features (e.g., genes, proteins) and columns represent samples. The list should contain at least two matrices for pairwise group comparison.
`group`	A character string indicating the name of the group variable in`meta.info` to be used in the analysis.
`meta.info`	A data frame containing the metadata for the samples. This includes sample grouping and any covariates to be included in the model.
`formula.str`	A string specifying the formula to be used in model fitting. It should follow the standard R formula syntax (e.g., `~ covariate1 + covariate2`). If the formula contains a random effects term (i.e., includes `|`, e.g., `~ 0 + group + (1|subject)`), the function automatically routes to DREAM analysis via `dream`. The DREAM path is restricted to two-group comparisons; passing more than two groups with a random term raises an error.
`trend`	A logical value indicating whether to allow for an intensity-dependent trend in the prior variance.
`robust`	A logical value indicating whether to use a robust fitting procedure to protect against outliers.
`correlation_block`	Character or NULL. The name of a column in `meta.info` that defines correlation blocks. When not NULL, within-block correlation is estimated via `duplicateCorrelation` and accounted for in `lmFit`. If NULL, standard independent fitting is used.

Details

This function combines the data matrices from different groups and fits a linear model using covariates provided in the `meta.info`. For two-group comparisons, the function computes contrasts between the two groups and applies empirical Bayes moderation. For multi-group analysis with a single covariate, pairwise contrasts are computed, and the moderated F-statistic is calculated for each feature.

When `formula.str` contains a `|` character (random effects term), the function bypasses the standard limma workflow and instead uses `dream` to fit a linear mixed model. This path is only supported for two-group comparisons. Passing more than two groups with a random-effects formula raises an informative error directing the user to use the `correlation_block` argument with a fixed-effects formula instead.

Value

A list containing the following elements:

`d`	A vector of the test statistics (log-fold changes or F-statistics) for each feature.
`s`	A vector of the standard deviations for each feature, adjusted by t he empirical Bayes procedure.
`corrected.logfc`	The log-fold changes for each feature after fitting the model.

See Also

`lmFit`, `eBayes`, `topTable`, `makeContrasts`, `dream`, `makeContrastsDream`

`Limma_permutating` *Perform Permutation-Based Linear Modeling with Covariates using Limma*

Description

This function performs linear modeling using the Limma package with permutation of the covariates to evaluate the test statistics under random assignments. It handles two-group comparisons and multi-group settings.

Usage

```
Limma_permutating(x, group, meta.info, formula.str, correlation_block = NULL)
```

Arguments

`x`	A data matrices where rows represent features (e.g., genes, proteins) and columns represent samples. The list should contain at least two matrices for pairwise group comparison.
`group`	A character string indicating the name of the group variable in `meta.info` to be used in the analysis.
`meta.info`	A data frame containing the metadata for the samples. This includes sample grouping and any covariates to be included in the model.

`formula.str` A string specifying the formula to be used in model fitting. It should follow the standard R formula syntax (e.g., ~ `covariate1` + `covariate2`).

`correlation_block`
Character or NULL. The name of a column in `meta.info` that defines correlation blocks. When not NULL, within-block correlation is estimated via `duplicateCorrelation` and accounted for in `lmFit`. If NULL, standard independent fitting is used.

Details

This function combines the data matrices from different groups and permutes the covariates from `meta.info`before fitting a linear model using Limma. Permutation helps assess how the covariates behave under random conditions, providing a null distribution of the test statistics. For two-group comparisons, the function computes contrasts between the two groups and applies empirical Bayes moderation. For multi-group analysis with a single covariate, pairwise contrasts are computed, and the moderated F-statistic is calculated for each feature.

Value

A list containing the following elements:

`d` A vector of the test statistics (log-fold changes or F-statistics) for each feature.

`s` A vector of the standard deviations for each feature, adjusted by the empirical Bayes procedure.

See Also

`lmFit`, `eBayes`, `topTable`, `makeContrasts`

`LimROTS` `LimROTS`*: A Hybrid Method Integrating Empirical Bayes and Reproducibility-Optimized Statistics for Robust Differential Expression Analysis*

Description

`LimROTS`: A Hybrid Method Integrating Empirical Bayes and Reproducibility-Optimized Statistics for Robust Differential Expression Analysis

Usage

```
LimROTS(
  x,
  assay.type = NULL,
  niter = 1000,
  K = NULL,
  a1 = NULL,
  a2 = NULL,
  log = TRUE,
  verbose = TRUE,
  meta.info,
```

```
  BPPARAM = NULL,
  group,
  formula.str,
  robust = TRUE,
  trend = TRUE,
  permutating.group = FALSE,
  correlation_block = NULL
)
```

Arguments

x	A SummarizedExperiment object, where rows represent features (e.g., proteins, metabolites) and columns represent samples. The values should be log-transformed.
assay.type	A character string or numeric index specifying the assay to use if x is a SummarizedExperiment. Default is NULL
niter	An integer representing the amount of bootstrap iterations. Default is 1000.
K	An optional integer representing the top list size for ranking. If not specified, it is set to one-fourth of the number of features.
a1	Optional numeric value used in the optimization process. If defined by the user, no optimization occurs.
a2	Optional numeric value used in the optimization process. If defined by the user, no optimization occurs.
log	Logical, indicating whether the data is already log-transformed. Default is TRUE.
verbose	Logical, indicating whether to display messages during the function's execution. Default is TRUE.
meta.info	a character vector of the metadata needed for the model to run and can be retrieved using colData().
BPPARAM	A BiocParallelParam object specifying the parallelization backend (e.g., MulticoreParam, SnowParam). The default depends on the operating system: if the user is on Windows, SnowParam(workers = 2) is used; otherwise, MulticoreParam(workers = 2).
group	A string specifying the column in meta.info that represents the groups or conditions for comparison. group should be retrieved using colData() as factor.
formula.str	A formula string for modeling. It should include "~ 0 + ..." to exclude the intercept from the model. All the model parameters must be present in meta.info.
robust	indicating whether robust fitting should be used. Default is TRUE, see eBayes.
trend	indicating whether to include trend fitting in the differential expression analysis. Default is TRUE. see eBayes.
permutating.group	Logical, If TRUE, the permutation for calculating the null distribution is performed by permuting the target group only specified in group Preserving all the other sample information. If FALSE, the entire sample information retrieved from meta.info will be permuted (recommended to be set to FALSE).
correlation_block	Character or NULL. The name of a column in meta.info that defines correlation blocks. Samples sharing the same value in this column are always resampled together as a unit, and within-block correlation is accounted for during model fitting via duplicateCorrelation. If NULL, standard independent resampling is used.

Details

The **LimROTS** approach initially uses **limma** package functionality to simulate the intensity data of proteins and metabolites. A linear model is subsequently fitted using the design matrix. Empirical Bayes variance shrinking is then implemented. To obtain the moderated t-statistics, the adjusted standard error $SE_{post} = \sqrt{s^2_{\text{post}}} \times unscaledSD$ for each feature is computed, along with the regression coefficient for each feature (indicating the impact of variations in the experimental settings). Then, by adapting a reproducibility-optimized technique known as ROTS to establish an optimality based on the largest overlap of top-ranked features within group-preserving bootstrap datasets, Finally based on the optimized parameters $\alpha 1$ and $\alpha 2$ this equation used to calculates the final statistics:

$$t_{\alpha_{(p)}} = \frac{\beta_{(p)}}{\alpha 1 + \alpha 2 \times SEpost_{(p)}}$$

where $t_{\alpha_{(p)}}$ is the final statistics for each feature, $\beta_{(p)}$ is the coefficient, and $SEpost_{(p)}$ is the the adjusted standard error. LimROTS generates p-values from permutation samples using the implementation available in qvalue package, along with internal implementation of FDR adapted from ROTS package. Additionally, the qvalue package is used to calculate q-values, were the proportion of true null p-values is set to the bootstrap method pi0est.

This function processes a dataset using parallel computation. It leverages the **BiocParallel** framework to distribute tasks across multiple workers, which can significantly reduce runtime for large datasets.

Value

An object of class `"SummarizedExperiment"` with the following elements:

`data`	The original data matrix.
`niter`	The number of bootstrap samples used.
`statistics`	The optimized statistics for each feature.
`pvalue`	P-values computed based on the permutation samples.
`FDR`	False discovery rate estimates.
`a1`	Optimized parameter used in differential expression ranking.
`a2`	Optimized parameter used in differential expression ranking.
`k`	Top list size used for ranking.
`corrected.logfc`	estimate of the log2-fold-change corresponding to the effect corrected by the s model see topTable.
`q_values`	Estimated q-values using the `qvalue` package.
`BH.pvalue`	Benjamini-Hochberg adjusted p-values.
`null.statistics`	The optimized null statistics for each feature.

References

Ritchie, M.E., Phipson, B., Wu, D., Hu, Y., Law, C.W., Shi, W., and Smyth, G.K. (2015). limma powers differential expression analyses for RNA-sequencing and microarray studies. Nucleic Acids Research 43(7), e47

Suomi T, Seyednasrollah F, Jaakkola M, Faux T, Elo L (2017). "ROTS: An R package for reproducibility-optimized statistical testing. " *PLoS computational biology*, *13*(5), e1005562. doi:10.1371/journal.pcbi.1005562 https://doi.org/10.1371/journal.pcbi.1005562, http://www.ncbi.nlm.nih.gov/pubmed/28542205

Elo LL, Filen S, Lahesmaa R, Aittokallio T. Reproducibility-optimized test statistic for ranking genes in microarray studies. IEEE/ACM Trans Comput Biol Bioinform. 2008;5(3):423-431. doi:10.1109/tcbb.2007.1078

Examples

```
# Example usage:

data <- data.frame(matrix(rnorm(500), nrow = 100, ncol = 10))
# Simulated data
meta.info <- data.frame(
    group = factor(rep(1:2, each = 5)),
    row.names = colnames(data)
)
formula.str <- "~ 0 + group"
result <- LimROTS(data,
    meta.info = meta.info, group = "group",
    formula.str = formula.str, niter = 10
)
```

`LimROTS_survival` `LimROTS_survival`*: A Hybrid Method Integrating Empirical Bayes and Reproducibility-Optimized Statistics for Robust survival analysis in Omics Data*

Description

`LimROTS_survival`: A Hybrid Method Integrating Empirical Bayes and Reproducibility-Optimized Statistics for Robust survival analysis in Omics Data

Usage

```
LimROTS_survival(
  x,
  assay.type = NULL,
  niter = 1000,
  K = NULL,
  a1 = NULL,
  a2 = NULL,
  verbose = TRUE,
  meta.info,
  BPPARAM = NULL,
  formula.str,
  competing_risks = FALSE,
  correlation_block = NULL
)
```

Arguments

x	A SummarizedExperiment object, where rows represent features (e.g., proteins, metabolites) and columns represent samples. The values should be log-transformed.
assay.type	A character string or numeric index specifying the assay to use if x is a SummarizedExperiment. Default is NULL (uses the first assay).
niter	An integer representing the amount of bootstrap iterations. Default is 1000.
K	An optional integer representing the top list size for ranking. If not specified, it is set to one-fourth of the number of features.
a1	Optional numeric value used in the optimization process. If defined by the user, no optimization occurs.
a2	Optional numeric value used in the optimization process. If defined by the user, no optimization occurs.
verbose	Logical, indicating whether to display messages during the function's execution. Default is TRUE.
meta.info	a character vector of the metadata needed for the model to run and can be retrieved using colData().
BPPARAM	A BiocParallelParam object specifying the parallelization backend (e.g., MulticoreParam, SnowParam). The default depends on the operating system: if the user is on Windows, SnowParam(workers = 2) is used; otherwise, MulticoreParam(workers = 2).
formula.str	A formula string for modeling. It should include "~ 0 + ..." to exclude the intercept from the model. All the model parameters must be present in meta.info.
competing_risks	Logical. If TRUE, the function will fit a competing risks model using the crr function from the cmprsk package instead of the standard Cox proportional hazards model. Default is FALSE.
correlation_block	Character or NULL. The name of a column in meta.info that defines correlation blocks. Samples sharing the same value in this column are always resampled together as a unit. If NULL, the function behaves identically to bootstrapSamples_limRots.

Details

LimROTS_survival applies the reproducibility-optimized statistic framework to survival analysis. For each bootstrap resample, a Cox proportional hazards model (or a competing risks model via crr from **cmprsk** when competing_risks = TRUE) is fitted per feature using the supplied formula.str. The coefficient $\beta_{(p)}$ and its standard error $SE_{(p)}$ are extracted for each feature across bootstrap datasets.

Reproducibility-optimized parameters α_1 and α_2 are then selected by maximizing the overlap of top-ranked features across group-preserving bootstrap datasets, following the ROTS approach. The final statistic for each feature is:

$$t_{\alpha_{(p)}} = \frac{\beta_{(p)}}{\alpha_1 + \alpha_2 \times SE_{(p)}}$$

where $t_{\alpha_{(p)}}$ is the final statistic, $\beta_{(p)}$ is the Cox model coefficient, and $SE_{(p)}$ is its standard error.

P-values are computed from permutation-based null distributions using empPvals from the **qvalue** package, alongside an internal FDR implementation adapted from the ROTS package. Q-values are estimated via qvalue with the proportion of true null p-values set using the bootstrap method pi0est.

This function processes a dataset using parallel computation. It leverages the **BiocParallel** framework to distribute tasks across multiple workers, which can significantly reduce runtime for large datasets.

Value

A `SummarizedExperiment` when `x` is a `SummarizedExperiment` (results added to `rowData` and `metadata`), or a list with the following elements:

`data`	The original data matrix.
`niter`	The number of bootstrap samples used.
`statistics`	The optimized statistics for each feature.
`pvalue`	P-values computed based on the permutation samples.
`FDR`	False discovery rate estimates.
`a1`	Optimized parameter used in survival ranking.
`a2`	Optimized parameter used in survival ranking.
`k`	Top list size used for ranking.
`R`	Reproducibility score for the optimized parameters.
`Z`	Z-score corresponding to the optimized parameters.
`ztable`	Table of z-scores across the parameter grid.
`exp_coef`	Exponentiated coefficient (hazard ratio estimate) from the Cox proportional hazards model for each feature.
`q_values`	Estimated q-values using the `qvalue` package.
`BH.pvalue`	Benjamini-Hochberg adjusted p-values.
`null.statistics`	The optimized null statistics for each feature.

References

Ritchie, M.E., Phipson, B., Wu, D., Hu, Y., Law, C.W., Shi, W., and Smyth, G.K. (2015). limma powers differential expression analyses for RNA-sequencing and microarray studies. Nucleic Acids Research 43(7), e47

Suomi T, Seyednasrollah F, Jaakkola M, Faux T, Elo L (2017). "ROTS: An R package for reproducibility-optimized statistical testing. " *PLoS computational biology*, *13*(5), e1005562. doi:10.1371/journal.pcbi.1005562 https://doi.org/10.1371/journal.pcbi.1005562, http://www.ncbi.nlm.nih.gov/pubmed/28542205

Elo LL, Filen S, Lahesmaa R, Aittokallio T. Reproducibility-optimized test statistic for ranking genes in microarray studies. IEEE/ACM Trans Comput Biol Bioinform. 2008;5(3):423-431. doi:10.1109/tcbb.2007.1078

Examples

```
set.seed(123)
nsamples <- 20
nfeatures <- 50
sim_data <- matrix(rnorm(nfeatures * nsamples), nrow = nfeatures)
colnames(sim_data) <- paste0("sample", seq_len(nsamples))
rownames(sim_data) <- paste0("gene", seq_len(nfeatures))

meta_data <- data.frame(
    time = abs(rnorm(nsamples, mean = 5, sd = 2)),
    event = sample(0:1, nsamples, replace = TRUE),
    group = factor(rep(seq_len(2), each = nsamples / 2)),
    row.names = colnames(sim_data)
)

formula.str <- "~ Surv(time, event) + group"
result <- LimROTS_survival(
    x = sim_data,
    meta.info = meta_data,
    formula.str = formula.str,
    niter = 10,
    verbose = FALSE,
    competing_risks = FALSE
)
```

`Optimizing` *Optimize Parameters Based on Overlap Calculations*

Description

This function optimizes parameters by calculating overlaps between observed and permuted data for multiple values of a smoothing constant (`ssq`) and a single-label replicate (SLR) comparison.

Usage

```
Optimizing(niter, ssq, N, D, S, pD, pS, verbose)
```

Arguments

`niter`	Integer. Number of bootstrap samples or resampling iterations.
`ssq`	Numeric vector. Smoothing constants to be evaluated.
`N`	Integer vector. Number of top values to consider for overlap calculation.
`D`	Numeric matrix. Observed data values.
`S`	Numeric matrix. Standard errors or related values for observed data.
`pD`	Numeric matrix. Permuted data values.
`pS`	Numeric matrix. Standard errors or related values for permuted data.
`verbose`	Logical. If `TRUE`, progress messages will be displayed.

Details

The function calculates overlaps for a range of smoothing constants and identifies the optimal set of parameters by maximizing a z-score-based metric, which compares the overlap of observed data to permuted data. It computes overlap matrices for both observed (`D` and `S`) and permuted (`pD` and `pS`) data and returns the optimal parameters based on the highest z-score.

Value

A list containing the optimal parameters:

- `a1`: Optimal smoothing constant or 1 for SLR.
- `a2`: SLR flag (1 if smoothing constant is optimal, 0 if SLR is optimal).
- `k`: Optimal number of top values to consider for overlap.
- `R`: Optimal overlap value.
- `Z`: Optimal z-score.
- `ztable`: Matrix of z-scores for all evaluated parameters.

---

permutating_survival *Perform Per-Feature Survival Modeling on Permuted Data*

Description

This function fits a per-feature survival model to the original data matrix using permuted sample metadata, generating a null distribution of test statistics for downstream FDR and p-value estimation.

Usage

```
permutating_survival(x, meta.info, formula.str, competing_risks)
```

Arguments

`x`	A data matrix where rows represent features (e.g., proteins, metabolites) and columns represent samples.
`meta.info`	A data frame of permuted sample metadata, where each row corresponds to a sample. Must include `time`, `event`, and any additional covariates used in `formula.str`.
`formula.str`	A string specifying the formula to be used in model fitting. Must include a `Surv(time, event)` term. The per-feature coefficient term (`y`) is prepended automatically.
`competing_risks`	Logical. If `FALSE` (default), a Cox proportional hazards model is fitted per feature using `coxph`. If `TRUE`, a competing risks model is fitted per feature using `crr` from the `cmprsk` package.

Details

For each feature (row), the function appends the feature expression values as `y` to the permuted metadata and fits either a Cox proportional hazards model (`competing_risks = FALSE`) or a sub-distribution hazard model (`competing_risks = TRUE`). Because the metadata is permuted, the resulting statistics form the null distribution used to compute empirical p-values and FDR.

Unlike `bootstrap_survival`, this function operates directly on the full data matrix without group splitting or resampling.

Value

A list containing the following elements:

`d`	A numeric vector of absolute Cox coefficients ($\|\beta\|$) for each feature.
`s`	A numeric vector of standard errors of the coefficients for each feature.

See Also

`coxph`, `crr`

`SanityChecK` *Sanity Check for Input Data and Parameters*

Description

This function performs a series of checks and initial setups for input data, metadata, and parameters, ensuring everything is correctly formatted for downstream analysis.

Usage

```
SanityChecK(
  x,
  assay.type = NULL,
  niter = 1000,
  K = NULL,
  meta.info,
  group,
  formula.str,
  verbose = TRUE,
  log = TRUE,
  survival = FALSE
)
```

Arguments

`x`	A matrix-like object or a `SummarizedExperiment` containing the data to be analyzed.
`assay.type`	A character string or numeric index specifying the assay to use if `x` is a `SummarizedExperiment`. Default is `NULL`.
`niter`	Integer. Number of bootstrap samples or resampling iterations. Default is 1000.

K	Integer. Top list size. If NULL, it will be set to a quarter of the number of rows in the data matrix. Default is NULL.
meta.info	Data frame. Metadata associated with the samples (columns of x). If x is a SummarizedExperiment, meta.info can be a vector of colData column names to use.
group	Character. Column name in meta.info that defines the groups or conditions for comparison.
formula.str	Optional character string representing the formula for the model.
verbose	Logical, indicating whether to display messages during the function's execution. Default is TRUE.
log	Logical, indicating whether the data is already log-transformed. Default is TRUE.
survival	Logical, indicating whether the analysis is survival analysis. Default is FALSE. If TRUE, meta.info must contain time and event columns.

Details

This function checks whether the input data and metadata are in the correct format, processes metadata from a SummarizedExperiment object if provided, and ensures that group information is correctly specified. If no top list size (K) is provided, it defaults to a quarter of the number of rows in the data.

Value

A list containing:

- meta.info: Processed metadata.
- data: Processed data matrix.
- groups: Numeric or factor vector indicating group assignments.
- K: Top list size to be used in the analysis.

UPS1.Case4 *Spectronaut and ScaffoldDIA UPS1 Spiked Dataset case 4*

Description

A SummarizedExperiment object containing DIA proteomics data from a UPS1-spiked E. coli proteins, processed using both Spectronaut and ScaffoldDIA separately, then merged.

Format

An instance of the SummarizedExperiment class with the following assays:

norm This assay includes log2 protein intensities calculated by averaging the peptides derived from the same protein

The object also contains colData and rowData:

colData A DataFrame with metadata for samples.

rowData A DataFrame with metadata for proteins.

The `colData` contains the following columns:

SampleID Unique identifier for each sample.

Conc Experimental condition or group for each sample , representing different conc. of UPS1-spiked proteins.

tool software tool used, Spectronaut or ScaffoldDIA.

fake.batch A fake digestion batch.

Source

Generated with both spectronaut and ScaffoldDIA separately. using a mixed mode acquisition method and FASTA mode for demonstration purposes.

References

Gotti, C., Roux-Dalvai, F., Joly-Beauparlant, C., Mangnier, L., Leclercq, M., & Droit, A. (2022). DIA proteomics data from a UPS1-spiked E.coli protein mixture processed with six software tools. In Data in Brief (Vol. 41, p. 107829). Elsevier BV. https://doi.org/10.1016/j.dib.2022.107829

Index