Package 'DOTSeq'

May 8, 2026

Type Package

Title Genome-wide Detection of Differential ORF Usage

Version 1.1.0

Description Differential open reading frame (ORF) translation analysis framework for ribosome profiling (Ribo-seq) with matched RNA-seq. Implements (i) Differential ORF Usage (DOU), a beta-binomial generalized linear model that models the expected proportion of Ribo-seq versus RNA-seq reads mapping to each ORF within a gene, and (ii) ORF-level Differential Translation Efficiency (DTE), a negative binomial GLM that capture changes in translation efficiency of individual ORFs across experimental conditions. Supports ORF-level read summarization for bulk and single-cell Ribo-seq.

URL https://github.com/compgenom/DOTSeq

BugReports https://github.com/compgenom/DOTSeq/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

VignetteBuilder knitr

LinkingTo Rcpp

Suggests BSgenome.Hsapiens.UCSC.hg38, TxDb.Hsapiens.UCSC.hg38.knownGene, TxDb.Dmelanogaster.UCSC.dm3.ensGene, org.Hs.eg.db, curl, pasillaBamSubset, BiocStyle, biomaRt, DHARMa, eulerr, ggplot2, ggsignif, knitr, rmarkdown, testthat, withr, magick

Imports ashr, boot, data.table, emmeans, glmmTMB, Matrix, methods, Rcpp, stats, utils, graphics, grDevices, pbapply, AnnotationDbi, BiocGenerics, BiocParallel, Biostrings, BSgenome, txdbmaker, DESeq2, GenomicAlignments, GenomicFeatures, GenomeInfoDb, GenomeInfoDbData, GenomicRanges, IRanges, rtracklayer, Rsamtools, S4Vectors, SummarizedExperiment

biocViews RiboSeq, SingleCell, GeneRegulation, GeneExpression, DifferentialExpression, Genetics, Sequencing, Software, RNASeq, Bayesian, Regression, MultipleComparison

License MIT + file LICENSE

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

git_branch devel

git_last_commit 8d37873

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-05-08

Author Chun Shen Lim [aut, cre] (ORCID:
<https://orcid.org/0000-0001-7015-0125>),
Gabrielle Chieng [aut, ctb] (ORCID:
<https://orcid.org/0009-0008-8710-9979>),
Marsden [fnd]

Maintainer Chun Shen Lim <lim.bioinfo@gmail.com>

Contents

`annotate_orf_type` *Annotate ORF Types from BED File*

Description

This function reads a BED file and converts it into a `GRanges` object. It then finds overlaps with a reference set of ORFs (provided as a `GRanges` object, typically derived from a GFF or DEXSeqDataSet), and annotates each overlapping ORF with a label (e.g., score or type) from the BED file.

Usage

```
annotate_orf_type(bed, gff_granges)
```

Arguments

bed	Character string. Path to a BED file (tab-delimited, no header) with six columns: **chr** Chromosome name **start** Start coordinate (0-based) **end** End coordinate (exclusive) **name** Feature name **score** Annotation label to assign (e.g., ORF type) **strand** Strand information ("+" or "-")
gff_granges	A GRanges object containing ORF coordinates to be annotated.

Value

A GRanges object with an added metadata column orf_type, containing the BED score for overlapping ORFs. Non-overlapping ORFs will have NA.

Examples

```
## Not run:
# Annotate a GRanges object using a BED file's score column
gff_granges <- parseBed("example.bed", granges)
head(orfs)

## End(Not run)
```

---

assign_strategy_levels

Assign strategy levels for RNA-seq and Ribo-seq

Description

This function identifies and assigns the reference and target levels for RNA-seq and Ribo-seq strategies from a given input data frame. It uses a flexible regular expression to match common representations of RNA-seq (e.g., "rna", "RNA", "RNA-seq", or "0") and assigns the remaining unique value(s) as Ribo-seq.

Usage

```
assign_strategy_levels(input_df, strategy_col = "strategy")
```

Arguments

input_df	A data frame containing a column that encodes strategy labels.
strategy_col	A string specifying the name of the column in input_df that contains strategy labels. Default is "strategy".

Value

A named list with two elements:

ribo_level The detected level corresponding to Ribo-seq.

rna_level The detected level corresponding to RNA-seq.

Examples

```
## Not run:
df <- data.frame(strategy = c("RNA", "Ribo"))
assign_strategy_levels(df)

## End(Not run)
```

---

bm_use_ensembl *Internal wrapper for BioMart functions*

Description

Internal wrapper for BioMart functions

Usage

```
bm_use_ensembl(biomart, dataset, host = NULL)
```

---

calculateTE *Calculate Translational Efficiency and Shifts in ORF Usage*

Description

This function computes translational efficiency (TE) and ORF usage for a set of normalized RNA-seq and Ribo-seq counts. TE is calculated as the ratio of Ribo-seq counts to RNA-seq counts. ORF usage is aproximnated as the fraction of total signal (Ribo-proportion and RNA-proportion) that comes from Ribo-proportion, per ORF and condition.

Usage

```
calculateTE(
  counts,
  rna_suffix = ".rna",
  ribo_suffix = ".ribo",
  sample_delim = ".",
  pseudocount = 1e-06
)
```

Arguments

counts	A numeric matrix or data frame of normalized counts, where rows correspond to ORFs and columns correspond to samples. RNA-seq and Ribo-seq reads are distinguished by suffixes.
rna_suffix	Character string indicating the suffix of RNA-seq samples in the column names. Default is ".rna".
ribo_suffix	Character string indicating the suffix of Ribo-seq samples in the column names. Default is ".ribo".
sample_delim	Character. Delimiter used in sample names. Default is ".".
pseudocount	Numeric value added to counts to avoid division by zero. Default is 1e-6.

Value

A matrix of translational efficiency (TE), calculated as the ratio of Ribo-seq to RNA-seq counts for each matched sample (by prefix).

---

calculateUsage *Calculate ORF usage across conditions from a SummarizedExperiment*

Description

This function extracts post hoc results for a given gene ID and computes condition-specific ORF usage using emmeans contrasts.

Usage

```
calculateUsage(data, gene_id)
```

Arguments

data	A DOUData-class object containing DOUResults
gene_id	A character string specifying the gene ID of interest

Details

Usage is computed as the inverse logit of the difference between ribo and RNA strategy emmeans per condition. Only valid emmGrid objects are used. Strategy levels are assigned using a helper function assign_strategy_levels.

Value

A data.frame with ORF IDs, conditions, and usage estimates

Examples

```
# Load a required library
library(SummarizedExperiment)

# Load test data
dir <- system.file("extdata", package = "DOTSeq")
cnt <- read.table(
    file.path(dir, "featureCounts.cell_cycle_subset.txt.gz"),
    header = TRUE,
    comment.char = "#"
)
names(cnt) <- gsub(".*(SRR[0-9]+).*", "\\1", names(cnt))

flat <- file.path(dir, "gencode.v47.orf_flattened_subset.gtf.gz")
bed <- file.path(dir, "gencode.v47.orf_flattened_subset.bed.gz")

meta <- read.table(file.path(dir, "metadata.txt.gz"))
names(meta) <- c("run", "strategy", "replicate", "treatment", "condition")
cond <- meta[meta$treatment == "chx", ]
cond$treatment <- NULL

d <- DOTSeqDataSetsFromFeatureCounts(
    count_table = cnt,
    condition_table = cond,
    flattened_gtf = flat,
    flattened_bed = bed
)

dou <- getDOU(d)

dou <- dou[rowData(dou)$is_kept == TRUE, ]
# Subset only one gene
dou <- dou[rowData(dou)$gene_id == "ENSG00000119402.18", ]

getDOU(d) <- dou

d <- DOTSeq(datasets = d, modules = "DOU")

usage_df <- calculateUsage(
    getDOU(d),
    gene_id = "ENSG00000119402.18"
)
print(usage_df)
```

`calculate_mean_dispersion` *Calculate Moment-Based Dispersion Estimate*

Description

This helper function computes a moment-based estimate of dispersion (intra-class correlation) for a given set of ORF counts and total gene counts across samples. It avoids division by zero and clamps negative dispersion estimates to zero.

Usage

```
calculate_mean_dispersion(counts, totals)
```

Arguments

counts	A numeric vector of ORF counts across samples.
totals	A numeric vector of total gene counts across the same samples.

Value

A list containing:

mean_prop Mean proportion of ORF counts relative to total gene counts.

rawRho Unclamped dispersion estimate (can be negative).

raw_intra_class_corr Clamped dispersion estimate (non-negative).

---

cistronic_data *Prepare Data and Metadata for DOU Heatmap*

Description

Prepares a matrix of differential ORF usage (DOU) estimates for heatmap visualization, comparing mORFs with either uORFs or dORFs. Filters significant ORFs based on local false sign rate (LFSR), clusters genes based on DOU profiles, and retrieves gene symbols for labeling.

Usage

```
cistronic_data(
  results,
  rowdata,
  id_mapping = NULL,
  estimates_col = "posterior",
  estimates_thresh = 1,
  signif_col = "lfsr",
  signif_thresh = 0.05,
  rank_by = "significance",
  top_hits = NULL,
  flip_sign = FALSE,
  sorf_type = "uORF",
  colors = list(low = "blue", middle = "white", high = "red")
)
```

Arguments

results	A data frame from testDOT containing DOU estimates and significance values. Must include columns for ORF IDs, gene IDs, effect size estimates, and LFSR values.
rowdata	A data frame containing ORF-level metadata, including ORF type (e.g., mORF, uORF).

`id_mapping` Optional data frame with gene symbols (e.g., from biomaRt). Used to label heatmap rows with gene symbols.

`estimates_col` Character string specifying the column name for DOU effect size estimates. Default is `"posterior"`.

`estimates_thresh` Numeric threshold for effect size. Default is `1`.

`signif_col` Character string specifying the column name for significance values (LFSR). Default is `"lfsr"`.

`signif_thresh` Numeric threshold for filtering significant ORFs based on LFSR. Default is `0.05`.

`rank_by` Character string specifying how genes should be ranked for visualization. Options are:

- `"score"`: ranks genes by a composite score defined as `abs(effect size) × (1 - lfsr)`, which prioritizes genes with large and confident differential ORF usage.
- `"significance"`: ranks genes by the median significance values across ORFs, prioritizing statistical confidence regardless of effect size. Default is `"significance"`.

`top_hits` Optional numeric. If specified, limits the heatmap to the top N genes ranked by effect size magnitude and significance.

`flip_sign` Logical; if `TRUE`, flips the sign of DOU estimates to align directionality with differential translation. Default is `TRUE`.

`sorf_type` Character string specifying the short ORF type to compare with mORFs. Accepts `"uORF"` or `"dORF"`. Default is `"uORF"`.

`colors` A named list of three colors for heatmap gradient:

- `low`: color for low values (e.g., "blue")
- `middle`: color for mid values (e.g., "white")
- `high`: color for high values (e.g., "red")

Details

This function supports visualization of differential ribosome loading across ORFs within genes. It compares mORFs with a specified short ORF type (e.g., uORFs), filters significant ORFs using local false sign rate (LFSR), and clusters genes based on DOU profiles. Gene symbols are optionally retrieved from Ensembl and used for labeling. The output is designed for downstream heatmap plotting.

Value

A list containing:

ordered_matrix Matrix of DOU estimates for significant genes, ordered by hierarchical clustering.

row_dend_clean Dendrogram object for clustered gene rows.

highlight_df Data frame indicating which ORF-gene tiles to highlight in the heatmap.

gene_labels Vector of gene symbols or Ensembl IDs for heatmap row labeling.

color_palette Color palette used for heatmap visualization.

color_breaks Breaks used for color scaling in the heatmap.

abs_max Maximum absolute value used for symmetric color scaling.

`contrast_vectors` *Generate Contrast Vectors for Pairwise and Baseline Comparisons*

Description

Constructs a named list of contrast vectors for differential analysis using a DESeq2 `DESeqDataSet` object. It identifies interaction terms (e.g., `condition:strategy`) from the model design and builds pairwise contrasts between them, as well as contrasts against a specified baseline condition.

This is useful for extracting custom contrasts not directly available via `resultsNames(dds)` and for post hoc comparisons in complex designs involving interaction terms.

Usage

```
contrast_vectors(dds, formula = NULL, baseline = NULL, delim = ".")
```

Arguments

`dds`	A `DESeqDataSet` object containing count data and sample annotations.
`formula`	Optional. A model formula used to generate the design matrix (e.g., `~ condition * strategy`). If not provided, the function will try to extract it from `design(dds)`.
`baseline`	Optional. A character string specifying the baseline condition for comparisons. If `NULL`, the first level of `condition` in `colData(dds)` is used.
`delim`	A character string used to identify interaction terms in coefficient names. Default is `"."`.

Value

A named list of numeric contrast vectors. Each vector represents a pairwise or baseline comparison. These can be used with `results(dds, contrast = ...)` for custom differential testing.

References

Love, M.I., Huber, W., Anders, S. (2014) Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biology, 15:550. DOI: 10.1186/s13059-014-0550-8

Examples

```
## Not run:
# Generate contrast vectors from a DESeqDataSet
contrast_list <- contrast_vectors(
    dds,
    formula = ~ condition * strategy
)

# Use a contrast vector with DESeq2
res <- lfcShrink(
    dot$dds,
    contrast = contrast_list[[1]],
    type = "ashr"
)

## End(Not run)
```

countReads *Count reads from BAM files over genomic features*

Description

Uses summarizeOverlaps to count reads overlapping genomic ranges, handling both single-end and paired-end BAM files.

Usage

```
countReads(
  gr,
  bam_files,
  ignore.strand = list(single = TRUE, paired = FALSE),
  verbose = TRUE
)
```

Arguments

gr	A GRanges object representing genomic features (e.g., ORFs).
bam_files	Character vector. Paths to BAM files.
ignore.strand	A named list with logical values for single and paired indicating whether to ignore strand information.
verbose	Logical. Whether to print progress messages.

Value

A matrix of read counts with features as rows and samples as columns.

Examples

```
library(GenomicRanges)
library(pasillaBamSubset)

bam_list <- c(untreated1_chr4(), untreated3_chr4())

gr <- GRanges(seqnames = "chr4",
    ranges = IRanges(start = 233, end = 2300))

countReads(gr = gr, bam_files = bam_list)
```

`countReadsSingleCell` *Count single-cell reads/UMIs from BAM over genomic features (optimized)*

Description

Vectorized, memory-efficient counting from BAMs with CB/UB tags. Overlaps are computed against per-seqname NCList indexes; UMIs are deduplicated per (cell, feature) if provided. Results are sparse matrices (features × cells).

Usage

```
countReadsSingleCell(
  gr,
  bam,
  seqlevels_style = "UCSC",
  tags = list(cell = "CB", umi = "UB"),
  ignore.strand = TRUE,
  yieldSize = 1e+06,
  mapq = 10,
  dedup = TRUE,
  mode = c("Union", "IntersectionStrict", "IntersectionNotEmpty"),
  cells = NULL,
  verbose = TRUE,
 BPPARAM = MulticoreParam(workers = 12, stop.on.error = FALSE, progressbar = TRUE)
)
```

Arguments

`gr`	GRanges of genomic features (e.g., ORFs). Must be on the same reference naming style as BAMs or convertible via seqlevelsStyle.
`bam`	Character vector of BAM paths (STARsolo/CellRanger or similar).
`seqlevels_style`	Character; the naming style for chromosome identifiers (e.g., `"UCSC"`, `"NCBI"`). This is applied to both the GRanges object and the TxDb annotation. Default is `"UCSC"`.
`tags`	Named list with `cell` and `umi` tag names (default CB/UB).
`ignore.strand`	Logical: whether to ignore strand in overlaps.
`yieldSize`	Integer chunk size for streaming from BAM.
`mapq`	Minimum MAPQ to keep (default 10).
`dedup`	Logical: if TRUE and UMI tag present, deduplicate UMIs per (cell, feature).
`mode`	Overlap mode; currently supports "Union" (others placeholder for extension).
`cells`	Optional character vector of barcodes to keep; if provided, columns are restricted to these and ordered accordingly.
`verbose`	Logical for progress messages.
`BPPARAM`	A BiocParallelParam. Default: `BiocParallel::MulticoreParam(workers = 12, stop.on.error = FALSE, progressbar = TRUE)`.

Value

A sparse dgCMatrix (features × cells), with rownames taken from `names(gr)` if present.

Examples

```
suppressPackageStartupMessages({
  library(GenomicRanges)
  library(Rsamtools)
  library(GenomeInfoDb)
  library(Matrix)
  library(BiocParallel)
})
# Minimal GRanges with two ORF-like features on chr1
gr <- GRanges("chr1", IRanges(c(90, 140), width = 40))
names(gr) <- c("orf1", "orf2")

# Write a tiny SAM with CB/UB tags for two cells (cellA, cellB)
# r1 and r2 share the same (cell, UMI) and overlap orf1 -> should deduplicate to 1
# r3 overlaps orf2 in cellA; r4 overlaps orf2 in cellB
sam <- c(
  "@HD\tVN:1.6\tSO:coordinate",
  "@SQ\tSN:chr1\tLN:1000",
 "r1\t0\tchr1\t100\t60\t20M\t*\t0\t0\tACGTACGTACGTACGTACGT\tFFFFFFFFFFFFFFFFFFFF\tCB:Z:cellA\tUB:Z:umi1",
 "r2\t0\tchr1\t110\t60\t20M\t*\t0\t0\tACGTACGTACGTACGTACGT\tFFFFFFFFFFFFFFFFFFFF\tCB:Z:cellA\tUB:Z:umi1", #
 "r3\t0\tchr1\t150\t60\t20M\t*\t0\t0\tACGTACGTACGTACGTACGT\tFFFFFFFFFFFFFFFFFFFF\tCB:Z:cellA\tUB:Z:umi2",
 "r4\t0\tchr1\t150\t60\t20M\t*\t0\t0\tACGTACGTACGTACGTACGT\tFFFFFFFFFFFFFFFFFFFF\tCB:Z:cellB\tUB:Z:umi1"
)
td <- tempdir()
sam_path <- file.path(td, "toy.sam")
writeLines(sam, sam_path)
dest_base <- file.path(td, "toy")
bam_path <- Rsamtools::asBam(sam_path, destination = dest_base, overwrite = TRUE)
bam_path <- paste0(dest_base, ".bam")
Rsamtools::indexBam(bam_path)

# Count per (feature × cell) with UMI deduplication; use SerialParam for portability
mat <- countReadsSingleCell(
  gr = gr,
  bam = bam_path,
  seqlevels_style = "UCSC",
  tags = list(cell = "CB", umi = "UB"),
  mapq = 10,
  dedup = TRUE,
  BPPARAM = BiocParallel::SerialParam()
)

# Inspect the sparse matrix
print(dim(mat))
print(colnames(mat))
as.matrix(mat)
```

`create_datasets` *Create DOU and DESeq2 datasets for differential translation analysis*

Description

This internal function constructs a DOUData object and a DESeqDataSet object from raw count data, sample metadata, and ORF annotations. It applies filtering based on count thresholds and singlet status, prepares metadata for modeling, and stores formulas and contrast specifications for downstream analysis.

Usage

```
create_datasets(
  count_table,
  condition_table,
  annotation,
  reduced_formula,
  emm_specs,
  deseq_formula,
  min_count = 1,
  stringent = TRUE,
  verbose = TRUE
)
```

Arguments

count_table	A matrix or data frame of raw counts. Columns must match sample names in condition_table.
condition_table	A data frame containing sample metadata. Must include columns: run, strategy, condition, and replicate.
annotation	A GRanges object containing ORF annotations, typically parsed from a flattened GTF or BED file.
reduced_formula	A formula object specifying the reduced model used for estimating marginal means (e.g., ~ condition + strategy).
emm_specs	A list of specifications for estimated marginal means contrasts, typically generated using emmeans::contrast().
deseq_formula	A formula object specifying the design for DESeq2 modeling (e.g., ~ condition * strategy).
min_count	Integer specifying the minimum count threshold for filtering ORFs. Default is 1.
stringent	Logical or NULL; determines the filtering strategy: TRUE Keep ORFs where all replicates in at least one condition pass min_count. FALSE Keep ORFs where all replicates in at least one condition-strategy group pass min_count. NULL Keep ORFs where total counts across all samples pass min_count.
verbose	Logical; if TRUE, prints progress and runtime messages. Default is TRUE.

Value

A list containing:

sumExp A DOUData object containing raw counts, metadata, and filtering status.

dds A DESeqDataSet object prepared for differential translation efficiency analysis.

create_read_numbers *Generate a simulated data set based on known model parameters*

Description

Generate a simulated data set based on known model parameters

Usage

```
create_read_numbers(mu, fit, p0, m = NULL, n = NULL, mod = NULL, beta = NULL)
```

Arguments

mu	Baseline mean expression for negative binomial model.
fit	Fitted relationship between log mean and log size.
p0	A vector of the probabilities a count is zero.
m	Number of genes/transcripts to simulate (not necessary if mod, beta are specified).
n	Number of samples to simulate (not necessary if mod, beta are specified).
mod	Model matrix you would like to simulate from without an intercept.
beta	Set of coefficients for the model matrix (must have same number of columns as mod).

Value

A list containing:

counts Data matrix with counts for genes in rows and samples in columns.

Author(s)

Jeff Leek

DOTSeq *DOTSeq: Differential Analysis of Translation Efficiency and Usage of Open Reading Frames*

Description

DOTSeq provides a flexible beta-binomial generalized linear model (GLM) framework for modeling the expected proportion of ribosome profiling (Ribo-seq) to RNA-seq counts for individual open reading frames (ORFs) relative to other ORFs within the same gene. It also includes a negative binomial GLM framework for detecting changes in translation efficiency across experimental conditions.

DOTSeq is a statistical framework for modeling Differential ORF Translation using ribosome profiling (Ribo-seq) and RNA-seq data. It supports two modes of input:

- A named list of raw input components: `count_table`, `condition_table`, `flattened_gtf`, and `bed`.
- A pre-constructed `DOTSeqDataSets-class` object containing processed data.

The function automatically detects the input type and proceeds with the appropriate workflow. It performs ORF-level filtering, model fitting, post hoc contrasts, and adaptive shrinkage of effect sizes. Plotting and downstream analysis are handled separately via the `plotDOT` function.

Usage

```
DOTSeq(
  datasets,
  formula = ~condition * strategy,
  modules = c("DOU", "DTE"),
  target = NULL,
  baseline = NULL,
  min_count = 1,
  stringent = TRUE,
  dispersion_modeling = "auto",
  dispformula = NULL,
  lrt = FALSE,
  diagnostic = FALSE,
  parallel = list(n = 4L, autopar = TRUE),
  optimizers = FALSE,
  nullweight = 500,
  contrasts_method = "revpairwise",
  verbose = TRUE
)
```

Arguments

`datasets`	Either: `DOTSeqDataSets-class` **object** A pre-constructed `DOTSeqDataSets-class` object created using `DOTSeqDataSets-class`. It must include: `DOU` A `DOUData-class` object containing filtered raw counts, sample metadata, and ORF-level annotations. `DTE` A `DESeqDataSet-class` object used for modeling DTE via DESeq2. If a `DOTSeqDataSets-class` object is provided, the function skips raw input parsing and uses these objects directly.
`formula`	A formula object specifying the design. Default is `~ condition * strategy`.
`modules`	Character vector specifying which **DOTSeq** modules to run. Options include `"DOU"` and `"DTE"`. Default is `c("DOU", "DTE")`.
`target`	Character string specifying the non-reference condition level to extract the corresponding interaction term. Default is `NULL`.
`baseline`	Character string specifying the desired reference level. Default is `NULL`.
`min_count`	Minimum count threshold for filtering ORFs. Default is `1`.
`stringent`	Logical or `NULL`; determines the filtering strategy: `TRUE` Keep ORFs where all replicates in at least one condition pass `min_count`. `FALSE` Keep ORFs where all replicates in at least one condition-strategy group pass `min_count`.

NULL Keep ORFs where total counts across replicates pass min_count.

dispersion_modeling
: String specifying the dispersion modeling approach for DOU. Options include "auto", "shared", or "custom". Default is "auto".

dispformula
: Optional formula object for custom dispersion modeling.

lrt
: Logical; if TRUE, performs a likelihood ratio test comparing full vs reduced models in DOU. Default is FALSE.

diagnostic
: Logical; if TRUE, enables model diagnostics in DOU. Default is FALSE.

parallel
: A list passed to glmmTMBControl to configure parallel optimization in DOU. Default is list(n = 4L, autopar = TRUE).

optimizers
: Logical; if TRUE, enables brute-force optimization using multiple optimizers in glmmTMBControl. Default is FALSE.

nullweight
: Numeric. Prior weight on the null hypothesis for empirical Bayes shrinkage in DOU. Default is 500.

contrasts_method
: Character string specifying the method for post hoc contrasts in DOU. Default is "revpairwise".

verbose
: Logical; if TRUE, prints progress messages. Default is TRUE.

Value

A DOTSeqDataSets-class object containing:

DOU A DOUData-class object with DOU results.

DTE A DTEData-class object with DTE results.

Author(s)

Maintainer: Chun Shen Lim <lim.bioinfo@gmail.com> (ORCID)

Authors:

- Gabrielle Chieng (ORCID) [contributor]

Other contributors:

- Marsden [funder]

References

Brooks, M. E., Kristensen, K., van Benthem, K. J., Magnusson, A., Berg, C. W., Nielsen, A., Skaug, H. J., Mächler, M. and Bolker, B. M. (2017). glmmTMB balances speed and flexibility among packages for zero-inflated generalized linear mixed modeling. The R Journal, 378–400. DOI: 10.32614/RJ-2017-066

Love, M. I., Huber, W., Anders, S. (2014). Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biology, 15:550. DOI: 10.1186/s13059-014-0550-8

Lenth, R., Piaskowski, J. (2025). emmeans: Estimated Marginal Means, aka Least-Squares Means. R package version 2.0.0. https://rvlenth.github.io/emmeans/

Stephens, M. (2016). False discovery rates: a new deal. Biostatistics, 18(2). DOI: 10.1093/biostatistics/kxw041

Hartig, F. (2025). DHARMa: Residual Diagnostics for Hierarchical (Multi-Level / Mixed) Regression Models. R package version 0.4.7. https://github.com/florianhartig/dharma

See Also

Useful links:

- https://github.com/compgenom/DOTSeq
- Report bugs at https://github.com/compgenom/DOTSeq/issues

DOTSeqDataSets-class, fitDOU, testDOU, plotDOT

Examples

```
# Load test data
dir <- system.file("extdata", package = "DOTSeq")

cnt <- read.table(
    file.path(dir, "featureCounts.cell_cycle_subset.txt.gz"),
    header = TRUE,
    comment.char = "#"
)
names(cnt) <- gsub(".*(SRR[0-9]+).*", "\\1", names(cnt))

gtf <- file.path(dir, "gencode.v47.orf_flattened_subset.gtf.gz")
bed <- file.path(dir, "gencode.v47.orf_flattened_subset.bed.gz")

meta <- read.table(file.path(dir, "metadata.txt.gz"))
names(meta) <- c("run", "strategy", "replicate", "treatment", "condition")
cond <- meta[meta$treatment == "chx", ]
cond$treatment <- NULL

# Use raw input list
d <- DOTSeqDataSetsFromFeatureCounts(
    count_table = cnt,
    condition_table = cond,
    flattened_gtf = gtf,
    flattened_bed = bed
)

d <- DOTSeq(datasets = d, modules = "DTE")

show(d)

# Get the DOUData object
dou <- getDOU(d)

# Subset DOUData and edit DOTSeqDataSets in place
set.seed(42)
random_rows <- sample(seq_len(nrow(dou)), size = 50)
getDOU(d) <- dou[random_rows, ]

# Run the DOU module
d <- DOTSeq(datasets = d)

# Create a DOTSeqDataSets object and use it as input
d <- DOTSeqDataSetsFromFeatureCounts(
    count_table = cnt,
    condition_table = cond,
    flattened_gtf = gtf,
```

```
    flattened_bed = bed
)
d <- DOTSeq(datasets = d, modules = "DTE")
```

DOTSeqDataSets-class *DOTSeqDataSets-class*

Description

A wrapper class to store both DOU and DTE results from **DOTSeq** analysis.

Displays a summary of the DOTSeqDataSets object.

Usage

```
## S4 method for signature 'DOTSeqDataSets'
show(object)
```

Arguments

object	A DOTSeqDataSets-class object.

Value

A DOTSeqDataSets-class S4 object containing DOU and DTE results.

Slots

DOU A DOUData-class object.

DTE A DTEData-class object.

Examples

```
# Read in count matrix, condition table, and annotation files
dir <- system.file("extdata", package = "DOTSeq")

cnt <- read.table(
    file.path(dir, "featureCounts.cell_cycle_subset.txt.gz"),
    header = TRUE,
    comment.char = "#"
)
names(cnt) <- gsub(".*(SRR[0-9]+).*", "\\1", names(cnt))

flat <- file.path(dir, "gencode.v47.orf_flattened_subset.gtf.gz")
bed <- file.path(dir, "gencode.v47.orf_flattened_subset.bed.gz")

meta <- read.table(file.path(dir, "metadata.txt.gz"))
names(meta) <- c("run", "strategy", "replicate", "treatment", "condition")
cond <- meta[meta$treatment == "chx", ]
cond$treatment <- NULL

# Create a DOTSeqDataSets object
d <- DOTSeqDataSetsFromFeatureCounts(
```

```
    count_table = cnt,
    condition_table = cond,
    flattened_gtf = flat,
    flattened_bed = bed
)

getDOU(d)

getDTE(d)
```

DOTSeqDataSetsFromFeatureCounts *Construct DOTSeqDatasets from featureCounts output for Differential ORF Translation Analysis*

Description

This function initialize and construct the DOTSeqDataSets-class object. This includes loading count and metadata tables, parsing ORF annotations, filtering ORFs based on count thresholds, and preparing objects for downstream differential translation analysis using beta-binomial and negative binomial GLM.

This function initialize and construct the DOTSeqDataSets-class object. This includes loading count and metadata tables, read a GRanges object with ORF-level annotation, filtering ORFs based on count thresholds, and preparing objects for downstream differential translation analysis using beta-binomial and negative binomial GLM.

Usage

```
DOTSeqDataSetsFromFeatureCounts(
  count_table,
  condition_table,
  flattened_gtf,
  flattened_bed,
  formula = ~condition * strategy,
  target = NULL,
  baseline = NULL,
  min_count = 1,
  stringent = TRUE,
  verbose = TRUE
)

DOTSeqDataSetsFromSummarizeOverlaps(
  count_table,
  condition_table,
  annotation,
  formula = ~condition * strategy,
  target = NULL,
  baseline = NULL,
  min_count = 1,
  stringent = TRUE,
  verbose = TRUE
)
```

Arguments

count_table	A dataframe of read counts with features as rows and samples as columns. Generated from countReads based on summarizeOverlaps.
condition_table	Path to a sample metadata file or a data frame. Must include columns: run, strategy, condition, replicate.
flattened_gtf	Path to a flattened GFF/GTF file containing ORF annotations.
flattened_bed	Path to a flattened BED file with ORF annotations.
formula	A formula object specifying the design. Default is ~ condition * strategy.
target	Character string specifying the non-reference condition level to extract the corresponding interaction term. Contrasted against the baseline condition. Default is NULL.
baseline	Character string specifying the desired reference level. Default is NULL.
min_count	Minimum count threshold for filtering ORFs. Default is 1.
stringent	Logical or NULL; determines the filtering strategy: TRUE Keep ORFs where all replicates in at least one condition pass min_count. FALSE Keep ORFs where all replicates in at least one condition-strategy group pass min_count. NULL Keep ORFs where total counts across replicates pass min_count.
verbose	Logical; if TRUE, prints progress messages. Default is TRUE.
annotation	A GRanges object with ORF level annotation, typically obtained from getORFs.

Value

A DOTSeqDataSets-class object containing:

DOU A DOUData-class object containing pre-filtered raw counts (assay slot), sample metadata (colData slot), and ORF-level annotation (rowRanges) used for modeling Differential ORF Usage (DOU).

DTE A DTEData-class object used for modeling Differential Translation Efficiency (DTE). Stores all data above except for rowRanges.

A DOTSeqDataSets-class object containing:

DOU A DOUData-class object containing pre-filtered raw counts (assay slot), sample metadata (colData slot), and ORF-level annotation (rowRanges) used for modeling Differential ORF Usage (DOU).

DTE A DTEData-class object used for modeling Differential Translation Efficiency (DTE). Stores all data above except for rowRanges.

See Also

DOTSeqDataSetsFromFeatureCounts

DOTSeqDataSetsFromFeatureCounts

Examples

```
# Read in count matrix, condition table, and annotation files
dir <- system.file("extdata", package = "DOTSeq")

cnt <- read.table(
    file.path(dir, "featureCounts.cell_cycle_subset.txt.gz"),
    header = TRUE,
    comment.char = "#"
)
names(cnt) <- gsub(".*(SRR[0-9]+).*", "\\1", names(cnt))

gtf <- file.path(dir, "gencode.v47.orf_flattened_subset.gtf.gz")
bed <- file.path(dir, "gencode.v47.orf_flattened_subset.bed.gz")

meta <- read.table(file.path(dir, "metadata.txt.gz"))
names(meta) <- c("run", "strategy", "replicate", "treatment", "condition")
cond <- meta[meta$treatment == "chx", ]
cond$treatment <- NULL

# Create a DOTSeqDataSets object
d <- DOTSeqDataSetsFromFeatureCounts(
    count_table = cnt,
    condition_table = cond,
    flattened_gtf = gtf,
    flattened_bed = bed
)

show(d)
```

`DOUData-class` *DOUData-class*

Description

A `RangedSummarizedExperiment` object extended to store modeling components for Differential ORF Usage (DOU) analysis in the **DOTSeq** framework.

Details

This class contains raw counts, sample metadata, and additional slots for the model formula, `emmeans` specifications, and contrast results. It supports flexible modeling of translation-specific effects using GLM / GLMM and post hoc contrasts.

Slots

`formula` A `formula` object specifying the model design (e.g., ~ condition * strategy).

`specs` A `formula` object used to generate `emmeans` specifications for post hoc contrasts.

`interaction` A `DFrame` containing results from interaction-specific contrasts.

`strategy` A `DFrame` containing results from strategy-specific contrasts.

See Also

RangedSummarizedExperiment, glmmTMB, emmeans

Examples

```
# Read in count matrix, condition table, and annotation files
dir <- system.file("extdata", package = "DOTSeq")

cnt <- read.table(
    file.path(dir, "featureCounts.cell_cycle_subset.txt.gz"),
    header = TRUE,
    comment.char = "#"
)
names(cnt) <- gsub(".*(SRR[0-9]+).*", "\\1", names(cnt))

flat <- file.path(dir, "gencode.v47.orf_flattened_subset.gtf.gz")
bed <- file.path(dir, "gencode.v47.orf_flattened_subset.bed.gz")

meta <- read.table(file.path(dir, "metadata.txt.gz"))
names(meta) <- c("run", "strategy", "replicate", "treatment", "condition")
cond <- meta[meta$treatment == "chx", ]
cond$treatment <- NULL

# Create a DOTSeqDataSets object
d <- DOTSeqDataSetsFromFeatureCounts(
    count_table = cnt,
    condition_table = cond,
    flattened_gtf = flat,
    flattened_bed = bed
)

getDOU(d)

getDTE(d)
```

DOUData-validity *Validity method for DOUData*

Description

Checks that the slots in a DOUData object are correctly populated.

Arguments

object	A DOUData-class object.

Value

TRUE if valid, or a character string describing the error.

DTEData-class *DTEData-class*

Description

A DESeqDataSet object extended to store modeling components for Differential Translation Efficiency (DTE) analysis in the **DOTSeq** framework.

Details

This class contains raw counts, sample metadata, and additional slots for the emmeans specifications, and contrast results.

Inherits all slots from DESeqDataSet, and adds:

- specs: A formula object used to generate emmeans specifications for post hoc contrasts.
- interaction: A DFrame or data.frame containing results interaction-specific contrasts
- strategy: A DFrame or data.frame containing results strategy-specific contrasts

See Also

DESeqDataSet-class

Examples

```
# Read in count matrix, condition table, and annotation files
dir <- system.file("extdata", package = "DOTSeq")

cnt <- read.table(
    file.path(dir, "featureCounts.cell_cycle_subset.txt.gz"),
    header = TRUE,
    comment.char = "#"
)
names(cnt) <- gsub(".*(SRR[0-9]+).*", "\\1", names(cnt))

flat <- file.path(dir, "gencode.v47.orf_flattened_subset.gtf.gz")
bed <- file.path(dir, "gencode.v47.orf_flattened_subset.bed.gz")

meta <- read.table(file.path(dir, "metadata.txt.gz"))
names(meta) <- c("run", "strategy", "replicate", "treatment", "condition")
cond <- meta[meta$treatment == "chx", ]
cond$treatment <- NULL

# Create a DOTSeqDataSets object
d <- DOTSeqDataSetsFromFeatureCounts(
    count_table = cnt,
    condition_table = cond,
    flattened_gtf = flat,
    flattened_bed = bed
)

getDOU(d)

getDTE(d)
```

extract_results *Extract Model Results into a Unified Data Frame*

Description

Extracts scalar results from a named list of model objects from glmmTMB fits into a tidy data frame. Handles nested lists, missing or NULL models, and ensures consistent columns across all entries.

Usage

```
extract_results(sumExp, verbose = TRUE)
```

Arguments

sumExp	A SummarizedExperiment object containing fitted model objects stored in rowData(sumExp)[['DOUF
verbose	Logical. If TRUE, messages will be printed for skipped or failed models.

Details

The function uses a recursive helper (flatten_scalars) to extract scalar values from nested lists. It supports model type "glmmTMB", and gracefully handles NULL or unsupported model types by returning minimal rows.

Missing columns across models are filled with NA to ensure a consistent structure.

Value

A data frame where each row corresponds to an ORF and its associated model results. Columns include scalar parameters extracted from the model, orf_id, and model_type.

filter_gtf *Filter GTF file for transcript features with required metadata*

Description

This function imports a GTF file and filters transcript features based on the presence of required metadata columns and an optional source filter.

Usage

```
filter_gtf(
  gtf_file,
  require_ids = c("hgnc_id", "protein_id", "ccdsid"),
  source_filter = NULL,
  verbose = TRUE
)
```

Arguments

gtf_file	Character. Path to the GTF file.
require_ids	Character vector. Metadata column names that must be present and non-NA. Default: c("hgnc_id", "protein_id","ccdsid").
source_filter	Character or NULL. If provided, filters transcripts by the 'source' column. Default: NULL.
verbose	Logical. Whether to print summary messages. Default: TRUE.

Value

A filtered GRanges object containing transcript features.

---

findORFsFasta *Find ORFs in FASTA sequences using ORFik's C++ engine*

Description

This function identifies ORFs in DNA sequences from FASTA files, DNAStringSet, or BSgenome objects. It supports both linear and circular genomes, and can detect ORFs on the sense (+) or both strands.

Usage

```
findORFsFasta(
  sequences,
  start_codons = "ATG",
  stop_codons = "TAA",
  min_len = 0,
  longest_orf = TRUE,
  is_circular = FALSE,
  plus_strand_only = TRUE
)
```

Arguments

sequences	Character path to a FASTA file, or a DNAStringSet or BSgenome object.
start_codons	Character string of start codons (e.g., "ATG|GTG")
stop_codons	Character string of stop codons (e.g., "TAA|TAG")
min_len	Integer. Minimum ORF length in bases. Default is 0.
longest_orf	Logical. If TRUE, return only the longest ORF per sequence.
is_circular	Logical. Whether the genome is circular (e.g., bacterial genomes).
plus_strand_only	Logical. If TRUE, scan only the forward strand; if FALSE, scan both strands.

Details

This method is optimized for prokaryotic genomes or transcript sequences. Direct use for eukaryotic whole genomes is not suitable due to the presence of splicing. See also **ORFik**'s `findMapORFs`.

Each FASTA header is treated independently, and the name (up to the first space) is used as the `seqnames` in the returned `GRanges` object. Circular genome support is included, and ORFs that span the start/end boundary are handled.

Note: Ensure your FASTA file is valid and headers are formatted as `>name info`, with the name first and no hidden characters, as this affects coordinate parsing.

Value

A `GRanges` object containing the ORFs found.

Author(s)

Haakon Tjeldnes et al. (original), Chun Shen Lim (modification).

References

Tjeldnes, H., Labun, K., Torres Cleuren, Y. et al. ORFik: a comprehensive R toolkit for the analysis of translation. BMC Bioinformatics 22, 336 (2021). DOI: 10.1186/s12859-021-04254-w

---

`fitDOU` *Fit Differential ORF Usage Models*

Description

This function fits beta-binomial generalized (mixed) linear models (GLM or GLMM) for Differential ORF Usage (DOU) across all genes (via `glmmTMB`). It supports multiple dispersion modeling approaches and optional diagnostics using **DHARMa**.

Usage

```
fitDOU(
  data,
  formula = ~condition * strategy,
  specs = ~condition * strategy,
  dispformula = NULL,
  dispersion_modeling = "auto",
  lrt = FALSE,
  diagnostic = FALSE,
  parallel = list(n = 4L, autopar = TRUE),
  optimizers = FALSE,
  verbose = TRUE
)
```

Arguments

data	A DOUData-class object containing raw ORF-level counts and sample metadata. This object is used as the input for modeling DOU within the **DOTSeq** framework.
formula	A formula object specifying the model design, e.g., ~ condition * strategy.
specs	A formula specifying the structure of the estimated marginal means. Default is ~condition * strategy.
dispformula	Optional formula object specifying a custom dispersion model (used when dispersion_modeling = "custom").
dispersion_modeling	Character string specifying the dispersion modeling strategy. Options are: "auto" Fit both strategy-dependent and shared dispersion models, and select the best via likelihood ratio test. "strategy" Model dispersion as a function of sequencing strategy. "shared" Assume constant dispersion across all predictor levels. "custom" Use a user-specified dispersion formula via dispformula.
lrt	Logical; if TRUE, performs a likelihood ratio test to compare the full model (with interaction) against a reduced model (without interaction) to assess translation-specific effects. Default is FALSE.
diagnostic	Logical; if TRUE, runs **DHARMa** diagnostics to assess model fit. Default is FALSE.
parallel	A list passed to glmmTMBControl to configure parallel optimization, e.g., list(parallel = TRUE, ncpus = 4). Default is list(n = 4L, autopar = TRUE).
optimizers	Logical; if TRUE, enables brute-force optimization using multiple optimizers in glmmTMBControl. Default is FALSE.
verbose	Logical; if TRUE, prints progress messages. Default is TRUE.

Value

A named list of PostHoc objects, one per ORF.

References

Brooks, M. E., Kristensen, K., van Benthem, K. J., Magnusson, A., Berg, C. W., Nielsen, A., Skaug, H. J., Mächler, M. and Bolker, B. M. (2017). glmmTMB balances speed and flexibility among packages for zero-inflated generalized linear mixed modeling. The R Journal, 378–400. DOI: 10.32614/RJ-2017-066

Lenth R, Piaskowski J (2025). emmeans: Estimated Marginal Means, aka Least-Squares Means. R package version 2.0.0. https://rvlenth.github.io/emmeans/

Hartig F (2025). DHARMa: Residual Diagnostics for Hierarchical (Multi-Level / Mixed) Regression Models. R package version 0.4.7. https://github.com/florianhartig/dharma

See Also

DOTSeq, DOTSeqDataSets-class, testDOU

Examples

```
# Load SummarizedExperiment to enable subsetting and access to components
# like rowRanges and rowData
library(SummarizedExperiment)

# Read in count matrix, condition table, and annotation files
dir <- system.file("extdata", package = "DOTSeq")

cnt <- read.table(
    file.path(dir, "featureCounts.cell_cycle_subset.txt.gz"),
    header = TRUE,
    comment.char = "#"
)
names(cnt) <- gsub(".*(SRR[0-9]+).*", "\\1", names(cnt))

gtf <- file.path(dir, "gencode.v47.orf_flattened_subset.gtf.gz")
bed <- file.path(dir, "gencode.v47.orf_flattened_subset.bed.gz")

meta <- read.table(file.path(dir, "metadata.txt.gz"))
names(meta) <- c("run", "strategy", "replicate", "treatment", "condition")
cond <- meta[meta$treatment == "chx", ]
cond$treatment <- NULL

# Create a DOTSeqDataSets object
d <- DOTSeqDataSetsFromFeatureCounts(
    count_table = cnt,
    condition_table = cond,
    flattened_gtf = gtf,
    flattened_bed = bed
)

dou <- getDOU(d)

# Keep ORFs where all replicates in at least one condition pass min_count
# Single-ORF genes are removed
dou <- dou[rowRanges(dou)$is_kept == TRUE, ]

# Randomly sample 100 ORFs for fitDOU
set.seed(42)
random_rows <- sample(seq_len(nrow(dou)), size = 100)
dou <- dou[random_rows, ]

# Model fitting using fitDOU
rowData(dou)[["DOUResults"]] <- fitDOU(
    data = dou,
    formula = ~ condition * strategy,
    specs = ~ condition * strategy,
    dispersion_modeling = "auto",
    lrt = FALSE,
    optimizers = FALSE,
    diagnostic = FALSE,
    parallel = list(n = 4L, autopar = TRUE),
    verbose = TRUE
)
```

`fit_beta_binomial` *Fit Beta-Binomial Models for a Single Gene*

Description

This internal worker function fits beta-binomial generalized (mixed) linear models (GLM or GLMM) for all ORFs within a single gene using `glmmTMB`. It supports multiple dispersion modeling approaches and optional model diagnostics.

Usage

```
fit_beta_binomial(
  ribo_mat,
  rna_mat,
  coldata,
  formula = ~condition * strategy,
  specs = ~condition * strategy,
  dispersion_modeling = c("auto", "shared", "custom"),
  dispformula = NULL,
  lrt = FALSE,
  diagnostic = FALSE,
  optimizers = FALSE,
  parallel = list(n = 4L, autopar = TRUE),
  verbose = FALSE
)
```

Arguments

`ribo_mat`	A numeric matrix of Ribo-seq counts for a single gene (rows = ORFs, columns = samples).
`rna_mat`	A numeric matrix of RNA-seq counts for the same gene (same dimensions as `ribo_mat`).
`coldata`	A data frame containing sample annotations. Must include columns such as `condition`, `strategy`, and `replicate`.
`formula`	A formula object specifying the model design, e.g., `~ condition * strategy`.
`dispersion_modeling`	Character string specifying the dispersion modeling strategy. Options are: `"auto"` Fit both strategy-dependent and shared dispersion models, and select the best via likelihood ratio test. `"shared"` Assume constant dispersion across all predictor levels. `"custom"` Use a user-specified dispersion formula via `dispformula`.
`dispformula`	Optional formula object specifying a custom dispersion model (used when `dispersion_modeling = "custom"`).
`lrt`	Logical; if `TRUE`, performs a likelihood ratio test to compare the full model (with interaction) against a reduced model (without interaction) to assess translation-specific effects. Default is `FALSE`.
`diagnostic`	Logical; if `TRUE`, runs **DHARMa** diagnostics to assess model fit. This requires **DHARMa** to be installed. Default is `FALSE`.

optimizers	Logical; if TRUE, enables brute-force optimization using multiple optimizers in glmmTMBControl. Default is FALSE.
parallel	A list passed to glmmTMBControl to configure parallel optimization, e.g., list(parallel = TRUE, ncpus = 4). Default is list(n = 4L, autopar = TRUE).
verbose	Logical; if TRUE, prints progress messages. Default is FALSE.

Value

A named list of PostHoc objects, one for each ORF in the gene.

References

Brooks, M. E., Kristensen, K., van Benthem, K. J., Magnusson, A., Berg, C. W., Nielsen, A., Skaug, H. J., Mächler, M. and Bolker, B. M. (2017). glmmTMB balances speed and flexibility among packages for zero-inflated generalized linear mixed modeling. The R Journal, 378–400. doi:10.32614/RJ2017066

Lenth R, Piaskowski J (2025). emmeans: Estimated Marginal Means, aka Least-Squares Means. R package version 2.0.0. https://rvlenth.github.io/emmeans/

Hartig F (2025). DHARMa: Residual Diagnostics for Hierarchical (Multi-Level / Mixed) Regression Models. R package version 0.4.7. https://github.com/florianhartig/dharma

---

fit_glmm *Fit a Beta-Binomial GLMM with Multiple Optimizers*

Description

This internal function fits a beta-binomial GLM or GLMM using glmmTMB, with support for multiple optimizers to improve convergence. It is used as a backend for fitting models to ORF-level count data in DOTSeq.

Usage

```
fit_glmm(
  formula,
  dispformula,
  data,
  family = betabinomial(),
  parallel = list(n = 4L, autopar = TRUE),
  optimizers = c("nlminb", "bobyqa", "optim"),
  max_iter = 1000
)
```

Arguments

formula	A formula object specifying the fixed effects structure of the model.
dispformula	A formula object specifying the dispersion model (e.g., ~strategy).
data	A data frame containing the model data, including counts and covariates.
optimizers	A character vector of optimizer names to try sequentially. Supported values include "nlminb", "bobyqa", and "optim". Default is c("nlminb", "bobyqa", "optim").
max_iter	An integer specifying the maximum number of iterations for the optimizer. Default is 1000.

Details

The function attempts to fit the model using each optimizer in the specified order. If a model converges successfully (i.e., `model$fit$convergence == 0` and `model$sdr$pdHess == TRUE`), it is returned immediately. Otherwise, the function continues trying the next optimizer.

Value

A fitted `glmmTMB` model object if convergence is successful. If all optimizers fail, the last attempted model (with convergence failure) is returned.

References

Brooks, M. E., Kristensen, K., van Benthem, K. J., Magnusson, A., Berg, C. W., Nielsen, A., Skaug, H. J., Mächler, M. and Bolker, B. M. (2017). glmmTMB balances speed and flexibility among packages for zero-inflated generalized linear mixed modeling. The R Journal, 378–400. DOI: 10.32614/RJ-2017-066

`fmla` *Access the model formula used in* **DOTSeq** *analysis*

Description

Retrieves the conditional formula stored in a `DOUData-class` or `DTEData-class` object. This formula defines the fixed effects structure used in GLM / GLMM modeling.

Usage

```
fmla(object)

## S4 method for signature 'DOUData'
fmla(object)

## S4 method for signature 'DTEData'
fmla(object)
```

Arguments

`object`	A `DOUData-class` or `DTEData-class` object.

Value

A `formula` object.

fragmentize_pairs *Convert paired-end alignments to fragment ranges*

Description

Given a GAlignmentPairs object, construct a GRanges representing the full inferred fragment span for each pair by taking the minimum start and maximum end across the first and last mate. This is useful for counting overlaps at the fragment level rather than at the read level.

Usage

```
fragmentize_pairs(gap, ignore.strand = TRUE)
```

Arguments

gap	A GAlignmentPairs object (typically produced by GenomicAlignments::readGAlignmentPairs()
ignore.strand	Logical. If TRUE, returned ranges have strand set to "*" regardless of input. If FALSE, the strand from the first mate is used.

Details

This function assumes mates are on the same reference sequence. For unusual cases where mates map to different chromosomes, the resulting behavior may be undefined and should be handled upstream (e.g., by filtering such pairs).

Value

A GRanges object with one range per input pair, with:

- seqnames from the first mate,
- start = min(start(first), start(last)),
- end = max(end(first), end(last)),
- strand depending on ignore.strand.

generate_coefficients *Generate Coefficients for Simulated Differential ORF Translation*

Description

Simulates log-fold change coefficients for differential ORF translation (DOT) analysis. This function assigns baseline coefficients to all ORFs and applies specific log-fold changes to multi-cistronic ORFs based on a selected regulatory scenario.

Usage

```
generate_coefficients(
  orfs,
  scenario = "uORF_up_mORF_down",
  gcoeff = 1.5,
  shape = 0.6,
  scale = 0.5
)
```

Arguments

orfs	A data frame containing ORF annotations. Row names must align with the filtered count matrices. Must include columns gene_id and orf_type.
scenario	A character string specifying the regulatory scenario to simulate. Must be one of: • "uORF_up_mORF_down" • "dORF_up_mORF_up" • "uORF_up_mORF_flat" • "uORF_down_mORF_flat" • "uORF_down_mORF_down" • "uORF_down_mORF_up" • "uORF_flat_mORF_up" • "uORF_flat_mORF_down" • "dORF_up_mORF_down" • "dORF_up_mORF_flat" • "dORF_down_mORF_flat" • "dORF_down_mORF_down" • "dORF_down_mORF_up" • "dORF_flat_mORF_up" • "dORF_flat_mORF_down"
gcoeff	Numeric. The log-fold change magnitude to apply to regulated ORFs.
shape	Numeric. Shape parameter for the gamma distribution used to simulate baseline coefficients.
scale	Numeric. Scale parameter for the gamma distribution used to simulate baseline coefficients.

Value

A list with the following components:

gcoeffs_ribo Named numeric vector of log-fold change coefficients for ribosome profiling data.

gcoeffs_rna Named numeric vector of baseline log-fold change coefficients for RNA-seq data.

labels Named binary vector indicating truly regulated ORFs (1) vs. non-regulated (0).

Examples

```
## Not run:
generate_coefficients(orfs_df, scenario = "uORF_up_mORF_down")

## End(Not run)
```

getContrasts *Access and replace contrast results from post hoc analysis*

Description

These methods retrieve or replace either interaction-specific or strategy-specific contrast results from a DOUData-class, DTEData-class, or DOTSeqDataSets-class object.

Usage

```
getContrasts(object, type = c("interaction", "strategy"))

## S4 method for signature 'DOUData'
getContrasts(object, type = c("interaction", "strategy"))

## S4 method for signature 'DTEData'
getContrasts(object, type = c("interaction", "strategy"))

## S4 method for signature 'DOTSeqDataSets'
getContrasts(object, type = c("interaction", "strategy"))

getContrasts(object, type = c("interaction", "strategy")) <- value

## S4 replacement method for signature 'DOUData'
getContrasts(object, type = c("interaction", "strategy")) <- value

## S4 replacement method for signature 'DTEData'
getContrasts(object, type = c("interaction", "strategy")) <- value
```

Arguments

object	A DOUData-class or DTEData-class object.
type	A character string, either "interaction" or "strategy".
value	A DFrame or data.frame to replace the contrast results.

Value

For the accessor, a DFrame or data.frame containing contrast results. For the replacement, an updated DOUData-class, DTEData-class or DOTSeqDataSets-class object.

Examples

```
# Read in count matrix, condition table, and annotation files
dir <- system.file("extdata", package = "DOTSeq")

cnt <- read.table(
    file.path(dir, "featureCounts.cell_cycle_subset.txt.gz"),
    header = TRUE,
    comment.char = "#"
)
names(cnt) <- gsub(".*(SRR[0-9]+).*", "\\1", names(cnt))
```

```
flat <- file.path(dir, "gencode.v47.orf_flattened_subset.gtf.gz")
bed <- file.path(dir, "gencode.v47.orf_flattened_subset.bed.gz")

meta <- read.table(file.path(dir, "metadata.txt.gz"))
names(meta) <- c("run", "strategy", "replicate", "treatment", "condition")
cond <- meta[meta$treatment == "chx", ]
cond$treatment <- NULL

# Create a DOTSeqDataSets object
d <- DOTSeqDataSetsFromFeatureCounts(
    count_table = cnt,
    condition_table = cond,
    flattened_gtf = flat,
    flattened_bed = bed
)

getDOU(d)

getDTE(d)
```

---

getDOU *Accessor and replacement methods for DOU slot*

Description

These methods allow access to and replacement of the DOUData-class object stored within a DOTSeqDataSets-class container.

Usage

```
getDOU(object)

## S4 method for signature 'DOTSeqDataSets'
getDOU(object)

getDOU(object) <- value

## S4 replacement method for signature 'DOTSeqDataSets'
getDOU(object) <- value
```

Arguments

object	A DOTSeqDataSets-class object.
value	A replacement object (e.g., a DOUData-class).

Value

For the accessor, a DOUData-class object. For the replacement, an updated DOTSeqDataSets-class object.

Examples

```
# Read in count matrix, condition table, and annotation files
dir <- system.file("extdata", package = "DOTSeq")

cnt <- read.table(
    file.path(dir, "featureCounts.cell_cycle_subset.txt.gz"),
    header = TRUE,
    comment.char = "#"
)
names(cnt) <- gsub(".*(SRR[0-9]+).*", "\\1", names(cnt))

gtf <- file.path(dir, "gencode.v47.orf_flattened_subset.gtf.gz")
bed <- file.path(dir, "gencode.v47.orf_flattened_subset.bed.gz")

meta <- read.table(file.path(dir, "metadata.txt.gz"))
names(meta) <- c("run", "strategy", "replicate", "treatment", "condition")
cond <- meta[meta$treatment == "chx", ]
cond$treatment <- NULL

# Create a DOTSeqDataSets object
d <- DOTSeqDataSetsFromFeatureCounts(
    count_table = cnt,
    condition_table = cond,
    flattened_gtf = gtf,
    flattened_bed = bed
)

getDOU(d)

getDTE(d)
```

---

getDTE *Accessor and replacement methods for DTE slot*

Description

These methods allow access to and replacement of the DTEData-class object stored within a DOTSeqDataSets-class container.

Usage

```
getDTE(object)

## S4 method for signature 'DOTSeqDataSets'
getDTE(object)

getDTE(object) <- value

## S4 replacement method for signature 'DOTSeqDataSets'
getDTE(object) <- value
```

Arguments

object	A DOTSeqDataSets-class object.
value	A replacement object (e.g., a DTEData-class).

Value

For the accessor, a DTEData-class object. For the replacement, an updated DOTSeqDataSets-class object.

Examples

```
# Read in count matrix, condition table, and annotation files
dir <- system.file("extdata", package = "DOTSeq")

cnt <- read.table(
    file.path(dir, "featureCounts.cell_cycle_subset.txt.gz"),
    header = TRUE,
    comment.char = "#"
)
names(cnt) <- gsub(".*(SRR[0-9]+).*", "\\1", names(cnt))

flat <- file.path(dir, "gencode.v47.orf_flattened_subset.gtf.gz")
bed <- file.path(dir, "gencode.v47.orf_flattened_subset.bed.gz")

meta <- read.table(file.path(dir, "metadata.txt.gz"))
names(meta) <- c("run", "strategy", "replicate", "treatment", "condition")
cond <- meta[meta$treatment == "chx", ]
cond$treatment <- NULL

# Create a DOTSeqDataSets object
d <- DOTSeqDataSetsFromFeatureCounts(
    count_table = cnt,
    condition_table = cond,
    flattened_gtf = flat,
    flattened_bed = bed
)

getDOU(d)

getDTE(d)
```

---

getExonicReads *Filter BAM files to retain only reads overlapping exonic regions*

Description

Filters one or more BAM files to retain reads overlapping exonic regions defined in a TxDb object stored in the metadata of a GRanges object. Optionally restricts filtering to coding genes only (genes with CDS). Filtered BAMs are sorted and saved with the suffix `.exonic.sorted.bam` in a user-specified or temporary directory.

Usage

```
getExonicReads(
  gr,
  seqlevels_style = "UCSC",
  bam_files,
  bam_output_dir = tempdir(),
  coding_genes_only = TRUE,
  verbose = TRUE
)
```

Arguments

gr	A GRanges object with a TxDb SQLite file path stored in its metadata under metadata(gr)$txdb.
seqlevels_style	Character; the naming style for chromosome identifiers (e.g., "UCSC", "NCBI"). This is applied to both the GRanges object and the TxDb annotation. Default is "UCSC".
bam_files	A character vector of paths to BAM files to be filtered.
bam_output_dir	A writable directory where filtered BAM files will be saved. Defaults to tempdir().
coding_genes_only	Logical; if TRUE, restrict filtering to coding genes only (i.e., genes with CDS). Default is TRUE.
verbose	Logical; if TRUE, print progress and runtime messages. Default is TRUE.

Details

The function uses Rsamtools::filterBam() to extract reads overlapping exonic regions and Rsamtools::sortBam() to sort the filtered BAM. The output files are named based on the input BAM file with the suffix .exonic.sorted.bam.

Value

This function is called for its side effect of creating filtered and sorted BAM files in bam_output_dir. It returns NULL invisibly.

Examples

```
library(TxDb.Dmelanogaster.UCSC.dm3.ensGene)
library(pasillaBamSubset)
library(AnnotationDbi)
library(GenomeInfoDb)
library(withr)

# Save a subset of TxDb as an SQLite file
txdb_chr4 <- keepSeqlevels(
    TxDb.Dmelanogaster.UCSC.dm3.ensGene,
    "chr4",
    pruning.mode = "coarse"
)
txdb_path <- file.path(tempdir(), "dm3_chr4.sqlite")
saveDb(txdb_chr4, file = txdb_path)
```

```
# Create a GRanges object with a link to the TxDb SQLite file
gr <- GRanges(seqnames = "chr4", ranges = IRanges(start = 233, end = 2300))
metadata(gr)$txdb <- txdb_path

# Filter BAM file and save output in a temporary directory
temp_dir <- tempdir()
getExonicReads(gr,
    bam_files = untreated1_chr4(),
    bam_output_dir = temp_dir
)

# Clean up
withr::defer(unlink(txdb_path))
withr::defer(unlink(list.files(temp_dir, pattern = "exonic", full.names = TRUE)))
```

`getORFs` *Extract Genomic ORFs from Transcript Sequences*

Description

Identifies open reading frames (ORFs) from transcript sequences and maps them to genomic coordinates using a GTF/GFF annotation file or a TxDb object. Supports input sequences as a FASTA file, a `DNAStringSet`, or a `BSgenome` object. Classifies small ORFs (sORFs) as upstream (uORF), downstream (dORF), or overlapping (oORF) relative to the main ORFs (mORFs).

Usage

```
getORFs(
  sequences,
  annotation,
  txdb_output_dir = NULL,
  organism = "Homo sapiens",
  require_ids = c("hgnc_id", "protein_id", "ccdsid"),
  source_filter = NULL,
  circ_seqs = NULL,
  start_codons = "ATG",
  stop_codons = "TAA|TAG|TGA",
  min_len = 0,
  longest_orf = TRUE,
  verbose = TRUE
)
```

Arguments

`sequences`	Transcript sequences. Can be a character string (path to a FASTA file), a `DNAStringSet` object, or a `BSgenome` object.
`annotation`	Transcript annotation. Can be a character string (path to a GTF or GFF file) or a `TxDb` object.
`txdb_output_dir`	TxDb output directory. The TxDb file path is linked to the GRanges returned in the metadata slot. Default: `NULL`.

organism	Character string specifying the organism name (used only when building a TxDb from a GTF/GFF file). Default is "Homo sapiens".
require_ids	Character vector. Metadata column names that must be present and non-NA. Default: c("hgnc_id", "protein_id","ccdsid").
source_filter	Character or NULL. If provided, filters transcripts by the 'source' column. Default: NULL.
circ_seqs	Character vector of circular sequences to exclude (e.g., "chrM"). Default is "chrM".
start_codons	Character vector of start codons to search for (e.g., "ATG"). Default is "ATG".
stop_codons	Character string of stop codons separated by "|" (e.g., "TAA|TAG|TGA"). Default is "TAA|TAG|TGA".
min_len	Integer specifying the minimum ORF length in bases. Default is 0.
longest_orf	Logical. If TRUE, only the longest ORF per transcript is returned. Default is TRUE.
verbose	Logical. If TRUE, prints progress messages and timing information. Default is TRUE.

Details

- ORFs are identified in transcript space using findORFsFasta().
- Coordinates are mapped to the genome using mapFromTranscripts() and exon annotations.
- Main ORFs are defined by overlap with annotated CDS regions.
- Small ORFs are classified relative to mORFs based on strand-aware genomic position.

Value

A GRanges object containing genomic coordinates of ORFs, with metadata columns gene_id and orf_type. Main ORFs are labeled as "mORF", and small ORFs are classified as "uORF", "dORF", or "oORF".

References

Lawrence, M., Huber, W., Pagès, H., Aboyoun, P., Carlson, M., Gentleman, R., Morgan, M., Carey, V. (2013). Software for Computing and Annotating Genomic Ranges. PLoS Computational Biology, 9. DOI: 10.1371/journal.pcbi.1003118

Tjeldnes, H., Labun, K., Torres Cleuren, Y. et al. ORFik: a comprehensive R toolkit for the analysis of translation. BMC Bioinformatics 22, 336 (2021). DOI: 10.1186/s12859-021-04254-w

Examples

```
library(BSgenome.Hsapiens.UCSC.hg38)
library(TxDb.Hsapiens.UCSC.hg38.knownGene)
library(GenomicFeatures)

# Load genome and TxDb
genome <- BSgenome.Hsapiens.UCSC.hg38
txdb <- TxDb.Hsapiens.UCSC.hg38.knownGene

# Get exons grouped by transcript
exons_by_tx <- exonsBy(txdb, by = "tx", use.names = TRUE)
```

```
# Select a single transcript for demonstration
tx1 <- head(exons_by_tx, 100)

# Extract transcript sequence
tx_seqs <- extractTranscriptSeqs(genome, tx1)

# Run getORFs on the transcript sequence
txdb_output_dir <- tempdir()
gr <- getORFs(
    sequences = tx_seqs,
    annotation = txdb,
    txdb_output_dir = txdb_output_dir
)
print(gr)

# Clean up
sqlite_files <- list.files(txdb_output_dir, pattern = "\\.sqlite$", full.names = TRUE)
unlink(sqlite_files)
```

`get_lfsr_annotation` *Retrieve and format adjusted p-value for a specific ORF and contrast*

Description

This helper function extracts the adjusted p-value (e.g., LFSR or padj) for a given ORF and contrast from a results data frame, and formats it to three decimal places. If the value is missing or NA, it returns "N/A".

Usage

```
get_lfsr_annotation(rowdata, orf, contrast_name, dou_signif_col = "lfsr")
```

Arguments

`rowdata`	A data frame containing differential results, including columns for `orf_id`, `contrast`, and the specified p-value column.
`orf`	A character string specifying the ORF ID to look up.
`contrast_name`	A character string specifying the contrast name (e.g., "Treatment - Control").
`dou_signif_col`	A character string specifying the column name to extract the adjusted p-value from. Defaults to `"lfsr"`.

Value

A character string representing the formatted p-value (e.g., `"0.023"`).

`get_params` *Estimate zero-inflated negative binomial parameters from a real dataset*

Description

This function estimates the parameters of a zero-inflated negative binomial distribution based on a real count data set using the method of moments. The function also returns a spline fit of log mean to log size which can be used when generating new simulated data.

Usage

```
get_params(
  counts,
  threshold = NULL,
  size_factor = NULL,
  min_size = NULL,
  scale_p0 = NULL
)
```

Arguments

`counts`	A matrix of counts.
`threshold`	Only estimate parameters from transcripts with row means greater than this threshold.
`size_factor`	Optional numeric scalar to scale the estimated size parameters.
`min_size`	Optional numeric value to enforce a minimum size parameter.
`scale_p0`	Optional numeric scalar to scale the zero-inflation probabilities.

Value

A list containing:

p0 A vector of probabilities that the count will be zero, one for each gene/transcript.

mu The estimated negative binomial mean by method of moments for the non-zero counts.

size The estimated negative binomial size by method of moments for the non-zero counts.

fit A fit relating log mean to log size for use in simulating new data.

Author(s)

Jeff Leek (original), Chun Shen Lim (modifications)

get_significant_genes *Extract Significant Genes Based on LFSR Threshold*

Description

Identifies genes with significant differential ORF usage (DOU) based on a local false sign rate (LFSR) threshold. Extracts gene IDs from ORF-level results and filters those with LFSR below the specified threshold.

Usage

```
get_significant_genes(results, padj_col = "lfsr", padj_threshold = 0.05)
```

Arguments

`results`	A data frame containing ORF-level DOU results. Must include columns `orf_id` and the specified `padj_col`.
`padj_col`	Character string specifying the column name for LFSR values. Default is `"lfsr"`.
`padj_threshold`	Numeric threshold for filtering significant ORFs. Default is `0.05`.

Value

A character vector of Ensembl gene IDs corresponding to significant ORFs.

See Also

plotDOT

Examples

```
## Not run:
sig_genes <- get_significant_genes(
  results_df,
  padj_col = "lfsr",
  padj_threshold = 0.05
)

## End(Not run)
```

group_bam_files *Group BAM files by read type (paired-end or single-end)*

Description

Splits a list of BAM file paths into paired-end and single-end groups based on the read flags.

Usage

```
group_bam_files(bam_files)
```

Arguments

bam_files	Character vector. Paths to BAM files.

Value

A list with two elements:

paired Character vector of paired-end BAM file paths

single Character vector of single-end BAM file paths

---

gSort *Sort a GRangesList, helper.*

Description

A helper for sortPerGroup. A faster, more versatile reimplementation of GenomicRanges::sort() Normally not used directly. Groups first each group, then either decreasing or increasing (on starts if byStarts == T, on ends if byStarts == F)

Usage

```
gSort(grl, decreasing = FALSE, byStarts = TRUE)
```

Arguments

grl	a GRangesList
decreasing	should the first in each group have max(start(group)) ->T or min-> default(F) ?
byStarts	a logical T, should it order by starts or ends F.

Value

an equally named GRangesList, where each group is sorted within group.

Author(s)

Haakon Tjeldnes et al.

References

Tjeldnes, H., Labun, K., Torres Cleuren, Y. et al. ORFik: a comprehensive R toolkit for the analysis of translation. BMC Bioinformatics 22, 336 (2021). DOI: 10.1186/s12859-021-04254-w

is_paired_end *Check if a BAM file is paired-end*

Description

Quickly checks whether a BAM file contains paired-end reads by inspecting the first 1000 flags. Assumes that all reads in the file are consistently either paired-end or single-end.

Usage

```
is_paired_end(bam_file)
```

Arguments

bam_file	Character. Path to a BAM file.

Value

Logical. TRUE if any of the first 1000 reads have the 'paired' flag set (0x1), FALSE otherwise.

lastExonEndPerGroup *Get last end per granges group*

Description

Get last end per granges group

Usage

```
lastExonEndPerGroup(grl, keep.names = TRUE)
```

Arguments

grl	a GRangesList
keep.names	a boolean, keep names or not, default: (TRUE)

Value

a Rle(keep.names = T), or integer vector(F)

Author(s)

Haakon Tjeldnes et al.

References

Tjeldnes, H., Labun, K., Torres Cleuren, Y. et al. ORFik: a comprehensive R toolkit for the analysis of translation. BMC Bioinformatics 22, 336 (2021). DOI: 10.1186/s12859-021-04254-w

Examples

```
## Not run:
gr_plus <- GRanges(seqnames = c("chr1", "chr1"),
                   ranges = IRanges(c(7, 14), width = 3),
                   strand = c("+", "+"))
gr_minus <- GRanges(seqnames = c("chr2", "chr2"),
                    ranges = IRanges(c(4, 1), c(9, 3)),
                    strand = c("-", "-"))
grl <- GRangesList(tx1 = gr_plus, tx2 = gr_minus)
lastExonEndPerGroup(grl)

## End(Not run)
```

---

lastExonPerGroup *Get last exon per GRangesList group*

Description

grl must be sorted, call ORFik:::sortPerGroup if needed

Usage

```
lastExonPerGroup(grl)
```

Arguments

grl	a GRangesList

Value

a GRangesList of the last exon per group

Author(s)

Haakon Tjeldnes et al.

References

Tjeldnes, H., Labun, K., Torres Cleuren, Y. et al. ORFik: a comprehensive R toolkit for the analysis of translation. BMC Bioinformatics 22, 336 (2021). DOI: 10.1186/s12859-021-04254-w

Examples

```
## Not run:
gr_plus <- GRanges(seqnames = c("chr1", "chr1"),
                   ranges = IRanges(c(7, 14), width = 3),
                   strand = c("+", "+"))
gr_minus <- GRanges(seqnames = c("chr2", "chr2"),
                    ranges = IRanges(c(4, 1), c(9, 3)),
                    strand = c("-", "-"))
grl <- GRangesList(tx1 = gr_plus, tx2 = gr_minus)
lastExonPerGroup(grl)
```

```
## End(Not run)
```

`lastExonStartPerGroup` *Get last start per granges group*

Description

Get last start per granges group

Usage

```
lastExonStartPerGroup(grl, keep.names = TRUE)
```

Arguments

`grl`	a `GRangesList`
`keep.names`	a boolean, keep names or not, default: (TRUE)

Value

a Rle(keep.names = T), or integer vector(F)

Author(s)

Haakon Tjeldnes et al.

References

Tjeldnes, H., Labun, K., Torres Cleuren, Y. et al. ORFik: a comprehensive R toolkit for the analysis of translation. BMC Bioinformatics 22, 336 (2021). DOI: 10.1186/s12859-021-04254-w

Examples

```
## Not run:
gr_plus <- GRanges(seqnames = c("chr1", "chr1"),
                   ranges = IRanges(c(7, 14), width = 3),
                   strand = c("+", "+"))
gr_minus <- GRanges(seqnames = c("chr2", "chr2"),
                    ranges = IRanges(c(4, 1), c(9, 3)),
                    strand = c("-", "-"))
grl <- GRangesList(tx1 = gr_plus, tx2 = gr_minus)
lastExonStartPerGroup(grl)

## End(Not run)
```

`longestORFs` *Get longest ORF per stop site*

Description

Rule: if seqname, strand and stop site is equal, take longest one. Else keep. If IRangesList or IRanges, seqnames are groups, if GRanges or GRangesList seqnames are the seqlevels (e.g. chromosomes/transcripts)

Usage

```
longestORFs(grl)
```

Arguments

`grl`	a GRangesList/IRangesList, GRanges/IRanges of ORFs

Value

a GRangesList/IRangesList, GRanges/IRanges (same as input)

Author(s)

Haakon Tjeldnes et al.

References

Tjeldnes, H., Labun, K., Torres Cleuren, Y. et al. ORFik: a comprehensive R toolkit for the analysis of translation. BMC Bioinformatics 22, 336 (2021). DOI: 10.1186/s12859-021-04254-w

See Also

Other ORFHelpers: `stopSites()`

Examples

```
## Not run:
ORF1 = GRanges("1", IRanges(10,21), "+")
ORF2 = GRanges("1", IRanges(1,21), "+") # <- longest
grl <- GRangesList(ORF1 = ORF1, ORF2 = ORF2)
longestORFs(grl) # get only longest

## End(Not run)
```

mapIDs *Retrieve Gene Symbols from Ensembl or Ensembl Genomes*

Description

Queries Ensembl or Ensembl Genomes BioMart databases to retrieve gene symbols, descriptions, and optionally Gene Ontology (GO) terms. Supports multiple organism groups including vertebrates, plants, fungi, protists, metazoa, and bacteria.

If the specified `symbol_col` returns only `NA` values, the function automatically falls back to using the `description` field instead.

Usage

```
mapIDs(
  ensembl_ids,
  dataset,
  symbol_col = "external_gene_name",
  include_go = FALSE,
  mart_source = "ensembl",
  host = NULL
)
```

Arguments

`ensembl_ids`	A character vector of Ensembl gene IDs to query.
`dataset`	A string specifying the dataset name (e.g., `"hsapiens_gene_ensembl"`, `"athaliana_eg_gene"`).
`symbol_col`	A string specifying the attribute to use as the gene symbol. Common options include `hgnc_symbol`, `"external_gene_name"`, `description`. The default is `"external_gene_name"`, which is widely used in vertebrate datasets such as human and mouse.
`include_go`	Logical; if `TRUE`, includes GO annotations (`go_id`, `name_1006`, `namespace_1003`) in the output.
`mart_source`	A string indicating the BioMart source. One of `"ensembl"`, `"plants"`, `"fungi"`, `"protists"`, `"metazoa"`, or `"bacteria"`.
`host`	Optional. A custom host URL (e.g., for archived Ensembl versions).

Value

A data frame containing gene symbols for the input Ensembl IDs. If `symbol_col` is unavailable, the `description` field is used instead and renamed to match `symbol_col`.

References

Durinck S, Spellman P, Birney E, Huber W (2009). Mapping identifiers for the integration of genomic datasets with the R/Bioconductor package biomaRt. Nature Protocols, 4, 1184–1191. DOI: 10.1038/nprot.2009.97

Durinck S, Moreau Y, Kasprzyk A, Davis S, De Moor B, Brazma A, Huber W (2005). BioMart and Bioconductor: a powerful link between biological databases and microarray data analysis. Bioinformatics, 21, 3439–3440. DOI: 10.1093/bioinformatics/bti525

Examples

```
# Ping Ensembl REST to avoid timeouts when the service is down.
is_ensembl_up <- function(timeout = 3) {
  url <- "https://rest.ensembl.org/info/ping"
  out <- try(
    curl::curl_fetch_memory(url, handle = curl::new_handle(timeout = timeout)),
    silent = TRUE
  )
  is.list(out) && out$status_code >= 200 && out$status_code < 500
}

if (is_ensembl_up()) {
  # Human gene example
  res <- mapIDs(
    ensembl_ids = c("ENSG00000139618"),
    dataset     = "hsapiens_gene_ensembl",
    mart_source = "ensembl"
  )
  head(res)

  # Arabidopsis example (uncomment to try)
  # res <- mapIDs(
  #   ensembl_ids = c("AT1G01010"),
  #   dataset     = "athaliana_eg_gene",
  #   symbol_col  = "tair_symbol",
  #   mart_source = "plants"
  # )
  # head(res)

  # Plasmodium falciparum example (uncomment to try)
  # res <- mapIDs(
  #   ensembl_ids = c("PF3D7_0100100"),
  #   dataset     = "pfalciparum_eg_gene",
  #   mart_source = "protists"
  # )
  # head(res)
} else {
  message("Ensembl appears unavailable; skipping online example.")
}
```

---

match_runs *Match and align sample metadata with count table*

Description

This internal function matches sample identifiers between the count table and the condition table, ensuring consistency in sample naming and ordering. It verifies that sufficient replicates exist, renames columns to include metadata, and prepares the count and condition tables for downstream differential translation analysis. It also assigns binary strategy labels (e.g., RNA = 0, Ribo = 1), sets factor levels, and optionally relevels the condition factor to a specified baseline.

Usage

```
match_runs(cnt, cond, num_feat_cols = 0, baseline = NULL, verbose = TRUE)
```

Arguments

cnt	A data frame containing count data. Columns must include sample names matching the run column in cond. If num_feat_cols is 6, the first six columns must be: Geneid, Chr, Start, End, Strand, Length.
cond	A data frame containing sample metadata. Must include columns: run, strategy, condition, and replicate.
num_feat_cols	Integer specifying the number of feature columns in cnt before sample columns. Typically 6 for featureCounts output, or 0 for raw count matrices.
baseline	Optional character string specifying the reference level for the condition factor.

Value

A list with two elements:

cnt A reordered count table with renamed sample columns.

cond A metadata data frame with binary strategy labels and factor columns, ready for modeling.

---

modelType *Access the model type from a PostHoc object*

Description

Retrieves the model type string from a PostHoc object.

Retrieves model results, parameters, and diagnostics.

Retrieves the post hoc summary object (e.g. from emmeans).

Usage

```
modelType(object)

## S4 method for signature 'PostHoc'
modelType(object)

fitResults(object)

## S4 method for signature 'PostHoc'
fitResults(object)

posthoc(object)

## S4 method for signature 'PostHoc'
posthoc(object)
```

Arguments

object	A PostHoc object.

Value

A `character(1)` string indicating the model type.

A `list` containing model results and diagnostics.

A post hoc summary object.

Functions

- `modelType(PostHoc)`: Access the model type from a PostHoc object.
- `fitResults(PostHoc)`: Access the results list.
- `posthoc(PostHoc)`: Access the post hoc summary.

Examples

```
ph <- PostHoc(type = "glmmTMB")
modelType(ph)
ph <- PostHoc(results = list(aic = 100))
fitResults(ph)
ph <- PostHoc(posthoc = "dummy_emmeans")
posthoc(ph)
```

---

numExonsPerGroup *Get list of the number of exons per group*

Description

Can also be used generaly to get number of GRanges object per GRangesList group

Usage

```
numExonsPerGroup(grl, keep.names = TRUE)
```

Arguments

`grl`	a `GRangesList`
`keep.names`	a logical, keep names or not, default: (TRUE)

Value

an integer vector of counts

Author(s)

Haakon Tjeldnes et al.

References

Tjeldnes, H., Labun, K., Torres Cleuren, Y. et al. ORFik: a comprehensive R toolkit for the analysis of translation. BMC Bioinformatics 22, 336 (2021). DOI: 10.1186/s12859-021-04254-w

pad_cols *Pad a sparse count matrix with missing cell barcode columns*

Description

Align a sparse matrix by a desired set/order of column names (cell barcodes), adding zero-filled columns for any barcodes not currently present. This is primarily used when summing sparse matrices generated from different chunks that may contain different subsets of cells.

Usage

```
pad_cols(M, cols)
```

Arguments

M	A sparse matrix (typically a dgCMatrix) with non-NULL column names representing cell barcodes. Row names are preserved.
cols	Character vector of target column names (barcodes). The returned matrix will have columns exactly in this order. Missing barcodes are added as all-zero columns.

Details

This function assumes that colnames(M) uniquely identify columns and can be used as a key for alignment across matrices. It will error if column names are NULL.

When repeatedly called in a loop, this can incur overhead due to repeated sparse matrix reallocation and subsetting; consider accumulating triplets (i, j, x) and building a single sparse matrix when performance is critical.

Value

A sparse matrix with the same number of rows as M and columns exactly equal to cols (same order). If cols contains names not present in M, they are appended as zero columns before reordering.

parse_condition_table *Parse and validate sample metadata for DOTSeq analysis*

Description

This internal helper function reads and validates the sample metadata provided either as a file path or a data frame. It ensures that the required columns (run, strategy, condition, and replicate) are present (case-insensitive match), and normalizes column names for consistency. If a file path is provided, the function reads the file and checks the header before loading the full table. If a data frame is provided, it performs similar validation directly.

Usage

```
parse_condition_table(condition_table)
```

Arguments

condition_table

A character string specifying the path to a tab-delimited metadata file, or a data frame containing sample metadata. The metadata must include the following columns (case-insensitive): run, strategy, condition, and replicate.

Value

A data frame with normalized column names and row names set to the run column. Used internally to construct the DOTSeqDataSets-class object.

---

plotDOT *Generate Differential ORF Translation (DOT) Visualization Suite*

Description

Generates a suite of visualizations to explore Differential ORF Usage (DOU) and Translation Efficiency (DTE) results. Supports volcano plots, Venn diagrams, composite scatter plots, heatmaps, and usage plots. Integrates Ensembl gene symbols and highlights significant ORFs based on empirical Bayes shrinkage (via the **ashr** package's ash function).

Usage

```
plotDOT(
  plot_type = "volcano",
  results = NULL,
  data = NULL,
  id_mapping = FALSE,
  include_go = FALSE,
  gene_id = NULL,
  dou_params = list(est_col = "posterior", est_thresh = 1, signif_col = "lfsr",
    signif_thresh = 0.05, signif_ceil = 10, extreme_thresh = NULL),
 dte_params = list(est_col = "log2FoldChange", signif_col = "padj", signif_thresh =
    0.05),
 plot_params = list(top_hits = 20, color_by = "significance", rank_by = "significance",
    legend_position = "topright", order_by = NULL, flip_sign = FALSE),
 annotation_params = list(sorf_type = "uORF", dataset = "hsapiens_gene_ensembl",
    symbol_col = "hgnc_symbol", mart_source = "ensembl"),
 colors = list(dte = adjustcolor("#0072B2", alpha.f = 0.6), dou = adjustcolor("#E69F00",
    alpha.f = 0.6), both = adjustcolor("#CC79A7", alpha.f = 0.6), none =
  adjustcolor("grey80", alpha.f = 0.6), uorf = adjustcolor("#D73027", alpha.f = 0.6),
  morf = adjustcolor("#4575B4", alpha.f = 0.6), dorf = adjustcolor("#A6A6A6", alpha.f =
    0.6), low = "blue", middle = "white", high = "red", usage = "Set2"),
  force_new_device = TRUE,
  verbose = TRUE
)
```

Arguments

plot_type	Type of plot to generate. Options include "volcano", "venn", "composite", "heatmap", and "usage". Default is "volcano".
results	A data frame containing DOU and DTE estimates and significance values. Required for all plots except "usage". Must include ORF-level identifiers and columns specified in dou_params and dte_params.
data	A DOUData-class object required for all plots except for "venn". Should contain DOUResults in rowData() and post hoc results in the interaction slot.
id_mapping	Optional input controlling gene symbol annotation. Can be one of: • FALSE (default): disables annotation. • TRUE: triggers automatic annotation via BioMart using mapIDs(). • a data.frame: a precomputed mapping of Ensembl IDs to gene symbols, which can be reused across plots. Used in volcano and heatmap plots to annotate genes with symbols. If TRUE, annotation is attempted and fallback to Ensembl IDs occurs if symbol coverage is low. Default is FALSE
include_go	Logical; if TRUE, includes GO annotations in id_mapping. Default is FALSE.
gene_id	Character string specifying the gene ID for usage plots. Default is NULL.
dou_params	A named list of parameters for DOU filtering and display: • est_col: Column name for DOU effect size (e.g., "posterior") • est_thresh: Threshold for effect size significance • signif_col: Column name for DOU significance (e.g., "lfsr") • signif_thresh: Threshold for significance • signif_ceil: Ceiling for -log10 significance axis • extreme_thresh: Optional threshold for labeling extreme points
dte_params	A named list of parameters for DTE filtering: • est_col: Column name for DTE effect size (e.g., "log2FoldChange") • signif_col: Column name for DTE significance (e.g., "padj") • signif_thresh: Threshold for significance
plot_params	A named list controlling labeling of top hits: • top_hits: Number of top hits to label or show in volcano plot and heatmap • color_by: How to color points. Options: – "significance": Based on DTE-only, DOU-only, or both – "orf_type": Based on ORF type (requires rowdata) . • rank_by: Ranking method ("significance" or "score") • legend_position: Character string specifying the position of the legend in the volcano and composite scatter plots. Options include: "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right", "center". Default is "topright". • order_by: Optional character vector specifying the order of conditions on the "usage" plot x-axis. • flip_sign: Logical; if TRUE, flips the sign of DOU estimates to align directionality with DTE. Default is TRUE.
annotation_params	A named list for BioMart annotation:

- `sorf_type`: Short ORF type (`"uORF"` or `"dORF"`)
- `dataset`: Ensembl dataset name
- `symbol_col`: Column name for gene symbols
- `mart_source`: BioMart source (`"ensembl"`, `"plants"`, etc.)

`colors` A named list of colors used across plots:

- `dte`, `dou`, `both`, `none`: for significance-based coloring
- `uorf`, `morf`, `dorf`: for ORF type-based coloring
- `low`, `middle`, `high`: for heatmap gradient
- `usage`: for condition-specific coloring

`force_new_device`
Logical; if `TRUE`, detects graphics error and resets graphics state unconditionally. Default is `TRUE`.

`verbose` Logical; if `TRUE`, prints progress messages. Default is `TRUE`.

Details

This function orchestrates multiple visualization components to explore differential translation across ORFs. It uses `testDOU` output to identify significant ORFs, retrieves gene symbols via `getBM`, and generates plots to summarize DOU and DTE relationships. The composite scatter plot includes marginal distributions by ORF type, helping to visualize the overlap and divergence between DTE and DOU signals. The volcano plot highlights extreme and top-ranked ORFs, while the heatmap summarizes DOU across top genes.

Value

A data frame containing gene symbols retrieved from Ensembl, used for labeling and heatmap visualization.

References

Durinck S, Spellman P, Birney E, Huber W (2009). Mapping identifiers for the integration of genomic datasets with the R/Bioconductor package biomaRt. Nature Protocols, 4, 1184–1191. DOI: 10.1038/nprot.2009.97

Durinck S, Moreau Y, Kasprzyk A, Davis S, De Moor B, Brazma A, Huber W (2005). BioMart and Bioconductor: a powerful link between biological databases and microarray data analysis. Bioinformatics, 21, 3439–3440. DOI: 10.1093/bioinformatics/bti525

Larsson J, Gustafsson P (2018). "A Case Study in Fitting Area-Proportional Euler Diagrams with Ellipses Using eulerr." In Proceedings of International Workshop on Set Visualization and Reasoning, volume 2116, 84–91. https://ceur-ws.org/Vol-2116/paper7.pdf

Examples

```
# Example ORF-level results
results_df <- data.frame(
  orf_id = c(
    "ENSG00000139618.19:O001",
    "ENSG00000139618.19:O002",
    "ENSG00000157764.15:O003"
  ),
  lfsr = c(0.01, 0.2, 0.03),
  padj = c(0.02, 0.01, 0.1)
```

```
)

plotDOT(plot_type = "venn", results = results_df)
```

`plot_composite` *Plot Composite Scatter and Marginal Plots for DTE vs DOU*

Description

Visualizes the relationship between differential translation efficiency (DTE) and differential ORF usage (DOU) using a scatter plot with marginal histograms or density curves. Points can be colored by significance (DTE, DOU, both) or by ORF type (uORF, mORF, dORF). This plot helps assess overlap and divergence between DTE and DOU signals across ORFs.

Usage

```
plot_composite(
  results,
  rowdata = NULL,
  color_by = c("significance", "orf_type"),
  marginal_plot_type = c("histogram", "density"),
  dou_estimates_col = "posterior",
  dou_signif_col = "lfsr",
  dte_estimates_col = "log2FoldChange",
  dte_signif_col = "padj",
  dou_signif_thresh = 0.05,
  dte_signif_thresh = 0.05,
  flip_sign = FALSE,
  lhist = 20,
 colors = list(dte = adjustcolor("#0072B2", alpha.f = 0.6), dou = adjustcolor("#E69F00",
    alpha.f = 0.6), both = adjustcolor("#CC79A7", alpha.f = 0.6), none =
   adjustcolor("grey80", alpha.f = 0.4), uorf = adjustcolor("#D73027", alpha.f = 0.6),
   morf = adjustcolor("#4575B4", alpha.f = 0.6), dorf = adjustcolor("#A6A6A6", alpha.f =
    0.6)),
  legend_position = "bottomright"
)
```

Arguments

`results`	A data frame containing DTE and DOU post hoc contrast results. Must include columns for DTE and DOU estimates and their significance values.
`rowdata`	Optional data frame containing ORF metadata, with row names corresponding to `orf_id`. Must include an `orf_type` column with values "uORF", "mORF", or "dORF" if `color_by = "orf_type"`.
`color_by`	Character string specifying how to color points: • `"significance"`: Colors by DTE-only, DOU-only, both significant • `"orf_type"`: Colors by ORF type (requires `rowdata`) Default is `"significance"`.

`marginal_plot_type`
: Character string specifying the type of marginal plot: `"histogram"` or `"density"`. Default is `"density"`.

`dou_estimates_col`
: Column name for DOU posterior estimates. Default is `"posterior"`.

`dou_signif_col`
: Column name for DOU significance values (e.g., local false sign rate). Default is `"lfsr"`.

`dte_estimates_col`
: Column name for DTE estimates (e.g., log2 fold-change). Default is `"log2FoldChange"`.

`dte_signif_col`
: Column name for DTE adjusted p-values. Default is `"padj"`.

`dou_signif_thresh`
: Numeric threshold for DOU significance. Default is `0.05`.

`dte_signif_thresh`
: Numeric threshold for DTE significance. Default is `0.05`.

`flip_sign`
: Logical; if `TRUE`, flips the sign of DOU estimates to align directionality with DTE. Default is `FALSE`.

`lhist`
: Integer; number of bins for marginal histograms. Default is `20`.

`colors`
: A named list of colors used for plotting. Must include:

  - `dte`, `dou`, `both`, `none`: for significance-based coloring
  - `uorf`, `morf`, `dorf`: for ORF type-based coloring

  Each value should be a valid color string or result of `adjustcolor()`.

`legend_position`
: Position of the legend. Options include `"bottomright"`, `"bottom"`, `"bottomleft"`, `"left"`, `"topleft"`, `"top"`, `"topright"`, `"right"`, `"center"`.

Details

The function uses base R graphics and a custom layout matrix to arrange the scatter and marginal plots. Coloring by significance requires DTE and DOU significance columns. Coloring by ORF type requires a metadata table with `orf_type` values.

Value

A composite plot consisting of:

Scatter plot Displays DTE (log2 fold-change) vs DOU (log-odds change) estimates, colored by significance or ORF type.

Marginal plots Show the distribution of DTE and DOU estimates along the top and right margins, using either histograms or density curves.

Additionally, the function returns the Spearman correlation between DTE and DOU estimates.

plot_heatmap *Plot Heatmap for Differential ORF Usage (DOU)*

Description

Generates a heatmap of differential ORF usage (DOU) estimates with hierarchical clustering and optional tile highlighting for significant ORFs. The function uses base R graphics and expects all required inputs to be passed via a named list produced by `cistronic_data()`.

Usage

```
plot_heatmap(paired_data)
```

Arguments

`paired_data` A named list containing heatmap data and metadata, typically the output from `cistronic_data()`. Must include the following elements:

ordered_matrix Matrix of DOU estimates for significant genes, ordered by hierarchical clustering.

row_dend_clean Dendrogram object for clustered gene rows.

highlight_df Data frame indicating which ORF-gene tiles to highlight in the heatmap.

gene_labels Vector of gene symbols or Ensembl IDs for heatmap row labeling.

color_palette Color palette used for heatmap visualization.

color_breaks Breaks used for color scaling in the heatmap.

abs_max Maximum absolute value used for symmetric color scaling.

Details

The heatmap displays log-odds changes in ORF usage across genes, comparing mORFs with short ORFs (e.g., uORFs or dORFs). Significant ORFs are highlighted using rectangles. The color scale is symmetric and centered at zero. The function uses base R plotting functions and is intended for interactive or scripted visualization.

Value

A heatmap with a dendrogram and color key.

plot_orf_usage *Plot ORF usage across conditions with significance annotations*

Description

Generates a faceted bar plot of ORF usage across experimental conditions, with optional significance annotations based on adjusted p-values (e.g., LFSR). Requires ggplot2 and ggsignif.

Usage

```
plot_orf_usage(
  data,
  gene_id = NULL,
  id_mapping = NULL,
  levels = NULL,
  dou_signif_thresh = 0.05,
  colors = "Set2"
)
```

Arguments

data	A DOUData-class object with an interaction slot storing interaction-specific post hoc contrast results.
gene_id	A character string specifying the gene ID of interest
id_mapping	Optional data frame with gene symbols (e.g., from biomaRt). Used to label heatmap rows with gene symbols.
levels	Optional character vector specifying the order of conditions on the x-axis.
dou_signif_thresh	Numeric threshold for significance annotation (default is 0.05).
colors	Color palette for scale_fill_brewer. Default is Set2.

Value

A ggplot object visualizing ORF usage across conditions, with significance annotations for contrasts passing dou_signif_thresh. ORFs with non-significant contrasts are not annotated.

---

plot_pca *Plot PCA for simulated RNA-seq or Ribo-seq data*

Description

Generates a PCA plot from simulated RNA-seq or Ribo-seq count data, highlighting sample-level variation across conditions and batches. The function applies variance-stabilizing transformation (VST), performs principal component analysis (PCA), and dynamically adjusts plot margins to accommodate legends based on device size and label width.

Usage

```
plot_pca(
  gcoeff,
  bcoeff,
  batch_scenario,
  num_batches,
  countdata,
  coldata,
  strategy = c("ribo", "rna"),
  formula1 = ~strategy,
  formula2 = ~condition + batch
)
```

Arguments

gcoeff	Numeric. Magnitude of log-fold change for DOT effects.
bcoeff	Numeric vector. Batch effect coefficients.
batch_scenario	Character. Describes the batch effect design (e.g., "balanced", "confounded").
num_batches	Integer. Number of batches.
countdata	A matrix or data frame of raw counts (genes x samples).
coldata	A data frame containing sample metadata. Must include condition, batch, and strategy columns.
strategy	Character string. Specifies which strategy to plot (e.g., "rna" or "ribo").
formula1	A formula object specifying the initial design for DESeq2 object construction. Default is ~strategy.
formula2	A formula object specifying the design for PCA modeling. Default is ~ condition + batch.

Value

A PCA plot is rendered to the active graphics device. The plot includes sample points colored by condition and shaped by batch, with legends placed dynamically to avoid overlap.

---

plot_venn *Plot Venn Diagram of DTE and DOU Significance Overlap*

Description

Generates a Venn diagram (Euler diagram) showing the overlap of significantly differentially translated ORFs (DTE) and differentially used ORFs (DOU). ORFs are classified as significant in DTE only, DOU only, or both. Requires eulerr.

Usage

```
plot_venn(
  results,
  dou_signif_col = "lfsr",
  dte_signif_col = "padj",
  dou_signif_thresh = 0.05,
  dte_signif_thresh = 0.05
)
```

Arguments

results	A data frame containing DTE and DOU results. Must include row names corresponding to ORF identifiers, and columns for adjusted p-values from both tests.
dou_signif_col	Character string specifying the column name for DOU significance values. Should correspond to local false sign rate (lfsr). Default is "lfsr".
dte_signif_col	Character string specifying the column name for DTE adjusted p-values. Default is "padj".
dou_signif_thresh	Numeric threshold for DOU LFSR significance. Default is 0.05.
dte_signif_thresh	Numeric threshold for DTE adjusted p-value significance. Default is 0.05.

Details

Significance is determined using a threshold of 0.05 on the adjusted p-values. The plot uses a color-blind friendly palette and includes counts for each region.

Value

A Venn diagram (Euler plot) showing the number of ORFs significant in DTE only, DOU only, or both.

References

Larsson J, Gustafsson P (2018). "A Case Study in Fitting Area-Proportional Euler Diagrams with Ellipses Using eulerr." In Proceedings of International Workshop on Set Visualization and Reasoning, volume 2116, 84–91. https://ceur-ws.org/Vol-2116/paper7.pdf

---

`plot_volcano` *Volcano Plot for Differential ORF Usage (DOU)*

Description

Generates a volcano plot to visualize differential ORF usage (DOU) results. The x-axis represents log-odds changes in ORF usage (effect sizes), and the y-axis shows the negative log10-transformed local false sign rate (LFSR). Points can be colored either by significance in DTE/DOU or by ORF type (uORF, mORF, dORF). Optional gene labeling is supported via an ID mapping table. The y-axis is capped at the nearest multiple of `dou_signif_ceil` for cleaner plotting.

Usage

```
plot_volcano(
  results,
  rowdata = NULL,
  id_mapping = NULL,
  color_by = c("significance", "orf_type"),
  dou_estimates_col = "posterior",
  dou_signif_col = "lfsr",
  dte_estimates_col = "log2FoldChange",
  dte_signif_col = "padj",
  dte_signif_thresh = 0.05,
  flip_sign = FALSE,
  dou_estimates_thresh = 1,
  dou_signif_thresh = 0.05,
  dou_signif_ceil = 10,
  extreme_thresh = NULL,
  top_hits = NULL,
  legend_position = "topright",
 colors = list(dte = adjustcolor("#0072B2", alpha.f = 0.6), dou = adjustcolor("#E69F00",
    alpha.f = 0.6), both = adjustcolor("#CC79A7", alpha.f = 0.6), none =
   adjustcolor("grey80", alpha.f = 0.6), uorf = adjustcolor("#D73027", alpha.f = 0.6),
   morf = adjustcolor("#4575B4", alpha.f = 0.6), dorf = adjustcolor("#A6A6A6", alpha.f =
    0.6)),
  verbose = TRUE
)
```

Arguments

results	A data frame containing DOU and optionally DTE results. Must include columns for DOU estimates and LFSR. If color_by = "orf_type", must also include an orf_type column with values "uORF", "mORF", or "dORF".
rowdata	Optional data frame containing ORF metadata, with row names corresponding to orf_id. Used to merge ORF type information.
id_mapping	Optional data frame with gene symbols. Used to label points with gene symbols. Must include ensembl_gene_id and symbol columns.
color_by	Character string specifying how to color points: • "significance": Colors by DTE-only, DOU-only, both significant • "orf_type": Colors by ORF type (requires orf_type column) Default is "significance".
dou_estimates_col	Column name for DOU effect size estimates. Default is "posterior".
dou_signif_col	Column name for DOU significance values (LFSR). Default is "lfsr".
dte_estimates_col	Column name for DTE effect size estimates. Default is "log2FoldChange".
dte_signif_col	Column name for DTE adjusted p-values. Default is "padj".
dte_signif_thresh	Numeric threshold for DTE adjusted p-value significance. Default is 0.05.
flip_sign	Logical. If TRUE, flips the sign of DOU estimates for plotting. Default is FALSE.
dou_estimates_thresh	Numeric threshold for DOU effect size significance. Default is 1.
dou_signif_thresh	Numeric threshold for DOU LFSR significance. Default is 0.05.
dou_signif_ceil	Numeric value to define the rounding ceiling for -log10(LFSR). The maximum y-axis value will be rounded up to the nearest multiple of this value. Default is 10.
extreme_thresh	Optional numeric threshold for labeling extreme points based on -log10(LFSR).
top_hits	Optional numeric. If provided, labels the top N most significant points.
legend_position	Position of the legend. Options include "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right", "center".
colors	A named list of colors used for plotting. Must include: • dte, dou, both, none: for significance-based coloring • uorf, morf, dorf: for ORF type-based coloring Each value should be a valid color string or result of adjustcolor().
verbose	Logical. If TRUE, prints messages about plot scaling and thresholds. Default is TRUE.

Value

A volcano plot is displayed in a new graphics device. Points are colored by significance or ORF type, depending on the color_by argument. Optionally, top or extreme points are labeled.

PostHoc *Construct a PostHoc Object*

Description

This function constructs a new PostHoc object, which stores the results of a statistical model fitted to ORF-level data. It is typically used internally by the **DOTSeq** workflow or manually for testing and diagnostics.

Usage

```
PostHoc(type = "fitError", results = list(), posthoc = NA_real_)
```

Arguments

type	A character(1) string indicating the model type. Default is "fitError".
results	A list containing model results, parameters, and test statistics.
posthoc	An optional object storing post hoc summary objects (e.g., from emmeans). Default is NA_real_.

Value

A PostHoc S4 object.

Examples

```
## Create a dummy PostHoc object
PostHocRes <- PostHoc(
    type = "glmmTMB",
    results = list(x = 3, y = 7, b = 4)
)
PostHocRes
```

PostHoc-class *The PostHoc class for DOTSeq*

Description

The PostHoc class represents post hoc summaries derived from a beta-binomial GLM/GLMM fitted to ORF-level data. It is also used to store diagnostics and metadata for each ORF.

Objects of this class are typically created by the user-level function fitDOU(), or manually using the PostHoc() constructor. In the **DOTSeq** workflow, each ORF is assigned a PostHoc object, which is stored in a DataFrame and embedded in the rowData slot of a SummarizedExperiment.

Slots

type A character(1) string indicating the model type. Default is "fitError". If the model is successfully fitted, the type is typically "glmmTMB".

results A list containing model results, parameters, and test statistics.

posthoc An object of class ANY storing post hoc summary objects (e.g., from emmeans).

Examples

```
## Create a dummy PostHoc object
PostHocRes <- PostHoc(
    type = "glmmTMB",
    results = list(
        model_fit = list(aic = 96.03),
        tests = list(pvalue_best = 0.355)
    )
)
PostHocRes
```

`reduce_formula` *Reduce a formula by removing terms with insufficient variation*

Description

This function takes a formula and a data frame, and returns a simplified formula that includes only terms corresponding to variables with at least two levels. Interaction terms using `*` or `:` are retained only if all involved variables are valid. If the formula is reduced, a message is printed to inform the user.

Usage

```
reduce_formula(formula_input, data)
```

Arguments

`formula_input`	A formula object (e.g., `~ batch + condition * strategy`) or a character string representing a formula.
`data`	A data frame containing the variables referenced in the formula.

Value

A reduced formula object that excludes terms with only one level in the data. If no valid terms remain, the function will throw an error.

Examples

```
## Not run:
  df <- data.frame(
    condition = factor(c(0, 0, 1, 1)),
    strategy = factor(c("ribo", "ribo", "rna", "rna")),
    batch = factor(rep(1, 4)) # single-level factor
  )
  formula_input <- ~ batch + condition * strategy
  reduce_formula(formula_input, df)
  # Returns: ~ condition * strategy, with a message about reduction

## End(Not run)
```

remove_random_effects *Remove random effects from a formula*

Description

This function takes an R formula object that may contain random effect terms and returns a new formula with all such terms removed.

Usage

```
remove_random_effects(formula)
```

Arguments

formula	An R formula object. Random effect terms are identified by the pattern `(1 | group)`.

Value

A new R formula object containing only the fixed effect terms.

reset_graphics *Reset and Recover Graphics Device*

Description

A wrapper function that resets the graphics layout and optionally opens a new graphics device before executing a plotting function. It is designed to prevent layout persistence and recover from common base R graphics errors such as "invalid graphics state" or "figure margins too large", which can occur when plotting multiple figures sequentially.

Usage

```
reset_graphics(plot_fn, force_new_device = TRUE)
```

Arguments

plot_fn	A function object that performs plotting. This function will be executed within a clean graphics context.
force_new_device	Logical; if `TRUE`, attempts to close and reopen the graphics device before plotting. This helps isolate plots and prevent layout or overlay issues. Default is `TRUE`.

Details

This function is useful when plotting multiple complex figures (e.g., composite plots with `layout()`) in sequence. It resets the layout using `layout(1)` and `par(mfrow = c(1, 1))` before plotting, and catches common graphics errors to retry the plot in a clean device.

Value

Invisibly returns the result of `plot_fn()`, if successful. If an error occurs and recovery is triggered, the function attempts to replot in a fresh graphics device.

---

reverseMinusStrandPerGroup *Reverse minus strand*

Description

Reverse minus strand per group in a GRangesList Only reverse if minus strand is in increasing order

Usage

```
reverseMinusStrandPerGroup(grl, onlyIfIncreasing = TRUE)
```

Arguments

grl	a GRangesList
onlyIfIncreasing	logical, default (TRUE), only reverse if decreasing

Value

a GRangesList

Author(s)

Haakon Tjeldnes et al.

References

Tjeldnes, H., Labun, K., Torres Cleuren, Y. et al. ORFik: a comprehensive R toolkit for the analysis of translation. BMC Bioinformatics 22, 336 (2021). DOI: 10.1186/s12859-021-04254-w

---

runtime *Format elapsed time between two timestamps*

Description

Computes the elapsed time between a start and end timestamp, returning a list with minutes and seconds. If the duration is less than one minute, only seconds are returned. Useful for logging runtimes in a human-readable format.

Usage

```
runtime(end_time, start_time, units = "secs")
```

Arguments

end_time	POSIXct. The end time of the process.
start_time	POSIXct. The start time of the process.
units	Character. Units for difftime() calculation. Default is "secs".

Value

A named list containing:

- mins (optional) — integer number of minutes
- secs — integer number of seconds

Examples

```
## Not run:
    start <- Sys.time()
    Sys.sleep(3.5)
    end <- Sys.time()
    runtime(end, start)

## End(Not run)
```

---

run_diagnostic *Run DHARMa Diagnostics on a Fitted Model*

Description

This function performs residual diagnostics on a fitted glmmTMB model using the **DHARMa** package. It simulates residuals and tests for overdispersion, zero inflation, uniformity, residual dispersion, and outliers. The results are stored in the diagnostics element of the provided results list.

Usage

```
run_diagnostic(fitted_model, results_list, diagnostics_name, plot = FALSE)
```

Arguments

fitted_model	A fitted glmmTMB model object.
results_list	A named list to which diagnostic results will be added. Must contain a diagnostics element.
diagnostics_name	A character string specifying the name under which diagnostic results will be stored in results_list$diagnostics.
plot	Logical; if TRUE, diagnostic plots will be generated. Default is FALSE.

Value

A modified version of results_list with diagnostic results added under results_list$diagnostics[[diagnostics_

Note

This function requires the **DHARMa** package. If it is not installed, the function will stop with an informative error message.

`seqnamesPerGroup` *Get list of seqnames per granges group*

Description

Get list of seqnames per granges group

Usage

```
seqnamesPerGroup(grl, keep.names = TRUE)
```

Arguments

`grl`	a `GRangesList`
`keep.names`	a boolean, keep names or not, default: (TRUE)

Value

a character vector or Rle of seqnames(if seqnames == T)

Author(s)

Haakon Tjeldnes et al.

References

Tjeldnes, H., Labun, K., Torres Cleuren, Y. et al. ORFik: a comprehensive R toolkit for the analysis of translation. BMC Bioinformatics 22, 336 (2021). DOI: 10.1186/s12859-021-04254-w

Examples

```
## Not run:
gr_plus <- GRanges(seqnames = c("chr1", "chr1"),
                   ranges = IRanges(c(7, 14), width = 3),
                   strand = c("+", "+"))
gr_minus <- GRanges(seqnames = c("chr2", "chr2"),
                    ranges = IRanges(c(4, 1), c(9, 3)),
                    strand = c("-", "-"))
grl <- GRangesList(tx1 = gr_plus, tx2 = gr_minus)
seqnamesPerGroup(grl)

## End(Not run)
```

simDOT *Simulate Differential ORF Translation (DOT)*

Description

Simulates ribosome profiling and matched RNA-seq count matrices with specified differential ORF translation (DOT) effects. The simulation can include batch effects and supports multiple experimental conditions and replicates.

Usage

```
simDOT(
  ribo,
  rna,
  annotation = NULL,
  regulation_type = NULL,
  te_genes = 10,
  bgenes = 10,
  num_samples = 2,
  conditions = 2,
  gcoeff = 1.5,
  bcoeff = 0.9,
  num_batches = 2,
  size_factor = NULL,
  min_size = NULL,
  scale_p0 = NULL,
  shape = 0.6,
  scale = 0.5,
  batch_scenario = "balanced",
  diagplot_ribo = FALSE,
  diagplot_rna = FALSE
)
```

Arguments

ribo	A matrix or data frame of ribosome profiling counts (genes x samples).
rna	A matrix or data frame of RNA-seq counts (genes x samples).
annotation	A GRanges object with ORF level annotation, typically obtained from getORFs.
regulation_type	Character. Specifies the type of DOT effect to simulate. Passed to the scenario argument of generate_coefficients.
te_genes	Numeric. Percentage of genes to be assigned as differentially translated (default: 10).
bgenes	Numeric. Percentage of genes to carry a batch effect (default: 10).
num_samples	Integer. Number of biological replicates per condition (default: 2).
conditions	Integer. Number of experimental conditions (default: 2).
gcoeff	Numeric. Magnitude of log-fold change for DOT effects (default: 1.5).
bcoeff	Numeric. Magnitude of batch effect coefficient (default: 0.9).

num_batches	Integer. Number of batches (default: 2).
size_factor	Numeric scalar. A multiplicative factor applied to the estimated size parameter (r) for all transcripts. Since dispersion $\phi = 1/r$, a value greater than 1 (e.g., 1.5) will decrease biological dispersion (noise), making the simulated data less variable. A value less than 1 will increase dispersion (default: 1.5).
min_size	Numeric scalar. A lower bound for the modified size parameter (r). Any transcript whose modified r falls below this value will be set to min_size. This caps maximum dispersion and prevents unrealistic variability (default: 5).
scale_p0	Optional numeric scalar to scale the zero-inflation probabilities.
shape	Numeric. Shape parameter for gamma distribution used to simulate baseline coefficients (default: 0.6).
scale	Numeric. Scale parameter for gamma distribution used to simulate baseline coefficients (default: 0.5).
batch_scenario	Character. Specifies the batch effect design. Must be one of: • "balanced" • "confounded" • "random" • "unbalanced" • "nested" • "modality_specific"
diagplot_ribo	Logical. If TRUE, generate diagnostic plots for ribo data (default: FALSE).
diagplot_rna	Logical. If TRUE, generate diagnostic plots for RNA data (default: FALSE).

Value

A DOTSeqDataSets-class object containing:

DOU A DOUData-class object containing simulated count matrix (assay slot), sample metadata (colData slot), and ORF-level annotation (rowRanges slot). The rowRanges slot also stores labels (named binary vector indicating true positive (1)), and logFC (log-fold changes for the simulated DOU effect) for modeling Differential ORF Usage (DOU).

DTE A DTEData-class object used for modeling Differential Translation Efficiency (DTE). Stores all data above except for rowRanges

References

Frazee, A. C., Jaffe, A. E., Langmead, B., & Leek, J. T. (2015). Polyester: Simulating RNA-seq datasets with differential transcript expression. Bioinformatics, 31(17), 2778-2784. DOI: 10.1093/bioinformatics/btv272

Chothani, S., Adami, E., Ouyang, J. F., Viswanathan, S., Hubner, N., Cook, S. A., Schafer, S., Rackham, O. J. L. (2019). deltaTE: Detection of translationally regulated genes by integrative analysis of Ribo-seq and RNA-seq data. Current Protocols in Molecular Biology, 129, e108. DOI: 10.1002/cpmb.108

Examples

```
library(SummarizedExperiment)
dir <- system.file("extdata", package = "DOTSeq")

cnt <- read.table(file.path(dir, "featureCounts.cell_cycle_subset.txt.gz"),
```

```
    header = TRUE, comment.char = "#"
)
names(cnt) <- gsub(".*(SRR[0-9]+).*", "\\1", names(cnt))

flat <- file.path(dir, "gencode.v47.orf_flattened_subset.gtf.gz")
bed <- file.path(dir, "gencode.v47.orf_flattened_subset.bed.gz")

meta <- read.table(file.path(dir, "metadata.txt.gz"))
names(meta) <- c("run", "strategy", "replicate", "treatment", "condition")
cond <- meta[meta$treatment == "chx", ]
cond$treatment <- NULL

d <- DOTSeqDataSetsFromFeatureCounts(
    count_table = cnt,
    condition_table = cond,
    flattened_gtf = flat,
    flattened_bed = bed
)
raw_counts <- assay(getDOU(d))
raw_counts <- raw_counts[, grep("Cycling|Interphase",
    colnames(raw_counts))]
ribo <- raw_counts[, grep("ribo", colnames(raw_counts))]
rna <- raw_counts[, grep("rna", colnames(raw_counts))]
rowranges <- rowRanges(getDOU(d))
r <- "uORF_up_mORF_down"
g <- 1.5
d <- simDOT(
    ribo,
    rna,
    annotation = rowranges,
    regulation_type = r,
    gcoeff = g,
    num_samples = 1,
    num_batches = 2
)

show(d)

rowData(getDOU(d))
```

`sortPerGroup` *Sort a GRangesList*

Description

A faster, more versatile reimplementation of `sort.GenomicRanges` for GRangesList, needed since the original works poorly for more than 10k groups. This function sorts each group, where "+" strands are increasing by starts and "-" strands are decreasing by ends.

Usage

```
sortPerGroup(grl, ignore.strand = FALSE, quick.rev = FALSE)
```

Arguments

grl	a GRangesList
ignore.strand	a boolean, (default FALSE): should minus strands be sorted from highest to lowest ends. If TRUE: from lowest to highest ends.
quick.rev	default: FALSE, if TRUE, given that you know all ranges are sorted from min to max for both strands, it will only reverse coordinates for minus strand groups, and only if they are in increasing order. Much quicker

Details

Note: will not work if groups have equal names.

Value

an equally named GRangesList, where each group is sorted within group.

Author(s)

Haakon Tjeldnes et al.

References

Tjeldnes, H., Labun, K., Torres Cleuren, Y. et al. ORFik: a comprehensive R toolkit for the analysis of translation. BMC Bioinformatics 22, 336 (2021). DOI: 10.1186/s12859-021-04254-w

---

specs *Access the emmeans specification formula*

Description

Retrieves the emmeans specification formula used for post hoc contrast generation.

Usage

```
specs(object)

## S4 method for signature 'DOUData'
specs(object)

## S4 method for signature 'DTEData'
specs(object)
```

Arguments

object	A DOUData-class or DTEData-class object.

Value

A formula object.

stopSites *Get the stop sites from a GRangesList of orfs grouped by orfs*

Description

In ATGTTTTGC, get the position of the C.

Usage

```
stopSites(grl, asGR = FALSE, keep.names = FALSE, is.sorted = FALSE)
```

Arguments

grl	a GRangesList object
asGR	a boolean, return as GRanges object
keep.names	a logical (FALSE), keep names of input.
is.sorted	a speedup, if you know the ranges are sorted

Value

if asGR is False, a vector, if True a GRanges object

Author(s)

Haakon Tjeldnes et al.

References

Tjeldnes, H., Labun, K., Torres Cleuren, Y. et al. ORFik: a comprehensive R toolkit for the analysis of translation. BMC Bioinformatics 22, 336 (2021). DOI: 10.1186/s12859-021-04254-w

See Also

Other ORFHelpers: longestORFs()

Examples

```
## Not run:
gr_plus <- GRanges(seqnames = c("chr1", "chr1"),
                   ranges = IRanges(c(7, 14), width = 3),
                   strand = c("+", "+"))
gr_minus <- GRanges(seqnames = c("chr2", "chr2"),
                    ranges = IRanges(c(4, 1), c(9, 3)),
                    strand = c("-", "-"))
grl <- GRangesList(tx1 = gr_plus, tx2 = gr_minus)
stopSites(grl, is.sorted = FALSE)

## End(Not run)
```

strandBool *Get logical list of strands*

Description

Helper function to get a logical list of True/False, if GRangesList group have + strand = T, if - strand = F Also checks for * strands, so a good check for bugs

Usage

```
strandBool(grl)
```

Arguments

grl	a GRangesList or GRanges object

Value

a logical vector

Author(s)

Haakon Tjeldnes et al.

References

Tjeldnes, H., Labun, K., Torres Cleuren, Y. et al. ORFik: a comprehensive R toolkit for the analysis of translation. BMC Bioinformatics 22, 336 (2021). DOI: 10.1186/s12859-021-04254-w

Examples

```
## Not run:
gr <- GRanges(Rle(c("chr2", "chr2", "chr1", "chr3"), c(1, 3, 2, 4)),
              IRanges(1:10, width = 10:1),
              Rle(strand(c("-", "+", "*", "+", "-")), c(1, 2, 2, 3, 2)))
strandBool(gr)

## End(Not run)
```

strandPerGroup *Get list of strands per granges group*

Description

Get list of strands per granges group

Usage

```
strandPerGroup(grl, keep.names = TRUE)
```

Arguments

grl	a GRangesList
keep.names	a boolean, keep names or not, default: (TRUE)

Value

a vector named/unnamed of characters

Author(s)

Haakon Tjeldnes et al.

References

Tjeldnes, H., Labun, K., Torres Cleuren, Y. et al. ORFik: a comprehensive R toolkit for the analysis of translation. BMC Bioinformatics 22, 336 (2021). DOI: 10.1186/s12859-021-04254-w

---

tally_chunk_vec *Tally per-chunk single-cell feature counts using per-seqname NCList indexes*

Description

Vectorized overlap counting for a chunk of alignments with cell barcode (CB) and optional UMI (UB) annotations. Reads are split by seqnames and overlapped against pre-built GNCList indexes for the matching seqname. Counts are accumulated into a sparse matrix of dimension n_features x n_cells.

Usage

```
tally_chunk_vec(
  read_gr,
  cb,
  ub,
  mapq_vec,
  gr_common,
  feature_ids,
  n_features,
  nclist_by_seq,
  subj_global_idx_by_seq,
  ignore.strand,
  mapq_min,
  cells,
  dedup,
  mode
)
```

Arguments

read_gr	GRanges of read (or fragment) intervals for a single chunk. Must be aligned to the same reference naming as the feature indexes.
cb	Character vector of cell barcodes (one per read in read_gr). NA values are dropped.
ub	Optional character vector of UMIs (one per read). If NULL, UMI-based deduplication is disabled regardless of dedup.
mapq_vec	Optional integer/numeric vector of MAPQ values (one per read). If provided, reads with mapq < mapq_min are dropped.
gr_common	GRanges of all features used to define the global row space. (Not directly used for overlap here, but defines global indexing.)
feature_ids	Character vector of length n_features giving global feature identifiers used as row names of the returned matrix.
n_features	Integer number of global features (rows).
nclist_by_seq	Named list of GNCList objects, one per seqname, built from the per-seqname split of gr_common. Names must correspond to seqnames present in read_gr.
subj_global_idx_by_seq	Named list mapping each seqname to an integer vector of global feature indices (rows of gr_common) corresponding to the features used to build nclist_by_seq[[seq]]. This is used to translate subjectHits() (local per-seqname feature indices) into global row indices in 1:n_features.
ignore.strand	Logical; passed to findOverlaps().
mapq_min	Minimum MAPQ threshold. Only used if mapq_vec is not NULL.
cells	Optional character vector of allowed cell barcodes. If provided, only these cells are retained and output columns are returned in this order.
dedup	Logical. If TRUE and ub is provided, deduplicate UMIs per (cell, feature) within the chunk (i.e., each unique UMI contributes at most 1 count to a (cell, feature) pair).
mode	Character overlap mode placeholder. Currently unused in the implementation shown; reserved for future extensions (e.g., intersection logic).

Details

Deduplication semantics. When dedup=TRUE and ub is not NULL, counts are deduplicated by unique triplets (cell, feature, UMI) within the current chunk and seqname. If the same (cell, feature, UMI) appears multiple times in the chunk, it is counted once.

Column ordering. If cells is supplied, columns are aligned to this whitelist/order (via match()). Otherwise, columns are based on the set of barcodes present within the chunk (sorted).

Performance notes. This function currently constructs and merges per-seqname sparse matrices using repeated column union and padding. For very large numbers of cells, it can be faster to accumulate triplets and build a single sparse matrix per chunk.

Value

A sparse matrix (dgCMatrix) with n_features rows and one column per cell observed/retained in the chunk. Row names are feature_ids. Column names are either:

- cells (if provided; fixed order), or
- sorted unique barcodes observed in the chunk (if cells is NULL).

If no reads pass filtering or no overlaps are found, returns NULL.

See Also

findOverlaps, GNCList, sparseMatrix

---

testDOU *Compute Differential ORF Usage (DOU) Contrasts Using EMMs*

Description

Performs Differential ORF Usage (DOU) analysis by computing contrasts between ribosome profiling and RNA-seq modalities using estimated marginal means (EMMs) from fitted models. Supports interaction-specific and strategy-specific contrasts. Applies empirical Bayes shrinkage via ash to stabilize effect size estimates.

Usage

```
testDOU(
  data,
  contrasts_method = "revpairwise",
  nullweight = 500,
  verbose = TRUE
)
```

Arguments

data	A DOUData-class object containing emmGrid objects, typically stored in rowData(data)[['DOURes
contrasts_method	Character string specifying the method for computing contrasts. Default is "revpairwise".
nullweight	Numeric. Prior weight on the null hypothesis for empirical Bayes shrinkage. Higher values yield more conservative lfsr estimates. Default is 500.
verbose	Logical. If TRUE, prints progress messages. Default is TRUE.

Details

Results for post hoc contrasts are stored in long format using explicit contrast and/or strategy columns. Non-converged models are omitted.

Value

A DOUData-class object with two new S4Vectors::DataFrame slots. These tables contain long-format results for all computed contrasts:

interaction A long-format S4Vectors::DataFrame containing DOU effect sizes (log-odds of Ribo-seq minus RNA-seq) for all interaction-specific contrasts. Columns include: contrast, betahat (raw log-odds effect size), sebetahat (standard error), waldpval (Wald test p-value), waldpadj (adjusted p-value), posterior (posterior mean from shrinkage), and lfsr (Local False Sign Rate).

strategy A long-format S4Vectors::DataFrame containing strategy-specific effect sizes (e.g., Ribo-seq only) for all computed contrasts. Columns include: strategy, contrast, and the same shrunken and unshrunken metrics as above.

References

Lenth R, Piaskowski J (2025). emmeans: Estimated Marginal Means, aka Least-Squares Means. R package version 2.0.0, https://rvlenth.github.io/emmeans/

Stephens, M. (2016) False discovery rates: a new deal. Biostatistics, 18:2. DOI: 10.1093/biostatistics/kxw041

See Also

DOTSeq, DOTSeqDataSets-class, fitDOU, plotDOT

Examples

```
# Load SummarizedExperiment to enable subsetting and access to
# components like rowRanges and rowData
library(SummarizedExperiment)

# Read in count matrix, condition table, and annotation files
dir <- system.file("extdata", package = "DOTSeq")

cnt <- read.table(
    file.path(dir, "featureCounts.cell_cycle_subset.txt.gz"),
    header = TRUE,
    comment.char = "#"
)
names(cnt) <- gsub(".*(SRR[0-9]+).*", "\\1", names(cnt))

gtf <- file.path(dir, "gencode.v47.orf_flattened_subset.gtf.gz")
bed <- file.path(dir, "gencode.v47.orf_flattened_subset.bed.gz")

meta <- read.table(file.path(dir, "metadata.txt.gz"))
names(meta) <- c("run", "strategy", "replicate", "treatment", "condition")
# extract only samples processed using cyclohexamide
cond <- meta[meta$treatment == "chx", ]
cond$treatment <- NULL # remove the treatment column

# Create a DOTSeqDataSets object
d <- DOTSeqDataSetsFromFeatureCounts(
    count_table = cnt,
    condition_table = cond,
    flattened_gtf = gtf,
    flattened_bed = bed
)

# Keep ORFs where all replicates in at least one condition pass min_count
# Single-ORF genes are removed
dou <- getDOU(d)
dou <- dou[rowRanges(dou)$is_kept == TRUE, ]

# Randomly sample 100 ORFs for fitDOU
set.seed(42)
random_rows <- sample(seq_len(nrow(dou)), size = 100)
dou <- dou[random_rows, ]

# Model fitting using fitDOU
rowData(dou)[["DOUResults"]] <- fitDOU(
    data = dou,
```

```
    formula = ~ condition * strategy,
    specs = ~ condition * strategy,
    dispersion_modeling = "auto",
    lrt = FALSE,
    optimizers = FALSE,
    diagnostic = FALSE,
    parallel = list(n = 4L, autopar = TRUE),
    verbose = TRUE
)

# Run post hoc contrasts, Wald tests, and effect size shrinkage
dou <- testDOU(dou, verbose = TRUE)
```

`widthPerGroup` *Get list of widths per granges group*

Description

Get list of widths per granges group

Usage

```
widthPerGroup(grl, keep.names = TRUE)
```

Arguments

`grl`	a GRangesList
`keep.names`	a boolean, keep names or not, default: (TRUE)

Value

an integer vector (named/unnamed) of widths

Author(s)

Haakon Tjeldnes et al.

References

Tjeldnes, H., Labun, K., Torres Cleuren, Y. et al. ORFik: a comprehensive R toolkit for the analysis of translation. BMC Bioinformatics 22, 336 (2021). DOI: 10.1186/s12859-021-04254-w

Examples

```
## Not run:
gr_plus <- GRanges(seqnames = c("chr1", "chr1"),
                   ranges = IRanges(c(7, 14), width = 3),
                   strand = c("+", "+"))
gr_minus <- GRanges(seqnames = c("chr2", "chr2"),
                    ranges = IRanges(c(4, 1), c(9, 3)),
                    strand = c("-", "-"))
grl <- GRangesList(tx1 = gr_plus, tx2 = gr_minus)
```

```
widthPerGroup(grl)

## End(Not run)
```

Index