Package: DOTSeq (via r-universe)

May 16, 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

Config/pak/sysreqs cmake make libbz2-dev libicu-dev liblzma-dev libpng-dev libxml2-dev libssl-dev xz-utils zlib1g-dev

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

Date/Publication 2026-04-28 13:06:05 UTC

RemoteUrl https://github.com/bioc/DOTSeq

RemoteRef HEAD

RemoteSha 8d37873b5e61305cc47f44c11a05b542a0f5763a

Contents

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)
```

`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", # dup
 "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)
```

`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)
```

`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
)
```

`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)
```

---

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.")
}
```

---

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

---

`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)
```

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
```

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.

`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))
```

`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)[['DOUResults`
`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)
```

Index