Package 'inDAGO'

July 5, 2025

Title A GUI for Dual and Bulk RNA-Sequencing Analysis

Version 1.0.0

Description A 'shiny' app that supports both dual and bulk RNA-seq, with the dual RNA-seq functionality offering the flexibility to perform either a sequential approach (where reads are mapped separately to each genome) or a combined approach (where reads are aligned to a single merged genome). The user-friendly interface automates the analysis process, providing step-by-step guidance, making it easy for users to navigate between different analysis steps, and download intermediate results and publication-ready plots.

License GPL (>= 3)

URL https://github.com/inDAGOverse/inDAGO

BugReports https://github.com/inDAGOverse/inDAGO/issues

Encoding UTF-8

RoxygenNote 7.3.2

Imports bigtabulate, BiocGenerics, Biostrings, bsicons, bslib, callr, checkmate, data.table, dplyr, DT, edgeR, fs, ggplot2, ggrepel, grDevices, heatmaply, Hmisc, htmltools, HTSFilter, limma, magrittr, matrixStats, memuse, methods, paletteer, parallel, pheatmap, plotly, R.devices, readr, reshape2, Rfastp, rintrojs, Rsamtools, Rsubread, rtracklayer, S4Vectors, seqinr, shiny, shinycssloaders, shinyFiles, shinyjs, shinyWidgets, ShortRead, spsComps, stats, tibble, tidyr, tools, upsetjs, UpSetR, utils, XVector

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Carmine Fruggiero [aut, cre], Gaetano Aufiero [aut]

Maintainer Carmine Fruggiero <fruggierocarmine3@gmail.com>

Repository CRAN

Date/Publication 2025-07-05 15:00:02 UTC

Contents

`barplotExp` *barplotExp*

Description

Create a barplot of library sizes per sample, optionally using effective library sizes.

Usage

```
barplotExp(x, palette, main, selectOrder, effecLibSize)
```

Arguments

`x`	A DGEList object from "edgeR".
`palette`	Character. Name of a discrete color palette from the "paletteer" package.
`main`	Character. Title for the barplot.
`selectOrder`	Character. Either "Groups" (order samples by group) or "Samples" (order by sample name).
`effecLibSize`	Logical. If TRUE, use effective library size (norm factors × raw size); otherwise use raw size.

Details

This function extracts library size information from an "edgeR" "DGEList", computes effective library sizes if requested, orders samples by group or name, and plots library sizes (in millions) colored by group.

1. Extracts or computes (effecLibSize = TRUE) the library size for each sample.
2. Orders samples by group or sample name per selectOrder.
3. Plots bar heights as library size (×10^6) with white fill and colored borders.

Value

A "ggplot" object showing per-sample barplots of library size in millions.

`BaseAverageQualityPlot`
BaseAverageQualityPlot

Description

BaseAverageQualityPlot

Usage

```
BaseAverageQualityPlot(input_data)
```

Arguments

`input_data`	folder containing data

`BaseAverageQualityPlotly`
interactive BaseAverageQualityPlot

Description

interactive BaseAverageQualityPlot

Usage

```
BaseAverageQualityPlotly(input_data)
```

Arguments

`input_data`	folder containing data

`BaseCompositionAreaChartPlot` *BaseCompositionAreaChartPlot*

Description

BaseCompositionAreaChartPlot

Usage

```
BaseCompositionAreaChartPlot(input_data)
```

Arguments

`input_data`	folder containing data

`BaseCompositionLinePlot` *BaseCompositionLinePlot*

Description

BaseCompositionLinePlot

Usage

```
BaseCompositionLinePlot(input_data)
```

Arguments

`input_data`	folder containing data

`BaseQualityBoxplotPlot` *BaseQualityBoxplotPlot*

Description

BaseQualityBoxplotPlot

Usage

```
BaseQualityBoxplotPlot(input_data)
```

Arguments

`input_data`	folder containing data

boxplotExp *boxplotExp*

Description

Generate a boxplot of log-CPM expression values per sample, colored by group.

Usage

```
boxplotExp(x, y, palette, main, selectOrder)
```

Arguments

x	A DGEList object from "edgeR".
y	Numeric matrix of log-CPM values (genes × samples), e.g., from edgeR::cpm().
palette	Character. Name of a discrete palette from the paletteer package.
main	Character. Title for the boxplot.
selectOrder	Character. Either "Groups" (order samples by group) or "Samples" (order by sample name).

Details

This function orders samples by group or sample name, and produces a ggplot2 boxplot with a horizontal line at the overall median.

1. Extract sample metadata (Samples, Groups) from "x$samples".
2. Order columns of y by group or sample name per "selectOrder".
3. Melt the ordered matrix to long format and join with metadata.
4. Plot boxplots with no outliers, colored by group, and include a dashed line at the overall median.

Value

A ggplot object showing per-sample boxplots of log-CPM values.

BulkAlignment *Bulk alignment function*

Description

Bulk alignment function

Usage

```
BulkAlignment(
  lalista,
  nodes,
  readsPath,
  GenomeIndex,
  outBam,
  threads,
  outFormat,
  phredScore,
  maxExtractedSubreads,
  consensusVote,
  mismatchMax,
  uniqueOnly,
  maxMultiMapped,
  indelLength,
  fragmentMinLength,
  fragmentMaxLength,
  matesOrientation,
  readOrderConserved,
  coordinatesSorting,
  allJunctions,
  tempfolder
)
```

Arguments

lalista	list of samples
nodes	logic cores
readsPath	sample folders
GenomeIndex	genome index
outBam	output folder
threads	processes
outFormat	BAM or SAM
phredScore	quality score
maxExtractedSubreads	number of subreads

consensusVote	consensus
mismatchMax	mismatch
uniqueOnly	no multimapping
maxMultiMapped	multimapping
indelLength	indel
fragmentMinLength	fragment minumum length
fragmentMaxLength	fragment maximum length
matesOrientation	mate orientation
readOrderConserved	read order
coordinatesSorting	sorting
allJunctions	junctions
tempfolder	temporary folder

---

checkMetadata *checkMetadata*

Description

Validate and extract non-empty annotation fields from a GTF file.

Usage

```
checkMetadata(gtfPath, typeFilter)
```

Arguments

gtfPath	Character. Path to the directory or file location of the GTF file.
typeFilter	Character. The feature type to filter on (e.g., "gene", "exon").

Details

This function imports a GTF file, filters entries by a specified feature type, and identifies metadata columns that contain at least one non-missing value.

1. Imports the GTF into a data frame via "rtracklayer::import()".
2. Filters rows by "type" == typeFilter.
3. Tests each column for all-NA or empty-string entries.
4. Returns names of columns with at least one non-missing, non-empty value.

Value

Character vector of column names in the GTF annotation that are not entirely NA or empty.

CombinedAlignment *Title*

Description

Title

Usage

```
CombinedAlignment(
  lalista,
  nodes,
  readsPath,
  GenomeConcIndex,
  outBam,
  threads,
  outFormat,
  phredScore,
  maxExtractedSubreads,
  consensusVote,
  mismatchMax,
  uniqueOnly,
  maxMultiMapped,
  indelLength,
  fragmentMinLength,
  fragmentMaxLength,
  matesOrientation,
  readOrderConserved,
  coordinatesSorting,
  allJunctions,
  tempfolder,
  readsAlignedBlock
)
```

Arguments

lalista	list of samples
nodes	logic cores
readsPath	sample folders
GenomeConcIndex	genome index
outBam	output folder
threads	processes
outFormat	BAM or SAM
phredScore	quality score

`maxExtractedSubreads`
: number of subreads

`consensusVote`
: consensus

`mismatchMax`
: mismatch

`uniqueOnly`
: no multimapping

`maxMultiMapped`
: multimapping

`indelLength`
: indel

`fragmentMinLength`
: fragment minumum length

`fragmentMaxLength`
: fragment maximum length

`matesOrientation`
: mate orientation

`readOrderConserved`
: read order

`coordinatesSorting`
: sorting

`allJunctions`
: junctions

`tempfolder`
: temporary folder

`readsAlignedBlock`
: chunks

`CorrPlotHeatmap` *CorrPlotHeatmap*

Description

Plot a correlation heatmap of top variable genes across samples.

Usage

```
CorrPlotHeatmap(
  x,
  scale,
  Color,
  type,
  display,
  round_number,
  cutree_rows,
  cutree_cols,
  cluster,
  show_names,
  NumGenes
)
```

Arguments

x	Numeric matrix of log-CPM values (genes × samples), e.g., from "edgeR::cpm()".
scale	Character. Scaling mode for the heatmap: "row", "column", or "none".
Color	Character. Name of a continuous palette from the "paletteer" package.
type	Character. Correlation method passed to "Hmisc::rcorr()": "pearson", "spearman", or "kendall".
display	Character. Which matrix to display: "correlation" (coefficients) or "pvalue".
round_number	Integer. Number of decimal places to round displayed numbers.
cutree_rows	Integer. Number of clusters to cut for row dendrogram.
cutree_cols	Integer. Number of clusters to cut for column dendrogram.
cluster	Character. Clustering mode: one of "both", "row", "column", or "none".
show_names	Character. One of "both", "row", "column", or "none" to display row/column labels.
NumGenes	Integer. Number of top-variance genes to include in the correlation.

Details

This function selects the highest-variance genes from a log-CPM matrix, computes pairwise correlation coefficients (or p-values) with "Hmisc::rcorr()", and renders a heatmap via "pheatmap", with options for clustering, scaling, and number display.

1. Compute per-gene variance and select the top "NumGenes".
2. Subset the matrix and compute correlations (and p-values) via "Hmisc::rcorr()".
3. Choose to display correlation coefficients or p-values, rounded to "round_number".
4. Determine clustering and label visibility from cluster and "show_names".
5. Render the heatmap with "pheatmap::pheatmap()", passing in custom distance, color, clustering, and "display" number settings, saving to a temporary file to suppress autosave.

Value

A "pheatmap" object representing the correlation heatmap with clustering.

CorrPlotHeatmaply *CorrPlotHeatmaply*

Description

Create an interactive correlation heatmap of top variable genes using Heatmaply.

Usage

```
CorrPlotHeatmaply(x, Color, type, cluster, scale, show_names, NumGenes)
```

Arguments

x	Numeric matrix of log-CPM values (genes × samples), e.g., from "edgeR::cpm()".
Color	Character. Name of a continuous palette from the "paletteer" package.
type	Character. Correlation method passed to "Hmisc::rcorr()": "pearson", "spearman", or "kendall".
cluster	Character or logical. Clustering option for dendrogram: "both", "row", "column", or "none".
scale	Character. Scaling mode for the heatmap: "row", "column", or "none".
show_names	Character. One of "both", "row", "column", or "none" to display row/column labels.
NumGenes	Integer. Number of top-variance genes to include in the correlation.

Details

This function selects the highest-variance genes from a log-CPM matrix, computes pairwise correlation coefficients (and p-values) with "Hmisc::rcorr()", and renders an interactive correlation heatmap via "heatmaply::heatmaply_cor()", using clustering and scaling options derived from "pheatmap" call.

1. Compute per-gene variance and select the top "NumGenes".
2. Subset the matrix and compute correlations (and p-values) via "Hmisc::rcorr()".
3. Generate a temporary static heatmap with "pheatmap" to extract dendrograms.
4. Render an interactive heatmap with "heatmaply::heatmaply_cor()", passing in color, clustering, scaling, tick-label visibility, and point size based on -log10(p-value).

Value

A Plotly object (heatmaply) representing the interactive correlation heatmap.

---

counting_Reads *COUNTING SEQUENCES*

Description

COUNTING SEQUENCES

Usage

```
counting_Reads(input_data)
```

Arguments

input_data	sample folder

DEGsServerLogic *Server function for DEGs module in Shiny application*

Description

Server function for DEGs module in Shiny application

Usage

```
DEGsServerLogic(id)
```

Arguments

id	Shiny module identifier

DEGsUserInterface *UI function for DEGs module in Shiny application*

Description

UI function for DEGs module in Shiny application

Usage

```
DEGsUserInterface(id)
```

Arguments

id	Shiny module identifier

EDAServerLogic *Server function for EDA module in Shiny application*

Description

Server function for EDA module in Shiny application

Usage

```
EDAServerLogic(id)
```

Arguments

id	Shiny module identifier

EDAUserInterface *UI function for EDA module in Shiny application*

Description

UI function for EDA module in Shiny application

Usage

```
EDAUserInterface(id)
```

Arguments

id	Shiny module identifier

EdgerDEG *EdgerDEG*

Description

Perform differential expression analysis on RNA-seq count data using edgeR.

Usage

```
EdgerDEG(
  gr,
  WD_samples,
  WD_DEGs,
  colIDgene,
  colCounts,
  skip_preN,
  grContrast,
  filter,
  model,
  normMethod,
  min_count,
  min_total_count,
  large_n,
  min_prop,
  adjustPvalue,
  Th_logFC,
  Th_Pvalue
)
```

Arguments

gr	Data frame. Sample metadata with columns Samples and Groups.
WD_samples	Character. Directory containing raw count .tab files.
WD_DEGs	Character. Directory in which to write results and logs.
colIDgene	Integer. Column index in each count file for gene IDs.
colCounts	Integer. Column index in each count file for raw counts.
skip_preN	Integer. Number of header lines to skip when reading count files.
grContrast	Data frame. Two-column table with Test and Baseline group names for contrasts.
filter	Character. Filtering method: "filterByExpr" or "HTSFilter".
model	Character. Statistical test: "exactTest", "glmQLFTest", or "glmLRT".
normMethod	Character. Normalization method for edgeR (e.g., "TMM", "RLE").
min_count	Numeric. Minimum count per gene for "filterByExpr".
min_total_count	Numeric. Minimum total count per gene for "filterByExpr".
large_n	Integer. Sample size threshold for "filterByExpr".
min_prop	Numeric. Proportion threshold for "filterByExpr".
adjustPvalue	Character. P-value adjustment method (e.g., "fdr", "holm", "none").
Th_logFC	Numeric. Absolute log-fold-change threshold to call differential expression.
Th_Pvalue	Numeric. Adjusted p-value threshold to call differential expression.

Details

This function reads raw count tables, applies expression filtering (via "filterByExpr" or "HTSFilter"), normalizes library sizes, estimates dispersion, fits statistical models ("exactTest", "glmQLFTest", or "glmLRT"), and writes per-contrast results and diagnostic plots.

1. Reads in per-sample count files and generate a DGEList.
2. Builds the design matrix and contrast definitions from "grContrast".
3. Filters lowly expressed genes, normalizes library sizes, and logs filtering summary.
4. Estimates dispersion (standard or quasi-likelihood).
5. Runs chosen differential test per contrast, annotates each gene as "UP", "DOWN", or "NO", and writes CSV output files named by filter, model, and contrast.
6. Captures and saves BCV and QL dispersion plots as SVGs in WD_DEGs.

Value

A list invisibly returned containing any captured plots and log messages; primary results are written to CSV files in "WD_DEGs".

`Filtering` *Filtering*

Description

Filter paired-end FASTQ files in parallel based on quality and adapter trimming criteria.

Usage

```
Filtering(
  Nodes,
  X,
  UploadPath,
  DownloadPath,
  qualityType,
  minLen,
  trim,
  trimValue,
  n,
  Adapters,
  Lpattern,
  Rpattern,
  max.Lmismatch,
  max.Rmismatch,
  kW,
  left,
  right,
  halfwidthAnalysis,
  halfwidth,
  compress
)
```

Arguments

`Nodes`	Integer. Number of parallel processing nodes (e.g., CPU cores).
`X`	List of character vectors. Each element is a character vector of paired file names (e.g., c("sample_1.fq", "sample_2.fq")).
`UploadPath`	Character. Path to directory containing raw FASTQ files.
`DownloadPath`	Character. Path to directory where filtered files will be saved.
`qualityType`	Character. Type of quality score encoding, e.g., "Sanger" or "Illumina".
`minLen`	Integer. Minimum length of reads to retain after filtering.
`trim`	Logical. Whether to perform quality-based trimming of reads.
`trimValue`	Integer. Minimum Phred score threshold for trimming.
`n`	Integer. Number of reads to stream per chunk (default typically set to 1e6).

Adapters	Logical. Whether to remove adapters from reads.
Lpattern	Character. Adapter sequence to remove from the 5' end (left).
Rpattern	Character. Adapter sequence to remove from the 3' end (right).
max.Lmismatch	Integer. Maximum mismatches allowed for the left adapter.
max.Rmismatch	Integer. Maximum mismatches allowed for the right adapter.
kW	Integer. Minimum number of low-quality scores in a window to trigger trimming (sliding window analysis).
left	Logical. Whether to allow trimming from the left end.
right	Logical. Whether to allow trimming from the right end.
halfwidthAnalysis	Logical. Whether to perform sliding window-based trimming.
halfwidth	Integer. Half-width of the sliding window.
compress	Logical. Whether to compress the output FASTQ files.

Details

This function processes raw paired-end FASTQ files to remove low-quality bases, trim adapters, and filter out short reads. It supports quality-based end trimming, sliding window trimming, and adapter removal. The processing is done in parallel across multiple nodes to enhance performance when working with large datasets.

- Paired FASTQ files must be named consistently, distinguished by "_1" and "_2" for forward and reverse reads.
- This function uses the "ShortRead" and "Biostrings" packages for FASTQ processing and quality filtering.
- Filtered files in FASTQ format".
- Log files containing read counts before and after filtering are written per sample.

Value

Filtered FASTQ files written to "DownloadPath"; one log file per sample.

FilteringServerLogic *Server function for filtering module in Shiny application*

Description

Server function for filtering module in Shiny application

Usage

```
FilteringServerLogic(id)
```

Arguments

id	Shiny module identifier

`FilteringUserInterface` *UI function for filtering module in Shiny application*

Description

UI function for filtering module in Shiny application

Usage

```
FilteringUserInterface(id)
```

Arguments

`id`	Shiny module identifier

`GCcontentDistributionPlot` *GCcontentDistributionPlot*

Description

GCcontentDistributionPlot

Usage

```
GCcontentDistributionPlot(input_data)
```

Arguments

`input_data`	samples folder

`GCcontentDistributionPlotly` *interactive GCcontentDistributionPlot*

Description

interactive GCcontentDistributionPlot

Usage

```
GCcontentDistributionPlotly(input_data)
```

Arguments

`input_data`	samples folder

getDegMerged *getDegMerged*

Description

Merge multiple DEG result CSVs with GTF annotations into a single data frame.

Usage

```
getDegMerged(path, gtfPath, columns, collapseName, typeFilter, selectUpDown)
```

Arguments

path	Character. Directory containing DEG result CSV files.
gtfPath	Character. Path to the GTF annotation file.
columns	Character vector. Names of annotation columns to include from the GTF.
collapseName	Logical. If TRUE, strip method/model prefixes from file names when prefixing columns.
typeFilter	Character. GTF feature type to filter (e.g., "gene" or "transcript").
selectUpDown	Logical. If TRUE, only include IDs with "diffExp" == UP or DOWN.

Details

This function reads all CSV files in a directory, validates presence of required columns ("ID", and optionally "diffExp"), filters for up/down regulated genes if requested, extracts annotation fields from a GTF, and returns a merged table of selected annotation columns alongside all DEG metrics (with optional file-based column prefixes).

Value

A combined data frame

GetEdgerY *GetEdgerY*

Description

Calculate and return filtered DGEList object and log-CPM matrices using edgeR and optional HTS-Filter

Usage

```
GetEdgerY(
  gr,
  WDpn,
  colIDgene,
  colCounts,
  skip_preN,
  filterMethod,
  min_count,
  min_total_count,
  large_n,
  min_prop,
  normMethod
)
```

Arguments

gr	Data frame with sample metadata, including sample names and group labels
WDpn	Directory containing count files (*.tab)
colIDgene	Column index of gene IDs in count files
colCounts	Column index of counts in count files
skip_preN	Number of header lines to skip in count files
filterMethod	Either "filterByExpr" or "HTSFilter"
min_count	Minimum count per gene (filterByExpr)
min_total_count	Minimum total count per gene (filterByExpr)
large_n	Number of samples per group to consider as "large" (filterByExpr)
min_prop	Minimum proportion of samples with expression (filterByExpr)
normMethod	Normalization method (e.g., "TMM", "RLE")

Value

A list with total/kept gene counts, filtered DGEList objects, and log-CPM matrices

---

HeatmapExp *HeatmapExp*

Description

Plot a heatmap of the top variable genes across samples.

Usage

```
HeatmapExp(
  x,
  ColorPanel,
  scale,
  cutree_rows,
  cutree_cols,
  cluster,
  show_names,
  NumGenes
)
```

Arguments

x	Numeric matrix of log-CPM values (genes × samples), e.g., from edgeR::cpm().
ColorPanel	Character. Name of a continuous palette from the paletteer package.
scale	Character. Scaling mode for heatmap: "row", "column", or "none".
cutree_rows	Integer. Number of clusters for rows (genes).
cutree_cols	Integer. Number of clusters for columns (samples).
cluster	Character. One of "both", "row", "column", or "none" to specify clustering.
show_names	Character. One of "both", "row", "column", or "none" to show row/col names.
NumGenes	Integer. Number of top-variance genes to include in the heatmap.

Details

This function selects the highest-variance genes from a log-CPM matrix, transposes the data, and renders a heatmap with customizable clustering, scaling, and color palettes using pheatmap.

1. Compute per-gene variance and select the top "NumGenes".
2. Transpose the subsetted matrix so samples are rows.
3. Apply the specified color palette (n = 50) via paletteer::paletteer_c().
4. Determine clustering and name-display options from "cluster" and "show_names".
5. Render the heatmap with "pheatmap::pheatmap()", saving to a temporary file to suppress autosave.

Value

A "pheatmap" object containing the heatmap and clustering information.

HeatmapExpPlotly *HeatmapExpPlotly*

Description

Create an interactive heatmap of top variable genes using Heatmaply.

Usage

```
HeatmapExpPlotly(x, ColorPanel, scale, cluster, show_names, NumGenes)
```

Arguments

x	Numeric matrix of log-CPM values (genes × samples), e.g., from edgeR::cpm().
ColorPanel	Character. Name of a continuous palette from the paletteer package.
scale	Character. Scaling mode: "row", "column", or "none".
cluster	Character or logical. Clustering option for dendrogram: "both", "row", "column", or "none".
show_names	Character. One of "both", "row", "column", or "none" to display row/column labels.
NumGenes	Integer. Number of top-variance genes to include in the heatmap.

Details

This function selects the highest-variance genes from a log-CPM matrix, transposes the data, and renders an interactive heatmap via "heatmaply", using "pheatmap" call.

1. Compute per-gene variance and select the top NumGenes.
2. Transpose the subsetted matrix so samples are rows.
3. Generate a temporary static heatmap with pheatmap to extract dendrograms.
4. Render an interactive heatmap with heatmaply::heatmaply().

Value

A Plotly object (heatmaply) representing the interactive heatmap.

inDAGO *inDAGO*

Description

A Shiny app for dual and bulk RNA-sequencing analysis.

Usage

```
inDAGO()
```

Details

This function allows to launch inDAGO Shiny interface.

Value

No return value, called for side effects

IndexingBulk *Bulk indexing*

Description

Bulk indexing

Usage

```
IndexingBulk(basename, reference, gappedIndex, indexSplit, memory, TH_subread)
```

Arguments

basename	output basename
reference	reference genome
gappedIndex	gapped structure
indexSplit	split structure
memory	handling memory
TH_subread	threshold memory usage

IndexingBulkServerLogic *Indexing bulk server logic*

Description

Indexing bulk server logic

Usage

```
IndexingBulkServerLogic(id)
```

Arguments

id	Shiny module identifier

IndexingBulkUserInterface *Indexing bulk ui*

Description

Indexing bulk ui

Usage

```
IndexingBulkUserInterface(id)
```

Arguments

id	Shiny module identifier

IndexingComb *Combined indexing*

Description

Combined indexing

Usage

```
IndexingComb(
  basename,
  reference,
  gappedIndex,
  indexSplit,
  memory,
  TH_subread,
  gen1,
  gen2,
  outfolder,
  tempfolder = file.path(fs::path_temp(), "TempDirSum_3738"),
  tag1,
  tag2
)
```

Arguments

basename	output basename
reference	reference genome
gappedIndex	gapped structure
indexSplit	split structure
memory	handling memory
TH_subread	threshold memory usage
gen1	first reference genome
gen2	second reference genome
outfolder	output folder
tempfolder	temporary folder
tag1	first genome label
tag2	second genome label

IndexingCombinedServerLogic *Indexing combined server logic*

Description

Indexing combined server logic

Usage

```
IndexingCombinedServerLogic(id)
```

Arguments

id	Shiny module identifier

IndexingCombinedUserInterface *Indexing combined ui*

Description

Indexing combined ui

Usage

```
IndexingCombinedUserInterface(id)
```

Arguments

id	Shiny module identifier

IndexingSequentialParallel *Indexing sequential parallel*

Description

Indexing sequential parallel

Usage

```
IndexingSequentialParallel(
  basename,
  reference,
  gappedIndex,
  indexSplit,
  memory,
  TH_subread
)
```

Arguments

basename	output basename
reference	reference genome
gappedIndex	gapped structure
indexSplit	split structure
memory	handling memory
TH_subread	threshold memory usage

IndexingSequentialProgressive *Indexing sequential progressive*

Description

Indexing sequential progressive

Usage

```
IndexingSequentialProgressive(
  outfolder1,
  outfolder2,
  refgen1,
  refgen2,
  gappedIndex,
```

```
  indexSplit,
  memory,
  TH_subread
)
```

Arguments

outfolder1	first output folder
outfolder2	second output folder
refgen1	first reference genome
refgen2	second reference genome
gappedIndex	gapped structure
indexSplit	split structure
memory	handling memory
TH_subread	threshold memory usage

IndexingSequentialServerLogic *Indexing sequential server logic*

Description

Indexing sequential server logic

Usage

```
IndexingSequentialServerLogic(id)
```

Arguments

id	Shiny module identifier

IndexingSequentialUserInterface *Indexing sequential ui*

Description

Indexing sequential ui

Usage

```
IndexingSequentialUserInterface(id)
```

Arguments

id	Shiny module identifier

`mappingBulkServerLogic` *Mapping bulk server logic*

Description

Mapping bulk server logic

Usage

```
mappingBulkServerLogic(id)
```

Arguments

`id`	Shiny module identifier

`mappingBulkUserInterface` *Mapping bulk ui*

Description

Mapping bulk ui

Usage

```
mappingBulkUserInterface(id)
```

Arguments

`id`	Shiny module identifier

`mappingCombinedServerLogic` *Mapping combined server logic*

Description

Mapping combined server logic

Usage

```
mappingCombinedServerLogic(id)
```

Arguments

`id`	Shiny module identifier

mappingCombinedUserInterface *Mapping combined ui*

Description

Mapping combined ui

Usage

```
mappingCombinedUserInterface(id)
```

Arguments

id	Shiny module identifier

mappingSequentialServerLogic *Mapping sequential server logic*

Description

Mapping sequential server logic

Usage

```
mappingSequentialServerLogic(id)
```

Arguments

id	Shiny module identifier

mappingSequentialUserInterface *Mapping sequential ui*

Description

Mapping sequential ui

Usage

```
mappingSequentialUserInterface(id)
```

Arguments

id	Shiny module identifier

mdsinfo *mdsinfo*

Description

Compute MDS coordinates for expression data using limma's plotMDS.

Usage

```
mdsinfo(matrix, top, gene.selection)
```

Arguments

matrix	A DGEList object.
top	Integer. Number of top most variable genes to include in MDS.
gene.selection	Method for gene selection: one of "pairwise", "common", or "logFC".

Details

This function performs multidimensional scaling (MDS) on a DGEList or log-expression matrix using limma's "plotMDS() " function. It returns the MDS object containing coordinates and eigenvalues without generating a plot.

Value

A list object from "plotMDS() " containing MDS coordinates and eigenvalues.

mdsPlot *mdsPlot*

Description

Generate a multidimensional scaling (MDS) plot based on expression data.

Usage

```
mdsPlot(
  x,
  Sample,
  Group,
  title,
  palette,
  maxOverlaps,
  sizeLabel,
  top,
  gene.selection
)
```

Arguments

x	DGEList object from edgeR.
Sample	A character vector of sample labels (one per column in "x ").
Group	A factor or character vector specifying the group/class of each sample.
title	Plot title as a character string.
palette	Name of a palette from the "paletteer " package for coloring groups.
maxOverlaps	Maximum number of overlapping labels allowed by "geom_text_repel ".
sizeLabel	Numeric value for label font size.
top	Integer. Number of top most variable genes to include in MDS.
gene.selection	Method for gene selection: one of "pairwise", "common", or "logFC".

Details

This function performs MDS analysis using limma's "plotMDS() " and visualizes the sample relationships in two dimensions using "ggplot2 " and "ggrepel ".

Value

A "ggplot " object representing the MDS plot.

---

mdsPlottly *mdsPlottly*

Description

Generate an interactive MDS plot using Plotly based on expression data.

Usage

```
mdsPlottly(x, Sample, Group, title, palette, top, gene.selection)
```

Arguments

x	A DGEList object from edgeR.
Sample	Character vector. Sample names corresponding to columns of "x ".
Group	Factor or character vector. Group or condition for each sample.
title	Character. Title for the plot.
palette	Character. Name of a discrete palette from the "paletteer " package.
top	Integer. Number of top most variable genes (by logFC) to include in MDS.
gene.selection	Character. Gene selection method: one of ""pairwise" ", ""common" ", or ""logFC" ".

Details

This function computes multidimensional scaling (MDS) coordinates with limma's "plotMDS() " and then renders an interactive scatterplot via "plotly::ggplotly() ".

1. Compute MDS on the input data with "limma::plotMDS() ".
2. Extract eigenvalues and first two dimensions for variance annotation.
3. Build a ggplot2 scatterplot with axis labels showing percent variance explained.
4. Convert the ggplot to an interactive Plotly graph.

Value

A Plotly object ("plotly::ggplotly ") representing the interactive MDS scatterplot.

`pcainfo` *pcainfo*

Description

Perform Principal Component Analysis (PCA) on log-expression data.

Usage

```
pcainfo(logcounts, center, scale)
```

Arguments

`logcounts`	Numeric matrix. Log-CPM values (genes × samples), e.g., from edgeR::cpm..
`center`	Logical. If TRUE, center variables by subtracting the mean (default: TRUE).
`scale`	Logical. If TRUE, scale variables to unit variance (default: FALSE).

Details

This function transposes a log-count matrix (samples as columns, genes as rows) and runs PCA using "stats::prcomp() ", with options to center and scale variables.

Value

An object of class "prcomp " containing the PCA results, including loadings, scores, and explained variance.

pcaPlot *pcaPlot*

Description

Create a PCA scatter plot from log-expression data with sample labels.

Usage

```
pcaPlot(
  logcounts,
  Sample,
  Group,
  title,
  palette,
  maxOverlaps,
  sizeLabel,
  center,
  scale
)
```

Arguments

logcounts	Numeric matrix of log-CPM values (genes × samples), e.g., from edgeR::cpm.
Sample	Character vector of sample names corresponding to the columns of "logcounts".
Group	Factor or character vector denoting group/condition for each sample.
title	Character. Title for the PCA plot.
palette	Character. Name of a discrete color palette from the "paletteer" package.
maxOverlaps	Integer. Maximum number of overlapping labels allowed by "ggrepel".
sizeLabel	Numeric. Font size for sample labels.
center	Logical. If TRUE, center variables before PCA.
scale	Logical. If TRUE, scale variables to unit variance before PCA.

Details

This function performs Principal Component Analysis (PCA) on a log-count matrix and visualizes the first two principal components using ggplot2 and ggrepel. Each point represents a sample, colored by group, with hover labels.

1. Transposes the "logcounts" matrix so samples are rows.
2. Runs PCA via "stats::prcomp()" with centering and scaling options.
3. Calculates percent variance explained by PC1 and PC2.
4. Builds a scatter plot with black-bordered points and non-overlapping labels.

Value

A "ggplot" object displaying the PCA scatter plot of PC1 vs PC2.

`pcaPlottly` *pcaPlottly*

Description

Create an interactive PCA scatter plot using Plotly from log-expression data.

Usage

```
pcaPlottly(logcounts, Sample, Group, title, palette, center, scale)
```

Arguments

`logcounts`	Numeric matrix of log-CPM values (genes × samples), e.g., from edgeR::cpm.
`Sample`	Character vector of sample names corresponding to the columns of "logcounts ".
`Group`	Factor or character vector denoting group/condition for each sample.
`title`	Character. Title for the PCA plot.
`palette`	Character. Name of a discrete color palette from the "paletteer" package.
`center`	Logical. If TRUE, center variables (genes) before PCA.
`scale`	Logical. If TRUE, scale variables to unit variance before PCA.

Details

This function performs Principal Component Analysis (PCA) on a log-count matrix and generates an interactive plot of the first two principal components via "plotly::ggplotly()".

1. Transposes the "logcounts " matrix so samples are rows.
2. Runs PCA with "stats::prcomp() ", using centering and scaling as specified.
3. Computes percent variance explained by PC1 and PC2.
4. Builds a ggplot2 scatterplot and converts it to an interactive Plotly graph.

Value

A Plotly object ("plotly::ggplotly ") representing the interactive PCA scatterplot.

QualityCheckAnalysis *QUALITY CONTROL ANALYSIS*

Description

QUALITY CONTROL ANALYSIS

Usage

```
QualityCheckAnalysis(
  directoryInput,
  inputFormat,
  Nodes,
  ReadsNumber,
  directoryOutput,
  tempFolder
)
```

Arguments

directoryInput	sample directory
inputFormat	raw read format
Nodes	cores
ReadsNumber	chunk
directoryOutput	output folder
tempFolder	temporary folder

qualityControlServerLogic *Quality control server logic*

Description

Quality control server logic

Usage

```
qualityControlServerLogic(id)
```

Arguments

id	Shiny module identifier

qualityControlUserInterface *Quality control ui*

Description

Quality control ui

Usage

```
qualityControlUserInterface(id)
```

Arguments

id	Shiny module identifier

Saturation *Saturation*

Description

Generate a saturation curve plot showing gene detection versus sequencing depth.

Usage

```
Saturation(matrix, method, max_reads, palette)
```

Arguments

matrix	Numeric matrix or object coercible to matrix (genes × samples), e.g., log-counts or raw counts. Genes are rows; samples are columns.
method	Character. Estimation method: "division" or "sampling".
max_reads	Numeric. Maximum number of reads to include in the rarefaction (default: Inf).
palette	Character. Name of a discrete color palette from the "paletteer " package for curve colors.

Details

This function estimates how many genes are detected at increasing read depths using a rarefaction-based approach ("estimate_saturation() from RNAseQC package https://github.com/BenaroyaResearch/RNAseQC.git"), and plots the saturation curves for each sample. It supports two estimation methods: “division” for a fast analytic approximation and “sampling” for more realistic approach.

1. Internally, "extract_counts() " (from countSubsetNorm) extracts a counts matrix from various input classes (matrix, DGEList, EList, ExpressionSet).

2. "estimate_saturation() " (from RNAseQC package https://github.com/BenaroyaResearch/RNAseQC.git) rarefies each library at multiple depths:

- “division” divides counts by scale factors;
- “sampling” performs repeated random sampling to simulate read down sampling.

1. The resulting data frame contains one row per sample per depth, with the number of detected genes ("sat ") and, for sampling, its variance ("sat.var ").
2. The function then plots gene saturation curves ("sat" vs. "depth") colored by sample.

Extract counts matrix from different types of expression objects

Estimate saturation of genes based on rarefaction of reads

Value

A "ggplot " object showing saturation (genes detected) versus sequencing depth for each sample.

`SequenceLengthDistributionPlot` *SequenceLengthDistributionPlot*

Description

SequenceLengthDistributionPlot

Usage

```
SequenceLengthDistributionPlot(input_data)
```

Arguments

`input_data`	result tables folder

`SequenceLengthDistributionPlotly` *interactive SequenceLengthDistributionPlot*

Description

interactive SequenceLengthDistributionPlot

Usage

```
SequenceLengthDistributionPlotly(input_data)
```

Arguments

`input_data`	result tables folder

SequentialAlignment *Sequential alignment function*

Description

Sequential alignment function

Usage

```
SequentialAlignment(
  lalista,
  nodes,
  readsPath,
  GenomeFirstIndex,
  GenomeSecondIndex,
  outBam1,
  outBam2,
  threads,
  outFormat,
  phredScore,
  maxExtractedSubreads,
  consensusVote,
  mismatchMax,
  uniqueOnly,
  maxMultiMapped,
  indelLength,
  fragmentMinLength,
  fragmentMaxLength,
  matesOrientation,
  readOrderConserved,
  coordinatesSorting,
  allJunctions,
  tempfolder,
  readsAlignedBlock
)
```

Arguments

lalista	list of samples
nodes	logic cores
readsPath	sample folders
GenomeFirstIndex	first genome index
GenomeSecondIndex	second genome index
outBam1	first output folder

outBam2	second output folder
threads	processes
outFormat	BAM or SAM
phredScore	quality score
maxExtractedSubreads	number of subreads
consensusVote	consensus
mismatchMax	mismatch
uniqueOnly	no multimapping
maxMultiMapped	multimapping
indelLength	indel
fragmentMinLength	fragment minumum length
fragmentMaxLength	fragment maximum length
matesOrientation	mate orientation
readOrderConserved	read order
coordinatesSorting	sorting
allJunctions	junctions
tempfolder	temporary folder
readsAlignedBlock	chunks

`Summarization` *Summarization*

Description

Summarizes read counts from multiple BAM/SAM files in parallel using feature annotations.

Usage

```
Summarization(
  NodesSum,
  Xsum,
  UploadPathSum,
  DownloadPathSum,
  annot.ext,
  isGTFAnnotationFile,
  GTF.featureType,
```

```
  GTF.attrType,
  useMetaFeatures,
  allowMultiOverlap,
  minOverlap,
  fracOverlap,
  fracOverlapFeature,
  largestOverlap,
  countMultiMappingReads,
  fraction,
  minMQS,
  primaryOnly,
  ignoreDup,
  strandSpecific,
  requireBothEndsMapped,
  checkFragLength,
  minFragLength,
  maxFragLength,
  countChimericFragments,
  autosort,
  nthreads,
  tmpDir,
  verbose
)
```

Arguments

`NodesSum`	Integer. Number of parallel R nodes (e.g., CPU cores) to spawn.
`Xsum`	Character vector. Filenames of BAM or SAM files to process.
`UploadPathSum`	Character. Directory containing the raw input files.
`DownloadPathSum`	Character. Directory into which all output files will be written.
`annot.ext`	Character. Path to an external annotation file (e.g., GTF/GFF).
`isGTFAnnotationFile`	Logical. Should `annot.ext` be treated as a GTF file?
`GTF.featureType`	Character. Feature type (e.g., "exon").
`GTF.attrType`	Character. GTF attribute (e.g., "gene_id").
`useMetaFeatures`	Logical. Collapse sub-features into meta-features before counting.
`allowMultiOverlap`	Logical. Allow reads overlapping multiple features to be counted.
`minOverlap`	Integer. Minimum number of overlapping bases to assign a read.
`fracOverlap`	Numeric. Minimum fraction of read that must overlap a feature.
`fracOverlapFeature`	Numeric. Minimum fraction of feature that must be covered by a read.

largestOverlap Logical. When overlapping multiple features, assign based on largest overlap.

countMultiMappingReads
Logical. Count reads that map to multiple locations.

fraction Logical. Distribute counts fractionally for multi-mapping reads.

minMQS Integer. Minimum mapping quality score for reads to be counted.

primaryOnly Logical. Count only the primary alignments of multi-mapping reads.

ignoreDup Logical. Exclude PCR duplicates from counting.

strandSpecific Integer. Strand-specific counting mode (0 = unstranded, 1 = stranded, 2 = reversely stranded).

requireBothEndsMapped
Logical. In paired-end mode, require both mates to map.

checkFragLength
Logical. Enforce fragment length checks on paired-end reads.

minFragLength Numeric. Minimum fragment length to keep.

maxFragLength Numeric. Maximum fragment length to keep.

countChimericFragments
Logical. Count discordant or chimeric read pairs.

autosort Logical. Automatically sort input files if not already sorted.

nthreads Integer. Number of threads per featureCounts call.

tmpDir Character. Directory for temporary files (e.g., large intermediate files).

verbose Logical. Print verbose messages during execution.

Details

This function run `Rsubread::featureCounts()` on each input file, capturing count statistics, annotation data, and per-sample summary logs. Results are written to the specified output directory.

1. A socket cluster of `NodesSum` workers is created.
2. Each worker invokes `featureCounts()` on one sample, using the annotation and counting parameters.
3. Outputs per sample:
   - A text summary (`*_summary.txt`) capturing the console output.
   - A CSV of count statistics (`*_stat.csv`).
   - A CSV of feature annotations (`*_annotation.csv`).
   - A tab-delimited count matrix saved under `Counts/<sample>.tab`.
4. The cluster is terminated once all samples complete.

Value

Writes files to `DownloadPathSum`.

`SummarizationServerLogic` *Server function for Summarization module in Shiny application*

Description

Server function for Summarization module in Shiny application

Usage

```
SummarizationServerLogic(id)
```

Arguments

`id`	Shiny module identifier

`SummarizationUserInterface` *UI function for Summarization module in Shiny application*

Description

UI function for Summarization module in Shiny application

Usage

```
SummarizationUserInterface(id)
```

Arguments

`id`	Shiny module identifier

UpsetjsPlot *UpsetjsPlot*

Description

Create an interactive UpSet plot of overlapping DEGs using "UpsetJS".

Usage

```
UpsetjsPlot(
  WD_samples,
  Th_logFC,
  Th_Pvalue,
  collapseName,
  nintersects,
  st_significance
)
```

Arguments

WD_samples	Character. Directory containing DEG result CSV files.
Th_logFC	Numeric. Absolute log2 fold-change threshold to include a gene.
Th_Pvalue	Numeric. P-value threshold for significance (0 < Th_Pvalue <= 1).
collapseName	Logical. If TRUE, strip method/model prefixes from file names when labeling sets.
nintersects	Integer. Maximum number of intersections to display.
st_significance	Character. Which p-value to use: "adjustPvalue" (FDR or FWER) or "PValue".

Details

This function reads DEG CSV files from a directory, filters genes by log-FC and p-value thresholds (adjusted or raw), optionally simplifies file names, and visualizes the intersections of gene sets using the "UpsetJS" package.

1. Lists all CSV files in "WD_samples" and reads each into a data frame.
2. Checks for duplicate IDs and selects "ID", "logFC", and either "adjustPvalue" or "PValue".
3. Filters each set by "|logFC| >= Th_logFC" and p-value < "Th_Pvalue".
4. Renames each gene-ID list to the (optionally collapsed) file name.
5. Feeds the list of gene sets into "upsetjs::upsetjs()"

Value

An interactive "UpsetJS" object.

UpSetPlot *UpSetPlot*

Description

Generate an UpSet plot of overlapping DEGs across multiple contrasts.

Usage

```
UpSetPlot(
  WD_samples,
  Th_logFC,
  Th_Pvalue,
  collapseName,
  nintersects,
  st_significance,
  scale
)
```

Arguments

WD_samples	Character. Directory containing DEG result CSV files.
Th_logFC	Numeric. Absolute log2 fold-change threshold to include a gene.
Th_Pvalue	Numeric. P-value threshold for significance (0 < Th_Pvalue <= 1).
collapseName	Logical. If TRUE, strip method/model prefixes from file names when labeling sets.
nintersects	Integer. Maximum number of intersections to display.
st_significance	Character. Which p-value to use: "adjustPvalue" (FDR or FWER) or "PValue".
scale	Numeric. Text scaling factor for plot labels and annotations.

Details

This function reads DEG CSV files from a directory, filters genes by log-FC and p-value thresholds (adjusted or raw), optionally simplifies file names, and visualizes the intersections of gene sets using an UpSet plot.

1. Validates thresholds (Th_logFC >= 0, 0 < Th_Pvalue <= 1).
2. Lists all CSV files in WD_samples and reads each into a data frame.
3. Checks for duplicate IDs and standardizes to columns ID, logFC, and adjustPvalue or PValue.
4. Filters each set of results by |logFC| >= Th_logFC and p-value < Th_Pvalue.
5. Renames each gene-ID column to the (optionally collapsed) file name.
6. Converts the list of filtered ID sets to an UpSetR input and calls UpSetR::upset().

Value

An UpSet plot.

`volcanoPlot` *volcanoPlot*

Description

Create a volcano plot of differential expression results.

Usage

```
volcanoPlot(
  x,
  palettePoint,
  maxOverlaps,
  sizeLabel,
  Th_logFC,
  Th_Pvalue,
  subsetGenes,
  st_significance
)
```

Arguments

`x`	Character. File path to a CSV containing DEG results, with at least columns "ID", "logFC", and one of "PValue", "FDR", or "FWER".
`palettePoint`	Character. Name of a discrete palette from the "paletteer" package, supplying colors for "UP", "DOWN", and "NO".
`maxOverlaps`	Integer. Maximum allowed label overlaps passed to "ggrepel::geom_text_repel()".
`sizeLabel`	Numeric. Font size for gene labels in the plot.
`Th_logFC`	Numeric. Absolute log2 fold-change threshold to call a gene "UP" or "DOWN".
`Th_Pvalue`	Numeric. P-value threshold to call significance (uses "FDR"/"FWER" if "st_significance = "adjustPvalue"", otherwise raw "PValue").
`subsetGenes`	Integer or "Inf". If numeric, only the top "subsetGenes" genes by p-value are shown and labeled.
`st_significance`	Character. Which p-value column to use: "adjustPvalue" (FDR or FWER) or "PValue".

Details

This function reads a CSV of DEGs, classifies genes as up/down/no change based on log-fold change and p-value thresholds, and plots –log10(p-value) versus log-FC using ggplot2.

1. Reads the input CSV and checks for duplicate IDs.
2. Standardizes columns to "ID", "logFC", and "adjustPvalue" or "PValue".
3. Optionally subsets to the top N genes by p-value.
4. Classifies each gene as "UP", "DOWN", or "NO" based on thresholds.
5. Plots points with manual fill, size, and alpha scales, adds threshold lines, and repels labels using "ggrepel".

Value

A "ggplot" object displaying the volcano plot.

`volcanoPlottly` *volcanoPlottly*

Description

Create an interactive volcano plot of differential expression results using "Plotly".

Usage

```
volcanoPlottly(
  x,
  palettePoint,
  Th_logFC,
  Th_Pvalue,
  subsetGenes,
  st_significance
)
```

Arguments

`x`	Character. File path to a CSV containing DEG results, with at least columns "ID", "logFC", and one of "PValue", "FDR", or "FWER".
`palettePoint`	Character. Name of a discrete palette from the "paletteer" package, supplying colors for "UP", "DOWN", and "NO".
`Th_logFC`	Numeric. Absolute log2 fold-change threshold to call a gene "UP" or "DOWN".
`Th_Pvalue`	Numeric. P-value threshold to call significance (uses "FDR"/"FWER" if "st_significance = "adjustPvalue"", otherwise raw "PValue").
`subsetGenes`	Integer or "Inf". If numeric, only the top "subsetGenes" genes by p-value are included in the plot.
`st_significance`	Character. Which p-value column to use: "adjustPvalue" (FDR or FWER) or "PValue".

Details

This function reads a CSV of DEGs, classifies genes as up/down/no change based on log-fold change and p-value thresholds, and renders an interactive volcano plot via "plotly::ggplotly()".

1. Reads the input CSV and checks for duplicate IDs.
2. Standardizes columns to "ID", "logFC", and "adjustPvalue" or "PValue".
3. Optionally subsets to the top N genes by p-value.
4. Classifies each gene as "UP", "DOWN", or "NO" based on thresholds.
5. Plots points with manual fill, size, and alpha scales, adds threshold lines, and converts to an interactive Plotly graph.

Value

A Plotly object ("plotly::ggplotly") representing the interactive volcano plot.

`WorkflowServerLogic` *Server function for workflow module in Shiny application*

Description

Server function for workflow module in Shiny application

Usage

```
WorkflowServerLogic(id)
```

Arguments

`id`	Shiny module identifier

`WorkflowUserInterface` *UI function for workflow module in Shiny application*

Description

UI function for workflow module in Shiny application

Usage

```
WorkflowUserInterface(id)
```

Arguments

`id`	Shiny module identifier

Index