Package 'doubletrouble'

January 10, 2025

Title Identification and classification of duplicated genes

Version 1.7.0

Date 2024-03-19

Description doubletrouble aims to identify duplicated genes from whole-genome protein sequences and classify them based on their modes of duplication. The duplication modes are i. segmental duplication (SD); ii. tandem duplication (TD); iii. proximal duplication (PD); iv. transposed duplication (TRD) and; v. dispersed duplication (DD). Transposon-derived duplicates (TRD) can be further subdivided into rTRD (retrotransposon-derived duplication) and dTRD (DNA transposon-derived duplication). If users want a simpler classification scheme, duplicates can also be classified into SD- and SSD-derived (small-scale duplication) gene pairs. Besides classifying gene pairs, users can also classify genes, so that each gene is assigned a unique mode of duplication. Users can also calculate substitution rates per substitution site (i.e., Ka and Ks) from duplicate pairs, find peaks in Ks distributions with Gaussian Mixture Models (GMMs), and classify gene pairs into age groups based on Ks peaks.

License GPL-3

URL https://github.com/almeidasilvaf/doubletrouble

BugReports https://support.bioconductor.org/t/doubletrouble

biocViews Software, WholeGenome, ComparativeGenomics, FunctionalGenomics, Phylogenetics, Network, Classification

Encoding UTF-8

LazyData false

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports syntenet, GenomicRanges, Biostrings, mclust, MSA2dist (>= 1.1.5), ggplot2, rlang, stats, utils, AnnotationDbi, GenomicFeatures

Depends R (>= 4.2.0)

Suggests txdbmaker, testthat (>= 3.0.0), knitr, feature, patchwork, BiocStyle, rmarkdown, covr, sessioninfo

Config/testthat/edition 3

VignetteBuilder knitr

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

git_branch devel

git_last_commit 4db21ce

git_last_commit_date 2024-10-29

Repository Bioconductor 3.21

Date/Publication 2025-01-10

Author Fabrício Almeida-Silva [aut, cre] (ORCID: <https://orcid.org/0000-0002-5314-2964>), Yves Van de Peer [aut] (ORCID: <https://orcid.org/0000-0003-4327-3730>)

Maintainer Fabrício Almeida-Silva <fabricio_almeidasilva@hotmail.com>

Contents

cds_scerevisiae . . . 3
classify_genes . . . 3
classify_gene_pairs . . . 4
diamond_inter . . . 6
diamond_intra . . . 7
duplicates2counts . . . 7
find_ks_peaks . . . 8
fungi_kaks . . . 9
get_anchors_list . . . 10
get_intron_counts . . . 11
get_segmental . . . 12
get_tandem_proximal . . . 13
get_transposed . . . 14
get_transposed_classes . . . 15
gmax_ks . . . 17
pairs2kaks . . . 17
plot_duplicate_freqs . . . 18
plot_ks_distro . . . 19
plot_ks_peaks . . . 20
plot_rates_by_species . . . 21
split_pairs_by_peak . . . 22
yeast_annot . . . 23
yeast_seq . . . 23

Index **25**

cds_scerevisiae *Coding sequences (CDS) of S. cerevisiae*

Description

Data were obtained from Ensembl Fungi, and only CDS of primary transcripts were included.

Usage

```
data(cds_scerevisiae)
```

Format

A DNAStringSet object with CDS of S. cerevisiae.

Examples

```
data(cds_scerevisiae)
```

classify_genes *Classify genes into unique modes of duplication*

Description

Classify genes into unique modes of duplication

Usage

```
classify_genes(gene_pairs_list = NULL)
```

Arguments

`gene_pairs_list`
: List of classified gene pairs as returned by `classify_gene_pairs()`.

Details

If a gene is present in pairs with different duplication modes, the gene is classified into a unique mode of duplication following the order of priority indicated in the levels of the factor **type**.

For scheme "binary", the order is SD > SSD. For scheme "standard", the order is SD > TD > PD > DD. For scheme "extended", the order is SD > TD > PD > TRD > DD. For scheme "full", the order is SD > TD > PD > rTRD > dTRD > DD.

Value

A list of 2-column data frames with variables **gene** and **type** representing gene ID and duplication type, respectively.

Examples

```
data(fungi_kaks)
scerevisiae_kaks <- fungi_kaks$saccharomyces_cerevisiae

cols <- c("dup1", "dup2", "type")
gene_pairs_list <- list(Scerevisiae = scerevisiae_kaks[, cols])

class_genes <- classify_genes(gene_pairs_list)
```

`classify_gene_pairs` *Classify duplicate gene pairs based on their modes of duplication*

Description

Classify duplicate gene pairs based on their modes of duplication

Usage

```
classify_gene_pairs(
  annotation = NULL,
  blast_list = NULL,
  scheme = "standard",
  blast_inter = NULL,
  intron_counts,
  evalue = 1e-10,
  anchors = 5,
  max_gaps = 25,
  proximal_max = 10,
  collinearity_dir = NULL,
  outgroup_coverage = 70
)
```

Arguments

`annotation`	A processed GRangesList or CompressedGRangesList object as returned by `syntenet::process_input()`.
`blast_list`	A list of data frames containing BLAST tabular output for intraspecies comparisons. Each list element corresponds to the BLAST output for a given species, and names of list elements must match the names of list elements in **annotation**. BLASTp, DIAMOND or simular programs must be run on processed sequence data as returned by `process_input()`.
`scheme`	Character indicating which classification scheme to use. One of "binary", "standard", "extended", or "full". See details below for information on what each scheme means. Default: "standard".

`blast_inter` (Only valid if `scheme == "extended"` or `"full"`). A list of data frames containing BLAST tabular output for the comparison between target species and outgroups. Names of list elements must match the names of list elements in `annotation`. BLASTp, DIAMOND or simular programs must be run on processed sequence data as returned by `process_input()`.

`intron_counts` (Only valid if `scheme == "full"`). A list of 2-column data frames with the number of introns per gene as returned by `get_intron_counts()`. Names of list elements must match names of **annotation**.

`evalue` Numeric scalar indicating the E-value threshold. Default: 1e-10.

`anchors` Numeric indicating the minimum required number of genes to call a syntenic block, as in `syntenet::infer_syntenet`. Default: 5.

`max_gaps` Numeric indicating the number of upstream and downstream genes to search for anchors, as in `syntenet::infer_syntenet`. Default: 25.

`proximal_max` Numeric scalar with the maximum distance (in number of genes) between two genes to consider them as proximal duplicates. Default: 10.

`collinearity_dir` Character indicating the path to the directory where .collinearity files will be stored. If NULL, files will be stored in a subdirectory of `tempdir()`. Default: NULL.

`outgroup_coverage` Numeric indicating the minimum percentage of outgroup species to use to consider genes as transposed duplicates. Only valid if multiple outgroup species are present (see details below). Values should range from 0 to 100. Default: 70.

Details

The classification schemes increase in complexity (number of classes) in the order 'binary', 'standard', 'extended', and 'full'.

For classification scheme "binary", duplicates are classified into one of 'SD' (segmental duplications) or 'SSD' (small-scale duplications).

For classification scheme "standard" (default), duplicates are classified into 'SD' (segmental duplication), 'TD' (tandem duplication), 'PD' (proximal duplication), and 'DD' (dispersed duplication).

For classification scheme "extended", duplicates are classified into 'SD' (segmental duplication), 'TD' (tandem duplication), 'PD' (proximal duplication), 'TRD' (transposon-derived duplication), and 'DD' (dispersed duplication).

Finally, for classification scheme "full", duplicates are classified into 'SD' (segmental duplication), 'TD' (tandem duplication), 'PD' (proximal duplication), 'rTRD' (retrotransposon-derived duplication), 'dTRD' (DNA transposon-derived duplication), and 'DD' (dispersed duplication).

Value

A list of 3-column data frames of duplicated gene pairs (columns 1 and 2), and their modes of duplication (column 3).

Examples

```
# Load example data
data(diamond_intra)
data(diamond_inter)
data(yeast_annot)
data(yeast_seq)

# Get processed annotation data
annotation <- syntenet::process_input(yeast_seq, yeast_annot)$annotation

# Get list of intron counts
library(txdbmaker)
txdb_list <- lapply(yeast_annot, txdbmaker::makeTxDbFromGRanges)
intron_counts <- lapply(txdb_list, get_intron_counts)

# Classify duplicates - full scheme
dup_class <- classify_gene_pairs(
    annotation = annotation,
    blast_list = diamond_intra,
    scheme = "full",
    blast_inter = diamond_inter,
    intron_counts = intron_counts
)

# Check number of gene pairs per class
table(dup_class$Scerevisiae$type)
```

---

diamond_inter *Interspecies DIAMOND output for yeast species*

Description

This list contains a similarity search of S. cerevisiae against C. glabrata, and it was obtained with `run_diamond()`.

Usage

```
data(diamond_inter)
```

Format

A list of data frames (length 1) containing the output of a DIAMOND search of S. cerevisiae against C. glabrata (outgroup).

Examples

```
data(diamond_inter)
```

diamond_intra *Intraspecies DIAMOND output for S. cerevisiae*

Description

List obtained with `run_diamond()`.

Usage

```
data(diamond_intra)
```

Format

A list of data frames (length 1) containing the whole paranome of S. cerevisiae resulting from intragenome similarity searches.

Examples

```
data(diamond_intra)
```

duplicates2counts *Get a duplicate count matrix for each genome*

Description

Get a duplicate count matrix for each genome

Usage

```
duplicates2counts(duplicate_list, shape = "long")
```

Arguments

`duplicate_list`	A list of data frames with the duplicated genes or gene pairs and their modes of duplication as returned by `classify_gene_pairs()` or `classify_genes()`.
`shape`	Character specifying the shape of the output data frame. One of "long" (data frame in the long shape, in the tidyverse sense), or "wide" (data frame in the wide shape, in the tidyverse sense). Default: "long".

Value

If **shape = "wide"**, a count matrix containing the frequency of duplicated genes (or gene pairs) by mode for each species, with species in rows and duplication modes in columns. If **shape = "long"**, a data frame in long format with the following variables:

type Factor, type of duplication.

n Numeric, number of duplicates.

species Character, species name

Examples

```
data(fungi_kaks)

# Get unique duplicates
duplicate_list <- classify_genes(fungi_kaks)

# Get count table
counts <- duplicates2counts(duplicate_list)
```

`find_ks_peaks` *Find peaks in a Ks distribution with Gaussian Mixture Models*

Description

Find peaks in a Ks distribution with Gaussian Mixture Models

Usage

```
find_ks_peaks(ks, npeaks = 2, min_ks = 0.01, max_ks = 4, verbose = FALSE)
```

Arguments

`ks`	A numeric vector of Ks values.
`npeaks`	Numeric scalar indicating the number of peaks in the Ks distribution. If you don't know how many peaks there are, you can include a range of values, and the number of peaks that produces the lowest BIC (Bayesian Information Criterion) will be selected as the optimal. Default: 2.
`min_ks`	Numeric scalar with the minimum Ks value. Removing very small Ks values is generally used to avoid the incorporation of allelic and/or splice variants and to prevent the fitting of a component to infinity. Default: 0.01.
`max_ks`	Numeric scalar indicating the maximum Ks value. Removing very large Ks values is usually performed to account for Ks saturation. Default: 4.
`verbose`	Logical indicating if messages should be printed on screen. Default: FALSE.

Value

A list with the following elements:

mean Numeric with the estimated means.

sd Numeric with the estimated standard deviations.

lambda Numeric with the estimated mixture weights.

ks Numeric vector of filtered Ks distribution based on arguments passed to min_ks and max_ks.

Examples

```
data(fungi_kaks)
scerevisiae_kaks <- fungi_kaks$saccharomyces_cerevisiae
ks <- scerevisiae_kaks$Ks

# Find 2 peaks in Ks distribution
peaks <- find_ks_peaks(ks, npeaks = 2)

# From 2 to 4 peaks, verbose = TRUE to show BIC values
peaks <- find_ks_peaks(ks, npeaks = c(2, 3, 4), verbose = TRUE)
```

---

fungi_kaks *Duplicate pairs and Ka, Ks, and Ka/Ks values for fungi species*

Description

This data set was obtained with `classify_gene_pairs()` followed by `pairs2kaks()`.

Usage

```
data(fungi_kaks)
```

Format

A list of data frame with elements named **saccharomyces_cerevisiae**, **candida_glabrata**, and **schizosaccharomyces_pombe**. Each data frame contains the following variables:

dup1 Character, duplicated gene 1.

dup2 Character, duplicated gene 2.

Ka Numeric, Ka values.

Ks Numeric, Ks values.

Ka_Ks Numeric, Ka/Ks values.

type Character, mode of duplication

Examples

```
data(fungi_kaks)
```

`get_anchors_list` *Get a list of anchor pairs for each species*

Description

Get a list of anchor pairs for each species

Usage

```
get_anchors_list(
  blast_list = NULL,
  annotation = NULL,
  evalue = 1e-10,
  anchors = 5,
  max_gaps = 25,
  collinearity_dir = NULL
)
```

Arguments

`blast_list`	A list of data frames containing BLAST tabular output for intraspecies comparisons. Each list element corresponds to the BLAST output for a given species, and names of list elements must match the names of list elements in `annotation`. BLASTp, DIAMOND or simular programs must be run on processed sequence data as returned by `process_input()`.
`annotation`	A processed GRangesList or CompressedGRangesList object as returned by `syntenet::process_input()`.
`evalue`	Numeric scalar indicating the E-value threshold. Default: 1e-10.
`anchors`	Numeric indicating the minimum required number of genes to call a syntenic block, as in `syntenet::infer_syntenet`. Default: 5.
`max_gaps`	Numeric indicating the number of upstream and downstream genes to search for anchors, as in `syntenet::infer_syntenet`. Default: 25.
`collinearity_dir`	Character indicating the path to the directory where .collinearity files will be stored. If NULL, files will be stored in a subdirectory of `tempdir()`. Default: NULL.

Value

A list of data frames representing intraspecies anchor pairs.

Examples

```
data(diamond_intra)
data(yeast_annot)
data(yeast_seq)
```

```
blast_list <- diamond_intra

# Get processed annotation for S. cerevisiae
annotation <- syntenet::process_input(yeast_seq, yeast_annot)$annotation

# Get list of intraspecies anchor pairs
anchorpairs <- get_anchors_list(blast_list, annotation)
```

`get_intron_counts` *Get a data frame of intron counts per gene*

Description

Get a data frame of intron counts per gene

Usage

```
get_intron_counts(txdb)
```

Arguments

`txdb`	A `TxDb` object with transcript annotations. See details below for examples on how to create `TxDb` objects from different kinds of input.

Details

The family of functions `makeTxDbFrom*` from the **txdbmaker** package can be used to create `TxDb` objects from a variety of input data types. You can create `TxDb` objects from e.g., `GRanges` objects (`makeTxDbFromGRanges()`), GFF files (`makeTxDbFromGFF()`), an Ensembl database (`makeTxDbFromEnsembl`), and a Biomart database (`makeTxDbFromBiomart`).

Value

A data frame with intron counts per gene, with variables:

gene Character with gene IDs.

introns Numeric with number of introns per gene.

Examples

```
data(yeast_annot)

# Create TxDb object from GRanges
library(txdbmaker)
txdb <- txdbmaker::makeTxDbFromGRanges(yeast_annot[[1]])

# Get intron counts
intron_counts <- get_intron_counts(txdb)
```

`get_segmental` *Classify gene pairs derived from segmental duplications*

Description

Classify gene pairs derived from segmental duplications

Usage

```
get_segmental(anchor_pairs = NULL, pairs = NULL)
```

Arguments

`anchor_pairs`	A 2-column data frame with anchor pairs in columns 1 and 2.
`pairs`	A 2-column data frame with all duplicate pairs. This is equivalent to the first 2 columns of the tabular output of BLAST-like programs.

Value

A 3-column data frame with the variables:

dup1 Character, duplicated gene 1

dup2 Character, duplicated gene 2

type Factor indicating duplication types, with levels "SD" (segmental duplication) or "DD" (dispersed duplication).

Examples

```
data(diamond_intra)
data(yeast_annot)
data(yeast_seq)
blast_list <- diamond_intra

# Get processed annotation for S. cerevisiae
annotation <- syntenet::process_input(yeast_seq, yeast_annot)$annotation[1]

# Get list of intraspecies anchor pairs
anchor_pairs <- get_anchors_list(blast_list, annotation)
anchor_pairs <- anchor_pairs[[1]][, c(1, 2)]

# Get duplicate pairs from DIAMOND output
duplicates <- diamond_intra[[1]][, c(1, 2)]
dups <- get_segmental(anchor_pairs, duplicates)
```

`get_tandem_proximal` *Classify gene pairs derived from tandem and proximal duplications*

Description

Classify gene pairs derived from tandem and proximal duplications

Usage

```
get_tandem_proximal(pairs = NULL, annotation_granges = NULL, proximal_max = 10)
```

Arguments

`pairs`	A 3-column data frame with columns **dup1**, **dup2**, and **type** indicating duplicated gene 1, duplicated gene 2, and the mode of duplication associated with the pair. This data frame is returned by `get_segmental()`.
`annotation_granges`	A processed GRanges object as in each element of the list returned by `syntenet::process_input()`.
`proximal_max`	Numeric scalar with the maximum distance (in number of genes) between two genes to consider them as proximal duplicates. Default: 10.

Value

A 3-column data frame with the variables:

dup1 Character, duplicated gene 1.

dup2 Character, duplicated gene 2.

type Factor of duplication types, with levels "SD" (segmental duplication), "TD" (tandem duplication), "PD" (proximal duplication), and "DD" (dispersed duplication).

Examples

```
data(yeast_annot)
data(yeast_seq)
data(fungi_kaks)
scerevisiae_kaks <- fungi_kaks$saccharomyces_cerevisiae

# Get processed annotation for S. cerevisiae
pdata <- annotation <- syntenet::process_input(yeast_seq, yeast_annot)
annot <- pdata$annotation[[1]]

# Get duplicated pairs
pairs <- scerevisiae_kaks[, c("dup1", "dup2", "type")]
pairs$dup1 <- paste0("Sce_", pairs$dup1)
pairs$dup2 <- paste0("Sce_", pairs$dup2)

# Get tandem and proximal duplicates
td_pd_pairs <- get_tandem_proximal(pairs, annot)
```

get_transposed *Classify gene pairs originating from transposon-derived duplications*

Description

Classify gene pairs originating from transposon-derived duplications

Usage

```
get_transposed(
  pairs,
  blast_inter,
  annotation,
  evalue = 1e-10,
  anchors = 5,
  max_gaps = 25,
  collinearity_dir = NULL,
  outgroup_coverage = 70
)
```

Arguments

pairs	A 3-column data frame with columns **dup1**, **dup2**, and **type** indicating duplicated gene 1, duplicated gene 2, and the mode of duplication associated with the pair. This data frame is returned by `get_tandem_proximal()`.
blast_inter	A list of data frames of length 1 containing BLAST tabular output for the comparison between the target species and an outgroup. Names of list elements must match the names of list elements in `annotation`. BLASTp, DIAMOND or simular programs must be run on processed sequence data as returned by `syntenet::process_input()`.
annotation	A processed GRangesList or CompressedGRangesList object as returned by `syntenet::process_input()`.
evalue	Numeric scalar indicating the E-value threshold. Default: 1e-10.
anchors	Numeric indicating the minimum required number of genes to call a syntenic block, as in `syntenet::infer_syntenet`. Default: 5.
max_gaps	Numeric indicating the number of upstream and downstream genes to search for anchors, as in `syntenet::infer_syntenet`. Default: 25.
collinearity_dir	Character indicating the path to the directory where .collinearity files will be stored. If NULL, files will be stored in a subdirectory of `tempdir()`. Default: NULL.
outgroup_coverage	Numeric indicating the minimum percentage of outgroup species to use to consider genes as transposed duplicates. Only valid if multiple outgroup species are present (see details below). Values should range from 0 to 100. Default: 70.

Details

If the list of interspecies DIAMOND tables contain comparisons of the same species to multiple outgroups (e.g., 'speciesA_speciesB', 'speciesA_speciesC'), this function will check if gene pairs are classified as transposed (i.e., only one gene is an ancestral locus) in each of the outgroup species, and then calculate the percentage of outgroup species in which each pair is considered 'transposed'. For instance, gene pair 1 is transposed based on 30\ on 100\ based on 0\ Parameter **outgroup_coverage** lets you choose a minimum percentage cut-off to classify pairs as transposed.

Value

A 3-column data frame with the following variables:

dup1 Character, duplicated gene 1.

dup2 Character, duplicated gene 2.

type Factor of duplication types, with levels "SD" (segmental duplication), "TD" (tandem duplication), "PD" (proximal duplication), "TRD" (transposon-derived duplication), and "DD" (dispersed duplication).

Examples

```
data(diamond_inter)
data(diamond_intra)
data(yeast_seq)
data(yeast_annot)
data(fungi_kaks)
scerevisiae_kaks <- fungi_kaks$saccharomyces_cerevisiae

# Get processed annotation
pdata <- syntenet::process_input(yeast_seq, yeast_annot)
annotation <- pdata$annotation

# Get duplicated pairs
pairs <- scerevisiae_kaks[, c("dup1", "dup2", "type")]
pairs$dup1 <- paste0("Sce_", pairs$dup1)
pairs$dup2 <- paste0("Sce_", pairs$dup2)

# Classify pairs
trd <- get_transposed(pairs, diamond_inter, annotation)

annotation <- c(annotation, list(Cglabrata2 = annotation$Cglabrata))
blast_inter <- c(diamond_inter, list(Scerevisiae_Cglabrata2 = diamond_inter[[1]]))
```

---

get_transposed_classes

Classify TRD genes as derived from either DNA transposons or retrotransposons

---

Description

Classify TRD genes as derived from either DNA transposons or retrotransposons

Usage

```
get_transposed_classes(pairs, intron_counts)
```

Arguments

pairs	A 3-column data frame with columns **dup1**, **dup2**, and **type** indicating duplicated gene 1, duplicated gene 2, and the mode of duplication associated with the pair. This data frame is returned by `get_transposed()`.
intron_counts	A 2-column data frame with columns **gene** and **introns** indicating the number of introns for each gene, as returned by `get_intron_counts`.

Value

A 3-column data frame with the following variables:

dup1 Character, duplicated gene 1.

dup2 Character, duplicated gene 2.

type Factor of duplication types, with levels "SD" (segmental duplication), "TD" (tandem duplication), "PD" (proximal duplication), "dTRD" (DNA transposon-derived duplication), "rTRD" (retrotransposon-derived duplication), and "DD" (dispersed duplication).

Examples

```
data(diamond_inter)
data(diamond_intra)
data(yeast_seq)
data(yeast_annot)
data(fungi_kaks)
scerevisiae_kaks <- fungi_kaks$saccharomyces_cerevisiae

# Get processed annotation
pdata <- syntenet::process_input(yeast_seq, yeast_annot)
annotation <- pdata$annotation

# Get duplicated pairs
pairs <- scerevisiae_kaks[, c("dup1", "dup2", "type")]
pairs$dup1 <- paste0("Sce_", pairs$dup1)
pairs$dup2 <- paste0("Sce_", pairs$dup2)

# Classify pairs
trd <- get_transposed(pairs, diamond_inter, annotation)

# Create TxDb object from GRanges
library(txdbmaker)
txdb <- txdbmaker::makeTxDbFromGRanges(yeast_annot[[1]])
```

```
# Get intron counts
intron_counts <- get_intron_counts(txdb)

# Get TRD classes
trd_classes <- get_transposed_classes(trd, intron_counts)
```

---

gmax_ks *Duplicate pairs and Ks values for Glycine max*

Description

This data set was obtained with `classify_gene_pairs()` followed by `pairs2kaks()`.

Usage

```
data(gmax_ks)
```

Format

A data frame with the following variables:

dup1 Character, duplicated gene 1.

dup2 Character, duplicated gene 2.

Ks Numeric, Ks values.

type Factor, duplication mode.

Examples

```
data(gmax_ks)
```

---

pairs2kaks *Calculate Ka, Ks, and Ka/Ks from duplicate gene pairs*

Description

Calculate Ka, Ks, and Ka/Ks from duplicate gene pairs

Usage

```
pairs2kaks(gene_pairs_list, cds, model = "MYN", threads = 1, verbose = FALSE)
```

Arguments

gene_pairs_list	List of data frames containing duplicated gene pairs as returned by `classify_gene_pairs()`.
cds	List of DNAStringSet objects containing the coding sequences of each gene.
model	Character scalar indicating which codon model to use. Possible values are "Li", "NG86", "NG", "LWL", "LPB", "MLWL", "MLPB", "GY", "YN", "MYN", "MS", "MA", "GNG", "GLWL", "GLPB", "GMLWL", "GMLPB", "GYN", and "GMYN". Default: "MYN".
threads	Numeric indicating the number of threads to use. Default: 1.
verbose	Logical indicating whether progress messages should be printed on screen. Default: FALSE.

Value

A list of data frames containing gene pairs and their Ka, Ks, and Ka/Ks values.

Examples

```
data(diamond_intra)
data(diamond_inter)
data(yeast_annot)
data(yeast_seq)
data(cds_scerevisiae)
blast_list <- diamond_intra
blast_inter <- diamond_inter

pdata <- syntenet::process_input(yeast_seq, yeast_annot)
annot <- pdata$annotation["Scerevisiae"]

# Binary classification scheme
pairs <- classify_gene_pairs(annot, blast_list)
td_pairs <- pairs[[1]][pairs[[1]]$type == "TD", ]
gene_pairs_list <- list(
    Scerevisiae = td_pairs[seq(1, 3, by = 1), ]
)

cds <- list(Scerevisiae = cds_scerevisiae)

kaks <- pairs2kaks(gene_pairs_list, cds)
```

---

plot_duplicate_freqs *Plot frequency of duplicates per mode for each species*

Description

Plot frequency of duplicates per mode for each species

Usage

```
plot_duplicate_freqs(dup_counts, plot_type = "facet", remove_zero = TRUE)
```

Arguments

dup_counts	A data frame in long format with the number of duplicates per mode for each species, as returned by the function duplicates2counts.
plot_type	Character indicating how to plot frequencies. One of 'facet' (facets for each level of the variable **type**), 'stack' (levels of the variable **type** as stacked bars), or 'stack_percent' (levels of the variable **type** as stacked bars, with x-axis representing relative frequencies). Default: 'facet'.
remove_zero	Logical indicating whether or not to remove rows with zero values. Default: TRUE.

Value

A ggplot object.

Examples

```
data(fungi_kaks)

# Get unique duplicates
duplicate_list <- classify_genes(fungi_kaks)

# Get count table
dup_counts <- duplicates2counts(duplicate_list)

# Plot counts
plot_duplicate_freqs(dup_counts, plot_type = "stack_percent")
```

---

plot_ks_distro *Plot distribution of synonymous substitution rates (Ks)*

Description

Plot distribution of synonymous substitution rates (Ks)

Usage

```
plot_ks_distro(
  ks_df,
  min_ks = 0.01,
  max_ks = 2,
  bytype = FALSE,
  type_levels = NULL,
  plot_type = "histogram",
  binwidth = 0.03
)
```

Arguments

ks_df	A data frame with Ks values for each gene pair as returned by pairs2kaks().
min_ks	Numeric indicating the minimum Ks value to keep. Default: 0.01.
max_ks	Numeric indicating the maximum Ks value to keep. Default: 2.
bytype	Logical indicating whether or not to plot the distribution by type of duplication (requires a column named type).
type_levels	(Only valid if **bytype** is not NULL) Character indicating which levels of the variable specified in parameter **group_by** should be kept. By default, all levels are kept.
plot_type	Character indicating the type of plot to create. If **bytype = TRUE**, possible types are "histogram" or "violin". If **bytype = FALSE**, possible types are "histogram", "density", or "density_histogram". Default: "histogram".
binwidth	(Only valid if **plot_type = "histogram"**) Numeric indicating the bin width. Default: 0.03.

Value

A ggplot object.

Examples

```
data(fungi_kaks)
ks_df <- fungi_kaks$saccharomyces_cerevisiae

# Plot distro
plot_ks_distro(ks_df, bytype = TRUE)
```

---

plot_ks_peaks *Plot histogram of Ks distribution with peaks*

Description

Plot histogram of Ks distribution with peaks

Usage

```
plot_ks_peaks(peaks = NULL, binwidth = 0.05)
```

Arguments

peaks	A list with elements **mean**, **sd**, **lambda**, and **ks**, as returned by the function fins_ks_peaks().
binwidth	Numeric scalar with binwidth for the histogram. Default: 0.05.

Value

A ggplot object with a histogram and lines for each Ks peak.

Examples

```
data(fungi_kaks)
scerevisiae_kaks <- fungi_kaks$saccharomyces_cerevisiae
ks <- scerevisiae_kaks$Ks

# Find 2 peaks in Ks distribution
peaks <- find_ks_peaks(ks, npeaks = 2)

# Plot
plot_ks_peaks(peaks, binwidth = 0.05)
```

---

plot_rates_by_species *Plot distributions of substitution rates (Ka, Ks, or Ka/Ks) per species*

Description

Plot distributions of substitution rates (Ka, Ks, or Ka/Ks) per species

Usage

```
plot_rates_by_species(
  kaks_list,
  rate_column = "Ks",
  bytype = FALSE,
  range = c(0, 2),
  fill = "deepskyblue3",
  color = "deepskyblue4"
)
```

Arguments

kaks_list	A list of data frames with substitution rates per gene pair in each species as returned by `pairs2kaks()`.
rate_column	Character indicating the name of the column to plot. Default: "Ks".
bytype	Logical indicating whether or not to show distributions by type of duplication. Default: FALSE.
range	Numeric vector of length 2 indicating the minimum and maximum values to plot. Default: `c(0, 2)`.
fill	Character with color to use for the fill aesthetic. Ignored if **bytype = TRUE**. Default: "deepskyblue3".
color	Character with color to use for the color aesthetic. Ignored if **bytype = FALSE**. Default: "deepskyblue4".

Details

Data will be plotted using the species order of the list. To change the order of the species to plot, reorder the list elements in **kaks_list**.

Value

A ggplot object.

Examples

```
data(fungi_kaks)

# Plot rates
plot_rates_by_species(fungi_kaks, rate_column = "Ka_Ks")
```

---

split_pairs_by_peak *Split gene pairs based on their Ks peaks*

Description

The purpose of this function is to classify gene pairs by age when there are 2+ Ks peaks. This way, newer gene pairs are found within a certain number of standard deviations from the highest peak, and older genes are found close within smaller peaks.

Usage

```
split_pairs_by_peak(ks_df, peaks, nsd = 2, binwidth = 0.05)
```

Arguments

ks_df	A 3-column data frame with gene pairs in columns 1 and 2, and Ks values for the gene pair in column 3.
peaks	A list with mean, standard deviation, and amplitude of Ks peaks as generated by find_ks_peaks.
nsd	Numeric with the number of standard deviations to consider for each peak.
binwidth	Numeric scalar with binwidth for the histogram. Default: 0.05.

Value

A list with the following elements:

pairs A 4-column data frame with the variables **dup1** (character), **dup2** (character), **ks** (numeric), and **peak** (numeric), representing duplicate gene pair, Ks values, and peak ID, respectively.

plot A ggplot object with Ks peaks as returned by plot_ks_peaks, but with dashed red lines indicating boundaries for each peak.

Examples

```
data(fungi_kaks)
scerevisiae_kaks <- fungi_kaks$saccharomyces_cerevisiae

# Create a data frame of duplicate pairs and Ks values
ks_df <- scerevisiae_kaks[, c("dup1", "dup2", "Ks")]

# Create list of peaks
peaks <- find_ks_peaks(ks_df$Ks, npeaks = 2)

# Split pairs
spairs <- split_pairs_by_peak(ks_df, peaks)
```

---

yeast_annot *Genome annotation of the yeast species S. cerevisiae and C. glabrata*

Description

Data obtained from Ensembl Fungi. Only annotation data protein-coding genes (with associated mRNA, exons, CDS, etc) are included.

Usage

```
data(yeast_annot)
```

Format

A CompressedGRangesList containing the elements **Scerevisiae** and **Cglabrata**.

Examples

```
data(yeast_annot)
```

---

yeast_seq *Protein sequences of the yeast species S. cerevisiae and C. glabrata*

Description

Data obtained from Ensembl Fungi. Only translated sequences of primary transcripts were included.

Usage

```
data(yeast_seq)
```

Format

A list of AAStringSet objects with the elements **Scerevisiae** and **Cglabrata**.

Examples

```
data(yeast_seq)
```

Index

∗ **datasets**
cds_scerevisiae, 3
diamond_inter, 6
diamond_intra, 7
fungi_kaks, 9
gmax_ks, 17
yeast_annot, 23
yeast_seq, 23

cds_scerevisiae, 3
classify_gene_pairs, 4
classify_genes, 3

diamond_inter, 6
diamond_intra, 7
duplicates2counts, 7

find_ks_peaks, 8
fungi_kaks, 9

get_anchors_list, 10
get_intron_counts, 11
get_segmental, 12
get_tandem_proximal, 13
get_transposed, 14
get_transposed_classes, 15
gmax_ks, 17

pairs2kaks, 17
plot_duplicate_freqs, 18
plot_ks_distro, 19
plot_ks_peaks, 20
plot_rates_by_species, 21

split_pairs_by_peak, 22

yeast_annot, 23
yeast_seq, 23