Package ‘doubletrouble’

October 15, 2023

Title Identification and classification of duplicated genes

Version 1.0.0

Date 2022-05-29

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. whole-genome duplication (WGD);
ii. tandem duplication (TD);
iii. proximal duplication (PD);
iv. transposed duplication (TRD) and;
v. dispersed duplication (DD).
If users want a simpler classification scheme, duplicates can also be
classified into WGD- 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

Encoding UTF-8

LazyData false

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

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

Depends R (>= 4.2.0)

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

Config/testthat/edition 3

VignetteBuilder knitr

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

git_branch RELEASE_3_17

git_last_commit c7acec6

git_last_commit_date 2023-04-25

Date/Publication 2023-10-15

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

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

R topics documented:

`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()`.

Value

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

Examples

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

pdata <- syntenet::process_input(yeast_seq, yeast_annot)
annot <- pdata$annotation["Scerevisiae"]
duplicates <- classify_gene_pairs(diamond_intra, annot)
class_genes <- classify_genes(duplicates)
```

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(
  blast_list = NULL,
  annotation = NULL,
  evalue = 1e-10,
  anchors = 5,
  max_gaps = 25,
  binary = FALSE,
  proximal_max = 10,
  blast_inter = 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.
`binary`	Logical indicating whether to perform a binary classification (i.e., whole-genome and small-scale duplication) or not. If FALSE, small-scale duplications are subdivided into tandem, proximal, and dispersed duplications. Default: FALSE.
`proximal_max`	Numeric scalar with the maximum distance (in number of genes) between two genes to consider them as proximal duplicates. Default: 10.
`blast_inter`	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()`.

Value

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

Examples

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

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

# Binary classification scheme
dup_binary <- classify_gene_pairs(blast_list, annot, binary = TRUE)
table(dup_binary$Scerevisiae$type)

# Expanded classification scheme
dup_exp <- classify_gene_pairs(blast_list, annot)
table(dup_exp$Scerevisiae$type)

# Full classification scheme
annotation <- pdata$annotation
dup_full <- classify_gene_pairs(
    blast_list, annotation, blast_inter = blast_inter
)
table(dup_full$Scerevisiae$type)
```

`classify_ssd_pairs` *Classify small-scale duplication-derived gene pairs into subcategories*

Description

SSD-derived gene pairs are classified into tandem, proximal, and dispersed duplicates (TD, PD, and DD, respectively).

Usage

```
classify_ssd_pairs(
  ssd_pairs = NULL,
  annotation_granges = NULL,
  annotation = NULL,
  proximal_max = 10,
  blast_inter = NULL
)
```

Arguments

`ssd_pairs`	A 2-column data frame with SSD-derived gene pairs. This data frame can be obtained by filtering the output of `get_wgd_pairs()` to keep only rows where type == "SSD".
`annotation_granges`	A processed GRanges object as in each element of the list returned by `syntenet::process_input()`.
`annotation`	A processed GRangesList or CompressedGRangesList object as returned by `syntenet::process_input()`, which must contain the gene ranges for all species.
`proximal_max`	Numeric scalar with the maximum distance (in number of genes) between two genes to consider them as proximal duplicates. Default: 10.
`blast_inter`	A list of data frames containing the tabular output of interspecies BLAST/DIAMOND searches, as returned by `run_diamond()`. Each element must contain the pair-wise comparison between a target species and its outgroup, which will be used to identify duplicated genes derived from transpositions (TRD). If this parameter is NULL, this function will not identify TRD genes.

Value

A 3-column data frame with the variables:

dup1 Duplicated gene 1

dup2 Duplicated gene 2

type Duplication type, which can be "TD" (tandem duplication), "PD" (proximal duplication), "TRD" (transposed duplication), and "DD" (dispersed duplication).

Examples

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

# Get processed annotation for S. cerevisiae
pdata <- annotation <- syntenet::process_input(yeast_seq, yeast_annot)
annotation <- pdata$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 and classify them
duplicates <- diamond_intra[[1]][, c(1, 2)]
dups <- get_wgd_pairs(anchor_pairs, duplicates)
ssd_pairs <- dups[dups$type == "SSD", ]

# Get GRanges
annotation_granges <- pdata$annotation[["Scerevisiae"]]
```

```
# Get annotation list
annotation <- pdata$annotation

# Classify SSD-derived gene pairs
ssd_classes <- classify_ssd_pairs(
    ssd_pairs, annotation_granges, annotation, blast_inter = blast_inter
)
```

`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)
```

`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(scerevisiae_kaks)
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)
```

`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
)
```

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.

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_transposed *Get transposed duplicate pairs*

Description

Get transposed duplicate pairs

Usage

```
get_transposed(pairs, blast_inter, annotation)
```

Arguments

`pairs`	A 2-column data frame with duplicated gene 1 and 2 in columns 1 and 2, respectively.
`blast_inter`	A list of data frames of length 1 containing BLAST tabular output for the comparison between target species and 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 `process_input()`.
`annotation`	A processed GRangesList or CompressedGRangesList object as returned by `syntenet::process_input()`.

Value

A 3-column data frame with the following variables:

dup1 Duplicated gene 1

dup2 Duplicated gene 2

type Duplication type, which can be either "TRD" (transposed duplication) or "DD" (dispersed duplication).

Examples

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

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

# Get duplicated pairs
annot <- pdata$annotation["Scerevisiae"]
pairs_all <- classify_gene_pairs(diamond_intra, annot)
pairs <- pairs_all$Scerevisiae[pairs_all$Scerevisiae$type == "DD", c(1, 2)]

trd <- get_transposed(pairs, blast_inter, annotation)
```

`get_wgd_pairs` *Get gene pairs derived from whole-genome and small-scale duplications*

Description

Get gene pairs derived from whole-genome and small-scale duplications

Usage

```
get_wgd_pairs(anchor_pairs = NULL, duplicate_pairs = NULL)
```

Arguments

`anchor_pairs`	A 2-column data frame with anchor pairs in columns 1 and 2.
`duplicate_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 Duplicated gene 1

dup2 Duplicated gene 2

type Duplication type, which can be either "WGD" (whole-genome duplication) or "SSD" (small-scale 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_wgd_pairs(anchor_pairs, duplicates)
```

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.

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)
```

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 scalar indicating the number of threads to use. Default: 1.

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
gene_pairs_list <- classify_gene_pairs(blast_list, annot, binary = TRUE)
gene_pairs_list <- list(
    Scerevisiae = gene_pairs_list[[1]][seq(1, 5, by = 1), ]
)

cds <- list(Scerevisiae = cds_scerevisiae)

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

---

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(scerevisiae_kaks)
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)
```

`scerevisiae_kaks` *Duplicate pairs and Ka, Ks, and Ka/Ks values for S. cerevisiae*

Description

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

Usage

```
data(scerevisiae_kaks)
```

Format

A data frame with 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(scerevisiae_kaks)
```

`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(scerevisiae_kaks)

# 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 for primary transcripts were 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