Package 'syntenet'

July 16, 2025

Title Inference And Analysis Of Synteny Networks

Version 1.11.0

Date 2022-03-28

Description syntenet can be used to infer synteny networks from whole-genome protein sequences and analyze them. Anchor pairs are detected with the MCScanX algorithm, which was ported to this package with the Rcpp framework for R and C++ integration. Anchor pairs from synteny analyses are treated as an undirected unweighted graph (i.e., a synteny network), and users can perform: i. network clustering; ii. phylogenomic profiling (by identifying which species contain which clusters) and; iii. microsynteny-based phylogeny reconstruction with maximum likelihood.

License GPL-3

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

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

biocViews Software, NetworkInference, FunctionalGenomics, ComparativeGenomics, Phylogenetics, SystemsBiology, GraphAndNetwork, WholeGenome, Network

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports Rcpp (>= 1.0.8), BiocParallel, GenomicRanges, rlang, Biostrings, rtracklayer, utils, methods, igraph, stats, grDevices, RColorBrewer, pheatmap, ggplot2, ggnetwork, intergraph

Suggests BiocStyle, ggtree, labdsv, covr, knitr, rmarkdown, testthat (>= 3.0.0), xml2, networkD3

Config/testthat/edition 3

VignetteBuilder knitr

LinkingTo Rcpp, testthat

NeedsCompilation yes

Depends R (>= 4.2)

LazyData false

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

git_branch devel

git_last_commit b93b114

git_last_commit_date 2025-04-15

Repository Bioconductor 3.22

Date/Publication 2025-07-15

Author Fabrício Almeida-Silva [aut, cre] (ORCID: <https://orcid.org/0000-0002-5314-2964>),
Tao Zhao [aut] (ORCID: <https://orcid.org/0000-0001-7302-6445>),
Kristian K Ullrich [aut] (ORCID: <https://orcid.org/0000-0003-4308-9626>),
Yves Van de Peer [aut] (ORCID: <https://orcid.org/0000-0003-4327-3730>)

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

Contents

`syntenet-package` *syntenet: Inference And Analysis Of Synteny Networks*

Description

syntenet can be used to infer synteny networks from whole-genome protein sequences and analyze them. Anchor pairs are detected with the MCScanX algorithm, which was ported to this package with the Rcpp framework for R and C++ integration. Anchor pairs from synteny analyses are treated as an undirected unweighted graph (i.e., a synteny network), and users can perform: i. network clustering; ii. phylogenomic profiling (by identifying which species contain which clusters) and; iii. microsynteny-based phylogeny reconstruction with maximum likelihood.

Author(s)

Maintainer: Fabrício Almeida-Silva <`fabricio_almeidasilva@hotmail.com`> (ORCID)

Authors:

- Tao Zhao <`tao.zhao@nwafu.edu.cn`> (ORCID)
- Kristian K Ullrich <`ullrich@evolbio.mpg.de`> (ORCID)
- Yves Van de Peer <`yves.vandepeer@psb.vib-ugent.be`> (ORCID)

See Also

Useful links:

- `https://github.com/almeidasilvaf/syntenet`
- Report bugs at `https://support.bioconductor.org/t/syntenet`

`angiosperm_phylogeny` *Microsynteny-based angiosperm phylogeny.*

Description

Original tree file obtained from Zhao et al., 2021. The tree is an object of class 'phylo', which can be created by reading the tree file with `treeio::read.tree()`.

Usage

```
data(angiosperm_phylogeny)
```

Format

An object of class 'phylo'.

References

Zhao, T., Zwaenepoel, A., Xue, J. Y., Kao, S. M., Li, Z., Schranz, M. E., & Van de Peer, Y. (2021). Whole-genome microsynteny-based phylogeny of angiosperms. Nature Communications, 12(1), 1-14.

Examples

```
data(angiosperm_phylogeny)
```

`annotation` *Filtered genome annotation for Ostreococcus sp. species*

Description

Data obtained from Pico-PLAZA 3.0. Only annotation data for primary transcripts were included, and only genes for chromosomes 1, 2, and 3.

Usage

```
data(annotation)
```

Format

A CompressedGRangesList containing the elements `Olucimarinus`, `Osp_RCC809`, and `Otauri`.

References

Van Bel, M., Silvestri, F., Weitz, E. M., Kreft, L., Botzki, A., Coppens, F., & Vandepoele, K. (2021). PLAZA 5.0: extending the scope and power of comparative and functional genomics in plants. Nucleic acids research.

Examples

```
data(annotation)
```

`binarize_and_transpose` *Binarize and transpose the phylogenomic profile matrix*

Description

Binarize and transpose the phylogenomic profile matrix

Usage

```
binarize_and_transpose(profile_matrix = NULL)
```

Arguments

`profile_matrix`	A matrix with phylogenomic profiles obtained with `phylogenomic_profile`.

Value

A binary and transposed version of the profiles matrix.

Examples

```
data(clusters)
profile_matrix <- phylogenomic_profile(clusters)
tmat <- binarize_and_transpose(profile_matrix)
```

`blast_list` *List of data frames containing BLAST-like tabular output*

Description

The object was created by running `run_diamond` on the protein sequences for the Ostreococcus algae available in the **proteomes** example data. Hits with <50% identity were filtered out. Code to recreate this data is available at the script/ subdirectory.

Usage

```
data(blast_list)
```

Format

A list of data frames containing the pairwise comparisons between proteomes of Ostreococcus species.

Examples

```
data(blast_list)
```

`check_input` *Check if input objects are ready for further analyses*

Description

Check if input objects are ready for further analyses

Usage

```
check_input(seq = NULL, annotation = NULL, gene_field = "gene_id")
```

Arguments

`seq`	A list of AAStringSet objects, each list element containing protein sequences for a given species. This list must have names (not NULL), and names of each list element must match the names of list elements in **annotation**.
`annotation`	A GRangesList, CompressedGRangesList, or list of GRanges with the annotation for the sequences in **seq**. This list must have names (not NULL), and names of each list element must match the names of list elements in **seq**.
`gene_field`	Character, name of the column in the GRanges objects that contains gene IDs. Default: "gene_id".

Details

This function checks the input data for 3 required conditions:

1. Names of **seq** list (i.e., `names(seq)`) match the names of **annotation** GRangesList/CompressedGRangesList (i.e., `names(annotation)`)
2. For each species (list elements), the number of sequences in **seq** is not greater than the number of genes in **annotation**. This is a way to ensure users do not input the translated sequences for multiple isoforms of the same gene (generated by alternative splicing). Ideally, the number of sequences in **seq** should be equal to the number of genes in **annotation**, but this may not always stand true because of non-protein-coding genes.
3. For each species, sequence names (i.e., `names(seq[[x]])`, equivalent to FASTA headers) match gene names in **annotation**.

Value

TRUE if the objects pass the check.

Examples

```
data(annotation)
data(proteomes)
check_input(proteomes, annotation)
```

`clusters` *Synteny network clusters of BUSCO genes for 25 eudicot species*

Description

Data obtained from Zhao & Schranz, 2019.

Usage

```
data(clusters)
```

Format

A 2-column data frame containing the following variables:

Gene Gene ID

Cluster Cluster ID

References

Zhao, T., & Schranz, M. E. (2019). Network-based microsynteny analysis identifies major differences and genomic outliers in mammalian and angiosperm genomes. Proceedings of the National Academy of Sciences, 116(6), 2165-2174.

Examples

```
data(clusters)
```

`cluster_network` *Cluster the synteny network using the Infomap algorithm*

Description

Cluster the synteny network using the Infomap algorithm

Usage

```
cluster_network(
  network = NULL,
  clust_function = igraph::cluster_infomap,
  clust_params = NULL
)
```

Arguments

`network`	A network represented as an edge list, which is a 2-column data frame with node 1 in the first column and node 2 in the second column. In a synteny network, node 1 and node are the anchor pairs.
`clust_function`	Function to be used to cluster the network. It must be one the functions from the **cluster_*** family in the **igraph** package (e.g., cluster_infomap, cluster_leiden, etc). Default: igraph::cluster_infomap.
`clust_params`	A list with additional parameters (if any) to be passed to the igraph clustering function. Default: NULL (no additional parameters).

Value

A 2-column data frame with the following variables:

Gene Gene ID.

Cluster Cluster ID as identified by infomap.

Examples

```
data(network)
clusters <- cluster_network(network[1:500, ])
```

`collapse_protein_ids` *Collapse protein IDs into gene IDs in sequence names of AAStringSet objects*

Description

This function can be used if the sequence names of the AAStringSet objects contain protein IDs instead of gene IDs (what syntenet requires)

Usage

```
collapse_protein_ids(seq, protein2gene = NULL)
```

Arguments

`seq`	A list of AAStringSet objects, each list element containing protein sequences for a given species. This list must have names (not NULL), and names of each list element must match the names of list elements in **protein2gene**.
`protein2gene`	A list of 2-column data frames containing protein-to-gene ID correspondences, where the first column contains protein IDs, and the second column contains gene IDs. Names of list elements must match names of **seq**.

Details

For each species, this function will replace the protein IDs in sequence names with gene IDs using the protein-to-gene correspondence table in **protein2gene**. After replacing protein IDs with gene IDs, if there are multiple sequences with the same gene ID (indicating different isoforms of the same gene), only the longest sequence is kept, so that the number of sequences is not greater than the number of genes.

Value

A list of AAStringSet objects as in **seq**, but with protein IDs replaced with gene IDs.

Examples

```
# Load data
seq_path <- system.file(
    "extdata", "RefSeq_parsing_example", package = "syntenet"
)
seq <- fasta2AAStringSetlist(seq_path)
annot <- gff2GRangesList(seq_path)

# Clean sequence names
names(seq$Aalosa) <- gsub(" .*", "", names(seq$Aalosa))

# Create a correspondence data frame
cor_df <- as.data.frame(annot$Aalosa[annot$Aalosa$type == "CDS", ])
cor_df <- cor_df[, c("Name", "gene")]

# Create a list of correspondence data frames
protein2gene <- list(Aalosa = cor_df)

# Collapse IDs
new_seqs <- collapse_protein_ids(seq, protein2gene)
```

`create_species_id_table`

Create a data frame of species IDs (3-5-character abbreviations)

Description

Create a data frame of species IDs (3-5-character abbreviations)

Usage

```
create_species_id_table(species_names)
```

Arguments

`species_names`	A character vector of names extracted from the **seq** or **annotation** lists, which can be extracted with `names(seq)` or `names(annotation)`.

Value

A 2-column data frame with the following variables:

species_id Character, species ID consisting of 3-5 characters.

species_name Character, original names passed as input.

Examples

```
# Load 'seq' list (list of AAStringSet objects)
data(proteomes)

# Create ID table
create_species_id_table(names(proteomes))
```

`diamond_is_installed` *Check if DIAMOND is installed*

Description

Check if DIAMOND is installed

Usage

```
diamond_is_installed()
```

Value

Logical indicating whether DIAMOND is installed or not.

Examples

```
diamond_is_installed()
```

`edges` *Synteny network of Ostreococcus genomes represented as an edge list*

Description

The object was created by running `infer_syntenet` on the **blast_list** example data. Code to recreate this data set is available at the script/ subdirectory.

Usage

```
data(edges)
```

Format

A data frame containing anchor pairs between two Ostreococcus proteomes.

Examples

```
data(edges)
```

`export_sequences` *Export processed sequences as FASTA files*

Description

Export processed sequences as FASTA files

Usage

```
export_sequences(seq = NULL, outdir = tempdir())
```

Arguments

`seq`	A processed list of AAStringSet objects as returned by `process_input()`.
`outdir`	Path to output directory where FASTA files will be stored.

Value

Path to exported FASTA files.

Examples

```
# Load data
data(proteomes)
data(annotation)

# Process data
pdata <- process_input(proteomes, annotation)

# Export data
outdir <- file.path(tempdir(), "example_test")
export_sequences(pdata$seq, outdir)
```

`fasta2AAStringSetlist` *Read FASTA files in a directory as a list of AAStringSet objects*

Description

Read FASTA files in a directory as a list of AAStringSet objects

Usage

```
fasta2AAStringSetlist(fasta_dir)
```

Arguments

`fasta_dir`	Character indicating the path to the directory containing FASTA files.

Value

A list of AAStringSet objects, where each element represents a different FASTA file.

Examples

```
fasta_dir <- system.file("extdata", "sequences", package = "syntenet")
aastringsetlist <- fasta2AAStringSetlist(fasta_dir)
```

`find_GS_clusters` *Find group-specific clusters based on user-defined species classification*

Description

Find group-specific clusters based on user-defined species classification

Usage

```
find_GS_clusters(
  profile_matrix = NULL,
  species_annotation = NULL,
  min_percentage = 50
)
```

Arguments

`profile_matrix`	A matrix of phylogenomic profiles obtained with `phylogenomic_profile`.
`species_annotation`	A 2-column data frame with species IDs in the first column (same as column names of profile matrix), and species annotation (e.g., higher-level taxonomic information) in the second column.

`min_percentage`	Numeric scalar with the minimum percentage of species in a group to consider group specificity. For instance, if a given cluster is present in only 1 group of species, but in less than **min_percentage** of the species for this group, it will not be considered a group-specific cluster. This filtering criterion is useful to differentiate group-specific clusters (e.g., family-specific) from subgroup-specific clusters (e.g., genus-specific). Default: 50.

Value

A data frame with the following variables:

Group To which group of species the cluster is specific.

Percentage Percentage of species from the group that are represented by the cluster.

Cluster Cluster ID.

Examples

```
data(clusters)
profile_matrix <- phylogenomic_profile(clusters)

# Species annotation
species_order <- c(
    "vra", "van", "pvu", "gma", "cca", "tpr", "mtr", "adu", "lja",
    "Lang", "car", "pmu", "ppe", "pbr", "mdo", "roc", "fve",
    "Mnot", "Zjuj", "hlu", "jcu", "mes", "rco", "lus", "ptr"
)
species_annotation <- data.frame(
   Species = species_order,
   Family = c(rep("Fabaceae", 11), rep("Rosaceae", 6),
              "Moraceae", "Ramnaceae", "Cannabaceae",
               rep("Euphorbiaceae", 3), "Linaceae", "Salicaceae")
)
gs_clusters <- find_GS_clusters(profile_matrix, species_annotation)
```

`gff2GRangesList` *Read GFF/GTF files in a directory as a GRangesList object*

Description

Read GFF/GTF files in a directory as a GRangesList object

Usage

```
gff2GRangesList(gff_dir)
```

Arguments

`gff_dir`	Character indicating the path to the directory containing GFF/GTF files.

Value

A GRangesList object, where each element represents a different GFF/GTF file.

Examples

```
gff_dir <- system.file("extdata", "annotation", package = "syntenet")
grangeslist <- gff2GRangesList(gff_dir)
```

---

infer_microsynteny_phylogeny

Infer microsynteny-based phylogeny with IQTREE

Description

Infer microsynteny-based phylogeny with IQTREE

Usage

```
infer_microsynteny_phylogeny(
  transposed_profiles = NULL,
  bootr = 1000,
  alrtboot = 1000,
  threads = "AUTO",
  model = "MK+FO+R",
  outdir = tempdir(),
  outgroup = NULL,
  verbose = FALSE
)
```

Arguments

`transposed_profiles`	A binary and transposed profile matrix. The profile matrix can be obtained with `phylogenomic_profile()`.
`bootr`	Numeric scalar with the number of bootstrap replicates. Default: 1000.
`alrtboot`	Numeric scalar with the number of replicates for the SH-like approximate likelihood ratio test. Default: 1000.
`threads`	Numeric scalar indicating the number of threads to use or "AUTO", which allows IQTREE to automatically choose the best number of threads to use. Default: "AUTO".
`model`	Substitution model to use. If you are unsure, pick the default. Default: "MK+FO+R".
`outdir`	Path to output directory. By default, files are saved in a temporary directory, so they will be deleted when the R session closes. If you want to keep the files, specify a custom output directory.
`outgroup`	Name of outgroup clade to group the phylogeny. Default: NULL (unrooted phylogeny).
`verbose`	Logical indicating if progress messages should be prompted. Default: FALSE.

Value

A character vector of paths to output files.

Examples

```
data(clusters)
profile_matrix <- phylogenomic_profile(clusters)
tmat <- binarize_and_transpose(profile_matrix)

# Leave only some legumes and P. mume as an outgroup for testing purposes
included <- c("gma", "pvu", "vra", "van", "cca", "pmu")
tmat <- tmat[rownames(tmat) %in% included, ]

# Remove non-variable sites
tmat <- tmat[, colSums(tmat) != length(included)]

if(iqtree_is_installed()) {
    phylo <- infer_microsynteny_phylogeny(tmat, outgroup = "pmu",
                                          threads = 1)
}
```

`infer_syntenet` *Infer synteny network*

Description

Infer synteny network

Usage

```
infer_syntenet(
  blast_list = NULL,
  annotation = NULL,
  outdir = tempdir(),
  anchors = 5,
  max_gaps = 25,
  is_pairwise = TRUE,
  verbose = FALSE,
  bp_param = BiocParallel::SerialParam(),
  ...
)
```

Arguments

`blast_list`	A list of data frames, each data frame having the tabular output of BLASTp or similar programs, such as DIAMOND. This is the output of the function `run_diamond()`. If you performed pairwise comparisons on the command line, you can read the tabular output as data frames and combine them in a list. List names must be have species names separated by underscore. For instance, if the first list element is a data frame containing the comparison of speciesA (query) against speciesB (database), its name must be "speciesA_speciesB".
`annotation`	A processed GRangesList, CompressedGRangesList, or list of GRanges as returned by `process_input()`.
`outdir`	Path to the output directory. Default: tempdir().

`anchors`	Numeric indicating the minimum required number of genes to call a syntenic block. Default: 5.
`max_gaps`	Numeric indicating the number of upstream and downstream genes to search for anchors. Default: 25.
`is_pairwise`	specify if only pairwise blocks should be reported Default: TRUE
`verbose`	Logical indicating if log messages should be printed on screen. Default: FALSE.
`bp_param`	BiocParallel back-end to be used. Default: `BiocParallel::SerialParam()`.
`...`	Any additional arguments to the `MCScanX` algorithm. For a complete list of all available options, see the man page of `rcpp_mcscanx_file()`.

Value

A network represented as an edge list.

Examples

```
# Load data
data(proteomes)
data(annotation)
data(blast_list)

# Create processed annotation list
annotation <- process_input(proteomes, annotation)$annotation

# Infer the synteny network
net <- infer_syntenet(blast_list, annotation)
```

`interspecies_synteny` *Detect interspecies synteny*

Description

Detect interspecies synteny

Usage

```
interspecies_synteny(
  blast_inter = NULL,
  annotation = NULL,
  inter_dir = file.path(tempdir(), "inter"),
  anchors = 5,
  max_gaps = 25,
  is_pairwise = TRUE,
  verbose = FALSE,
  bp_param = BiocParallel::SerialParam(),
  ...
)
```

Arguments

`blast_inter`	A list of BLAST/DIAMOND data frames for interspecies comparisons as returned by `run_diamond()`.
`annotation`	A processed GRangesList or CompressedGRangesList object as returned by `process_input()`.
`inter_dir`	Path to output directory where .collinearity files will be stored.
`anchors`	Numeric indicating the minimum required number of genes to call a syntenic block. Default: 5.
`max_gaps`	Numeric indicating the number of upstream and downstream genes to search for anchors. Default: 25.
`is_pairwise`	specify if only pairwise blocks should be reported Default: TRUE.
`verbose`	Logical indicating if log messages should be printed on screen. Default: FALSE.
`bp_param`	BiocParallel back-end to be used. Default: `BiocParallel::SerialParam()`.
`...`	Any additional arguments to the `MCScanX` algorithm. For a complete list of all available options, see the man page of `rcpp_mcscanx_file()`.

Value

Paths to .collinearity files.

Examples

```
# Load data
data(proteomes)
data(blast_list)
data(annotation)

# Get DIAMOND and processed annotation lists
blast_inter <- blast_list[2]
annotation <- process_input(proteomes, annotation)$annotation

# Detect interspecies synteny
intersyn <- interspecies_synteny(blast_inter, annotation)
```

`intraspecies_synteny` *Detect intraspecies synteny*

Description

Detect intraspecies synteny

Usage

```
intraspecies_synteny(
  blast_intra = NULL,
  annotation = NULL,
  intra_dir = file.path(tempdir(), "intra"),
  anchors = 5,
  max_gaps = 25,
```

```
    is_pairwise = TRUE,
    verbose = FALSE,
    bp_param = BiocParallel::SerialParam(),
    ...
)
```

Arguments

`blast_intra`	A list of BLAST/DIAMOND data frames for intraspecies comparisons as returned by `run_diamond()`.
`annotation`	A processed GRangesList or CompressedGRangesList object as returned by `process_input()`.
`intra_dir`	Path to output directory where .collinearity files will be stored.
`anchors`	Numeric indicating the minimum required number of genes to call a syntenic block. Default: 5.
`max_gaps`	Numeric indicating the number of upstream and downstream genes to search for anchors. Default: 25.
`is_pairwise`	Logical indicating if only pairwise blocks should be reported. Default: TRUE.
`verbose`	Logical indicating if log messages should be printed on screen. Default: FALSE.
`bp_param`	BiocParallel back-end to be used. Default: `BiocParallel::SerialParam()`.
`...`	Any additional arguments to the `MCScanX` algorithm. For a complete list of all available options, see the man page of `rcpp_mcscanx_file()`.

Value

Paths to .collinearity files.

Examples

```
# Load data
data(scerevisiae_annot)
data(scerevisiae_diamond)

# Detect intragenome synteny
intra_syn <- intraspecies_synteny(
    scerevisiae_diamond, scerevisiae_annot
)
```

`iqtree_is_installed` *Check if IQTREE is installed*

Description

Check if IQTREE is installed

Usage

```
iqtree_is_installed()
```

Value

Logical indicating whether IQTREE is installed or not.

Examples

```
iqtree_is_installed()
```

`iqtree_version` *Get IQ-TREE version*

Description

Get IQ-TREE version

Usage

```
iqtree_version()
```

Value

Numeric indicating IQ-TREE version, with either 1 or 2.

Examples

```
iqtree_version()
```

`last_is_installed` *Check if last is installed*

Description

Check if last is installed

Usage

```
last_is_installed()
```

Value

Logical indicating whether last is installed or not.

Examples

```
last_is_installed()
```

`network` *Synteny network of BUSCO genes for 25 eudicot species*

Description

Data obtained from Zhao & Schranz, 2019.

Usage

```
data(network)
```

Format

An edgelist (i.e., a 2-column data frame with node 1 in column 1 and node 2 in column 2).

References

Zhao, T., & Schranz, M. E. (2019). Network-based microsynteny analysis identifies major differences and genomic outliers in mammalian and angiosperm genomes. Proceedings of the National Academy of Sciences, 116(6), 2165-2174.

Examples

```
data(network)
```

`parse_collinearity` *Parse .collinearity files obtained with MCScan*

Description

The .collinearity files can be obtained with `intraspecies_synteny` and `interspecies_synteny`, which execute a native version of the MCScan algorithm.

Usage

```
parse_collinearity(collinearity_paths = NULL, as = "anchors")
```

Arguments

`collinearity_paths`	Character vector of paths to .collinearity files.
`as`	Character specifying what to extract. One of "anchors" (default), "blocks", or "all".

Value

If **as** is "anchors", a data frame with variables "Anchor1", and "Anchor2". If **as** is "blocks", a data frame with variables "Block", "Block_score", "Chr", and "Orientation". If **as** is "all", a data frame with all aforementioned variables, which indicate:

Block Numeric, synteny block ID

Block_score Numeric, score of synteny block.

Chr Character, query and target chromosome of the synteny block formatted as "&".

Orientation Character, the orientation of genes within blocks, with "plus" indicating that genes are in the same direction, and "minus" indicating that genes are in opposite directions.

Anchor1 Character, gene ID of anchor 1.

Anchor2 Character, gene ID of anchor 2.

Examples

```
collinearity_paths <- system.file(
    "extdata", "Scerevisiae.collinearity", package = "syntenet"
)
net <- parse_collinearity(collinearity_paths)
```

`phylogenomic_profile` *Perform phylogenomic profiling for synteny network clusters*

Description

Perform phylogenomic profiling for synteny network clusters

Usage

```
phylogenomic_profile(clusters = NULL)
```

Arguments

`clusters`	A 2-column data frame with variables **Gene** and **Cluster** as returned by `cluster_network`.

Value

A matrix of i rows and j columns containing the number of genes in cluster i for each species j. The number of rows is equal to the number of clusters in **clusters**, and the number of columns is equal to the number of species in **clusters**.

Examples

```
data(clusters)
profiles <- phylogenomic_profile(clusters)
```

`plot_network` *Plot network*

Description

Plot network

Usage

```
plot_network(
  network = NULL,
  clusters = NULL,
  cluster_id = NULL,
  color_by = "cluster",
  interactive = FALSE,
  dim_interactive = c(600, 600)
)
```

Arguments

`network`	The synteny network represented as an edge list, which is a 2-column data frame with each member of the anchor pair in a column.
`clusters`	A 2-column data frame with the variables **Gene** and **Cluster** representing gene ID and cluster ID, respectively, exactly as returned by `cluster_network`.
`cluster_id`	Character scalar or vector with cluster ID. If more than one cluster is passed as input, clusters are colored differently.
`color_by`	Either "cluster" or a 2-column data frame with gene IDs in the first column and variable to be used for coloring (e.g., taxonomic information) in the second column.
`interactive`	Logical scalar indicating whether to display an interactive network or not. Default: FALSE.
`dim_interactive`	Numeric vector of length 2 with the window dimensions of the interactive plot. If **interactive** is set to FALSE, this parameter is ignored.

Value

A ggplot object with the network.

Examples

```
data(network)
data(clusters)
# Option 1: 1 cluster
cluster_id <- 25
plot_network(network, clusters, cluster_id)

# Option 2: 2 clusters
cluster_id <- c(25, 1089)
plot_network(network, clusters, cluster_id)
```

```
# Option 3: custom annotation for coloring
species_order <- c(
    "vra", "van", "pvu", "gma", "cca", "tpr", "mtr", "adu", "lja",
    "Lang", "car", "pmu", "ppe", "pbr", "mdo", "roc", "fve",
    "Mnot", "Zjuj", "jcu", "mes", "rco", "lus", "ptr"
)
species_annotation <- data.frame(
   Species = species_order,
   Family = c(rep("Fabaceae", 11), rep("Rosaceae", 6),
              "Moraceae", "Ramnaceae", rep("Euphorbiaceae", 3),
              "Linaceae", "Salicaceae")
)
genes <- unique(c(network$node1, network$node2))
gene_df <- data.frame(
    Gene = genes,
    Species = unlist(lapply(strsplit(genes, "_"), head, 1))
)
gene_df <- merge(gene_df, species_annotation)[, c("Gene", "Family")]

plot_network(network, clusters, cluster_id = 25, color_by = gene_df)
```

`plot_profiles` *Plot a heatmap of phylogenomic profiles*

Description

Plot a heatmap of phylogenomic profiles

Usage

```
plot_profiles(
  profile_matrix = NULL,
  species_annotation = NULL,
  palette = "Greens",
  dist_function = stats::dist,
  dist_params = list(method = "euclidean"),
  clust_function = stats::hclust,
  clust_params = list(method = "ward.D"),
  cluster_species = FALSE,
  show_colnames = FALSE,
  discretize = TRUE,
  ...
)
```

Arguments

`profile_matrix`	A matrix of phylogenomic profiles obtained with `phylogenomic_profile`.
`species_annotation`	A 2-column data frame with species IDs in the first column (same as column names of profile matrix), and species annotation (e.g., higher-level taxonomic information) in the second column.
`palette`	A character vector of colors or a character scalar with the name of an RColorBrewer palette. Default: "RdYlBu".

`dist_function`	Function to use to calculate a distance matrix for synteny clusters. Popular examples include `stats::dist`, `labdsv::dsvdis`, and `vegan::vegdist`. Default: stats::dist.
`dist_params`	A list with parameters to be passed to the function specified in parameter **dist_function**. Default: list(method = "euclidean").
`clust_function`	Function to use to cluster the distance matrix returned by the function specified in `dist_function`. Examples include `stats::hclust` and `Rclusterpp::Rclusterpp.hclust`. Default: stats::hclust.
`clust_params`	A list with additional parameters (if any) to be passed to the function specified in parameter **clust_function**. Default: list(method = "ward.D").
`cluster_species`	Either a logical scalar (TRUE or FALSE) or a character vector with the order in which species should be arranged. TRUE or FALSE indicate whether hierarchical clustering should be applied to rows (species). Ideally, the character vector should contain the order of species in a phylogenetically meaningful way. If users pass a named vector, vector names will be used to rename species. If users have a species tree, they can read it with `treeio::read.tree()`, plot it with `ggtree::ggtree()`, and get the species order from the ggtree object with `ggtree::get_taxa_name()`. Default: FALSE.
`show_colnames`	Logical indicating whether to show column names (i.e., cluster IDs) or not. Showing cluster IDs can be useful when visualizing a small subset of them. When visualizing all clusters, cluster IDs are impossible to read. Default: FALSE.
`discretize`	Logical indicating whether to discretize clusters in 4 categories: 0, 1, 2, and 3+. If FALSE, counts will be log2 transformed. Default: TRUE.
`...`	Additional parameters to `pheatmap::pheatmap()`.

Value

A pheatmap object.

Examples

```
data(clusters)
profile_matrix <- phylogenomic_profile(clusters)
species_order <- c(
    "vra", "van", "pvu", "gma", "cca", "tpr", "mtr", "adu", "lja",
    "Lang", "car", "pmu", "ppe", "pbr", "mdo", "roc", "fve",
    "Mnot", "Zjuj", "jcu", "mes", "rco", "lus", "ptr"
)
species_names <- c(
    "V. radiata", "V. angularis", "P. vulgaris", "G. max", "C. cajan",
    "T. pratense", "M. truncatula", "A. duranensis", "L. japonicus",
    "L. angustifolius", "C. arietinum", "P. mume", "P. persica",
    "P. bretschneideri", "M. domestica", "R. occidentalis",
    "F. vesca", "M. notabilis", "Z. jujuba", "J. curcas",
    "M. esculenta", "R. communis", "L. usitatissimum", "P. trichocarpa"
)
names(species_order) <- species_names
species_annotation <- data.frame(
   Species = species_order,
   Family = c(rep("Fabaceae", 11), rep("Rosaceae", 6),
              "Moraceae", "Ramnaceae", rep("Euphorbiaceae", 3),
              "Linaceae", "Salicaceae")
```

```
)
p <- plot_profiles(profile_matrix, species_annotation,
                   cluster_species = species_order)

p <- plot_profiles(profile_matrix, species_annotation,
                   cluster_species = species_order,
                   discretize = FALSE)
```

`process_input` *Process sequence data*

Description

Process sequence data

Usage

```
process_input(
  seq = NULL,
  annotation = NULL,
  gene_field = "gene_id",
  filter_annotation = FALSE
)
```

Arguments

`seq`	A list of AAStringSet objects, each list element containing protein sequences for a given species. This list must have names (not NULL), and names of each list element must match the names of list elements in **annotation**.
`annotation`	A GRangesList, CompressedGRangesList, or list of GRanges with the annotation for the sequences in **seq**. This list must have names (not NULL), and names of each list element must match the names of list elements in **seq**.
`gene_field`	Character, name of the column in the GRanges objects that contains gene IDs. Default: "gene_id".
`filter_annotation`	Logical indicating whether **annotation** should be filtered to keep only genes that are also in **seq**. This is particularly useful if users want to remove information on non-protein coding genes from **annotation**, since such genes are typically not present in sets of whole-genome protein sequences. Default: FALSE.

Details

This function processes the input sequences and annotation to:

1. Remove whitespace and anything after it in sequence names (i.e., `names(seq[[x]])`, which is equivalent to FASTA headers), if there is any.
2. Add a unique species identifier to sequence names. The species identifier consists of the first 3-5 strings of the element name. For instance, if the first element of the **seq** list is named "Athaliana", each sequence in it will have an identifier "Atha_" added to the beginning of each gene name (e.g., Atha_AT1G01010).
3. If sequences have an asterisk (*) representing stop codon, remove it.

4. Add a unique species identifier (same as above) to gene and chromosome names of each element of the **annotation** GRangesList/CompressedGRangesList.
5. Filter each element of the **annotation** GRangesList/CompressedGRangesList to keep only seqnames, ranges, and gene ID.

Value

A list of 2 elements:

seq The processed list of AAStringSet objects from **seq**.

annotation The processed GRangesList or CompressedGRangesList object from **annotation**.

Examples

```
data(annotation)
data(proteomes)
seq <- proteomes
clean_data <- process_input(seq, annotation)
```

`profiles2phylip` *Save the transposed binary profiles matrix to a file in PHYLIP format*

Description

Save the transposed binary profiles matrix to a file in PHYLIP format

Usage

```
profiles2phylip(transposed_profiles = NULL, outdir = tempdir())
```

Arguments

`transposed_profiles`	A binary and transposed profile matrix. The profile matrix can be obtained with `phylogenomic_profile()`.
`outdir`	Path to output directory. By default, files are saved in a temporary directory, so they will be deleted when the R session closes. If you want to keep the files, specify a custom output directory.

Value

Character specifying the path to the PHYLIP file.

Examples

```
data(clusters)
profile_matrix <- phylogenomic_profile(clusters)
tmat <- binarize_and_transpose(profile_matrix)
profiles2phylip(tmat)
```

`proteomes` *Filtered proteomes of Ostreococcus sp. species*

Description

Data obtained from Pico-PLAZA 3.0. Only the translated sequences of primary transcripts were included, and only genes from chromosomes 1, 2, and 3.

Usage

```
data(proteomes)
```

Format

A list of AAStringSet objects containing the elements `Olucimarinus`, `Osp_RCC809`, and `Otauri`.

References

Van Bel, M., Silvestri, F., Weitz, E. M., Kreft, L., Botzki, A., Coppens, F., & Vandepoele, K. (2021). PLAZA 5.0: extending the scope and power of comparative and functional genomics in plants. Nucleic acids research.

Examples

```
data(proteomes)
```

`rcpp_mcscanx_file` *rcpp_mcscanx_file*

Description

MCSCanX provides a clustering module for viewing the relationship of collinear segments in multiple genomes (or heavily redundant genomes). It takes the predicted pairwise segments from dynamic programming (DAGchainer in particular) and then tries to build consensus segments from a set of related, overlapping segments.

Usage

```
rcpp_mcscanx_file(
  blast_file,
  gff_file,
  prefix = "out",
  outdir = "",
  match_score = 50L,
  gap_penalty = -1L,
  match_size = 5L,
  e_value = 1e-05,
  max_gaps = 25L,
  overlap_window = 5L,
  is_pairwise = FALSE,
```

```
  in_synteny = 0L,
  species_id_length = 3L,
  verbose = FALSE
)
```

Arguments

`blast_file`	Character indicating the path to the BLAST/DIAMOND output file.
`gff_file`	Character indicating the path to the "gff" file, which is a tab-delimited file with 4 columns indicating the chromosome name, gene id, gene start position, and gene end position, respectively.
`prefix`	Character indicating the prefix to output files. Default: "out".
`outdir`	Character indicating the path to the output directory. Default: "".
`match_score`	Numeric indicating the match score. Default: 50.
`gap_penalty`	Numeric indicating the gap penalty. Default: -1.
`match_size`	Numeric indicating the minimum number of genes required to call synteny. Default: 5.
`e_value`	Numeric indicating the minimum e-value allowed. Default: 1e-5.
`max_gaps`	Numeric indicating the maximum number of gaps between genes allowed. The unit measure of gaps is number of genes, so `max_gaps = 20` indicates that a maximum of 20 genes can exist between two homologous genes for synteny to be called. Default: 25.
`overlap_window`	Numeric indicating the overlap window. Default: 5.
`is_pairwise`	Logical indicating whether only pairwise blocks should be reported. Default: FALSE.
`in_synteny`	Numeric indicating the patterns of collinear blocks, where 0 indicates intra and interspecies comparisons, 1 indicates intraspecies comparisons, and 2 indicates interspecies comparisons. Default: 0.
`species_id_length`	Integer indicating the length of the species IDs. Default: 3. 0: intra- and inter-species (default); 1: intra-species; 2: inter-species
`verbose`	Logical indicating whether to print progress messages to the screen. Default: FALSE.

Value

NULL, and a .collinearity file is created in the directory specified in `outdir`.

Author(s)

Kristian K Ullrich and Fabricio Almeida-Silva

References

Wang et al. (2012) MCScanX: a toolkit for detection and evolutionary analysis of gene synteny and collinearity. *Nucleic acids research*. **40.7**, e49-e49.

Haas et al. (2004) DAGchainer: a tool for mining segmental genome duplications and synteny. *Bioinformatics*. **20.18** 3643-3646.

`read_diamond` *Read DIAMOND/BLAST tables as a list of data frames*

Description

Read DIAMOND/BLAST tables as a list of data frames

Usage

```
read_diamond(diamond_dir = NULL)
```

Arguments

`diamond_dir`	Path to directory containing the tabular output of DIAMOND or similar programs (e.g., BLAST).

Value

A list of data frames with the tabular DIAMOND output.

Examples

```
# Path to output directory
diamond_dir <- system.file("extdata", package = "syntenet")

# Read output
l <- read_diamond(diamond_dir)
```

`run_diamond` *Wrapper to run DIAMOND from an R session*

Description

Wrapper to run DIAMOND from an R session

Usage

```
run_diamond(
  seq = NULL,
  top_hits = 5,
  verbose = FALSE,
  outdir = tempdir(),
  threads = NULL,
  compare = "all",
  ...
)
```

Arguments

`seq`	A processed list of AAStringSet objects as returned by `process_input()`.
`top_hits`	Number of top hits to keep in DIAMOND search. Default: 5.
`verbose`	Logical indicating if progress messages should be printed. Default: FALSE.
`outdir`	Output directory for DIAMOND results. By default, output files are saved to a temporary directory.
`threads`	Number of threads to use. Default: let DIAMOND auto-detect and use all available virtual cores on the machine.
`compare`	Character scalar indicating which comparisons should be made when running DIAMOND. Possible modes are "all" (all-vs-all comparisons), "intraspecies" (intraspecies comparisons only), or "interspecies" (interspecies comparisons only). Alternatively, users can pass a 2-column data frame as input with the names of species to be compared.
`...`	Any additional arguments to `diamond blastp`.

Value

A list of data frames containing DIAMOND's tabular output for each pairwise combination of species. For n species, the list length will be n^2.

Examples

```
data(proteomes)
data(annotation)
seq <- process_input(proteomes, annotation)$seq[1:2]
if(diamond_is_installed()) {
    diamond_results <- run_diamond(seq)
}
```

`run_last` *Wrapper to run last from an R session*

Description

Wrapper to run last from an R session

Usage

```
run_last(
  seq = NULL,
  verbose = FALSE,
  outdir = tempdir(),
  threads = 1,
  compare = "all",
  lastD = 1e+06,
  ...
)
```

Arguments

`seq`	A processed list of AAStringSet objects as returned by `process_input()`.
`verbose`	Logical indicating if progress messages should be printed. Default: FALSE.
`outdir`	Output directory for last results. By default, output files are saved to a temporary directory.
`threads`	Number of threads to use. Default: 1.
`compare`	Character scalar indicating which comparisons should be made when running last. Possible modes are "all" (all-vs-all comparisons), "intraspecies" (intraspecies comparisons only), or "interspecies" (interspecies comparisons only). Alternatively, users can pass a 2-column data frame as input with the names of species to be compared.
`lastD`	last option D: query letters per random alignment. Default: 1e6.
`...`	Any additional arguments to `lastal`.

Value

A list of data frames containing last's tabular output for each pairwise combination of species. For n species, the list length will be n^2.

Examples

```
data(proteomes)
data(annotation)
seq <- process_input(proteomes, annotation)$seq[1:2]
if(last_is_installed()) {
    last_results <- run_last(seq)
}
```

`scerevisiae_annot` *Genome annotation of the yeast species S. cerevisiae*

Description

Data obtained from Ensembl Fungi. Only annotation data for primary transcripts were included.

Usage

```
data(scerevisiae_annot)
```

Format

A GRangesList as returned by `process_input()` containing the element **Scerevisiae**.

Examples

```
data(scerevisiae_annot)
```

`scerevisiae_diamond` *Intraspecies DIAMOND output for S. cerevisiae*

Description

List obtained with `run_diamond()`.

Usage

```
data(scerevisiae_diamond)
```

Format

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

Examples

```
data(scerevisiae_diamond)
```

Index