Package 'cogeqc'

November 28, 2024

Title Systematic quality checks on comparative genomics analyses

Version 1.10.0

Date 2022-01-26

Description cogeqc aims to facilitate systematic quality checks on standard comparative genomics analyses to help researchers detect issues and select the most suitable parameters for each data set. cogeqc can be used to asses: i. genome assembly and annotation quality with BUSCOs and comparisons of statistics with publicly available genomes on the NCBI; ii. orthogroup inference using a protein domain-based approach and; iii. synteny detection using synteny network properties. There are also data visualization functions to explore QC summary statistics.

License GPL-3

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

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

biocViews Software, GenomeAssembly, ComparativeGenomics, FunctionalGenomics, Phylogenetics, QualityControl, Network

Encoding UTF-8

LazyData false

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

SystemRequirements BUSCO (>= 5.1.3) <https://busco.ezlab.org/>

Imports utils, graphics, stats, methods, reshape2, ggplot2, scales, ggtree, patchwork, igraph, rlang, ggbeeswarm, jsonlite, Biostrings

Depends R (>= 4.2.0)

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

Config/testthat/edition 3

VignetteBuilder knitr

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

git_branch RELEASE_3_20

git_last_commit 8ce711f

git_last_commit_date 2024-10-29

Repository Bioconductor 3.20

Date/Publication 2024-11-28

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>

Contents

cogeqc-package *cogeqc: Systematic quality checks on comparative genomics analyses*

Description

cogeqc aims to facilitate systematic quality checks on standard comparative genomics analyses to help researchers detect issues and select the most suitable parameters for each data set. cogeqc can be used to asses: i. genome assembly and annotation quality with BUSCOs and comparisons of statistics with publicly available genomes on the NCBI; ii. orthogroup inference using a protein domain-based approach and; iii. synteny detection using synteny network properties. There are also data visualization functions to explore QC summary statistics.

Author(s)

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

Authors:

- Yves Van de Peer <yves.vandepeer@psb.vib-ugent.be> (ORCID)

See Also

Useful links:

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

assess_orthogroups *Assess orthogroup inference based on functional annotation*

Description

Assess orthogroup inference based on functional annotation

Usage

```
assess_orthogroups(
  orthogroups = NULL,
  annotation = NULL,
  correct_overclustering = TRUE
)
```

Arguments

orthogroups	A 3-column data frame with columns **Orthogroup**, **Species**, and **Gene**. This data frame can be created from the 'Orthogroups.tsv' file generated by OrthoFinder with the function `read_orthogroups()`.

annotation A list of 2-column data frames with columns **Gene** (gene ID) and **Annotation** (annotation ID). The names of list elements must correspond to species names as in the second column of *orthogroups*. For instance, if there are two species in the *orthogroups* data frame named "SpeciesA" and "SpeciesB", *annotation* must be a list of 2 data frames, and each list element must be named "SpeciesA" and "SpeciesB".

correct_overclustering
Logical indicating whether to correct for overclustering in orthogroups. Default: TRUE.

Value

A data frame.

Examples

```
data(og)
data(interpro_ath)
data(interpro_bol)
# Subsetting annotation for demonstration purposes.
annotation <- list(Ath = interpro_ath[1:1000,], Bol = interpro_bol[1:1000,])
assess <- assess_orthogroups(og, annotation)
```

---

assess_synnet *Assess synteny network based on graph properties*

Description

Assess synteny network based on graph properties

Usage

```
assess_synnet(synnet = NULL, cc_type = "average")
```

Arguments

synnet Edge list for the synteny network in a 2-column data frame, with columns 1 and 2 representing names of loci in anchor 1 and anchor 2, respectively.

cc_type Type of clustering coefficient to be calculated. One of 'global' or 'average'. Default: 'average'.

Details

Network score is the product of the network's clustering coefficient, node count, and R squared for the scale-free topology fit.

Value

A data frame with the following variables:

CC Numeric representing clustering coefficient.

Node_count Numeric representing number of nodes in the network.

Rsquared Numeric indicating the coefficient of determination for the scale-free topology fit.

Score Numeric representing network score, which is the product of 'CC' and 'Node_number'.

Examples

```
data(synnet)
assess_synnet(synnet)
```

---

assess_synnet_list *Assess list of synteny networks as in* assess_synnet

---

Description

Assess list of synteny networks as in `assess_synnet`

Usage

```
assess_synnet_list(synnet_list = NULL, cc_type = "average")
```

Arguments

synnet_list	A list of networks, each network being an edge list as a 2-column data frame, with columns 1 and 2 representing names of loci in anchor 1 and anchor 2, respectively.
cc_type	Type of clustering coefficient to be calculated. One of 'global' or 'average'. Default: 'average'.

Value

A data frame with the following variables:

CC Numeric representing clustering coefficient.

Node_count Numeric representing number of nodes in the network.

Rsquared Numeric indicating the coefficient of determination for the scale-free topology fit.

Score Numeric representing network score, which is the product of 'CC' and 'Node_number'.

Network Character of network name.

Examples

```
set.seed(123)
data(synnet)
net1 <- synnet
net2 <- synnet[-sample(1:10000, 500), ]
net3 <- synnet[-sample(1:10000, 1000), ]
synnet_list <- list(net1 = net1, net2 = net2, net3 = net3)
assess_synnet_list(synnet_list)
```

batch_summary *BUSCO summary output for batch mode*

Description

This object was created with the function `read_busco()` using a batch run of BUSCO on the genomes of Herbaspirillum seropedicae SmR1 and Herbaspirillum rubrisubalbicans M1.

Usage

```
data(batch_summary)
```

Format

A 2-column data frame with the following variables:

Class Factor of BUSCO classes

Frequency Numeric with the percentage of BUSCOs in each class.

Lineage Character with the lineage dataset used.

File Character with the name of the FASTA file used.

Examples

```
data(batch_summary)
```

busco_is_installed *Check if BUSCO is installed*

Description

Check if BUSCO is installed

Usage

```
busco_is_installed()
```

Value

Logical indicating whether BUSCO is installed or not.

Examples

```
busco_is_installed()
```

`calculate_H` *Calculate homogeneity scores for orthogroups*

Description

Calculate homogeneity scores for orthogroups

Usage

```
calculate_H(
  orthogroup_df,
  correct_overclustering = TRUE,
  max_size = 200,
  update_score = TRUE
)
```

Arguments

`orthogroup_df`	Data frame with orthogroups and their associated genes and annotation. The columns **Gene**, **Orthogroup**, and **Annotation** are mandatory, and they must represent Gene ID, Orthogroup ID, and Annotation ID (e.g., Interpro/PFAM), respectively.
`correct_overclustering`	Logical indicating whether to correct for overclustering in orthogroups. Default: TRUE.
`max_size`	Numeric indicating the maximum orthogroup size to consider. If orthogroups are too large, calculating Sorensen-Dice indices for all pairwise combinations could take a long time, so setting a limit prevents that. Default: 200.
`update_score`	Logical indicating whether to replace scores with corrected scores or not. If FALSE, the dispersal term and corrected scores are returned as separate variables in the output data frame.

Details

Homogeneity is calculated based on pairwise Sorensen-Dice similarity indices between gene pairs in an orthogroup, and they range from 0 to 1. Thus, if all genes in an orthogroup share the same domain, the orthogroup will have a homogeneity score of 1. On the other hand, if genes in an orthogroup do not have any domain in common, the orthogroup will have a homogeneity score of 0. The percentage of orthogroups with size greater than **max_size** will be subtracted from the homogeneity scores, since too large orthogroups typically have very low scores. Additionally, users can correct for overclustering by penalizing protein domains that appear in multiple orthogroups (default).

Value

A 2-column data frame with the variables **Orthogroup** and **Score**, corresponding to orthogroup ID and orthogroup score, respectively. If **update_score = FALSE**, additional columns named **Dispersal** and **Score_c** are added, which correspond to the dispersal term and corrected scores, respectively.

Examples

```
data(og)
data(interpro_ath)
orthogroup_df <- merge(og[og$Species == "Ath", ], interpro_ath)
# Filter data to reduce run time
orthogroup_df <- orthogroup_df[1:10000, ]
H <- calculate_H(orthogroup_df)
```

`compare_genome_stats` *Compare user-defined assembly statistics with statistics of NCBI genomes*

Description

This function helps users analyze their genome assembly stats in a context by comparing metrics obtained by users with "reference" metrics in closely-related organisms.

Usage

```
compare_genome_stats(ncbi_stats = NULL, user_stats = NULL)
```

Arguments

`ncbi_stats`	A data frame of summary statistics for a particular taxon obtained from the NCBI, as obtained with the function `get_genome_stats`.
`user_stats`	A data frame with assembly statistics obtained by the user. A column named **accession** is mandatory, and it must contain unique identifiers for the genome(s) analyzed by the user. Dummy variables can be used as identifiers (e.g., "my_genome_001"), as long as they are unique. All other column containing assembly stats must have the same names as their corresponding columns in the data frame specified in **ncbi_stats**. For instance, stats on total number of genes and sequence length must be in columns named "gene_count_total" and "sequence_length", as in the **ncbi_stats** data frame.

Details

For each genome assembly statistic (e.g., "gene_count_total"), values in **user_stats** are compared to a distribution of values from **ncbi_stats**, and their percentile and rank in the distributions are reported.

Value

A data frame with the following variables:

accession character, unique identifier as in user_stats$accession.

variable character, name of the genome assembly metric (e.g., "CC_ratio").

percentile numeric, percentile in the distribution.

rank numeric, rank in the distribution (highest to lowest). For the variable "CC_ratio", ranks go from lowest to highest.

Examples

```
# Use case: user assembled a maize (Zea mays) genome

## Obtain stats for maize genomes on the NCBI
ncbi_stats <- get_genome_stats(taxon = "Zea mays")

## Create a data frame of stats for fictional maize genome
user_stats <- data.frame(
    accession = "my_lovely_maize",
    sequence_length = 2.4 * 1e9,
    gene_count_total = 50000,
    CC_ratio = 1
)

# Compare stats
compare_genome_stats(ncbi_stats, user_stats)
```

---

compare_orthogroups *Compare inferred orthogroups to a reference set*

Description

Compare inferred orthogroups to a reference set

Usage

```
compare_orthogroups(ref_orthogroups = NULL, test_orthogroups = NULL)
```

Arguments

`ref_orthogroups`
: Reference orthogroups in a 3-column data frame with columns **Orthogroup**, **Species**, and **Gene**. This data frame can be created from the 'Orthogroups.tsv' file generated by OrthoFinder with the function `read_orthogroups()`.

`test_orthogroups`
: Test orthogroups that will be compared to *ref_orthogroups* in the same 3-column data frame format.

Details

This function compares a test set of orthogroups to a reference set and returns which orthogroups in the reference set are fully preserved in the test set (i.e., identical gene repertoire) and which are not. Species names (column 2) must be the same between reference and test set. If some species are not shared between reference and test sets, they will not be considered for the comparison.

Value

A 2-column data frame with the following variables:

Orthogroup Character of orthogroup IDs.

Preserved A logical vector of preservation status. It is TRUE if the orthogroup in the reference set is fully preserved in the test set, and FALSE otherwise.

Examples

```
set.seed(123)
data(og)
og <- og[1:5000, ]
ref <- og
# Shuffle genes to simulate a different set
test <- data.frame(
    Orthogroup = sample(og$Orthogroup, nrow(og), replace = FALSE),
    Species = og$Species,
    Gene = og$Gene
)
comparison <- compare_orthogroups(ref, test)

# Calculating percentage of preservation
sum(comparison$Preserved) / length(comparison$Preserved)
```

`fit_sft` *Goodness of fit test for the scale-free topology model*

Description

Goodness of fit test for the scale-free topology model

Usage

```
fit_sft(edges)
```

Arguments

`edges`	A 2-column data frame with network edges represented in each. Columns 1 and 2 represent nodes 1 and 2 of each edge.

Value

A numeric scalar with the R squared for the scale-free topology fit.

Examples

```
data(synnet)
edges <- synnet
fit_sft(edges)
```

`get_genome_stats` *Get summary statistics for genomes on NCBI using the NCBI Datasets API*

Description

Get summary statistics for genomes on NCBI using the NCBI Datasets API

Usage

```
get_genome_stats(taxon = NULL, filters = NULL)
```

Arguments

`taxon`	Taxon for which summary statistics will be retrieved, either as a character scalar (e.g., "brassicaceae") or as a numeric scalar representing NCBI Taxonomy ID (e.g., 3700).
`filters`	(optional) A list of filters to use when querying the API in the form of key-value pairs, with keys in list names and values in list elements (e.g., `list(filters.reference_only = "true")`, see examples for details).

Details

Possible filters for the **filters** parameter can be accessed at https://www.ncbi.nlm.nih.gov/datasets/docs/v2/reference-docs/rest-api/#get-/genome/taxon/-taxons-/dataset_report.

Value

A data frame with the following variables:

accession character, accession number.

source character, data source.

species_taxid numeric, NCBI Taxonomy ID.

species_name character, species' scientific name.

species_common_name character, species' common name.

species_ecotype character, species' ecotype.

species_strain character, species' strain.

species_isolate character, species' isolate.

species_cultivar character, species' cultivar.

assembly_level factor, assembly level ("Complete", "Chromosome", "Scaffold", or "Contig").

assembly_status character, assembly status.

assembly_name character, assembly name.

assembly_type character, assembly type.

submission_date character, submission date (YYYY-MM-DD).

submitter character, submitter name.

sequencing_technology character, sequencing technology.

atypical logical, indicator of wheter the genome is atypical.

refseq_category character, RefSeq category.

chromosome_count numeric, number of chromosomes.

sequence_length numeric, total sequence length.

ungapped_length numeric, ungapped sequence length.

contig_count numeric, number of contigs.

contig_N50 numeric, contig N50.

contig_L50 numeric, contig L50.

scaffold_N50 numeric, contig N50.

scaffold_L50 numeric, contig L50.

GC_percent numeric, GC percentage (0-100).

annotation_provider character, name of annotation provider.

annotation_release_date character, annotation release date (YYYY-MM-DD).

gene_count_total numeric, total number of genes.

gene_count_coding numeric, number of protein-coding genes.

gene_count_noncoding numeric, number of non-coding genes.

gene_count_pseudogene numeric, number of pseudogenes.

gene_count_other numeric, number of other genes.

CC_ratio numeric, ratio of the number of contigs to the number of chromosomes.

Examples

```
# Example 1: Search for A. thaliana genomes by tax ID
ex1 <- get_genome_stats(taxon = 3702)

# Example 2: Search for A. thaliana genomes by name
ex2 <- get_genome_stats(taxon = "Arabidopsis thaliana")

# Example 3: Search for chromosome-level Brassicaeae genomes
ex3 <- get_genome_stats(
    taxon = "brassicaceae",
    filters = list(filters.assembly_level = "chromosome")
)
```

---

interpro_ath *Intepro annotation for Arabidopsis thaliana's genes*

Description

The annotation data were retrieved from PLAZA Dicots 5.0.

Usage

```
data(interpro_ath)
```

Format

A 2-column data frame:

Gene Character of gene IDs.

Annotation Character of Interpro domains.

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(interpro_ath)
```

---

interpro_bol *Intepro annotation for Brassica oleraceae's genes*

Description

The annotation data were retrieved from PLAZA Dicots 5.0.

Usage

```
data(interpro_bol)
```

Format

A 2-column data frame:

Gene Character of gene IDs.

Annotation Character of Interpro domains.

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(interpro_bol)
```

list_busco_datasets *List BUSCO data sets*

Description

List BUSCO data sets

Usage

```
list_busco_datasets()
```

Value

A hierarchically organized list of available data sets as returned by `busco --list-datasets`.

Examples

```
if(busco_is_installed()) {
    list_busco_datasets()
}
```

og *Orthogroups between Arabidopsis thaliana and Brassica oleraceae*

Description

Data obtained from PLAZA Dicots 5.0.

Usage

```
data(og)
```

Format

A 3-column data frame with the following variables:

Orthogroup Orthogroup ID.

Species Abbreviation for species' name.

Gene Gene ID

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(og)
```

plot_busco *Plot BUSCO summary output*

Description

Plot BUSCO summary output

Usage

```
plot_busco(summary_df = NULL)
```

Arguments

summary_df	Data frame with BUSCO summary output as returned by `read_busco()`.

Value

A ggplot object with a barplot of BUSCOs in each class.

Examples

```
# Single file
result_dir <- system.file("extdata", package = "cogeqc")
summary_df <- read_busco(result_dir)
# Batch mode
data(batch_summary)
plot_busco(summary_df)
plot_busco(batch_summary)
```

plot_duplications *Plot species-specific duplications*

Description

Plot species-specific duplications

Usage

```
plot_duplications(stats_list = NULL)
```

Arguments

stats_list	A list of data frames with Orthofinder summary stats as returned by the function `read_orthofinder_stats`.

Value

A ggplot object with a barplot of number of species-specific duplications.

Examples

```
dir <- system.file("extdata", package = "cogeqc")
stats_list <- read_orthofinder_stats(dir)
plot_duplications(stats_list)
```

`plot_genes_in_ogs` *Plot percentage of genes in orthogroups for each species*

Description

Plot percentage of genes in orthogroups for each species

Usage

```
plot_genes_in_ogs(stats_list = NULL)
```

Arguments

`stats_list`	A list of data frames with Orthofinder summary stats as returned by the function `read_orthofinder_stats`.

Value

A ggplot object with a barplot of percentages of genes in orthogroups for each species.

Examples

```
dir <- system.file("extdata", package = "cogeqc")
stats_list <- read_orthofinder_stats(dir)
plot_genes_in_ogs(stats_list)
```

`plot_genome_stats` *Plot statistics on genome assemblies on the NCBI*

Description

Plot statistics on genome assemblies on the NCBI

Usage

```
plot_genome_stats(ncbi_stats = NULL, user_stats = NULL)
```

Arguments

ncbi_stats	A data frame of summary statistics for a particular taxon obtained from the NCBI, as obtained with the function get_genome_stats.
user_stats	(Optional) A data frame with assembly statistics obtained by the user. Statistics in this data frame are highlighted in red if this data frame is passed. A column named **accession** is mandatory, and it must contain unique identifiers for the genome(s) analyzed by the user. Dummy variables can be used as identifiers (e.g., "my_genome_001"), as long as they are unique. All other column containing assembly stats must have the same names as their corresponding columns in the data frame specified in **ncbi_stats**. For instance, stats on total number of genes and sequence length must be in columns named "gene_count_total" and "sequence_length", as in the **ncbi_stats** data frame.

Value

A composition of ggplot objects made with patchwork.

Examples

```
# Example 1: plot stats on maize genomes on the NCBI
## Obtain stats for maize genomes on the NCBI
ncbi_stats <- get_genome_stats(taxon = "Zea mays")

plot_genome_stats(ncbi_stats)

## Plot stats
# Example 2: highlight user-defined stats in the distribution
## Create a data frame of stats for fictional maize genome
user_stats <- data.frame(
    accession = "my_lovely_maize",
    sequence_length = 2.4 * 1e9,
    gene_count_total = 50000,
    CC_ratio = 1
)

plot_genome_stats(ncbi_stats, user_stats)
```

---

plot_og_overlap *Plot pairwise orthogroup overlap between species*

Description

Plot pairwise orthogroup overlap between species

Usage

```
plot_og_overlap(stats_list = NULL, clust = TRUE)
```

Arguments

stats_list	A list of data frames with Orthofinder summary stats as returned by the function read_orthofinder_stats.
clust	Logical indicating whether to clust data based on overlap. Default: TRUE

Value

A ggplot object with a heatmap.

Examples

```
dir <- system.file("extdata", package = "cogeqc")
stats_list <- read_orthofinder_stats(dir)
plot_og_overlap(stats_list)
```

`plot_og_sizes` *Plot orthogroup sizes per species*

Description

Plot orthogroup sizes per species

Usage

```
plot_og_sizes(orthogroups = NULL, log = FALSE, max_size = NULL)
```

Arguments

`orthogroups`	A 3-column data frame with columns **Orthogroup**, **Species**, and **Gene**. This data frame can be created from the 'Orthogroups.tsv' file generated by OrthoFinder with the function `read_orthogroups()`.
`log`	Logical indicating whether to transform orthogroups sizes with natural logarithms. Default: FALSE.
`max_size`	Numeric indicating the maximum orthogroup size to consider. If this parameter is not NULL, orthogroups larger than `max_size` (e.g., 100) will not be considered. Default: NULL.

Value

A ggplot object with a violin plot.

Examples

```
data(og)
plot_og_sizes(og, log = TRUE)
plot_og_sizes(og, max_size = 100)
plot_og_sizes(og, log = TRUE, max_size = 100)
```

plot_orthofinder_stats

Plot a panel with a summary of Orthofinder stats

Description

This function is a wrapper for `plot_species_tree`, `plot_duplications`, `plot_genes_in_ogs`, `plot_species_specific_ogs`.

Usage

```
plot_orthofinder_stats(tree = NULL, stats_list = NULL, xlim = c(0, 1))
```

Arguments

`tree`	Tree object as returned by `treeio::read.*`, a family of functions in the **treeio** package to import tree files in multiple formats, such as Newick, Phylip, NEXUS, and others. If your species tree was inferred with Orthofinder (using STAG), the tree file is located in *Species_Tree/SpeciesTree_rooted_node_labels.txt*. Then, it can be imported with `treeio::read_tree(path_to_file)`.
`stats_list`	(optional) A list of data frames with Orthofinder summary stats as returned by the function `read_orthofinder_stats`. If this list is given as input, nodes will be labeled with the number of duplications.
`xlim`	Numeric vector of x-axis limits. This is useful if your node tip labels are not visible due to margin issues. Default: c(0, 1).

Value

A panel of ggplot objects.

Examples

```
data(tree)
dir <- system.file("extdata", package = "cogeqc")
stats_list <- read_orthofinder_stats(dir)
plot_orthofinder_stats(tree, xlim = c(0, 1.5), stats_list = stats_list)
```

plot_species_specific_ogs

Plot number of species-specific orthogroups

Description

Plot number of species-specific orthogroups

Usage

```
plot_species_specific_ogs(stats_list = NULL)
```

Arguments

stats_list	A list of data frames with Orthofinder summary stats as returned by the function read_orthofinder_stats.

Value

A ggplot object with a barplot of number of species-specific orthogroups for each species.

Examples

```
dir <- system.file("extdata", package = "cogeqc")
stats_list <- read_orthofinder_stats(dir)
plot_species_specific_ogs(stats_list)
```

---

plot_species_tree *Plot species tree*

Description

Plot species tree

Usage

```
plot_species_tree(tree = NULL, xlim = c(0, 1), stats_list = NULL)
```

Arguments

tree	Tree object as returned by treeio::read.*, a family of functions in the **treeio** package to import tree files in multiple formats, such as Newick, Phylip, NEXUS, and others. If your species tree was inferred with Orthofinder (using STAG), the tree file is located in *Species_Tree/SpeciesTree_rooted_node_labels.txt*. Then, it can be imported with treeio::read_tree(path_to_file).
xlim	Numeric vector of x-axis limits. This is useful if your node tip labels are not visible due to margin issues. Default: c(0, 1).
stats_list	(optional) A list of data frames with Orthofinder summary stats as returned by the function read_orthofinder_stats. If this list is given as input, nodes will be labeled with the number of duplications.

Value

A ggtree/ggplot object with the species tree.

Examples

```
data(tree)
plot_species_tree(tree)
```

read_busco *Read and parse BUSCO's summary report*

Description

Read and parse BUSCO's summary report

Usage

```
read_busco(result_dir = NULL)
```

Arguments

result_dir	Path to the directory where BUSCO results are stored. This function will look for the short_summary* file (single run) or short_summary* file (batch mode).

Value

A data frame with the following variables:

Class BUSCO class. One of **Complete_SC**, **Complete_duplicate**, **Fragmented**, or **Missing**

Frequency Frequency of BUSCOs in each class. If BUSCO was run in batch mode, this variable will contain relative frequencies. If BUSCO was run for a single file, it will contain absolute frequencies.

Lineage Name of the lineage dataset used.

File (batch mode only) Name of the input FASTA file.

Examples

```
result_dir <- system.file("extdata", package = "cogeqc")
df <- read_busco(result_dir)
```

read_orthofinder_stats *Read and parse Orthofinder summary statistics*

Description

Read and parse Orthofinder summary statistics

Usage

```
read_orthofinder_stats(stats_dir = NULL)
```

Arguments

stats_dir	Path to directory containing Orthofinder's comparative genomics statistics. In your Orthofinder results directory, this directory is named **Comparative_Genomics_Statistics**.

Value

A list of data frames with the following elements:

1. **stats** A data frame of summary stats per species with the following variables:

   Species Factor of species names.

   N_genes Numeric of number of genes.

   N_genes_in_OGs Numeric of number of genes in orthogroups.

   Perc_genes_in_OGs Numeric of percentage of genes in orthogroups.

   N_ssOGs Numeric of number of species-specific orthogroups.

   N_genes_in_ssOGs Numeric of number of genes in species-specific orthogroups.

   Perc_genes_in_ssOGs Numeric of percentage of genes in species-specific orthogroups.

   Dups Integer with number of duplications per species.
2. **og_overlap** A symmetric data frame of pairwise orthogroup overlap between species.
3. **duplications** A 2-column data frame with node IDs in the first column and number of gene duplications (50% support) in the second column.

Examples

```
stats_dir <- system.file("extdata", package = "cogeqc")
ortho_stats <- read_orthofinder_stats(stats_dir)
```

---

read_orthogroups *Read and parse orthogroups file created by OrthoFinder*

Description

This function converts the orthogroups file named **Orthogroups.tsv** to a parsed data frame.

Usage

```
read_orthogroups(orthogroups_path = NULL)
```

Arguments

orthogroups_path	Path to Orthogroups/Orthogroups.tsv file generated by OrthoFinder.

Value

A 3-column data frame with orthogroups, species IDs and gene IDs, respectively.

Author(s)

Fabricio Almeida-Silva

Examples

```
path <- system.file("extdata", "Orthogroups.tsv.gz", package = "cogeqc")
og <- read_orthogroups(path)
```

run_busco *Run BUSCO assessment of assembly and annotation quality*

Description

Run BUSCO assessment of assembly and annotation quality

Usage

```
run_busco(
  sequence = NULL,
  outlabel = NULL,
  mode = c("genome", "transcriptome", "proteins"),
  lineage = NULL,
  auto_lineage = NULL,
  force = FALSE,
  threads = 1,
  outpath = NULL,
  download_path = tempdir()
)
```

Arguments

sequence	An object of class DNAStringSet/AAStringSet/RNAStringSet or path to FASTA file with the genome, transcriptome, or protein sequences to be analyzed. If there are many FASTA files in a directory, you can input the path to this directory, so BUSCO will be run in all FASTA files inside it.
outlabel	Character with a recognizable short label for analysis directory and files.
mode	Character with BUSCO mode. One of 'genome', 'transcriptome', or 'proteins'.
lineage	Character with name of lineage to be used.
auto_lineage	Character indicating whether BUSCO should determine optimum lineage path automatically. One of 'euk', 'prok', 'all', or NULL. If 'euk', it will determine optimum lineage path on eukaryote tree. If 'prok', it will determine optimum lineage path on non-eukaryote trees. If 'all', it will determine optimum lineage path for all trees. If NULL, it will not automatically determine lineage, and *lineage* must be manually specified. Default: NULL.
force	Logical indicating whether existing runs with the same file names should be overwritten. Default: FALSE.
threads	Numeric with the number of threads/cores to use. Default: 1.
outpath	Path to results directory. If NULL, results will be stored in the current working directory. Default: NULL.
download_path	Path to directory where BUSCO datasets will be stored after downloading. Default: tempdir().

Value

A character vector with the names of subdirectories and files in the results directory.

Examples

```
sequence <- system.file("extdata", "Hse_subset.fa", package = "cogeqc")
download_path <- paste0(tempdir(), "/datasets")
if(busco_is_installed()) {
    run_busco(sequence, outlabel = "Hse", mode = "genome",
              lineage = "burkholderiales_odb10",
              outpath = tempdir(), download_path = download_path)
}
```

---

synnet *Synteny network for Brassica oleraceae, B. napus, and B. rapa*

Description

Synteny network for Brassica oleraceae, B. napus, and B. rapa

Usage

```
data(synnet)
```

Format

A 2-column data frame with the variables **anchor1** and **anchor2**, containing names of loci in anchor 1 and anchor 2, respectively.

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(synnet)
```

---

tree *Species tree for model species*

Description

The data used to create this object was retrieved from Orthofinder's example output for model species, available in https://bioinformatics.plants.ox.ac.uk/davidemms/public_data/.

Usage

```
data(tree)
```

Format

An object of class "phylo" as returned by `treeio::read.tree()`.

References

Emms, D. M., & Kelly, S. (2019). OrthoFinder: phylogenetic orthology inference for comparative genomics. Genome biology, 20(1), 1-14.

Examples

```
data(tree)
```

Index