

# Package ‘cogeqc’

April 10, 2023

**Title** Systematic quality checks on comparative genomics analyses

**Version** 1.2.1

**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 assess: i. genome assembly quality with BUSCOs; 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** 7.2.1

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

**Imports** utils, graphics, stats, methods, reshape2, ggplot2, ggtree, patchwork, igraph, 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\_16

**git\_last\_commit** 32e6bdb

**git\_last\_commit\_date** 2023-03-14

**Date/Publication** 2023-04-10

**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 <[fabricao\\_almeidasilva@hotmail.com](mailto:fabricao_almeidasilva@hotmail.com)>

## R topics documented:

assess_orthogroups . . . . .	3
assess_synnet . . . . .	4
assess_synnet_list . . . . .	5
batch_summary . . . . .	6
busco_is_installed . . . . .	6
calculate_H . . . . .	7
compare_orthogroups . . . . .	8
fit_sft . . . . .	9
interpro_ath . . . . .	9
interpro_bol . . . . .	10
list_busco_datasets . . . . .	11
og . . . . .	11
plot_busco . . . . .	12
plot_duplications . . . . .	12
plot_genes_in_ogs . . . . .	13
plot_og_overlap . . . . .	14
plot_og_sizes . . . . .	14
plot_orthofinder_stats . . . . .	15
plot_species_specific_ogs . . . . .	16
plot_species_tree . . . . .	16
read_busco . . . . .	17
read_orthofinder_stats . . . . .	18
read_orthogroups . . . . .	19
run_busco . . . . .	19
synnet . . . . .	20
tree . . . . .	21

**Index**

**22**

---

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	<i>Assess synteny network based on graph properties</i>
---------------	---

---

### 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	<i>BUSCO summary output for batch mode</i>
---------------	--

---

**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	<i>Check if BUSCO is installed</i>
--------------------	------------------------------------

---

**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	<i>Calculate homogeneity scores for orthogroups</i>
-------------	---

---

## Description

Calculate homogeneity scores for orthogroups

## Usage

```
calculate_H(orthogroup_df, correct_overclustering = 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.

## 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. 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.

## 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\_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)
```

---

**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	<i>Plot percentage of genes in orthogroups for each species</i>
-------------------	---

---

**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\_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 <b>treeio</b> 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 <i>Species_Tree/SpeciesTree_rooted_node_labels.txt</i> . 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	<i>Read and parse BUSCO's summary report</i>
------------	--

---

**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	<i>Read and parse orthogroups file created by OrthoFinder</i>
------------------	---

---

**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	<i>Run BUSCO assessment of assembly and annotation quality</i>
-----------	--

---

**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 <i>lineage</i> 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	<i>Species tree for model species</i>
------	---------------------------------------

---

**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/](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

## \* datasets

- batch\_summary, [6](#)
- interpro\_ath, [9](#)
- interpro\_bol, [10](#)
- og, [11](#)
- synnet, [20](#)
- tree, [21](#)

assess\_orthogroups, [3](#)

assess\_synnet, [4](#)

assess\_synnet\_list, [5](#)

batch\_summary, [6](#)

busco\_is\_installed, [6](#)

calculate\_H, [7](#)

compare\_orthogroups, [8](#)

fit\_sft, [9](#)

interpro\_ath, [9](#)

interpro\_bol, [10](#)

list\_busco\_datasets, [11](#)

og, [11](#)

plot\_busco, [12](#)

plot\_duplications, [12](#)

plot\_genes\_in\_ogs, [13](#)

plot\_og\_overlap, [14](#)

plot\_og\_sizes, [14](#)

plot\_orthofinder\_stats, [15](#)

plot\_species\_specific\_ogs, [16](#)

plot\_species\_tree, [16](#)

read\_busco, [17](#)

read\_orthofinder\_stats, [18](#)

read\_orthogroups, [19](#)

run\_busco, [19](#)

synnet, [20](#)

tree, [21](#)