Supplementary Text S1: Obtaining species trees for Ensembl instances

Fabricio Almeida-Silva[1] ***and Yves Van de Peer***[1]

[1]VIB-UGent Center for Plant Systems Biology, Ghent University, Ghent, Belgium

27 February 2024

Contents

1 Introduction

Here, we will describe the code to obtain a species tree for each Ensembl instance using BUSCO genes.

```
library(here)
## here() starts at /home/faalm/Dropbox/package_benchmarks/doubletrouble_paper
library(tidyverse)
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
library(biomaRt)
library(Herper)
## Loading required package: reticulate
##
## Attaching package: 'Herper'
##
## The following object is masked from 'package:reticulate':
##
##     conda_search
library(taxize)
library(Biostrings)
## Loading required package: BiocGenerics
##
## Attaching package: 'BiocGenerics'
##
## The following objects are masked from 'package:lubridate':
##
##     intersect, setdiff, union
##
## The following objects are masked from 'package:dplyr':
##
##     combine, intersect, setdiff, union
##
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
##
## The following objects are masked from 'package:base':
##
##     anyDuplicated, aperm, append, as.data.frame, basename, cbind,
##     colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
##     get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
##     match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
##     Position, rank, rbind, Reduce, rownames, sapply, setdiff, sort,
##     table, tapply, union, unique, unsplit, which.max, which.min
##
## Loading required package: S4Vectors
## Loading required package: stats4
##
## Attaching package: 'S4Vectors'
##
## The following objects are masked from 'package:lubridate':
##
##     second, second<-
##
## The following objects are masked from 'package:dplyr':
##
##     first, rename
##
## The following object is masked from 'package:tidyr':
##
##     expand
##
## The following object is masked from 'package:utils':
##
##     findMatches
##
## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname
##
## Loading required package: IRanges
##
## Attaching package: 'IRanges'
##
## The following object is masked from 'package:lubridate':
##
##     %within%
##
## The following objects are masked from 'package:dplyr':
##
##     collapse, desc, slice
##
## The following object is masked from 'package:purrr':
##
##     reduce
##
## Loading required package: XVector
##
## Attaching package: 'XVector'
##
## The following object is masked from 'package:purrr':
##
##     compact
```

```
##
## Loading required package: GenomeInfoDb
##
## Attaching package: 'Biostrings'
##
## The following object is masked from 'package:base':
##
##     strsplit
library(cogeqc)

set.seed(123) # for reproducibility
options(timeout = 1e6) # to allow download of big files

source(here("code", "utils.R"))
source(here("code", "utils_busco_phylogeny.R"))
```

2 Summary stats

To start with, let's get the number of species for each instance:

```
# Get number of species in Ensembl Genomes
instances <- c("fungi_mart", "plants_mart", "metazoa_mart", "protists_mart")
nspecies_ensemblgenomes <- unlist(lapply(instances, function(x) {
    return(nrow(listDatasets(useEnsemblGenomes(biomart = x))))
}))

# Get number of species in Ensembl
nspecies_ensembl <- nrow(listDatasets(useEnsembl(biomart = "genes")))

# Combine summary stats onto a data frame
nspecies_all <- data.frame(
    instance = c(gsub("_mart", "", instances), "ensembl"),
    n_genes = c(nspecies_ensemblgenomes, nspecies_ensembl)
)

nspecies_all
##   instance n_genes
## 1    fungi      70
## 2   plants     151
## 3  metazoa     280
## 4 protists      33
## 5  ensembl     214
```

3 Getting species metadata

Now, let's get species metadata for each Ensembl instance.

```r
# Metadata column names
col_names <- c(
    "name", "species", "division", "taxonomy_id", "assembly",
    "assembly_accession", "genebuild", "variation", "microarray", "pan_compara",
    "peptide_compara", "genome_alignments", "other_alignments", "core_db",
    "species_id"
)

to_remove <- c(
    "variation", "microarray", "pan_compara", "peptide_compara",
    "genome_alignments", "other_alignments", "core_db", "species_id"
)

# Ensembl Fungi
metadata_fungi <- read_tsv(
    "http://ftp.ebi.ac.uk/ensemblgenomes/pub/release-57/fungi/species_EnsemblFungi.txt",
    col_names = col_names, skip = 1, col_select = 1:15, show_col_types = FALSE
) |>
    dplyr::filter(!startsWith(core_db, "fungi_")) |>
    dplyr::select(!any_of(to_remove)) |>
    as.data.frame()

metadata_fungi <- cbind(
    metadata_fungi,
    classification(metadata_fungi$taxonomy_id, db = "ncbi") |>
        format_classification()
)

# Ensembl Plants
metadata_plants <- read_tsv(
    "http://ftp.ebi.ac.uk/ensemblgenomes/pub/release-57/plants/species_EnsemblPlants.txt",
    col_names = col_names, skip = 1, col_select = 1:15, show_col_types = FALSE
) |>
    dplyr::filter(species != "triticum_aestivum_kariega") |>
    dplyr::select(!any_of(to_remove)) |>
    as.data.frame()

metadata_plants <- cbind(
    metadata_plants,
    classification(metadata_plants$taxonomy_id, db = "ncbi") |>
        format_classification()
)

# Ensembl Metazoa
metadata_metazoa <- read_tsv(
    "http://ftp.ebi.ac.uk/ensemblgenomes/pub/release-57/metazoa/species_EnsemblMetazoa.txt",
    col_names = col_names, skip = 1, col_select = 1:15, show_col_types = FALSE
) |>
    dplyr::filter(!startsWith(core_db, "metazoa_")) |>
    dplyr::select(!any_of(to_remove)) |>
    as.data.frame()
```

```r
metadata_metazoa <- cbind(
    metadata_metazoa,
    classification(metadata_metazoa$taxonomy_id, db = "ncbi") |>
        format_classification()
)

# Ensembl Protists
metadata_protists <- read_tsv(
    "http://ftp.ebi.ac.uk/ensemblgenomes/pub/release-57/protists/species_EnsemblProtists.txt",
    col_names = col_names, skip = 1, col_select = 1:15, show_col_types = FALSE
) |>
    dplyr::filter(!startsWith(core_db, "protists_")) |>
    dplyr::select(!any_of(to_remove)) |>
    as.data.frame()

metadata_protists <- cbind(
    metadata_protists,
    classification(metadata_protists$taxonomy_id, db = "ncbi") |>
        format_classification()
)

# Ensembl
metadata_ensembl <- read_tsv(
    "https://ftp.ensembl.org/pub/release-110/species_EnsemblVertebrates.txt",
    col_names = col_names, skip = 1, col_select = 1:15, show_col_types = FALSE
) |>
    dplyr::select(!any_of(to_remove)) |>
    as.data.frame()

metadata_ensembl <- cbind(
    metadata_ensembl,
    classification(metadata_ensembl$taxonomy_id, db = "ncbi") |>
        format_classification()
)

# Combining all metadata data frames into a list and saving it
metadata_all <- list(
    fungi = metadata_fungi,
    plants = metadata_plants,
    metazoa = metadata_metazoa,
    protists = metadata_protists,
    ensembl = metadata_ensembl
)

save(
    metadata_all, compress = "xz",
    file = here("products", "result_files", "metadata_all.rda")
)
```

4 BUSCO-guided phylogeny inference

Here, for each Ensembl instance, we infer a species tree using the following workflow:

1. Run BUSCO in protein mode with **cogeqc**, using translated sequences for primary transcripts as input;
2. Get the sequences of the identified complete BUSCOs that are shared across all species;
3. Perform a multiple sequence alignment for each BUSCO gene family.
4. Trim the alignments to remove columns with >50% of gaps.
5. Infer a phylogeny with IQ-TREE2.

To start with, we will use the Bioconductor package **Herper** to create a Conda environment containing BUSCO and all its dependencies. Then, we will use this environment to run BUSCO from the R session.

```r
# Create Conda environment with BUSCO
my_miniconda <- "~/"

conda <- install_CondaTools(
    tools = "busco==5.5.0",
    env = "busco_env",
    pathToMiniConda = my_miniconda
)
```

4.1 Obtaining BUSCO sequences

To obtain sequences for BUSCO genes, we will run BUSCO in protein mode using the R/Bioconductor package **cogeqc**. Then, we will read the sequences for complete, single-copy BUSCOs, and keep only BUSCO genes that are shared by a certain % of the species. Ideally, this cut-off should be 100% of conservation (i.e., the BUSCO gene is found in all species), but it can be relaxed for some clades.

4.1.1 Ensembl Fungi

Here, we will obtain BUSCO genes for Ensembl Fungi species using the following parameters:

1. Lineage: **eukaryota_odb10**
2. Conservation: 100%

```r
# Download whole-genome protein sequences to a directory sequences
busco_fungi <- file.path("~/Downloads/busco_fungi")
seq_fungi <- file.path(busco_fungi, "seqs")
if(!dir.exists(seq_fungi)) { dir.create(seq_fungi, recursive = TRUE) }

download_filtered_proteomes(metadata_all$fungi, "fungi", seq_fungi)

# Run BUSCO in `protein` mode
with_CondaEnv(
    "busco_env",
    cogeqc::run_busco(
        sequence = seq_fungi,
        outlabel = "ensembl_fungi",
        mode = "protein",
        lineage = "eukaryota_odb10",
        outpath = busco_fungi,
        threads = 3,
        download_path = busco_fungi

    ),
    pathToMiniConda = my_miniconda
)

outdir <- file.path(busco_fungi, "ensembl_fungi")
fungi_busco_seqs <- read_busco_sequences(outdir, verbose = TRUE)
```

Saving BUSCO sequences:

```
# Save list of AAStringSet objects with conserved BUSCO sequences
save(
    fungi_busco_seqs, compress = "xz",
    file = here("products", "result_files", "busco_seqs", "fungi_busco_seqs.rda")
)
```

4.1.2 Ensembl Plants

Here, we will use the lineage data set **eukaryota_odb10**. We could use **viridiplantae_odb10**, but there are 3 Rhodophyta species (*Chondrus crispus*, *Galdieria sulphuraria*, and *Cyanidioschyzon merolae*). Because none of the BUSCO genes were shared by all species, we selected genes shared by >60% of the species, and then manually selected BUSCO genes in a way that all species are included. This was required because some taxa (in particular *Triticum* species) had very few BUSCO genes.

```
# Download whole-genome protein sequences to a directory sequences
busco_plants <- file.path("~/Downloads/busco_plants")
seq_plants <- file.path(busco_plants, "seqs")
if(!dir.exists(seq_plants)) { dir.create(seq_plants, recursive = TRUE) }

download_filtered_proteomes(metadata_all$plants, "plants", seq_plants)

# Run BUSCO in `protein` mode
with_CondaEnv(
    "busco_env",
    cogeqc::run_busco(
        sequence = seq_plants,
        outlabel = "ensemblplants",
        mode = "protein",
        lineage = "eukaryota_odb10",
        outpath = busco_plants,
        threads = 4,
        download_path = busco_plants

    ),
    pathToMiniConda = my_miniconda
```

```r
)

# Read sequences of BUSCOs preserved in >=60% of the species
outdir <- file.path(busco_plants, "ensemblplants")
plants_busco_seqs <- read_busco_sequences(outdir, conservation_freq = 0.6)

# Select 10 BUSCO genes so that all species are represented
plants_busco_pav <- get_busco_pav(plants_busco_seqs)

#' The following code was used to manually select BUSCOs in a way that
#' all species are represented

#> ht <- ComplexHeatmap::Heatmap(plants_busco_pav)
#> ht <- ComplexHeatmap::draw(ht)
#> InteractiveComplexHeatmap::htShiny(ht)


# Create a vector of selected BUSCOs
selected_buscos <- c(
    "549762at2759", "1003258at2759", "1247641at2759",
    "1200489at2759", "1398309at2759", "1346432at2759",
    "1266231at2759", "1094121at2759", "1421503at2759",
    "664730at2759", "1405073at2759", "450058at2759",
    "865202at2759", "901894at2759", "1450538at2759",
    "1284731at2759"
)

# Subset sequences to keep only selected BUSCOs
plants_busco_seqs <- plants_busco_seqs[selected_buscos]
```

Saving BUSCO sequences:

```r
# Save list of AAStringSet objects with conserved BUSCO sequences
save(
    plants_busco_seqs, compress = "xz",
    file = here("products", "result_files", "busco_seqs", "plants_busco_seqs.rda")
)
```

4.1.3 Ensembl Protists

Here, we will obtain BUSCO genes for Ensembl Protists species using the following parameters:

1. Lineage: **eukaryota_odb10**
2. Conservation: 100%

```r
# Download whole-genome protein sequences to a directory sequences
busco_protists <- file.path("~/Downloads/busco_protists")
seq_protists <- file.path(busco_protists, "seqs")
if(!dir.exists(seq_protists)) { dir.create(seq_protists, recursive = TRUE) }

download_filtered_proteomes(metadata_all$protists, "protists", seq_protists)
```

```
# Run BUSCO in `protein` mode
with_CondaEnv(
    "busco_env",
    cogeqc::run_busco(
        sequence = seq_protists,
        outlabel = "ensemblprotists",
        mode = "protein",
        lineage = "eukaryota_odb10",
        outpath = busco_protists,
        threads = 4,
        download_path = busco_protists

    ),
    pathToMiniConda = my_miniconda
)

# Read sequences of BUSCOs preserved in >=60% of the species
outdir <- file.path(busco_protists, "ensemblprotists")
protists_busco_seqs <- read_busco_sequences(outdir, verbose = TRUE)
```

Saving BUSCO sequences:

```
# Save list of AAStringSet objects with conserved BUSCO sequences
save(
    protists_busco_seqs, compress = "xz",
    file = here("products", "result_files", "busco_seqs", "protists_busco_seqs.rda")
)
```

4.1.4 Ensembl Metazoa

For the Metazoa instance, we used the *metazoa_odb10* lineage data set.

```
# Download whole-genome protein sequences to a directory sequences
busco_metazoa <- file.path("~/Downloads/busco_metazoa")
seq_metazoa <- file.path(busco_metazoa, "seqs")
if(!dir.exists(seq_metazoa)) { dir.create(seq_metazoa, recursive = TRUE) }

download_filtered_proteomes(metadata_all$metazoa, "metazoa", seq_metazoa)

# Run BUSCO in `protein` mode
with_CondaEnv(
    "busco_env",
    cogeqc::run_busco(
        sequence = seq_metazoa,
        outlabel = "ensemblmetazoa",
        mode = "protein",
        lineage = "metazoa_odb10",
        outpath = busco_metazoa,
        threads = 4,
        download_path = busco_metazoa
    ),
    pathToMiniConda = my_miniconda
)

# Read sequences of BUSCOs preserved in >=60% of the species
outdir <- file.path(busco_metazoa, "ensemblmetazoa")
metazoa_busco_seqs <- read_busco_sequences(outdir, conservation_freq = 0.9)

# Select 10 BUSCO genes so that all species are represented
metazoa_busco_pav <- get_busco_pav(metazoa_busco_seqs)

#' The following code was used to manually select BUSCOs in a way that
#' all species are represented

#> ht <- ComplexHeatmap::Heatmap(metazoa_busco_pav)
#> ht <- ComplexHeatmap::draw(ht)
#> InteractiveComplexHeatmap::htShiny(ht)


# Create a vector of selected BUSCOs
selected_buscos <- c(
    "351226at33208", "135294at33208",
    "517525at33208", "501396at33208",
    "464987at33208", "443518at33208",
    "495100at33208", "335107at33208",
    "454911at33208", "134492at33208"
)

# Subset sequences to keep only selected BUSCOs
metazoa_busco_seqs <- metazoa_busco_seqs[selected_buscos]
```

Saving BUSCO sequences:

```
# Save list of AAStringSet objects with conserved BUSCO sequences
save(
    metazoa_busco_seqs, compress = "xz",
    file = here("products", "result_files", "busco_seqs", "metazoa_busco_seqs.rda")
)
```

4.1.5 Ensembl Vertebrates

Here, because there are 3 non-vertebrate species (*C. elegans*, *D. melanogaster*, and *S. cerevisiae*), we will use the lineage data set **eukaryota_odb10**.

```
# Download whole-genome protein sequences to a directory sequences
busco_vertebrates <- file.path("~/Downloads/busco_vertebrates")
seq_vertebrates <- file.path(busco_vertebrates, "seqs")
if(!dir.exists(seq_vertebrates)) { dir.create(seq_vertebrates, recursive = TRUE) }

download_filtered_proteomes(metadata_all$ensembl, "ensembl", seq_vertebrates)
```

```r
# Run BUSCO in `protein` mode
with_CondaEnv(
    "busco_env",
    cogeqc::run_busco(
        sequence = seq_vertebrates,
        outlabel = "ensemblvertebrates",
        mode = "protein",
        lineage = "eukaryota_odb10",
        outpath = busco_vertebrates,
        threads = 4,
        download_path = busco_vertebrates

    ),
    pathToMiniConda = my_miniconda
)

# Read sequences of BUSCOs preserved in >=90% of the species
outdir <- file.path(busco_vertebrates, "ensemblvertebrates")
vertebrates_busco_seqs <- read_busco_sequences(outdir, conservation_freq = 0.9)

# Select 10 BUSCO genes so that all species are represented
vertebrates_busco_pav <- get_busco_pav(vertebrates_busco_seqs)

#' The following code was used to manually select BUSCOs in a way that
#' all species are represented

#> ht <- ComplexHeatmap::Heatmap(vertebrates_busco_pav)
#> ht <- ComplexHeatmap::draw(ht)
#> InteractiveComplexHeatmap::htShiny(ht)


# Create a vector of selected BUSCOs
selected_buscos <- c(
    "834694at2759", "551907at2759",
    "491869at2759", "1085752at2759",
    "801857at2759", "1398309at2759",
    "176625at2759", "1324510at2759",
    "1377237at2759", "1085752at2759"
)

# Subset sequences to keep only selected BUSCOs
vertebrates_busco_seqs <- vertebrates_busco_seqs[selected_buscos]
```

Saving BUSCO sequences:

```r
# Save list of AAStringSet objects with conserved BUSCO sequences
save(
    vertebrates_busco_seqs, compress = "xz",
    file = here("products", "result_files", "busco_seqs", "vertebrates_busco_seqs")
)
```

4.2 Tree inference from BUSCO genes

Now, we will infer species trees from MSAs for each family, and from a single concatenated MSA (when possible).

4.2.1 Ensembl Fungi

Performing MSA with MAFFT and trimming the alignment:

```
# Perform MSA with MAFFT
aln_fungi <- align_sequences(busco_seqs_fungi, threads = 4)

# Trim alignment to remove columns with >50% of gaps
aln_fungi_trimmed <- lapply(aln_fungi, trim_alignment, max_gap = 0.5)
```

Now, let's infer a species tree using IQ-TREE2.

```
outgroup <- "aphanomyces.astaci,aphanomyces.invadans,globisporangium.ultimum"
trees_fungi <- infer_species_tree(aln_fungi_trimmed, outgroup, threads = 4)
```

Finally, for comparative reasons, we will also infer a single tree from a concatenated multiple sequence alignment.

```
# Concatenate alignments
aln_fungi_conc <- Reduce(xscat, aln_fungi_trimmed)
names(aln_fungi_conc) <- names(aln_fungi_trimmed[[1]])

# Infer tree from concatenated alignment
tree_fungi_conc <- infer_species_tree(
    list(conc = aln_fungi_conc),
    outgroup, threads = 4
)
```

Combining the trees and saving them:

```
# Combine trees
fungi_busco_trees <- c(
    tree_fungi_conc, trees_fungi
)

save(
    fungi_busco_trees, compress = "xz",
    file = here("products", "result_files", "trees", "fungi_busco_trees.rda")
)
```

4.2.2 Ensembl Plants

Here, because no BUSCO gene is present in all species, we will only infer a single tree from concatenated alignments.

```
# Perform MSA with MAFFT
aln_plants <- align_sequences(plants_busco_seqs, threads = 4)
```

```
# Trim alignment to remove columns with >50% of gaps
aln_plants_trimmed <- lapply(aln_plants, trim_alignment, max_gap = 0.5)
```

Finally, let's infer a species tree from a concatenated alignment. As outgroups, we're going to use *Chondrus crispus*, *Galdieria sulphuraria*, and *Cyanidioschyzon merolae*.

```
outgroup <- "chondrus.crispus,galdieria.sulphuraria,cyanidioschyzon.merolae"

# Concatenate alignments
aln_plants_conc <- concatenate_alignments(aln_plants_trimmed)

# Infer tree from concatenated alignment
plants_busco_trees <- infer_species_tree(
    list(conc = aln_plants_conc),
    outgroup, threads = 4
)

# Save tree
save(
    plants_busco_trees, compress = "xz",
    file = here("products", "result_files", "trees", "plants_busco_trees.rda")
)
```

4.2.3 Ensembl Protists

For this instance, two BUSCO genes were conserved across all species, so we will infer trees for each family + a tree from a concatenated alignment.

```
# Perform MSA with MAFFT
aln_protists <- align_sequences(protists_busco_seqs, threads = 4)

# Trim alignment to remove columns with >50% of gaps
aln_protists_trimmed <- lapply(aln_protists, trim_alignment, max_gap = 0.5)
```

Now, let's infer species trees. As outgroup, we will use Fornicata (*Giardia lamblia*) based on this paper.

```
outgroup <- "giardia.lamblia"

# Path 1: a tree per BUSCO gene
protists_trees1 <- infer_species_tree(
    aln_protists_trimmed, outgroup, threads = 4
)

# Path 2: a single tree from a concatenated alignment
protists_trees2 <- infer_species_tree(
    list(conc = concatenate_alignments(aln_protists_trimmed)),
    outgroup, threads = 6
)
```

```r
# Combine trees and save them
protists_busco_trees <- c(protists_trees1, protists_trees2)

save(
    protists_busco_trees, compress = "xz",
    file = here("products", "result_files", "trees", "protists_busco_trees.rda")
)
```

However, even though we specified *Giardia lamblia*, IQ-TREE2 placed it as an ingroup. This suggests that, based on our data (BUSCO sequences), *Giardia lamblia* may not be a good outgroup.

Since protists are not actually a real phylogenetic group (not monophyletic), instead of digging deeper into the real phylogeny of the group and searching for a proper outgroup, we will simply use this phylogeny but acknowledging that it may not be completely accurate.

4.2.4 Ensembl Metazoa

For this instance, two BUSCO genes were conserved across all species, so we will infer trees for each family + a tree from a concatenated alignment.

```r
# Perform MSA with MAFFT
aln_metazoa <- align_sequences(metazoa_busco_seqs, threads = 4)

# Trim alignment to remove columns with >50% of gaps
aln_metazoa_trimmed <- lapply(aln_metazoa, trim_alignment, max_gap = 0.5)
```

Now, let's infer a species tree. As outgroup, we will use the ctenophore *Mnemiopsis leidyi*.

```r
outgroup <- "mnemiopsis.leidyi"

# Get a single tree from a concatenated alignment
metazoa_busco_trees <- infer_species_tree(
    list(conc = concatenate_alignments(aln_metazoa_trimmed)),
    outgroup, threads = 6
)

# Save tree
save(
    metazoa_busco_trees, compress = "xz",
    file = here("products", "result_files", "trees", "metazoa_busco_trees.rda")
)
```

4.2.5 Ensembl Vertebrates

For this instance, no BUSCO gene was conserved in all species. Thus, we will infer a single tree from a concatenated alignment of ten representative BUSCOs.

```r
# Perform MSA with MAFFT
aln_vertebrates <- align_sequences(vertebrates_busco_seqs, threads = 4)

```

```r
# Trim alignment to remove columns with >50% of gaps
aln_vertebrates_trimmed <- lapply(aln_vertebrates, trim_alignment, max_gap = 0.5)
```

Now, let's infer a species tree. As outgroup, we will use the yeast *Saccharomyces cerevisiae*.

```r
outgroup <- "saccharomyces.cerevisiae"

# Get a single tree from a concatenated alignment
vertebrates_busco_trees <- infer_species_tree(
    list(conc = concatenate_alignments(aln_vertebrates_trimmed)),
    outgroup, threads = 6
)

# Save tree
save(
    vertebrates_busco_trees, compress = "xz",
    file = here("products", "result_files", "trees", "vertebrates_busco_trees.rda")
)
```

5 Obtaining BUSCO scores

Finally, since we ran BUSCO to obtain single-copy gene families, we will also use BUSCO's output to explore gene space completeness across species in Ensembl instances.

```r
# Read BUSCO completeness stats
## Ensembl Fungi
fungi_busco_scores <- read_busco(
    "~/Downloads/busco_fungi/ensembl_fungi"
)

## Ensembl Plants
plants_busco_scores <- read_busco(
    "~/Downloads/busco_plants/ensemblplants"
)

## Ensembl Protists
protists_busco_scores <- read_busco(
    "~/Downloads/busco_protists/ensemblprotists"
)

## Ensembl Metazoa
metazoa_busco_scores <- read_busco(
    "~/Downloads/busco_metazoa/ensemblmetazoa"
)

## Ensembl Vertebrates
vertebrates_busco_scores <- read_busco(
    "~/Downloads/busco_vertebrates/ensemblvertebrates"
)
```

```
# Save files
save(
    fungi_busco_scores, compress = "xz",
    file = here(
        "products", "result_files", "busco_scores", "fungi_busco_scores.rda"
    )
)

save(
    plants_busco_scores, compress = "xz",
    file = here(
        "products", "result_files", "busco_scores", "plants_busco_scores.rda"
    )
)

save(
    protists_busco_scores, compress = "xz",
    file = here(
        "products", "result_files", "busco_scores", "protists_busco_scores.rda"
    )
)

save(
    metazoa_busco_scores, compress = "xz",
    file = here(
        "products", "result_files", "busco_scores", "metazoa_busco_scores.rda"
    )
)

save(
    vertebrates_busco_scores, compress = "xz",
    file = here(
        "products", "result_files", "busco_scores", "vertebrates_busco_scores.rda"
    )
)
```

Session info

This document was created under the following conditions:

```
## - Session info --------------------------------------------------------------
##  setting  value
##  version  R version 4.3.2 (2023-10-31)
##  os       Ubuntu 22.04.3 LTS
##  system   x86_64, linux-gnu
##  ui       X11
##  language (EN)
##  collate  en_US.UTF-8
##  ctype    en_US.UTF-8
##  tz       Europe/Brussels
##  date     2024-02-27
##  pandoc   3.1.1 @ /usr/lib/rstudio/resources/app/bin/quarto/bin/tools/ (via rmarkdown)
##
## - Packages -------------------------------------------------------------------
##  package          * version     date (UTC) lib source
##  AnnotationDbi      1.64.1      2023-11-03 [1] Bioconductor
##  ape                5.7-1       2023-03-13 [1] CRAN (R 4.3.2)
##  aplot              0.2.2       2023-10-06 [1] CRAN (R 4.3.2)
##  beeswarm           0.4.0       2021-06-01 [1] CRAN (R 4.3.2)
##  Biobase            2.62.0      2023-10-24 [1] Bioconductor
##  BiocFileCache      2.10.1      2023-10-26 [1] Bioconductor
##  BiocGenerics     * 0.48.1      2023-11-01 [1] Bioconductor
##  BiocManager        1.30.22     2023-08-08 [1] CRAN (R 4.3.2)
##  BiocStyle        * 2.30.0      2023-10-24 [1] Bioconductor
##  biomaRt          * 2.58.2      2024-01-30 [1] Bioconductor 3.18 (R 4.3.2)
##  Biostrings       * 2.70.2      2024-01-28 [1] Bioconductor 3.18 (R 4.3.2)
##  bit                4.0.5       2022-11-15 [1] CRAN (R 4.3.2)
##  bit64              4.0.5       2020-08-30 [1] CRAN (R 4.3.2)
##  bitops             1.0-7       2021-04-24 [1] CRAN (R 4.3.2)
##  blob               1.2.4       2023-03-17 [1] CRAN (R 4.3.2)
##  bold               1.3.0       2023-05-02 [1] CRAN (R 4.3.2)
##  bookdown           0.37        2023-12-01 [1] CRAN (R 4.3.2)
##  cachem             1.0.8       2023-05-01 [1] CRAN (R 4.3.2)
##  cli                3.6.2       2023-12-11 [1] CRAN (R 4.3.2)
##  codetools          0.2-19      2023-02-01 [4] CRAN (R 4.2.2)
##  cogeqc           * 1.6.2       2024-01-14 [1] Bioconductor 3.18 (R 4.3.2)
##  colorspace         2.1-0       2023-01-23 [1] CRAN (R 4.3.2)
##  conditionz         0.1.0       2019-04-24 [1] CRAN (R 4.3.2)
##  crayon             1.5.2       2022-09-29 [1] CRAN (R 4.3.2)
##  crul               1.4.0       2023-05-17 [1] CRAN (R 4.3.2)
##  curl               5.2.0       2023-12-08 [1] CRAN (R 4.3.2)
##  data.table         1.15.0      2024-01-30 [1] CRAN (R 4.3.2)
##  DBI                1.2.1       2024-01-12 [1] CRAN (R 4.3.2)
##  dbplyr             2.4.0       2023-10-26 [1] CRAN (R 4.3.2)
##  digest             0.6.34      2024-01-11 [1] CRAN (R 4.3.2)
##  dplyr            * 1.1.4       2023-11-17 [1] CRAN (R 4.3.2)
##  evaluate           0.23        2023-11-01 [1] CRAN (R 4.3.2)
##  fansi              1.0.6       2023-12-08 [1] CRAN (R 4.3.2)
##  fastmap            1.1.1       2023-02-24 [1] CRAN (R 4.3.2)
##  filelock           1.0.3       2023-12-11 [1] CRAN (R 4.3.2)
##  forcats          * 1.0.0       2023-01-29 [1] CRAN (R 4.3.2)
##  foreach            1.5.2       2022-02-02 [1] CRAN (R 4.3.2)
##  fs                 1.6.3       2023-07-20 [1] CRAN (R 4.3.2)
##  generics           0.1.3       2022-07-05 [1] CRAN (R 4.3.2)
##  GenomeInfoDb     * 1.38.6      2024-02-08 [1] Bioconductor 3.18 (R 4.3.2)
##  GenomeInfoDbData   1.2.11      2023-12-21 [1] Bioconductor
##  ggbeeswarm         0.7.2       2023-04-29 [1] CRAN (R 4.3.2)
##  ggfun              0.1.4       2024-01-19 [1] CRAN (R 4.3.2)
##  ggplot2          * 3.4.4       2023-10-12 [1] CRAN (R 4.3.2)
##  ggplotify          0.1.2       2023-08-09 [1] CRAN (R 4.3.2)
##  ggtree             3.10.0      2023-10-24 [1] Bioconductor
##  glue               1.7.0       2024-01-09 [1] CRAN (R 4.3.2)
##  gridGraphics       0.5-1       2020-12-13 [1] CRAN (R 4.3.2)
##  gtable             0.3.4       2023-08-21 [1] CRAN (R 4.3.2)
##  here             * 1.0.1       2020-12-13 [1] CRAN (R 4.3.2)
##  Herper           * 1.12.0      2023-10-24 [1] Bioconductor
##  hms                1.1.3       2023-03-21 [1] CRAN (R 4.3.2)
##  htmltools          0.5.7       2023-11-03 [1] CRAN (R 4.3.2)
##  httpcode           0.3.0       2020-04-10 [1] CRAN (R 4.3.2)
##  httr               1.4.7       2023-08-15 [1] CRAN (R 4.3.2)
##  igraph             2.0.1.1     2024-01-30 [1] CRAN (R 4.3.2)
##  IRanges          * 2.36.0      2023-10-24 [1] Bioconductor
##  iterators          1.0.14      2022-02-05 [1] CRAN (R 4.3.2)
##  jsonlite           1.8.8       2023-12-04 [1] CRAN (R 4.3.2)
##  KEGGREST           1.42.0      2023-10-24 [1] Bioconductor
##  knitr              1.45        2023-10-30 [1] CRAN (R 4.3.2)
##  lattice            0.22-5      2023-10-24 [4] CRAN (R 4.3.1)
##  lazyeval           0.2.2       2019-03-15 [1] CRAN (R 4.3.2)
##  lifecycle          1.0.4       2023-11-07 [1] CRAN (R 4.3.2)
##  lubridate        * 1.9.3       2023-09-27 [1] CRAN (R 4.3.2)
##  magrittr           2.0.3       2022-03-30 [1] CRAN (R 4.3.2)
##  Matrix             1.6-3       2023-11-14 [4] CRAN (R 4.3.2)
##  memoise            2.0.1       2021-11-26 [1] CRAN (R 4.3.2)
##  munsell            0.5.0       2018-06-12 [1] CRAN (R 4.3.2)
##  nlme               3.1-163     2023-08-09 [4] CRAN (R 4.3.1)
##  patchwork          1.2.0       2024-01-08 [1] CRAN (R 4.3.2)
##  pillar             1.9.0       2023-03-22 [1] CRAN (R 4.3.2)
##  pkgconfig          2.0.3       2019-09-22 [1] CRAN (R 4.3.2)
##  plyr               1.8.9       2023-10-02 [1] CRAN (R 4.3.2)
##  png                0.1-8       2022-11-29 [1] CRAN (R 4.3.2)
##  prettyunits        1.2.0       2023-09-24 [1] CRAN (R 4.3.2)
##  progress           1.2.3       2023-12-06 [1] CRAN (R 4.3.2)
##  purrr            * 1.0.2       2023-08-10 [1] CRAN (R 4.3.2)
##  R6                 2.5.1       2021-08-19 [1] CRAN (R 4.3.2)
##  rappdirs           0.3.3       2021-01-31 [1] CRAN (R 4.3.2)
##  Rcpp               1.0.12      2024-01-09 [1] CRAN (R 4.3.2)
##  RCurl              1.98-1.14   2024-01-09 [1] CRAN (R 4.3.2)
##  readr            * 2.1.5       2024-01-10 [1] CRAN (R 4.3.2)
##  reshape2           1.4.4       2020-04-09 [1] CRAN (R 4.3.2)
##  reticulate       * 1.35.0      2024-01-31 [1] CRAN (R 4.3.2)
##  rjson              0.2.21      2022-01-09 [1] CRAN (R 4.3.2)
##  rlang              1.1.3       2024-01-10 [1] CRAN (R 4.3.2)
##  rmarkdown          2.25        2023-09-18 [1] CRAN (R 4.3.2)
##  rprojroot          2.0.4       2023-11-05 [1] CRAN (R 4.3.2)
##  RSQLite            2.3.5       2024-01-21 [1] CRAN (R 4.3.2)
##  rstudioapi         0.15.0      2023-07-07 [1] CRAN (R 4.3.2)
##  S4Vectors        * 0.40.2      2023-11-23 [1] Bioconductor 3.18 (R 4.3.2)
##  scales             1.3.0       2023-11-28 [1] CRAN (R 4.3.2)
##  sessioninfo        1.2.2       2021-12-06 [1] CRAN (R 4.3.2)
##  stringi            1.8.3       2023-12-11 [1] CRAN (R 4.3.2)
##  stringr          * 1.5.1       2023-11-14 [1] CRAN (R 4.3.2)
##  taxize          * 0.9.100     2022-04-22 [1] CRAN (R 4.3.2)
##  tibble          * 3.2.1       2023-03-20 [1] CRAN (R 4.3.2)
##  tidyr           * 1.3.1       2024-01-24 [1] CRAN (R 4.3.2)
##  tidyselect        1.2.0       2022-10-10 [1] CRAN (R 4.3.2)
##  tidytree          0.4.6       2023-12-12 [1] CRAN (R 4.3.2)
##  tidyverse       * 2.0.0       2023-02-22 [1] CRAN (R 4.3.2)
##  timechange        0.3.0       2024-01-18 [1] CRAN (R 4.3.2)
##  treeio            1.26.0      2023-10-24 [1] Bioconductor
##  tzdb              0.4.0       2023-05-12 [1] CRAN (R 4.3.2)
##  utf8              1.2.4       2023-10-22 [1] CRAN (R 4.3.2)
##  uuid              1.2-0       2024-01-14 [1] CRAN (R 4.3.2)
##  vctrs             0.6.5       2023-12-01 [1] CRAN (R 4.3.2)
##  vipor             0.4.7       2023-12-18 [1] CRAN (R 4.3.2)
##  withr             3.0.0       2024-01-16 [1] CRAN (R 4.3.2)
##  xfun              0.42        2024-02-08 [1] CRAN (R 4.3.2)
##  XML               3.99-0.16.1 2024-01-22 [1] CRAN (R 4.3.2)
##  xml2              1.3.6       2023-12-04 [1] CRAN (R 4.3.2)
##  XVector         * 0.42.0      2023-10-24 [1] Bioconductor
##  yaml              2.3.8       2023-12-11 [1] CRAN (R 4.3.2)
##  yulab.utils       0.1.4       2024-01-28 [1] CRAN (R 4.3.2)
##  zlibbioc          1.48.0      2023-10-24 [1] Bioconductor
##  zoo               1.8-12      2023-04-13 [1] CRAN (R 4.3.2)
##
##  [1] /home/faalm/R/x86_64-pc-linux-gnu-library/4.3
##  [2] /usr/local/lib/R/site-library
##  [3] /usr/lib/R/site-library
##  [4] /usr/lib/R/library
##
## ------------------------------------------------------------------------------
```