Metarhizium ortholog trees

Simeon Rossmann

03/19/2021

Script description

This script was used to parse results of Orthofinder (version 2.5.2) run on the genomes of 11 *Metarhizium* species and 4 fungal outgroup genomes. It is implemented in an RMarkdown document, read more about RMarkdown here.

This code is documentation for an analysis that was conducted as part of the publication "Genomic signatures and insights into host niche adaptation of the entomopathogenic fungus *Metarhizium humberi*" by Iwanicki *et al.*, 2021. Attached packages and their versions are given on the bottom of the document.

Setup The path is specified here and required libraries are loaded. A seed is set so that randomized functions give a reproducible result

```
#CHANGE ME to base path that contains the results folder
path_base <- "/path/to/OrthoFinder/Results_Feb17"
#CHANGE ME to the output directory (same as path_base by default)
path_output <- path_base
#CHANGE ME to the seed you want to use
seed <- 1198872

#knitr options
knitr::opts_chunk$set(echo = TRUE)
#knitr::opts_knit$set(root.dir = path_base)

#load libraries
library(tidyverse)
library(Biostrings)

#libraries for tree generation
library(DECIPHER)
library(ape)
library(ggtree)
library(data.table)
library(phangorn)
library(treeio)

#Setting the chosen seed
set.seed(seed)
```

Orthofinder

Important note: In each genome, gene names were prefixed with genome name, e.g. `AnigerSH23_g1.t1`

This is the log file of the Orthofinder run, documenting version and input files. File paths are anonymized.

```
2021-02-17 12:50:27 : Started OrthoFinder version 2.5.2
Command Line: /path/to/OrthoFinder/orthofinder -f /path/to/genomes

WorkingDirectory_Base: /path/to/output/OrthoFinder/Results_Feb17/WorkingDirectory/

Species used:
0: AnigerSH23.faa
1: BbassianaARSEF28603.faa
2: Fverticillioides76003.faa
3: MacrCQMa1023.faa
4: MalbARSEF1941.faa
5: ManiARSEF5493.faa
6: ManiBRIP53293.faa
7: ManiE63.faa
8: MbruARSEF32973.faa
9: MguiARSEF9773.faa
10: MmajARSEF2973.faa
11: MrilRCEF48713.faa
12: MrobARSEF233.faa
13: TharzianumT67763.faa
14: metarhizium1638.faa

FN_Orthogroups: /path/to/output/OrthoFinder/Results_Feb17/WorkingDirectory/
                    clusters_OrthoFinder_I1.5.txt_id_pairs.txt
WorkingDirectory_Trees: /path/to/output/Results_Feb17/WorkingDirectory/
2021-02-17 13:10:25 : OrthoFinder run completed
```

Parsing gene counts

This chunk parses the table over gene counts per orthogroup and strain "Orthogroups.GeneCount.tsv" into a vector of 100 randomly selected orthogroups that contain exactly 1 gene per genome and a vector of all such genes per genome.

```
#Read in the table
ortho_gene_counts <- read_tsv(file = file.path(path_base, "Orthogroups",
                                               "Orthogroups.GeneCount.tsv"))
# Select rows with one gene for each strain and total == total number of genomes
genomes <- colnames(ortho_gene_counts)
genomes <- genomes[2:(length(genomes)-1)]

single_gene_orthogroups <- ortho_gene_counts %>%
  filter(if_all(genomes, ~ . == 1) &
         Total == length(genomes))

# Save all to one object and sample 100 random groups to a different object
# (change random selection by changing seed in the first chunk)
```

```
all_single_orthogroups <- single_gene_orthogroups$Orthogroup
hundred_random_orthogroups <- sample(single_gene_orthogroups$Orthogroup, 100)
```

Parsing orthogroup genes

This chunk parses the table over genes counts in the orthogroups "Orthogroups.tsv" into a key table to associate gene names with Orthogroups and Strains. Then, the sequence file for the given OGs is read in and the sequences are added to the key table. The table is then split into a list of tables per Strain.

```
#### NO CHANGES NEEDED in this chunk ####
# Read in table and pivot it longer to transform to a longer format
ortho_gene_ids <- read_tsv(file = file.path(path_base, "Orthogroups",
                                            "Orthogroups.tsv")) %>%
  pivot_longer(cols = genomes,
               names_to = "Strain",
               values_to = "Genes")

# Select the gene ids belonging to orthogroups
# CHANGE 'all_single_orthogroups' to 'hundred_random_orthogroups' to use that
# list of OGs instead of all!
gene_ids_of_orthogroups <- ortho_gene_ids %>%
  filter(Orthogroup %in% all_single_orthogroups)

#Read in sequences for the selected orthogroups
## Create empty object to receive sequences
sequences_orthogroups <- AAStringSet()

## Loop over orthogroups and read in the seqs, append to the existing object
for (orthogroup in all_single_orthogroups) {
  sequence_og <- Biostrings::readAAStringSet(
    filepath = file.path(path_base, "Orthogroup_Sequences",
                         paste0(orthogroup, ".fa")))
  sequences_orthogroups <- append(sequences_orthogroups, sequence_og)
}

#Turn sequences into table and add to the table with strains and gene names
seqs_tbl <- tibble::enframe(as.character(sequences_orthogroups),
                            name = "Genes", value = "Seqs") %>%
  unique()

gene_seqs_orthogroups <- gene_ids_of_orthogroups %>%
  left_join(seqs_tbl, by = "Genes") %>% unique()

# split into one table containing all sequences per strain in a list
strain_seqs_orthogroups <- gene_seqs_orthogroups %>%
  group_by(Strain, .drop = FALSE) %>%
  group_split()
```

Concatenating and exporting

This chunk turns the table back into sequence objects, one for the concatenated sequences (also generated in this chunk), one for the single gene sequences, identified by Orthogroup, Strain and gene name. Both are

then exported into FASTA files.

```
####NO CHANGES NEEDED IN THIS CHUNK####
# Operate on the lists per strain and make a sequence object that contains all
# sequences as well as a sequence object that contains all concatenations.
out_seqlists <- AAStringSet()
out_concats <- AAStringSet()
for (tbl_element in strain_seqs_orthogroups) {
  # Extract strain info
  strain <- tbl_element %>%
    select(Strain) %>%
    unique() %>%
    unlist() %>% unname()
  # Extract sequence info and unique identifiers
  out <- tbl_element %>%
    arrange(Orthogroup) %>%
    unite(col="OG_Str_Gene", Orthogroup, Genes)
  # Make table where all sequences per strain are merged
  concat <- tbl_element %>%
    arrange(Orthogroup) %>%
    group_by(Strain) %>%
    mutate(Seq = paste0(Seqs, collapse = "")) %>%
    select(Strain, Seq) %>%
    unique()

  # Make sequence object for all single seqs in table
  seqs_out <- out$Seqs
  names(seqs_out) <- out$OG_Str_Gene
  out_biostrings <- AAStringSet(seqs_out)
  # Append to the common sequence object for all sequences
  out_seqlists <- append(out_seqlists, out_biostrings)
  # Make sequence object from concatenated table and append to common object
  concat_seq <- concat$Seq
  names(concat_seq) <- concat$Strain
  out_concat <- AAStringSet(concat_seq)
  # Append to the common sequence object for concatenated sequences
  out_concats <- append(out_concats, out_concat)
}

# Change name of Mhumberi
genomes <- str_replace(genomes, "metarhizium1638", "Mhumberi1638")
names(out_concats) <- str_replace(names(out_concats),
                                  "metarhizium1638", "Mhumberi1638")
names(out_seqlists) <- str_replace(names(out_seqlists),
                                  "metarhizium1638", "Mhumberi1638")
# Write to files
out_concats
writeXStringSet(out_concats,
                filepath = file.path(
                  path_output, paste0("genes_concatenated_seed_", seed,".fa")),
                format = "fasta")

out_seqlists
writeXStringSet(out_seqlists,
```

```
                filepath = file.path(
                  path_output, paste0("genes_separate_seed_", seed,".fa")),
                format = "fasta")
```

```
#This chunk can be run from a FASTA file of concatenated seqs, to not run all
# the script above. The first chunk loading the libraries must still be run.
# To override/not override from file, uncomment/comment out the line below.
#override_from_file <- "/path/to/genes_concatenated_seed_1198872.fa"

#Define genome names to subset on (here: all starting with capital 'M')
metarh_subset <- genomes[str_detect(genomes, "^M")]

#CHANGE ME to the number of bootstraps, 0 to skip bootstrapping
bootstrapping <- 100

#CHANGE ME to the number of cores that should be used in multithreading
ncores = 16

####NO CHANGES NEEDED IN THIS CHUNK####
if (exists("override_from_file")) {
  out_concats <- readAAStringSet(override_from_file)
}
subset_concats <- out_concats[metarh_subset]

#Function to calculate trees
calculate_tree <- function(tree_seqs){
  alignment_concatenated <- AlignSeqs(AAStringSet(tree_seqs))
    seqs_phang <- phangorn::phyDat(as(alignment_concatenated, "matrix"),
                                   type="AA")
  seqs_dm <- phangorn::dist.ml(seqs_phang)
  seqs_treeNJ <- NJ(seqs_dm)
  seqs_fit = pml(seqs_treeNJ, data=seqs_phang)
  fitLGseqs <- update(seqs_fit, k=4, inv=0.2)
  fitLGseqs <- optim.pml(fitLGseqs, model="LG", optInv=TRUE, optGamma=TRUE,
                rearrangement = "stochastic", control = pml.control(trace = 1L))
  return(fitLGseqs)
}

# Varient of the author's package cuphyr::root_tree in outgroup
# github/simeross/CuPhyR
# Roots an unrooted tree in the most distant outgroup
root_generic_tree <- function(tree.unrooted){
      if (requireNamespace(c("ape", "data.table"), quietly = TRUE)) {
        phylo_tree <- tree.unrooted
        tips <- ape::Ntip(phylo_tree)
        tree_data <- base::cbind(data.table::data.table(phylo_tree$edge),
            data.table::data.table(length = phylo_tree$edge.length))[1:tips,
            ]
        tree_data <- base::cbind(tree_data, data.table::data.table(
          id = phylo_tree$tip.label))
        out_group <- dplyr::slice(tree_data, which.max(length)) %>%
            select(id) %>% as.character()
        new_tree <- ape::root(phylo_tree, outgroup = out_group,
```

```
            resolve.root = TRUE)
        message("Tree successfully rooted.")
    }else {
        stop("The function 'root_tree_in_outgroup' requires the packages
             'ape' and 'data.table' to be installed.
             Please make sure those packages can be loaded.") }
  return(new_tree)
}

# Calculate tree for whole set (may take multiple days)
fitLGseqs_all <- calculate_tree(out_concats)
# Root tree in longest branch (in this case A. niger)
fitLGseqs_all$tree <- root_generic_tree(fitLGseqs_all$tree)
# Write to file
saveRDS(fitLGseqs_all, file = file.path(path_output,
                                        paste0("tree_all_seed_",
                                               seed, ".rds")))
# Perform bootstrapping and write to file (may take multiple hours)
  if (bootstrapping > 0) {
    bs_fitLGseqs_all <- bootstrap.pml(fitLGseqs_all, bs = bootstrapping,
                                      optNni=TRUE, multicore = TRUE,
                                      mc.cores = ncores)
    saveRDS(bs_fitLGseqs_all, file = file.path(path_output,
                                        paste0("bs_all_",bootstrapping,
                                               "_bootstraps_seed_",
                                               seed, ".rds")))
  }

# Calculate tree for subset
fitLGseqs_pc <- calculate_tree(subset_concats)
# write to file
saveRDS(fitLGseqs_pc, file = file.path(path_output,
                                        paste0("tree_metarh_seed_",
                                               seed, ".rds")))
# Perform bootstrapping and write to file (may take multiple hours)
  if (bootstrapping > 0) {
    bs_fitLGseqs_pc <- bootstrap.pml(fitLGseqs_pc, bs = bootstrapping,
                                     optNni=TRUE, multicore = TRUE,
                                      mc.cores = ncores)
    saveRDS(bs_fitLGseqs_pc, file = file.path(path_output,
                                        paste0("bs_metarh_",bootstrapping,
                                               "_bootstraps_seed_",
                                               seed, ".rds")))
  }
```

Plot tree data

This chunk just plots the tree data that was calculated in the chunk above. Dimensions of the "concat_tree" objects can be adjusted by adjusting 'xlim'. The position of the legend can be adjusted by changing 'geom_treescale'.

```
##CHANGE me to the width of the output image (in cm)
pl_w <- 15
##CHANGE me to the height of the output image (in cm)
pl_h <- 10
##CHANGE me to the resolution of the output image (in dpi)
pl_res <- 300

# manual override to plot from files, please comment out for normal use
#fitLGseqs_all <- readRDS("/path/to/tree_all_seed_1198872.rds")
#fitLGseqs_pc <- readRDS("/path/to/tree_metarh_seed_1198872.rds")
#bs_fitLGseqs_all <- readRDS(
#"/path/to/bs_all_100_bootstraps_seed_1198872.rds")
#bs_fitLGseqs_pc <- readRDS(
#"/path/to/bs_metarh_100_bootstraps_seed_1198872.rds")
#seed <- 1198872

###NO CHANGES NEEDED BELOW###
concat_tree_all <- ggtree::ggtree(fitLGseqs_all$tree) + ggtree::geom_tree() +
  ggtree::geom_treescale(x=0.1) + ggtree::geom_tiplab() + xlim(0, 2)

concat_tree_pc <- ggtree::ggtree(fitLGseqs_pc$tree) + ggtree::geom_tree() +
  ggtree::geom_treescale(x=0.01) + ggtree::geom_tiplab() + xlim(0, 0.3)

plotBS(fitLGseqs_all$tree, bs_fitLGseqs_all, p = 0, type="p")
add.scale.bar(0, 0.8, 0.1)
bootstrap_plot_all <- recordPlot()

plotBS(fitLGseqs_pc$tree, bs_fitLGseqs_pc, p = 50, type="p")
add.scale.bar(0.05, 0.8, length = 0.01)
bootstrap_plot_pc <- recordPlot()

concat_tree_all
concat_tree_pc

#Save tree
ggsave(file.path(path_output, paste0("concat_tree_all_seed_", seed,".pdf")),
       concat_tree_all, width = pl_w, height = pl_h, dpi = pl_res)
ggsave(file.path(path_output, paste0("concat_tree_metarh_seed_", seed,".pdf")),
       concat_tree_pc, width = pl_w, height = pl_h, dpi = pl_res)
#Save bootstrap tree
pdf(file.path(path_output, paste0("bootstrap_tree_all_seed_", seed,".pdf")),
    width = pl_w, height = pl_h)
bootstrap_plot_all
dev.off()
pdf(file.path(path_output, paste0("bootstrap_tree_metarh_seed_", seed,".pdf")),
    width = pl_w, height = pl_h)
bootstrap_plot_pc
dev.off()
```

Session record

This chunk prints the packages used in this script and their versions via `sessionInfo()`

```
sessionInfo()
```

```
## R version 4.0.4 (2021-02-15)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 17763)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United Kingdom.1252
## [2] LC_CTYPE=English_United Kingdom.1252
## [3] LC_MONETARY=English_United Kingdom.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United Kingdom.1252
##
## attached base packages:
## [1] stats4    parallel  stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] treeio_1.12.0       phangorn_2.5.5      data.table_1.14.0
##  [4] ggtree_2.2.4        ape_5.4-1           DECIPHER_2.16.1
##  [7] RSQLite_2.2.4       Biostrings_2.56.0   XVector_0.28.0
## [10] IRanges_2.22.2      S4Vectors_0.26.1    BiocGenerics_0.34.0
## [13] forcats_0.5.1       stringr_1.4.0       dplyr_1.0.5
## [16] purrr_0.3.4         readr_1.4.0         tidyr_1.1.3
## [19] tibble_3.1.0        ggplot2_3.3.3       tidyverse_1.3.0
##
## loaded via a namespace (and not attached):
##  [1] nlme_3.1-152        fs_1.5.0            lubridate_1.7.10
##  [4] bit64_4.0.5         httr_1.4.2          tools_4.0.4
##  [7] backports_1.2.1     utf8_1.2.1          R6_2.5.0
## [10] DBI_1.1.1           lazyeval_0.2.2      colorspace_2.0-0
## [13] withr_2.4.1         tidyselect_1.1.0    bit_4.0.4
## [16] compiler_4.0.4      cli_2.3.1           rvest_1.0.0
## [19] xml2_1.3.2          scales_1.1.1        quadprog_1.5-8
## [22] digest_0.6.27       rmarkdown_2.7       pkgconfig_2.0.3
## [25] htmltools_0.5.1.1   dbplyr_2.1.0        fastmap_1.1.0
## [28] rlang_0.4.10        readxl_1.3.1        rstudioapi_0.13
## [31] generics_0.1.0      jsonlite_1.7.2      magrittr_2.0.1
## [34] patchwork_1.1.1     Matrix_1.3-2        Rcpp_1.0.6
## [37] munsell_0.5.0       fansi_0.4.2         lifecycle_1.0.0
## [40] stringi_1.5.3       yaml_2.2.1          zlibbioc_1.34.0
## [43] grid_4.0.4          blob_1.2.1          crayon_1.4.1
## [46] lattice_0.20-41     haven_2.3.1         hms_1.0.0
## [49] knitr_1.31          pillar_1.5.1        igraph_1.2.6
## [52] fastmatch_1.1-0     reprex_1.0.0        glue_1.4.2
## [55] evaluate_0.14       BiocManager_1.30.10 modelr_0.1.8
## [58] vctrs_0.3.6         cellranger_1.1.0    gtable_0.3.0
## [61] assertthat_0.2.1    cachem_1.0.4        xfun_0.22
## [64] broom_0.7.5         tidytree_0.3.3      aplot_0.0.6
## [67] rvcheck_0.1.8       memoise_2.0.0       ellipsis_0.3.1
```