Supplementary to: Developing a non-destructive metabarcoding protocol for detection of pest insects in bulk trap catches

Arthropod reference database builder

J. Batovska, A.M. Piper, I. Valenzuela, J.P. Cunningham & M.J. Blacket

2019/11/28

Introduction

This is the R based reproducible workflow for assembling the taxanomic reference database used in the manuscript "Developing a non-destructive metabarcoding protocol for detection of pest insects in bulk trap catches" by J. Batovska, A.M. Piper, I. Valenzuela, J.P. Cunningham & M.J. Blacket

For this study, 3 seperate primer sets were used to amplify COI, 18S and 12S genes, and therefore a reference database is required for each of these. To form this database, the below script was run seperately for the 3 gene regions, and the fasta files for the complete genes were merged at the end.

Load packages

```
##Load Necessary packages
sapply(c("rentrez","bold", "seqinr","taxonomizr",
         "tidyverse", "data.table","DECIPHER"),
       require, character.only = TRUE)
```

```
##    rentrez       bold     seqinr taxonomizr  tidyverse data.table   DECIPHER
##       TRUE       TRUE       TRUE       TRUE       TRUE       TRUE       TRUE
```

Fetch BOLD sequences for Arthropoda

In order to save memory a list of Arthropod orders on BOLD was used in a loop, rather than just the taxon "Arthropoda". The bold package downloads all loci by default, so a loci input is used to isolate the desired loci

```
# Read in bold orders
bold_taxon <- readLines(con = "Bold_Arthropoda_orders.txt")
bold_loci <- c("COI-5P")
output <- "Arthropoda_COI"

dir.create("reference/bold")

# Download BOLD sequences
for (k in 1:length(bold_taxon)){
  time <- Sys.time() # get time
```

```
  date <- Sys.Date()
  data <- bold_seqspec(taxon = bold_taxon[k])

  possibleError <- tryCatch( if(length(data)!=0){

    # delete old file
    cat(file=paste0("reference/bold/",bold_taxon[k], "_", date,"_BOLD.csv"))

    #Write column headers
    write.table(data[1,],
                file=paste0("reference/bold/",bold_taxon[k], "_", date,"_BOLD.csv"),
                append=T, sep="," , row.names = FALSE)

    #Write data
    for (i in 1:nrow(data)){
      write.table(data[i,], file=paste0("reference/bold/",
                                        bold_taxon[k], "_", date,"_BOLD.csv"),
                  append=T, sep=",", row.names = FALSE, col.names = FALSE)
    }
  } ,
  error=function(e)
    if(inherits(possibleError, "error")) next

  )
  time <- Sys.time() - time
  message(paste("Downloaded ", nrow(data)," sequences and specimen information for ",
                bold_taxon[k], " in ", format(time, digits=2), " from BOLD.", sep=""))
}


# Isolate COI-5P from all BOLD sequences and merge into 1 large fasta
bold_path <- "reference/bold/"
bold_dl <- sort(list.files(bold_path, pattern=".csv", full.names = TRUE) )
length(bold_dl)

## Delete old file
cat(file=paste(file=paste(output, "_BOLD.fa")))

l = 1
possibleError <- 1

for (l in 1:length(bold_dl)){
        time <- Sys.time() # get time
      possibleError <- tryCatch( if (file.size(bold_dl[l]) > 0){

      #Read in bold_seqspec CSV
      data <- read.csv(bold_dl[l], na.strings = c("","NA"))
      prefilt <- nrow(data)
      name <- bold_dl[l] %>%
        str_split_fixed("_", n=2)
      name <- name[[1]] %>%
        str_split_fixed("/", n=2)
```

```
      #Subset to necessary rows & Filter
      data <- data %>%
        subset(select=c("processid", "phylum_name", "class_name",
                        "order_name", "family_name", "genus_name",
                        "species_name","markercode", "nucleotides")) %>%
        na.omit() %>%
        dplyr::filter(grepl(bold_loci, markercode)) %>%
        dplyr::filter(!grepl("sp.", species_name)) %>%
        mutate(domain_name = "Eukaryota")

      #Get sequence names
      bold_seqname <- data %>%
        subset(select=c("processid", "domain_name", "phylum_name",
                        "class_name", "order_name", "family_name",
                        "genus_name", "species_name"))
      bold_seqname <- apply(bold_seqname, 1, paste, collapse=";")
      bold_seqname <- str_replace_all(bold_seqname," ","_")
      data <- as.character(data$nucleotides)

      #Write out fasta
      for (i in 1:length(data)){
        exp <- paste(">", bold_seqname[i], "\n", data[i], "\n", sep="")
        cat(exp, file=paste0("reference/bold/",output,"_BOLD.fa"), append=T)
      }

  time <- Sys.time() - time
  message(paste("Added ",length(bold_seqname)," ",
                name[[2]], " Sequences to fasta in ",
                format(time, digits=2), sep=""))},

  #Error handling
  error=function(e) {
    warning(paste("Error, no data for", bold_loci," in file :", bold_dl[l]))
    }, if(inherits(possibleError, "Error - Empty file")) next)
}
```

Fetch GenBank sequences for Arthropoda

For genbank, the taxon "Arthropoda" was used as the query, as the rentrez package supports chunking of queries

```
ncbi_taxon <- ("Arthropoda")
ncbi_loci <- c("COI", "CO1") ##Search terms to download
output <- "Arthropoda_COI"
maxlength <- 2000

# Set up search term
searchQ <- paste("(",ncbi_taxon, "[ORGN])",
                 " AND (", paste(c(ncbi_loci), collapse=" OR "),
                 ") AND 1:",maxlength ,"[Sequence Length]", sep="")

# Conduct entrez search
search_results <- entrez_search(db = "nuccore", term = searchQ,
```

```
                                retmax=9999999, use_history=TRUE)
message(paste(search_results$count," Sequences to be downloaded"))

destfile <- paste("reference/genbank/",output,"_gb.fa", sep="_")
cat(file = destfile, sep="") # delete old file

i <- 1
start <- 0
time <- Sys.time()

# Calculate number of chunks
chunks <- length(search_results$ids)/10000
if (!is.integer(chunks)){chunks <- as.integer(length(search_results$ids)/10000)+1}

# Download in chunks using efetch
for(i in 1:chunks){
  dl <- entrez_fetch(db="nuccore", web_history=search_results$web_history,
                     rettype="fasta", retmax=10000, retstart= start)

  cat(dl, file=destfile, sep=" ", append=T)
  message("Chunk", i, " of ",chunks, " downloaded\r")
  start <- start + 10000
  Sys.sleep(2.5)

  if (i >= chunks){
    time <- Sys.time() - time
    message(paste("Download complete for: ",
                  search_results$count," Sequences in ",
                  format(time, digits=2), "From Genbank"))
  }
}

# Check if Download worked
if (i < chunks){
  message(paste("Warning: Less sequences than expected, attempt download again"))
}
```

Trim sequences using Geneious

Sequences need to be trimmed to primer regions used in our study. As there are millions of sequences, a de-novo alignment of these is impractical. Instead sequences were mapped to Hemipteran reference sequences (Mitogenome or long rRNA sequence) in Geneious, and regions between primers were extracted.

Retrieve taxonomy for genbank sequences

While BOLD sequences were downloaded alongside their taxonomic annotations, the genbank files did not come with these. Instead, the taxonomy was added using a local copy of the NCBI taxonomy database

Get taxonomy database

Warning: this is a large download and assembles a very large SQL database. Approximately 60GB of HDD space is required.

```
# Download names and nodes
getNamesAndNodes()
getAccession2taxid()

#Convert acc2taxid to SQL
read.accession2taxid(list.files('.','accession2taxid.gz$'),'accessionTaxa.sql')
```

Add taxonomy

```
output <- "Arthropoda_COI"

##Get taxonomy
taxaNodes <- read.nodes('nodes.dmp')
taxaNames <- read.names('names.dmp')

genbank_dl <- read.fasta(file=paste0("reference/genbank/", output, "_gb_trimmed.fasta"),
                         strip.desc = FALSE, as.string = FALSE)

genbank_acc <- getName(genbank_dl)

##If there is extra in the getName ie: _(reversed)
genbank_acc <- genbank_acc %>%
  str_split_fixed( "_", n=2)
taxaId <- accessionToTaxa(genbank_acc[,1], "accessionTaxa.sql")

lineage <- as.data.frame(getTaxonomy(taxaId,taxaNodes,taxaNames))
lineage <- tibble::rownames_to_column(lineage)
taxlineage <- cbind(genbank_acc[,1], lineage)

genbank_taxname <- subset(taxlineage, select=c("genbank_acc[, 1]", "superkingdom",
                                               "phylum", "class", "order",
                                               "family", "genus", "species"))
genbank_taxname <- apply(genbank_taxname, 1, paste, collapse=";")
genbank_taxname <- str_replace_all(genbank_taxname, pattern=" ", replacement="_")

write.fasta(genbank_dl, genbank_taxname,
            paste0("reference/genbank/", output, "_gbtaxonomy.fa"),
            as.string=FALSE, nbchar=100)
rm (list= c("genbank_dl", "taxaNames", "taxaNodes", "lineage", "taxlineage"))
```

Merge Genbank & BOLD datasets

```
output <- "Arthropoda_COI"

# Merge genbank and boldfastas
```

```
bold <- read.fasta(file=paste0("reference/bold/", output, "_bold_trimmed.fasta"),
                   strip.desc = FALSE, as.string = FALSE)
bold_names <- getName(bold)

genbank <- read.fasta(file=paste("reference/genbank/", output, "_gbtaxonomy.fa"),
                      strip.desc = FALSE, as.string = FALSE)
genbank_names <- getName(genbank)

write.fasta(c(bold, genbank), c(bold_names, genbank_names),
            paste0("reference/", output, "_mergeddb.fa"),
            as.string=FALSE, nbchar=100)

# Read in Merged file
merged <- read.fasta(paste0("reference/", output, "_mergeddb.fa"),
                     strip.desc = FALSE, as.string = FALSE)
```

Text filters

The first set of filters were text filters, to remove:

- Sequences where adding taxonomy failed (ie: NA;NA present)
- Sequences of inapropriate length
- Duplicated sequences
- Any insufficiently identified sequences

```
# Filter failed taxonomy - NA;NA
merged_names <- getAnnot(merged)
merged_filtered <- merged[!grepl("NA;NA", merged_names)]
filt_na <- (length(merged) - length(merged_filtered))
message(filt_na, " sequences removed containing NA's")

# Filter sequences to between 200 and 3000bp
merged_filtered <- merged_filtered[which(getLength(merged_filtered) >200)]
merged_filtered <- merged_filtered[which(getLength(merged_filtered)<3000)]
filt_size <- ((length(merged)- length(merged_filtered)) - filt_na)
message(filt_size, " sequences removed outside of length filters")

# Filter duplicate sequences
merged_filtered<- unique(merged_filtered)
filt_uniq <- ((length(merged)- length(merged_filtered)) - filt_na - filt_size)
message(filt_uniq, " duplicate sequences removed")

# Filter any further erroneous or insufficiently identified sequences
keyword_filt <- as.tibble(unlist(getAnnot(merged_filtered)))
keyword_filt <- keyword_filt %>%
  dplyr::filter(!str_detect(value, fixed("sp."))) %>%
  dplyr::filter(!str_detect(value, fixed("aff."))) %>%
  dplyr::filter(!str_detect(value, fixed("nr."))) %>%
  dplyr::filter(!str_detect(value, fixed("cf."))) %>%
  dplyr::filter(!str_detect(value, fixed("nom."))) %>%
  dplyr::filter(!str_detect(value, fixed("nud."))) %>%
  dplyr::filter(!str_detect(value, fixed("environment"))) %>%
```

```
  dplyr::filter(!str_detect(value, fixed("undescribed"))) %>%
  dplyr::filter(!str_detect(value, fixed("unverified"))) %>%
  dplyr::filter(!str_detect(value, fixed("uncultured"))) %>%
  dplyr::filter(!str_detect(value, fixed("unidentif"))) %>%
  dplyr::filter(!str_detect(value, fixed("Bacterium"))) %>%
  dplyr::filter(!str_detect(value, fixed("wolbachia"))) %>%
  dplyr::filter(!str_detect(value, fixed("symbiont"))) %>%
  dplyr::filter(!str_detect(value, fixed("Bacterium"))) %>%
  dplyr::filter(!str_detect(value, fixed("NA"))) %>%
  dplyr::filter(!str_detect(value, fixed("error"))) %>%
  dplyr::filter(!str_detect(value, fixed("CO1_COnsensus")))

rm_keywords <- keyword_filt$value

name_filtered <- merged_filtered[getAnnot(merged_filtered) %in% rm_keywords]
message(paste((length(merged_filtered)- length(name_filtered)), "sequences removed"))

# Write out filtered fasta
name_filtered_annot <- getName(name_filtered)
write.fasta(name_filtered, name_filtered_annot,
            paste0("reference/",output,"_tempfilt1.fa"),
            as.string=FALSE, nbchar=100)

# Removed previous temp file
rem <- paste0("reference/", output, "_mergeddb.fa")
if (file.exists(rem)) file.remove(rem)
```

Filter Wolbachia contaminant sequences

The next stage was to filter out any contaminating wolbachia sequences that can occur in public reference data

Download local wolbachia database

```
# Download All Wolbachia sequences for target loci
# Set up search term
ncbi_loci <- c("COI", "CO1")
wolb_search <- paste("(wolbachia [ORGN]) AND (",
                     paste(c(ncbi_loci), collapse=" OR "),
                     ") AND 1:2000[Sequence Length]", sep="")

# Conduct entrez search
search_results <- entrez_search(db   = "nuccore", term = wolb_search,
                                retmax=9999999, use_history=TRUE)
message(paste(search_results$count," Sequences to be downloaded"))

destfile <- paste0("reference/",output,"_wolbachia_loci.fasta")
cat(file = destfile, sep="") # delete old file

i <- 1
start <- 0
```

```
time <- Sys.time() # get time

# Calculate chunks
chunks <- length(search_results$ids)/10000
if (!is.integer(chunks)){chunks <- as.integer(length(search_results$ids)/10000)+1}

# Download using efetch
for(i in 1:chunks){
  dl <- entrez_fetch(db="nuccore", web_history= search_results$web_history,
                     rettype="fasta", retmax=10000, retstart= start)

  cat(dl, file= destfile, sep=" ", append=T)
  message("Chunk", i, " of ",chunks, " downloaded\r")
  start <- start + 10000
  Sys.sleep(2.5)

  if (i >= chunks){
    time <- Sys.time() - time
    message(paste("Download complete for: ", search_results$count,
                  " Sequences in ",  format(time, digits=2), "From Genbank"))
  }
}

# Check if Download worked
if (i < chunks){
  message(paste("Warning: Less sequences than expected, attempt download again"))
}

# Read in wolbachia
wolb <- read.fasta(paste0("reference/",output,"_wolbachia_loci.fasta"),
                           strip.desc = FALSE, as.string = FALSE)
wolb <- unique(wolb)
names_wolb <- getName(wolb)
write.fasta(wolb, names_wolb,
            paste0("reference/",output,"_wolbachia_loci.fasta"))
```

Blast against wolbachia

The wolbachia sequences were turned into a BLAST DB and all insecta sequences were queried against it to identify any contaminant wolbachia sequences

```
# BASH
makeblastdb -in Arthropoda_COI_wolbachia_loci.fasta -parse_seqids -dbtype nucl
blastn -db wolbachia_loci.fasta -query Arthropoda_COI_tempfilt1.fa \
-out wolbachia_out.csv -outfmt 6 -perc_identity 95
```

Remove wolbachia matches from reference database

Any sequences matching wolbachia were then filtered from the sequence files

```
filtered_1 <- read.fasta(paste0("reference/",output,"_tempfilt1.fa"),
                         strip.desc = FALSE, as.string = FALSE)

# Read in blast results
wolbachia_filt <- fread("wolbachia_out.csv")

wolb <- wolbachia_filt$V1
wolb <- paste0(">", wolb)

# Remove matches to wolbachia
wolb_filtered <- filtered_1[!getAnnot(filtered_1) %in% wolb]
message(paste((length(filtered_1) - length(wolb_filtered)), "sequences removed"))
names_wolb_filtered <- getName(wolb_filtered)

# Write out wolbachia filtered
write.fasta(wolb_filtered, names_wolb_filtered,
            paste0("reference/",output,"_tempfilt2.fa"))

# Remove previous temp file
rem <- paste0("reference/",output,"_tempfilt1.fa")
if (file.exists(rem)) file.remove(rem)
```

Filter taxanomically mislabelled sequences

The third filter applied was to remove any taxonomically mislabelled sequnces. This was achieved by clustering the sequences at 99% and flagging any clusters that contained more than one order

Cluster sequences

```
#BASH
sumaclust_v1.0.31/sumaclust -t 0.99 Arthropoda_COI_tempfilt2.fa > clustered.fa
```

Flag clusters

Following clustering, the output was fed back into R to identify clusters containing putatively mislabelled sequences.

```
#read in Sumaclust clustered file & get headers
clustered <- read.fasta("clustered.fa",strip.desc = FALSE, as.string = FALSE)
clust_head <- getAnnot(clustered)

#clean up table
clust_split <- str_split_fixed(clust_head, ";", n=12)
clust_split <- as.data.table(clust_split)

clust_sub <- clust_split %>%
  select(c("V1", "V2", "V3", "V4", "V5", "V6", "V7", "V8", "V10")) %>%
  separate(V10, c("waste", "accession"), sep="=", extra="merge" ) %>%
  select(c("V1", "V2", "V3", "V4", "V5", "V6", "V7", "V8", "accession"))
```

```
colnames(clust_sub) <- c("seq_acc", "Domain","Phylum","Class",
                         "Order","Family","Genus",
                         "Genus_species", "clust_acc")

# Subset missanotated sequences
mis_annot <- clust_sub %>%
  group_by_(.dots = names(clust_sub)[9]) %>%
  filter(n_distinct(Order) > 1)
print(nrow(mis_annot))

# Write out putative misannotated seqs
write.csv(mis_annot, file=paste0("reference/",output,"_mis_annot.csv"))
```

Remove misannotated sequences

The flagged clusters were then manually explored using a blast against the NCBI nucleotide databse and the tree functionality, and the ID's of putatively misannotated sequences entered in the "remove_seq.txt" file for removal from dataset.

```
# Read in file of seqs to remove
remove_seq <- readLines(con = "reference/remove_seq.txt")
pre_cluster<- read.fasta(paste0("reference/",output,"_tempfilt2.fa"),
                         strip.desc = FALSE, as.string = FALSE)

# Filter seqs
pre_cluster_annot <- getAnnot(pre_cluster)
remove_rows <- str_split_fixed(pre_cluster_annot, ";", n=8)
remove_rows <- remove_rows[,1]
mis_filtered <- pre_cluster[!remove_rows %in% remove_seq]
message(paste((length(pre_cluster) - length(mis_filtered)),
              "mis-annotated sequences removed"))

# Write out filtered seqs
write.fasta(mis_filtered, getName(mis_filtered),
            paste0("reference/", output, "tempfilt3.fa"),
            as.string=FALSE, nbchar=5000)
```

Create seperate species and genus fastas

```
# Create Genus Fasta
rdp_genus <- read.fasta(paste0(output,"_tempfilt3.fa"),
                        strip.desc = FALSE, as.string = FALSE)

all_names <- getName(rdp_genus)
names_genus <- str_split_fixed(all_names, ";", n=8) %>%
  as_tibble() %>%
  select(c("V2","V3","V4","V5","V6","V7"))

# Collapse names
names_genus <- apply(names_genus, 1, paste, collapse=";")
```

```
# Add bold_loci to first level of taxonomy
names_genus <- paste0(bold_loci,"_",names_genus)

# Merge in our sequences - Change filepath for different loci
rdp_inhouse <- read.fasta("COI_inhouse.fa",strip.desc = FALSE, as.string = FALSE)
names_rdp_inhouse <- getName(rdp_inhouse)
names_inhouse_genus <- str_split_fixed(names_rdp_inhouse, ";", n=8)
names_inhouse_genus <- as.data.table(names_inhouse_genus)
names_inhouse_genus <- subset(names_inhouse_genus,
                              select=c("V2","V3","V4","V5","V6","V7"))

# Collapse names
names_inhouse_genus <- apply(names_inhouse_genus, 1, paste, collapse=";")

# Add loci to first level of taxonomy
names_inhouse_genus <- paste0(bold_loci,"_",names_inhouse_genus)

# Save final genus level merged reference DB
write.fasta(c(rdp_genus, rdp_inhouse), c(names_genus,names_inhouse_genus),
            paste0(output, "_rdp_genus.fa"), as.string=FALSE, nbchar=5000)

# Create species Fasta

names_species <- str_split_fixed(all_names, ";", n=8)
names_species <- as.data.table(names_species)
names_species <- subset(names_species, select=c("V1", "V8")) %>%
  separate(V8, c("genus","species"), sep="_", extra="merge" )

names_species <- apply(names_species, 1, paste, collapse=" ")

names_inhouse_species <- str_split_fixed(names_rdp_inhouse, ";", n=8)
names_inhouse_species <- as.data.table(names_inhouse_species)
names_inhouse_species <- subset(names_inhouse_species, select=c("V1", "V8")) %>%
  separate(V8, c("genus","species"), sep="_", extra="merge" )
names_inhouse_species <- apply(names_inhouse_species, 1, paste, collapse=" ")

# Save final species level merged reference DB
write.fasta(c(rdp_genus, rdp_inhouse), c(names_species,names_inhouse_species),
            paste0(output, "_rdp_species.fa"), as.string=FALSE, nbchar=5000)

# Remove previous temp file
rem <- paste0("reference/",output,"_tempfilt3.fa")
if (file.exists(rem)) file.remove(rem)
```

Merge 3 genes together into final DB

All above steps were conducted for the COI, 18S and 12S genes targetted in this study, and then fasta files were merged into one final database for Kingdom to genus classification, and a seperate for exact matching to species

```
# Merge Genus level classifier
genus_COI <- read.fasta("Arthropoda_COI_rdp_genus.fa",
```

```
                        strip.desc = FALSE, as.string = FALSE)
names_genus_COI <- getName(genus_COI)
genus_18S <- read.fasta("Arthropoda_18S_rdp_genus.fa",
                        strip.desc = FALSE, as.string = FALSE)
names_genus_18s <- getName(genus_18S)
genus_12S <- read.fasta("Arthropoda_12S_rdp_genus.fa",
                        strip.desc = FALSE, as.string = FALSE)
names_genus_12s <- getName(genus_12S)

write.fasta(c(genus_COI, genus_18S,genus_12S ),
            c(names_genus_COI,names_genus_18s,names_genus_12s),
            "merged_arthropoda_rdp_genus.fa", as.string=FALSE, nbchar=5000)

# Remove _ after loci
genus_merged <- read.fasta("merged_arthropoda_rdp_genus.fa",
                           strip.desc = FALSE, as.string = FALSE)
names_all <- getName(genus_merged)
names_all <- str_replace_all(names_all, "_", "-")
names_all <- str_replace_all(names_all, "-5P", "")
write.fasta(genus_merged,names_all, "merged_arthropoda_rdp_genus.fa",
            as.string=FALSE, nbchar=5000)

# Merge Species level classifier
species_COI <- read.fasta("Arthropoda_COI_rdp_species.fa",
                          strip.desc = TRUE, as.string = FALSE)
names_species_COI <- getAnnot(species_COI)
species_18S <- read.fasta("Arthropoda_18S_rdp_species.fa",
                          strip.desc = TRUE, as.string = FALSE)
names_species_18s <- getAnnot(species_18S)
species_12S <- read.fasta("Arthropoda_12S_rdp_species.fa",
                          strip.desc = TRUE, as.string = FALSE)
names_species_12s <- getAnnot(species_12S)

# Write merged species level reference fasta
write.fasta(c(species_COI, species_18S,species_12S ),
            c(names_species_COI,names_species_18s,names_species_12s),
            "merged_arthropoda_rdp_species.fa",
            as.string=FALSE, nbchar=5000)
```

Session info

```
sessionInfo()
```

```
## R version 3.6.1 (2019-07-05)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 17763)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_Australia.1252  LC_CTYPE=English_Australia.1252
```

```
## [3] LC_MONETARY=English_Australia.1252 LC_NUMERIC=C
## [5] LC_TIME=English_Australia.1252
##
## attached base packages:
## [1] stats4    parallel  stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] DECIPHER_2.14.0     RSQLite_2.1.2       Biostrings_2.54.0
##  [4] XVector_0.26.0      IRanges_2.20.0      S4Vectors_0.24.0
##  [7] BiocGenerics_0.32.0 data.table_1.12.6   forcats_0.4.0
## [10] stringr_1.4.0       dplyr_0.8.3         purrr_0.3.3
## [13] readr_1.3.1         tidyr_1.0.0         tibble_2.1.3
## [16] ggplot2_3.2.1       tidyverse_1.2.1     taxonomizr_0.5.3
## [19] seqinr_3.6-1        bold_0.9.0          rentrez_1.2.2
## [22] knitr_1.26
##
## loaded via a namespace (and not attached):
##  [1] httr_1.4.1       bit64_0.9-7      jsonlite_1.6     modelr_0.1.5
##  [5] assertthat_0.2.1 blob_1.2.0       cellranger_1.1.0 yaml_2.2.0
##  [9] pillar_1.4.2     backports_1.1.5  lattice_0.20-38  glue_1.3.1
## [13] digest_0.6.22    rvest_0.3.5      colorspace_1.4-1 htmltools_0.4.0
## [17] plyr_1.8.4       XML_3.98-1.20    pkgconfig_2.0.3  httpcode_0.2.0
## [21] broom_0.5.2      haven_2.1.1      zlibbioc_1.32.0  scales_1.0.0
## [25] generics_0.0.2   withr_2.1.2      lazyeval_0.2.2   cli_1.1.0
## [29] magrittr_1.5     crayon_1.3.4     readxl_1.3.1     memoise_1.1.0
## [33] evaluate_0.14    nlme_3.1-141     MASS_7.3-51.4    xml2_1.2.2
## [37] tools_3.6.1      hms_0.5.2        lifecycle_0.1.0  munsell_0.5.0
## [41] ade4_1.7-13      compiler_3.6.1   rlang_0.4.1      grid_3.6.1
## [45] rstudioapi_0.10  rmarkdown_1.17   gtable_0.3.0     DBI_1.0.0
## [49] reshape_0.8.8    curl_4.2         R6_2.4.1         lubridate_1.7.4
## [53] bit_1.1-14       zeallot_0.1.0    stringi_1.4.3    crul_0.9.0
## [57] Rcpp_1.0.2       vctrs_0.2.0      tidyselect_0.2.5 xfun_0.11
```