DADA2 with ITS sequences

Christine Hawkes

2/2/2022

Contents

References:
DADA2 ITS Tutorial
Data reference: Bakker (2018). A fungal mock community control for amplicon sequencing experiments
Schoch et al. (2012). Nuclear ribosomal internal transcribed spacer (ITS) region as a universal DNA barcode marker for Fungi
Pauvert et al.(2019).Bioinformatics matters: The accuracy of plant and soil fungal community data is highly dependent on the metabarcoding pipeline
UNITE ITS database

Setup

Install and load packages

```
#Install
#library(BiocManager);packageVersion("BiocManager")
#BiocManager::install("FastqCleaner")
#BiocManager::install("ShortRead")
#library(devtools)
#install_github("tobiasgf/lulu")

#load libraries
library(tidyverse);packageVersion("tidyverse")
```

```
## [1] '1.3.1'
```

```
library(ShortRead);packageVersion("ShortRead")
```

```
## [1] '1.52.0'
```

```
library(Biostrings);packageVersion("Biostrings")
```

```
## [1] '2.62.0'
```

```
library(dada2);packageVersion("dada2")
```

```
## [1] '1.22.0'
```

```
library(FastqCleaner);packageVersion("FastqCleaner")
```

```
## [1] '1.12.0'
```

```
library(lulu);packageVersion("lulu")
```

```
## [1] '0.1.0'
```

```
#library(phyloseq);packageVersion("phyloseq")
#library(DESeq2);packageVersion("DESeq2")
```

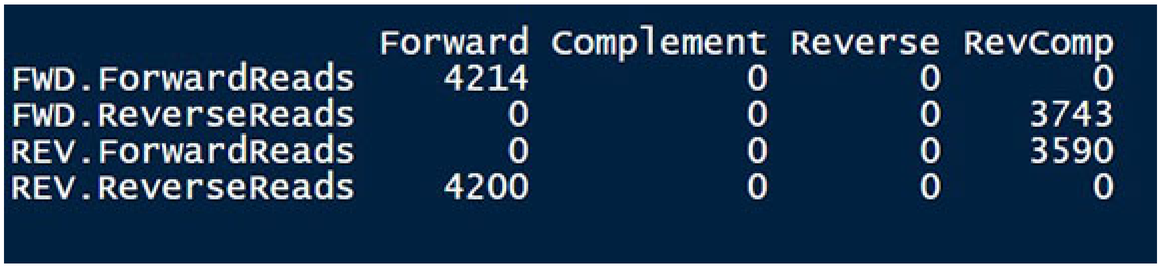
DADA 2 ITS tutorial

The DADA2 ITS Tutorial uses sequences from Data reference: Bakker (2018). A fungal mock community control for amplicon sequencing experiments. The tutorial links to Bioproject PRJNA377530 for these sequences on the European Nucleotide Archive.

Download sequences for the DADA2 ITS Tutorial at: https://drive.google.com/file/d/1745BQxXLaatCGol16zMaZqbYEAt4B view?usp=sharing, not from the link in the tutorial. Move the folder to your desired location and unzip. Make sure you can correctly identify the path name to the reads.

Follow the DADA2 ITS tutorial until you get to the Remove Primers step. For that section, follow the code below.
A note for Windows users: Set multithread=FALSE where this variable appears.

You should end up with the following primer reads.



```
                 Forward Complement Reverse RevComp
FWD.ForwardReads    4214          0       0       0
FWD.ReverseReads       0          0       0    3743
REV.ForwardReads       0          0       0    3590
REV.ReverseReads    4200          0       0       0
```

Remove primers

In the previous step, note that there are ~8400 cases with forward primer sequences and ~7300 cases with reverse complements in the tutorial dataset. These need to be removed before continuing.

We will be using FastqCleaner, not cutadapt. Cudadapt is used for primer removal in the DADA2 ITS tutorial, but we are using this R package today to avoid downloading extra software due to compatibility issues with Windows. For your own analyses, other software may be preferred. Such software includes:

- BBDuk
- Cutadapt
- Trimmomatic

Load sequences into R for FastqClean

```
fwdFiltSeqs <- purrr::map(fnFs.filtN, ~readFastq(.x))
revFiltSeqs <- purrr::map(fnRs.filtN, ~readFastq(.x))
```

Trim the primers with FastqCleaner + Biostrings
There is a bug in `adapter_filter()` function in the current version of FastqCleaner. This bug causes the function to be unable to trim the "left" (5') primers, so we will use a Biostrings function to trim primers from the left. We will still use the `adapter_filter()` function to trim primers from the right, because its matching algorithm to detect the primers within the sequences appears to be more sensitive.

```
# trim the forward primer from the left
# then trim the reverse primer from the right
```

```
fwdAdapterFilters <- purrr::map(fwdFiltSeqs, ~Biostrings::trimLRPatterns(
                                Lpattern=FWD, subject=.x, Lfixed = FALSE,
                                max.Lmismatch = 0.2, with.Lindels = TRUE)) %>%
                     purrr::map(., ~FastqCleaner::adapter_filter(.x,
                                   Rpattern = rc(REV), anchored=FALSE))

# trim the reverse primer from the left
# then trim the forward primer from the right

revAdapterFilters <- purrr::map(revFiltSeqs, ~Biostrings::trimLRPatterns(
                                Lpattern=REV, subject=.x, Lfixed = FALSE,
                                max.Lmismatch = 0.2, with.Lindels = TRUE)) %>%
                     purrr::map(., ~FastqCleaner::adapter_filter(.x,
                                  Rpattern = rc(FWD), anchored=FALSE))
```

Trim the primers with FastqCleaner (not today!)
The bug in the `adapter_filter()` function should be fixed in a few weeks, so in the future, if the bug is fixed you should be able to trim the primers from the left as well, using the `Lpattern =` variable in `adapter_filter()` function as shown below.

```
# trim the FWD primer from the left min_match_flank=7, min_match_flank=7,
# then trim the reverse complement of the reverse primer from the right

fwdAdapterFilters <- purrr::map(fwdFiltSeqs,
                              ~FastqCleaner::adapter_filter(.x,
                               Lpattern = FWD, anchored=FALSE)) %>%
                     purrr::map(., ~FastqCleaner::adapter_filter(.x,
                               Rpattern=rc(REV), anchored = FALSE))

# trim the reverse primer from the left of the reverse read
# then trim the reverse complement of the forward primer from the right
revAdapterFilters <- purrr::map(revFiltSeqs,
                              ~FastqCleaner::adapter_filter(.x,
                               Lpattern = REV, anchored=FALSE)) %>%
                     purrr::map(., ~FastqCleaner::adapter_filter(.x,
                               Rpattern=rc(FWD), anchored=FALSE))
```

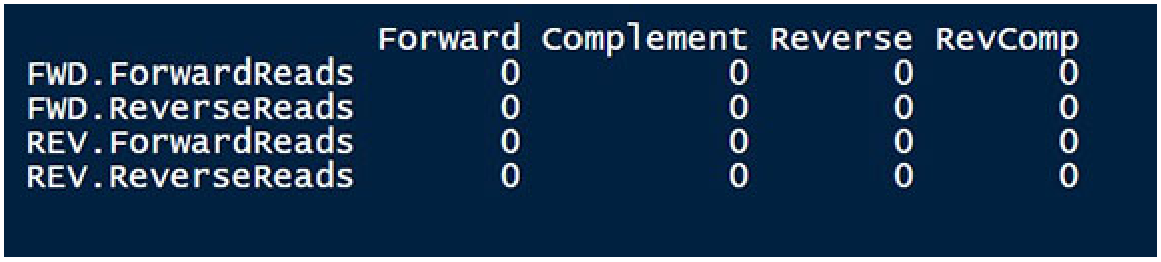
Create a new directory and save your trimmed fastq files

```
path.cut <- file.path(path, "cut") #error "mode is not a"? delete prior "cut" folder
if(!dir.exists(path.cut)) dir.create(path.cut)
fnFs.cut <- file.path(path.cut, basename(fnFs))
fnRs.cut <- file.path(path.cut, basename(fnRs))

purrr::map2(fwdAdapterFilters, fnFs.cut, ~writeFastq(.x, file = .y))
purrr::map2(revAdapterFilters, fnRs.cut, ~writeFastq(.x, file = .y))
```

Check that primer sequences were removed. There should be zero primer sequences identified.

```
rbind(FWD.ForwardReads = sapply(FWD.orients, primerHits, fn = fnFs.cut[[1]]),
      FWD.ReverseReads = sapply(FWD.orients, primerHits, fn = fnRs.cut[[1]]),
      REV.ForwardReads = sapply(REV.orients, primerHits, fn = fnFs.cut[[1]]),
      REV.ReverseReads = sapply(REV.orients, primerHits, fn = fnRs.cut[[1]]))
```


```
                  Forward Complement Reverse RevComp
FWD.ForwardReads        0          0       0       0
FWD.ReverseReads        0          0       0       0
REV.ForwardReads        0          0       0       0
REV.ReverseReads        0          0       0       0
```

Load the trimmed reads
The reads are primer free! Read in the names of the files with trimmed sequences and get a list of sample names.

```r
# Forward and reverse fastq filenames have the format:
cutFs <- sort(list.files(path.cut, pattern = "_1.fastq.gz", full.names = TRUE))
cutRs <- sort(list.files(path.cut, pattern = "_2.fastq.gz", full.names = TRUE))

# Extract sample names, assuming filenames have format:
get.sample.name <- function(fname) strsplit(basename(fname), "_")[[1]][1]
sample.names <- unname(sapply(cutFs, get.sample.name))
head(sample.names)
```

Continue the DADA2 Tutorial.
Stop before the Assign Taxonomy step - we will not run that today. Instead, we will run LULU curation.

LULU

LULU github page here for your reference

Save ASVs

Save ASVs in a fasta file for input into blastn.

```r
# create a DNA string set object to input into the writeXStringSet() command
ASVs <- colnames(seqtab.nochim) %>%
  DNAStringSet()
# set names of sequences for fasta file
names(ASVs) <- colnames(seqtab.nochim)

# save the file -- CHANGE FILE PATH to your desired location
writeXStringSet(ASVs, "~/ASVs.fasta", format="fasta")
```

The line `names(ASVs) <- colnames(seqtab.nochim)` sets the names of the sequences as the ASV sequences themselves. The ASVs are labeled as the ASV sequences in the ASV table, so that is why we have named as such here.

Make BLAST database

Open Powershell (Windows) or Terminal (Mac) and run blastn on ASVs as described in LULU tutorial
Note that the use of sequences as ASV names may produce a blastn message `FASTA-Reader: Title ends with at least 20 valid nucleotide characters. Was the sequence accidentally put in the title line?`, but blastn should still run despite this warning. If it does not, you may need to delete `-parse_seqids` from the `makeblastdb` command.

LULU Curation

```
# note for lulu - transposed orientation required for otu table and class dataframe not matrix
otutab <- as.data.frame(t(seqtab.nochim))

# import the match list - CHANGE FILE PATH to your desired location
matchlist <- read.table("~/match_list.txt", header=FALSE,as.is=TRUE, stringsAsFactors=FALSE)
curated_result <- lulu(otutab, matchlist)
curatedOTU <- curated_result$curated_table
```

If you wished to perform the taxonomic assignment, you can do it now with the curated ASV table from LULU. Performing the taxonomic assignment before the LULU curation will result in extra ASVs that were discarded by LULU in your taxa table. *Note: The taxonomic assignment is very computationally expensive. If you do this, be sure to save all of your necessary tables before running taxonomic assignment in case R crashes so that you don't have to rerun everything.*

Code for integrating LULU with original ps object

You do not need to run this code today.

```
#save lulu results
ASVlulu_discards <- curated_result$discarded_otus
write.csv(ASVlulu_discards, "ASVlulu_discards.csv")

#Prune taxa in phyloseq object based on lulu results
#Open discarded taxa file from lulu and convert from data.frame to vector
#note: alternatively, could use the retained file to subset
badTaxa <- read.csv(file="ASVlulu_discards.csv",
                    row.names=1, header=TRUE, sep=",")
badTaxa_v <- badTaxa$x

#prune original ps object by discarded taxa and rename to new ps object
ps
allTaxa <- taxa_names(ps)
allTaxa <- allTaxa[!(allTaxa %in% badTaxa_v)]
ps_lulu <- prune_taxa(allTaxa, ps)
ps_lulu
```

Code for converting sequences to ASV names

You do not need to run this code today.

```r
# For use at the end of the DADA2 tutorial, after lulu and taxonomic assignments
# Convert the full sequences as ASV names to "ASV1, ASV2, ..." for convenience
# To do this, When creating the phyloseq object, add a DNAStringSet object
# (accessed by the refseq function)

sequences <- Biostrings::DNAStringSet(taxa_names(ps))
names(sequences) <- taxa_names(ps)
# add to the ps object
ps <- merge_phyloseq(ps, sequences)
# rename the ASVs
taxa_names(ps) <- paste0("ASV", seq(ntaxa(ps)))
ps
```

Helpful hint for knitting!

You do not need to knit the file - just upload the raw Rmd to GitHub. However if you want to knit this code in the future, you can save necessary objects as an Rdata file and comment out or set `eval=FALSE` in chunks to avoid having to run DADA2, primer removal, taxanomic assignment, and lulu again when knitting. You can also save any necessary tables as csv files and then load in the tables in order to complete the exercises. Below is an example for .RData.

```r
#save Rdata - CHANGE FILE PATH to your desired location
save(seqtab.nochim, curated_result, file="~/ITS_outputs.Rdata")

#load objects into R - CHANGE FILE PATH to your desired location
load("~/ITS_outputs.Rdata")
```

Coding Exercises

1. Submit an unknitted R markdown document for the dada2 ITS + LULU pipeline that you used in class
2. Run the `decontam` tutorial and submit an unknitted R markdown document to GitHub https://benjjneb.github.io/decontam/vignettes/decontam_intro.html

- `decontam` was developed by Ben Callahan's lab to address contaminants in sequence data
- This requires that you have DNA quantitation for each sample and negative controls
- To install, use `BiocManager::install("decontam")`

Session Info

```r
sessionInfo()
```

```
## R version 4.1.2 (2021-11-01)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19042)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats4    stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] jpeg_0.1-9                  lulu_0.1.0
##  [3] FastqCleaner_1.12.0         dada2_1.22.0
##  [5] Rcpp_1.0.8                  ShortRead_1.52.0
##  [7] GenomicAlignments_1.30.0    SummarizedExperiment_1.24.0
##  [9] Biobase_2.54.0              MatrixGenerics_1.6.0
## [11] matrixStats_0.61.0          Rsamtools_2.10.0
## [13] GenomicRanges_1.46.1        Biostrings_2.62.0
## [15] GenomeInfoDb_1.30.0         XVector_0.34.0
## [17] IRanges_2.28.0              S4Vectors_0.32.3
## [19] BiocParallel_1.28.3         BiocGenerics_0.40.0
## [21] forcats_0.5.1               stringr_1.4.0
## [23] dplyr_1.0.7                 purrr_0.3.4
## [25] readr_2.1.1                 tidyr_1.1.4
## [27] tibble_3.1.6                ggplot2_3.3.5
## [29] tidyverse_1.3.1
##
## loaded via a namespace (and not attached):
##  [1] bitops_1.0-7           fs_1.5.2               lubridate_1.8.0
##  [4] RColorBrewer_1.1-2     httr_1.4.2             tools_4.1.2
##  [7] backports_1.4.1        DT_0.20                utf8_1.2.2
## [10] R6_2.5.1               DBI_1.1.2              colorspace_2.0-2
## [13] withr_2.4.3            tidyselect_1.1.1       compiler_4.1.2
## [16] cli_3.1.1              rvest_1.0.2            xml2_1.3.3
## [19] DelayedArray_0.20.0    scales_1.1.1           digest_0.6.29
## [22] shinyBS_0.61           rmarkdown_2.11         pkgconfig_2.0.3
## [25] htmltools_0.5.2        dbplyr_2.1.1           fastmap_1.1.0
## [28] htmlwidgets_1.5.4      rlang_0.4.12           readxl_1.3.1
## [31] rstudioapi_0.13        generics_0.1.1         hwriter_1.3.2
## [34] jsonlite_1.7.3         RCurl_1.98-1.5         magrittr_2.0.1
## [37] GenomeInfoDbData_1.2.7 Matrix_1.4-0           munsell_0.5.0
## [40] fansi_0.5.0            lifecycle_1.0.1        stringi_1.7.6
## [43] yaml_2.2.1             zlibbioc_1.40.0        plyr_1.8.6
## [46] grid_4.1.2             parallel_4.1.2         crayon_1.4.2
## [49] lattice_0.20-45        haven_2.4.3            hms_1.1.1
## [52] knitr_1.37             pillar_1.6.5           reshape2_1.4.4
## [55] reprex_2.0.1           glue_1.6.0             evaluate_0.14
## [58] latticeExtra_0.6-29    RcppParallel_5.1.5     modelr_0.1.8
## [61] vctrs_0.3.8            png_0.1-7              tzdb_0.2.0
## [64] cellranger_1.1.0       gtable_0.3.0           assertthat_0.2.1
## [67] xfun_0.29              broom_0.7.11           ellipsis_0.3.2
```