Chapter 4: 16S data analysis using the R package PhyloSeq

Emily Giroux

09/08/2020

Load packages into session, and print package versions:

```
sapply(c(.cran_packages, .bioc_packages), require, character.only = TRUE)
```

```
##    cowplot data.table    ggplot2      knitr  rprojroot  BiocStyle Biostrings
##       TRUE       TRUE       TRUE       TRUE       TRUE       TRUE       TRUE
##      dada2  ShortRead
##       TRUE       TRUE
```

Load the saved image from Chapter 1, then save it as a separate image to retain environment data specific to the 16S processing and analysis workflow.

```
load(paste(imageDirPath, startUpImage, sep = ""))
chptImageA <- "ecobiomics_16S.RData"
save.image(paste(imageDirPath, chptImageA, sep = ""))
```

When re-starting a session, you can quickly load up the image by running the chunk below:

```
sharedPath <- "/isilon/cfia-ottawa-fallowfield/users/girouxeml/PIRL_working_directory/"
analysis <- "ecobiomics/"
sharedPathAn <- paste(sharedPath, analysis, sep = "/")
imageDirPath <- "/home/CFIA-ACIA/girouxeml/GitHub_Repos/r_environments/ecobiomics/"
chptImageA   <- "ecobiomics_16S.RData"
load(paste(imageDirPath, chptImageA, sep = ""))
```

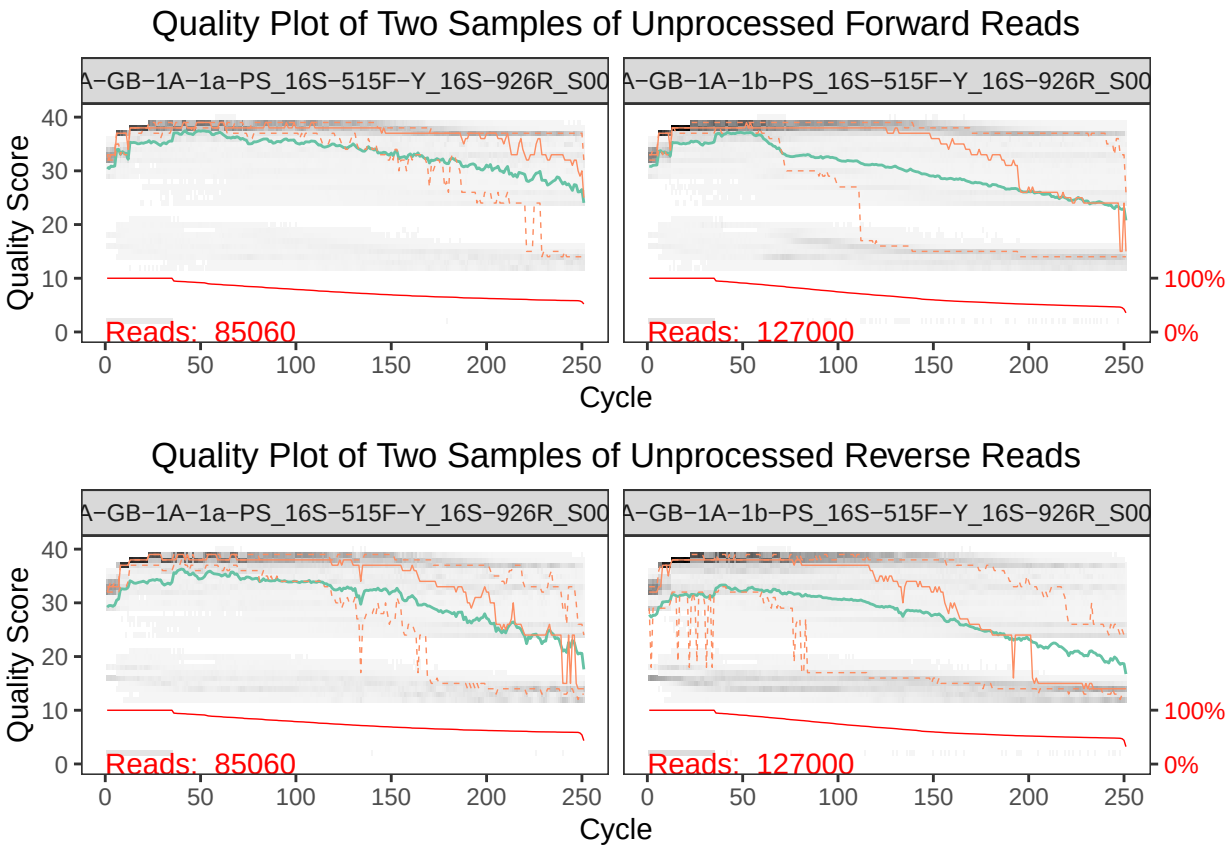
Create a variable shortcut to the region-specific analysis directory:
In this chapter, we are working with the 16S amplicon samples, so the 16S directory within our main project directory will contain all the region-specific output.

```
region <- "16S"
sharedPathReg <- paste(sharedPathAn, "/", region, "_analysis/", sep = "")
if(!dir.exists(sharedPathReg)) dir.create(sharedPathReg)

# Make the region-specific metadatable for this chapter the main metadata table:
metadataRegion <- metadata16S
```

Sample processing follows the steps outlined in Chapter 1

1. ggPlotRaw 16S:

```
library("dada2")
library("ggplot2")
plotRawQFwd <- dada2::plotQualityProfile(paste(metadataRegion$rawFastqPath[1:2],
                                                metadataRegion$rawR1[1:2], sep = "")) +
  ggplot2::ggtitle("Quality Plot of Two Samples of Unprocessed Forward Reads") +
  ggplot2::theme(plot.title = element_text(hjust = 0.5))
plotRawQRev <- dada2::plotQualityProfile(paste(metadataRegion$rawFastqPath[1:2],
                                                metadataRegion$rawR2[1:2], sep = "")) +
  ggplot2::ggtitle("Quality Plot of Two Samples of Unprocessed Reverse Reads") +
  ggplot2::theme(plot.title = element_text(hjust = 0.5))

library("cowplot")
cowplot::plot_grid(plotRawQFwd, plotRawQRev, nrow = 2)
```



Checking for Adapter Sequences

We're going to use Cutadapt to detect and trim off 16S-specific adapter sequences, V4-V5 hypervariable region.

515F-Y:

5' GTGYCAGCMGCCGCGGTAA 3'

926R:

5' CCGYCAATTYMTTTRAGTTT 3'

2. Cutadapt. We are replacing the code for SeqPrep with the use of cutadapt and workflow developed by Benjamin Callahan. See page: https://benjjneb.github.io/dada2/ITS_workflow.html

Capture your forward and reverse 16S primers:

```
fwdPrimer <- "GTGYCAGCMGCCGCGGTAA"
revPrimer <- "CCGYCAATTYMTTTRAGTTT"
```

Use the AllOrients (function defined in chapter2_ITSprocessing.Rmd) function to generate the primer sequences in all possible orientations in which they may be found:

```
fwdOrients <- AllOrients(fwdPrimer)
revOrients <- AllOrients(revPrimer)
fwdOrients
```

```
              Forward            Complement               Reverse
"GTGYCAGCMGCCGCGGTAA" "CACRGTCGKCGGCGCCATT" "AATGGCGCCGMCGACYGTG"
              RevComp
"TTACCGCGGCKGCTGRCAC"
```

Remove those sequences with ambiguous bases prior to running cutadapt:

```
library("dada2")
# Path to the processed fastq file directory:
processedFastq <- paste(sharedPathReg, "processedFQ", sep = "")
if(!dir.exists(processedFastq)) dir.create(processedFastq)

# Path to fastq filtered for ambiguous bases (Ns):
filtNsPath <- file.path(processedFastq, "A_removedAmbiguous")
if(!dir.exists(filtNsPath)) dir.create(filtNsPath)

# Define path and file names:
fwd   <- paste(metadataRegion$rawFastqPath, metadataRegion$rawR1, sep = "")
filtF <- file.path(filtNsPath,
```

```
                    paste(metadataRegion$LibraryName, "_F.fastq.gz", sep = ""))
rev   <- paste(metadataRegion$rawFastqPath, metadataRegion$rawR2, sep = "")
filtR <- file.path(filtNsPath,
                    paste(metadataRegion$LibraryName, "_R.fastq.gz", sep = ""))

# Run DADA2's filterAndTrim function to remove sequences with ambiguous bases:
dada2::filterAndTrim(fwd, filtF, rev, filtR, maxN = 0,
                     multithread = TRUE, verbose = TRUE)
```

We can now check the N-filtered sequences of just one sample set for primer hits using the PrimerHits function (function defined in chapter2_ITSprocessing.Rmd):

```
library("ShortRead")
rbind(FWD.ForwardReads = sapply(fwdOrients, PrimerHits, fn = filtF[1]),
      FWD.ReverseReads = sapply(fwdOrients, PrimerHits, fn = filtR[1]),
      REV.ForwardReads = sapply(revOrients, PrimerHits, fn = filtF[1]),
      REV.ReverseReads = sapply(revOrients, PrimerHits, fn = filtR[1]))
```

```
                 Forward Complement Reverse RevComp
FWD.ForwardReads   41451          0       0     822
FWD.ReverseReads     936          0       0    8006
REV.ForwardReads     662          0       0   13492
REV.ReverseReads   48359          0       0     519
```

As Expected, the vast majority of forward primer is found in its forward orientation, and in some of the reverse reads in its reverse-complement orientation (due to read-through when the 16S region is short). Similarly, the reverse primer is found with its expected orientations.

Set up the paths and fastq file input and output names in preparation for running cutadapt. **Note:** You may need to install cutadapt locally if it is not installed system wide.

```
cutadapt <- "/home/CFIA-ACIA/girouxeml/prog/anaconda3/envs/bio/bin/cutadapt"
```

We can use the `system2` call to run shell commands directly from R

```
system2(cutadapt, args = "--help")
```

Create a directory for the cutadapted fastq files, only if it doesn't already exist. Also create the output filenames for the cutadapt-ed fastq files.

```
cutAdaptDir <- file.path(processedFastq, "B_cutadapt", sep = "")
if(!dir.exists(cutAdaptDir)) dir.create(cutAdaptDir)

cutFqs <- paste(cutAdaptDir,
                metadataRegion$LibraryName, "_Fcut.fastq.gz", sep = "")
cutRqs <- paste(cutAdaptDir,
                metadataRegion$LibraryName, "_Rcut.fastq.gz", sep = "")
```

Now we can proceed to using cutadapt to remove primer sequences at the ends of our reads. See chapter2_ITSprocessing.Rmd for more details on Cutadapt flag parameters.

```
library("dada2")
fwdPrimerRC <- dada2:::rc(fwdPrimer)
revPrimerRC <- dada2:::rc(revPrimer)
# Trim fwdPrimer and the reverse-complement of the revPrimer off forward reads:
fwdFlags <- paste("-g", fwdPrimer, "-a", revPrimerRC)
# Trim revPrimer and the reverse-complement of the fwdPrimer off reverse reads:
revFlags <- paste("-G", revPrimer, "-A", fwdPrimerRC)
# Run Cutadapt
for(i in seq_along(metadataRegion$LibraryName)) {
  system2(cutadapt, args = c(fwdFlags, revFlags, "-n", 2,
                             "-o", cutFqs[i], "-p", cutRqs[i],
                             filtF[i], filtR[i]))
}
```

Note: There is a lot of output generated to the console when running cutadapt. One warning we may see often is about detection of incomplete adapter sequences:

```
WARNING:
One or more of your adapter sequences may be incomplete.
Please see the detailed output above.
```

Because our DNA fragments are generated from amplicon sequences and are not random, we can ignore this warning.

As a sanity check, we will count the presence of primers in the first cutadapt-ed sample:

```
rbind(FWD.ForwardReads = sapply(fwdOrients, PrimerHits, fn = cutFqs[1]),
      FWD.ReverseReads = sapply(fwdOrients, PrimerHits, fn = cutRqs[1]),
      REV.ForwardReads = sapply(revOrients, PrimerHits, fn = cutFqs[1]),
      REV.ReverseReads = sapply(revOrients, PrimerHits, fn = cutRqs[1]))
```

```
                 Forward Complement Reverse RevComp
FWD.ForwardReads       0          0       0     819
FWD.ReverseReads     934          0       0       0
REV.ForwardReads     662          0       0       0
REV.ReverseReads       0          0       0     519
```

3. filterAndTrimming. Raw read processing.

```
# Path to the final processed fastq file directory:
filtPathFinal <- file.path(processedFastq, "C_finalProcessed")
if(!dir.exists(filtPathFinal)) dir.create(filtPathFinal)
```

```
# Create output filenames for the final filtered and trimmed fastq files:
filtFwd <- paste(filtPathFinal, "/",
                 metadataRegion$LibraryName, "_F_filt.fastq.gz", sep = "")
filtRev <- paste(filtPathFinal, "/",
                 metadataRegion$LibraryName, "_R_filt.fastq.gz", sep = "")
```

For this dataset, we will use standard filtering paraments: maxN=0 (DADA2 requires sequences contain no Ns), truncQ = 2, rm.phix = TRUE and maxEE=2. The maxEE parameter sets the maximum number of "expected errors" allowed in a read, which is a better filter than simply averaging quality scores.

Run filterAndTrim using cutadapt-ed fastq files as input:

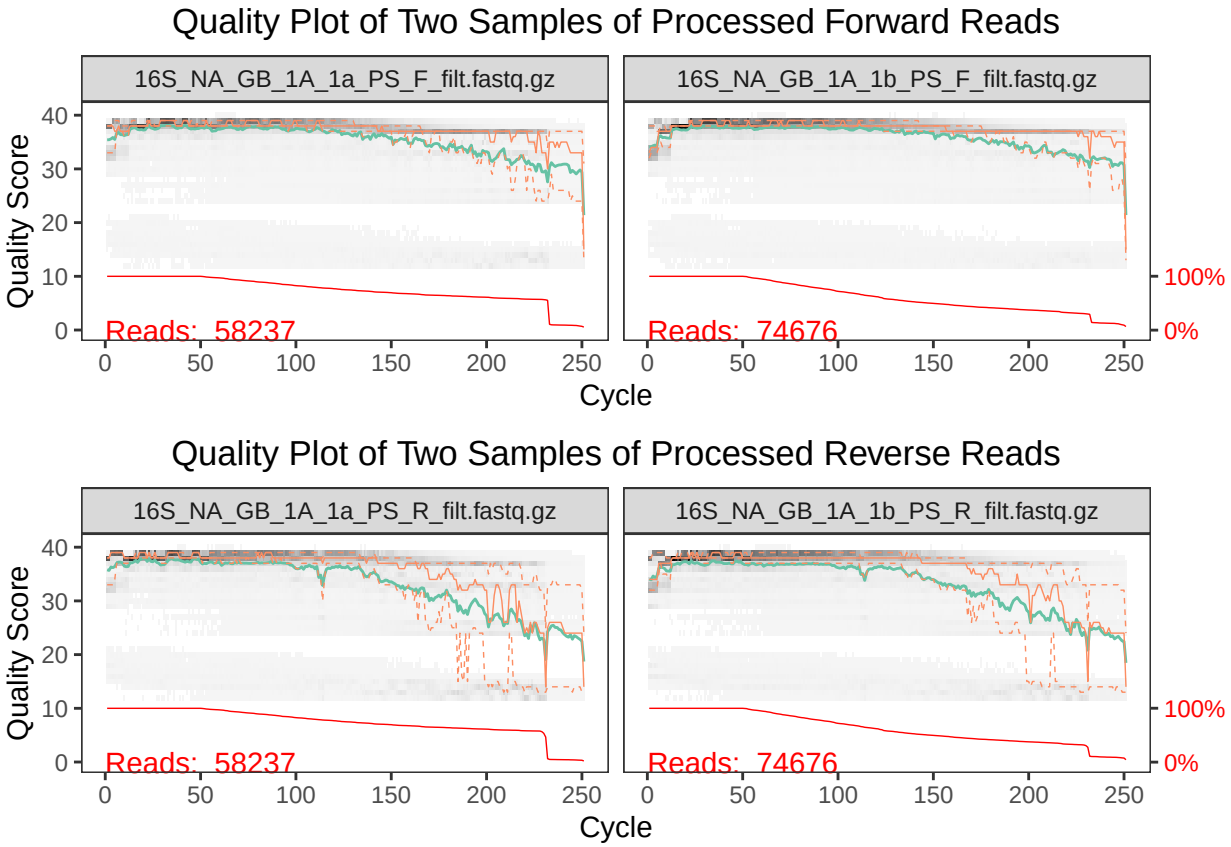
```
library("dada2")
trimOut <- dada2::filterAndTrim(cutFqs, filtFwd, cutRqs, filtRev, maxN = 0,
                                maxEE = c(3,3), truncQ = 2, minLen = 50,
                                rm.phix = TRUE, compress = TRUE, matchIDs = TRUE,
                                multithread = TRUE, verbose = TRUE)
```

Save the workspace image right after this step so it doesn't have to be repeated if R accidentally shuts down:

```
save.image(paste(imageDirPath, chptImageA, sep = ""))
```

4. ggPlotsProcessed. Let's look at the quality plots of our filtered and processed reads for our first 2 samples to see the effects of our trimming parameters:

```
library("dada2")
library("ggplot2")
plotQfwd <- dada2::plotQualityProfile(paste(filtPathFinal, "/",
                                            metadataRegion$LibraryName[1:2],
                                            "_F_filt.fastq.gz", sep = "")) +
  ggplot2::ggtitle("Quality Plot of Two Samples of Processed Forward Reads") +
  ggplot2::theme(plot.title = element_text(hjust = 0.5))
plotQrev <- dada2::plotQualityProfile(paste(filtPathFinal, "/",
                                            metadataRegion$LibraryName[1:2],
                                            "_R_filt.fastq.gz", sep = "")) +
  ggplot2::ggtitle("Quality Plot of Two Samples of Processed Reverse Reads") +
  ggplot2::theme(plot.title = element_text(hjust = 0.5))

library("cowplot")
cowplot::plot_grid(plotQfwd, plotQrev, nrow = 2)
```



5. keepTrimmed. Update metadata so that samples that did not have any reads that passed filtering are removed from further analysis, to avoid downstream processing errors.

We can take a brief look at the summary or reads in and out for the first 6 samples:

```
head(trimOut[order(-trimOut[,2]),])
```

```
                   reads.in reads.out
16S_NA_GB_K1C_2_FD   826918    625758
16S_NA_GB_K1C_1_PS   682689    526549
16S_NA_GB_SW_PS_16   363529    300549
16S_NA_GB_K1C_2_PS   365207    278495
16S_NA_GB_M7R_PS_4   296621    245237
16S_NA_GB_K1A_2_PW   308174    226683
```

The row names we see have retained the fastq file name of the input forward reads, yet the output sums are for both forward and reverse reads. Let's remove the file suffix so that the rownames will apply to both forward and reverse reads.

```
rownames(trimOut) <- gsub("_Fcut.fastq.gz", "", rownames(trimOut))
```

As a precaution, we still check for samples with no remaining reads and update the metadata table using those results. If you receive an error during the dereplication step later on: `Error in`

`open.connection(con, "rb") : cannot open the connection`, you need to update the files so that those that didn't pass filtering aren't attempted. See issue at:
https://github.com/benjjneb/dada2/issues/375

```
keepTrim <- trimOut[,"reads.out"] > 20 # Or other cutoff

filtFs <- file.path(filtPathFinal,
                    paste(metadataRegion$LibraryName,
                          "_F_filt.fastq.gz", sep = ""))[keepTrim]
filtRs <- file.path(filtPathFinal,
                    paste(metadataRegion$LibraryName,
                          "_R_filt.fastq.gz", sep = ""))[keepTrim]

filtNames <- basename(filtFs)[keepTrim]
filtNames <- gsub("_F_filt.fastq.gz", "", filtNames)
```

Run the `keepTrimmed2` chunk. We've updated our list of fastq file names, but we also need to update our metadata table so that rows that had read pairs where a direction no longer had reads after filtering are removed from the metadata table.

```
library("data.table")
data.table::setkey(metadataRegion, LibraryName)
metadatafilt <- metadataRegion[.(filtNames)]
metadatafilt$filtFwd <- paste(filtPathFinal, "/", metadatafilt$LibraryName,
                              "_F_filt.fastq.gz", sep = "")
metadatafilt$filtRev <- paste(filtPathFinal, "/", metadatafilt$LibraryName,
                              "_R_filt.fastq.gz", sep = "")
```

6. splitRunsMetadata. We need to do this for our 16S amplicons set, which was generated on two separate sequencing runs.

```
library("data.table")
metadatafiltPlate1 <- metadatafilt[SeqPlate == 1]
metadatafiltPlate4 <- metadatafilt[SeqPlate == 4]
```

Save the chapter image.

```
save.image(paste(imageDirPath, chptImageA, sep = ""))
```