Long Read Tutorial

Jorden Rabasco

6/29/2022

In the sequencing world there are many different technologies that have advantages and disadvantages, which should all be considered before a sequencing run has begun. Short read sequencing technologies, such as illumina pair-wise sequencing, have been at the head of the field for a number of years. Illumina sequencing can give an accurate sequence but is constrained in size, with the reads being anywhere from 200-300bp. The advent of long read sequencing attempts to solve this length constraint. Pacbio one of the leaders in the long read sequencing field can generate reads much longer than traditional short read sequencing but with an increaced error rate. To solve this techniques such as loop seq have been implemtented which sequence a genetic region many times and then creates a consensus sequence in order to bring down the per base error rate. However this again creates a limitation in the size of the locus that is being sequenced. To address this and other limitations bioinformatic tools and workflows, such as those included here, have been created. This tutorial is intended for those that wish to analyze their long read sequences, generated from Pacbio techniques. The output from this workflow will consist of assigned taxonomies based on the ASVs generated by dada2. This tutorial assumes a working version of R 4.2.0

Lets begin. This tutorial utilizes data from “High-throughput amplicon sequencing of the full-length 16S rRNA gene with single-nucleotide resolution” by Callahan et al. Nucleic Acids Research, 2019c which can be found here https://academic.oup.com/nar/article/47/18/e103/5527971?login=true (https://academic.oup.com/nar/article/47/18/e103/5527971?login=true). The data itself consists of sequencing measurements of a set of nine human fecal samples, from 3 different subjects. Additionally pacbio sequencing of a zymo mock community was included as a benchmark.

Initial Set up

In the initial set up we will have to prep the reads for filtering and primer trimming. To carry out the preprocessing steps prior to the dada2 workflow, run the below code at the command line. This will convert the bam file into a fastq file that can then be further analyzed downstream, by this workflow.

ccs –pbi –force –logLevel=DEBUG –numThreads=16 –minSnr=3.75 –minReadScore=0.65 –maxLength=7000 –minLength=10 –minPasses=3 –minZScore=-5 –maxDropFraction=0.34 –minPredictedAccuracy=0.999 subreads.bam ccs.bam

Prior to any R processing please make sure to install dada2 appropriately as well as the dependencies. The appropriate information can be found here: https://benjjneb.github.io/dada2/dada-installation.html (https://benjjneb.github.io/dada2/dada-installation.html)

Once dada2 is installed load the library and check the version. Make sure thea you are using the latest version of dada2 as some fixes may have been implemented in later releases. This tutorial ultitzes version 1.20.0 of the dada2 package.

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

```
## [1] '1.20.0'
```

Pathing setup

This section sets up the pathing for the tutorial and its output. When you first download the tutorial zip file, you will need to change the “base_path” variable to wherever the tutorial zip file is downloaded too. Note: When running your own data through this tutorial, it would be prudent to set up your data in the “data” folder of the tutorial while removing the test data present in there. This will prevent any needless errors from pathing to occur.

```
base_path <- "C:/Users/jorde/OneDrive/Desktop" #Informs all other pathing in the tutorial
```

Identification of Primers

In this section we need to input the primer information and initalize the rc funciton from the dada2 package. Note: After you have concluded the tutorial you will need to change “FW_pacbio” and “Rev_pacbio” to the appropriate primers used in your Pacbio sequencing.

```
Fw_pacbio<- "AGRGTTYGATYMTGGCTCAG" #change to fw primer in sequencing
Rev_pacbio <- "RGYTACCTTGTTACGACTT" #change to reverse primer in sequencing
rc <- dada2:::rc #intialized the rc function in the dada2 package
theme_set(theme_bw())
```

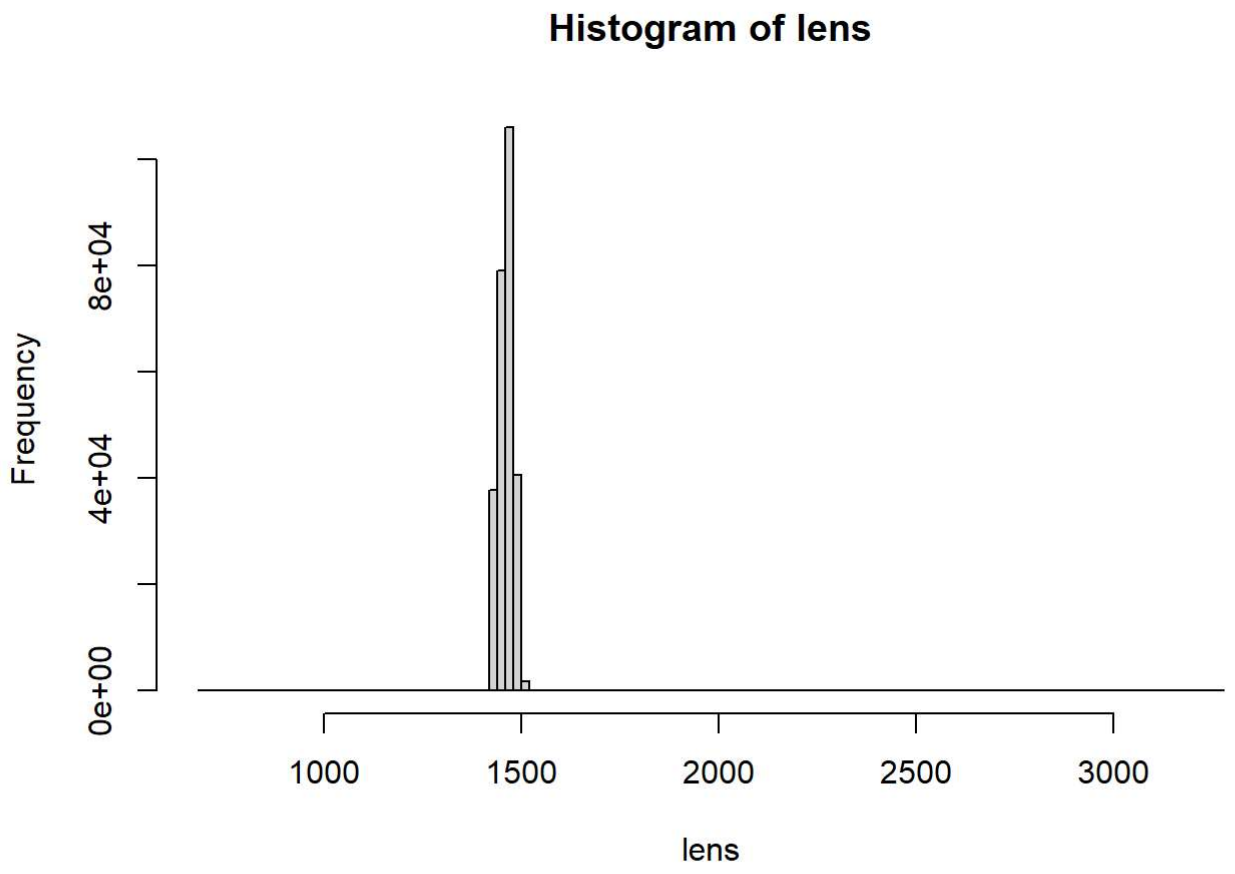
Filtering and Primer Removal

Here we will be removing the primers and filtering out poor qlaity reads. We want to remove the primers as they were not part of the original sequence and thus can alter and return wrong ASVs or taxonomic assinments. Furthermore, when you have your reads in a fastq style format which was generated in the first step of this workflow, each read has both an id and a sequence of quality indicators. These indicators show how sure we are that each base was identified correctly. That is to say that with low quality bases some abiguity in the idenity of the base, in that position is inherrent. Therefore, we want to filter out these poor quality reads from reads that have a higher quality to ensure that the results are accurate. These two steps will ensure the quality of the reads and assist in downstream taxonomic assignments. Additionally, Pacbio read orientation is mixed, meaning that some reads are in the 3’ to 5’ orientaiton and some are in the 5’ to 3’ orientation. This is corrected in the primer removal step. We will also want to vizualize the length of the seqeuences after trimming to ensure that the sequences are predominently of the lengths that we expect.

```
path<-file.path(base_path, "v5_long_read_tutorial/data")#sets locaiton of data for input into wo
rkflow
#The below code allows all files with the "pattern" suffix to be found in the path1 or path2 fol
der location
fn_r <- list.files(path, pattern="fastq.gz", full.names=TRUE)
nop <- file.path(base_path, "v5_long_read_tutorial/Pacbio/noprimers", basename(fn_r)) #generates
file in the noprimers folder
prim <- removePrimers(fn_r, nop, primer.fwd=Fw_pacbio, primer.rev=dada2:::rc(Rev_pacbio), orient
=TRUE,  verbose=TRUE) #removes primers from fastq input file and overwrites file in noprimers fo
lder
```

```
## 9727 sequences out of 19627 are being reverse-complemented.
## 15162 sequences out of 30983 are being reverse-complemented.
## 13574 sequences out of 27505 are being reverse-complemented.
## 9274 sequences out of 18783 are being reverse-complemented.
## 15890 sequences out of 32133 are being reverse-complemented.
## 13727 sequences out of 28156 are being reverse-complemented.
## 11851 sequences out of 24760 are being reverse-complemented.
## 10566 sequences out of 21621 are being reverse-complemented.
## 7371 sequences out of 15169 are being reverse-complemented.
## 39439 sequences out of 77453 are being reverse-complemented.
```

```
lens.fn <- lapply(nop, function(fn) nchar(getSequences(fn)))
lens <- do.call(c, lens.fn)
hist(lens, 100) #generates plot of sequences after primer removal
```



There is a strong peak around 1450, so that looks good. Next we are going to filter the reads for quality.

```
filt <- file.path(base_path,"v5_long_read_tutorial/Pacbio/filtered", basename(fn_r)) #generates
 file in the filtered folder
track <- filterAndTrim(nop, filt, minQ=3, minLen=1000, maxLen=1600, maxN=0, rm.phix=FALSE, maxEE
=2)
track
```

```
##                      reads.in reads.out
## R11_1_P3C3.fastq.gz    17436     17436
## R3_1_P3C3.fastq.gz     26605     26605
## R3_2_P3C3.fastq.gz     24477     24081
## R3_3_P3C3.fastq.gz     16738     16738
## R9_1_P3C3.fastq.gz     28623     28623
## R9_1B_P3C3.fastq.gz    24832     24831
## R9_2_P3C3.fastq.gz     21759     21759
## R9_3_P3C3.fastq.gz     18701     18700
## R9_4_P3C3.fastq.gz     13481     13258
## Zymo_P1C1.fastq.gz     73057     72940
```

Given in the table are the reads fed into the filtering step and the output reads after the filtering step. If the numbers in each column are the same then no reads were removed due to quality, however, it is not uncommon that a few reads will be removed due to quality concerns.

Denoising

The next step is arguable the most important part of the tutorial; denoising via dada2 package. This was already loaded into the R environment in the beginning of the workflow so there is no need to worry. We will also want to produce error plots and an ASV table which we can then use for taxonomic assingment. The first step in this process is to dereplicate amplicon sequences. Then we will learn error rates, save them in an R object, and plot the output error graph. If for any reason you need to stop the tutorial, the R object can then be loaded in and the workflow continued from this step. To check to see if the sequences were dereplicated correcly we can look at some of the unique sequence counts in the one of the samples.

```
drp <- derepFastq(filt, verbose=TRUE)
head(drp$R11_1_P3C3.fastq.gz$uniques)
```

```
## GATGAACGCTGGCGGCGTGCTTAACACATGCAAGTCGAACGAAGCGCTTGAGAAGATTTCTTCGGAATGAAACTTAAGTGACTGAGTGGCGGA
CGGGTGAGTAACGCGTGGGTAACCTGCCTCATACAGGGGGATAACAGTCAGAAATGACTGCTAATACCGCATAAGCGCACAGTACCGCATGGTACG
GTGTGAAAAACTCCGGTGGTATGGGATGGACCCGCGTCTGATTAGCTAGTTGGTGGGGTAACGGCCTACCAAGGCGACGATCAGTAGCCGGCCTGA
GAGGGCGACCGGCCACATTGGGACTGAGACACGGCCCAAACTCCTACGGGAGGCAGCAGTGGGGAATATTGCACAATGGGGGAAACCCTGATGCAG
CGACGCCGCGTGAGCGAAGAAGTATTTCGGTATGTAAAGCTCTATCAGCAGGGAAGAAAATGACGGTACCTGACTAAGAAGCCCCGGCTAACTACG
TGCCAGCAGCCGCGGTAATACGTAGGGGGCAAGCGTTATCCGGATTTACTGGGTGTAAAGGGAGCGTAGACGGACTGGCAAGTCTGATGTGAAAAC
CCGGGGCTCAACCCCGGGACTGCATTGGAAACTGTCAGTCTAGAGTGTCGGAGGGGTAAGTGGAATTCCTAGTGTAGCGGTGAAATGCGTAGATAT
TAGGAGGAACACCAGTGGCGAAGGCGGCTTACTGGACGATAACTGACGTTGAGGCTCGAAAGCGTGGGGAGCAAACAGGATTAGATACCCTGGTAG
TCCACGCCGTAAACGATGAATACTAGGTGTCGGGAAGCAAAGCTTTTCGGTGCCGCAGCAAACGCAATAAGTATTCCACCTGGGGAGTACGTTCGC
AAGAATGAAACTCAAAGGAATTGACGGGGACCCGCACAAGCGGTGGAGCATGTGGTTTAATTCGAAGCAACGCGAAGAACCTTACCAAGTCTTGAC
ATCGATCTGCCCGGACTGTAACGAGTCCTTTCCCTTCGGGGACAGAGAAGACAGGTGGTGCATGGTTGTCGTCAGCTCGTGTCGTGAGATGTTGGG
TTAAGTCCCGCAACGAGCGCAACCCTTATCCTTAGTAGCCAGCAGGTAAAGCTGGGCACTCTGGGGAGACTGCCAGGGATAACCTGGAGGAAGGTG
GGGATGACGTCAAATCATCATGCCCCTTATGATTTGGGCTACACACGTGCTACAATGGCGTAAACAAAGGGAAGCAGCCCCGCGAGGGTGAGCAAA
TCTCAAAAATAACGTCCCAGTTCGGATTGTAGTCTGCAACTCGACTACATGAAGCTGGAATCGCTAGTAATCGCAGGTCAGCATACTGCGGTGAAT
ACGTTCCCGGGTCTTGTACACACCGCCCGTCACACCATGGGAGTCGGATATGCCCGAAGCCAGTGACCCAACCTTAGGGAGGGAGCTGTCGAAGGT
GGAGCCGATAACTGGGGTG
##
1320
## GATGAACGCTGGCGGCGTGCTTAACACATGCAAGTCGAACGAAGCACTTGAGAAGATTTCTTCGGAATGAAACTTAAGTGACTGAGTGGCGGA
CGGGTGAGTAACGCGTGGGTAACCTGCCTCATACAGGGGGATAACAGTCAGAAATGACTGCTAATACCGCATAAGCGCACAGTACCGCATGGTACG
GTGTGAAAAACTCCGGTGGTATGGGATGGACCCGCGTCTGATTAGCTAGTTGGTGGGGTAACGGCCTACCAAGGCGACGATCAGTAGCCGGCCTGA
GAGGGCGACCGGCCACATTGGGACTGAGACACGGCCCAAACTCCTACGGGAGGCAGCAGTGGGGAATATTGCACAATGGGGGAAACCCTGATGCAG
CGACGCCGCGTGAGCGAAGAAGTATTTCGGTATGTAAAGCTCTATCAGCAGGGAAGAAAATGACGGTACCTGACTAAGAAGCCCCGGCTAACTACG
TGCCAGCAGCCGCGGTAATACGTAGGGGGCAAGCGTTATCCGGATTTACTGGGTGTAAAGGGAGCGTAGACGGACTGGCAAGTCTGATGTGAAAAC
CCGGGGCTCAACCCCGGGACTGCATTGGAAACTGTCAGTCTAGAGTGTCGGAGGGGTAAGTGGAATTCCTAGTGTAGCGGTGAAATGCGTAGATAT
TAGGAGGAACACCAGTGGCGAAGGCGGCTTACTGGACGATAACTGACGTTGAGGCTCGAAAGCGTGGGGAGCAAACAGGATTAGATACCCTGGTAG
TCCACGCCGTAAACGATGAATACTAGGTGTCGGGAAGCAAAGCTTTTCGGTGCCGCAGCAAACGCAATAAGTATTCCACCTGGGGAGTACGTTCGC
AAGAATGAAACTCAAAGGAATTGACGGGGACCCGCACAAGCGGTGGAGCATGTGGTTTAATTCGAAGCAACGCGAAGAACCTTACCAAGTCTTGAC
ATCGATCTGCCCGGACTGTAACGAGTCCTTTCCCTTCGGGGACAGAGAAGACAGGTGGTGCATGGTTGTCGTCAGCTCGTGTCGTGAGATGTTGGG
TTAAGTCCCGCAACGAGCGCAACCCTTATCCTTAGTAGCCAGCAGGTAAAGCTGGGCACTCTGGGGAGACTGCCAGGGATAACCTGGAGGAAGGTG
GGGATGACGTCAAATCATCATGCCCCTTATGATTTGGGCTACACACGTGCTACAATGGCGTAAACAAAGGGAAGCAGCCCCGCGAGGGTGAGCAAA
TCTCAAAAATAACGTCCCAGTTCGGATTGTAGTCTGCAACTCGACTACATGAAGCTGGAATCGCTAGTAATCGCAGGTCAGCATACTGCGGTGAAT
ACGTTCCCGGGTCTTGTACACACCGCCCGTCACACCATGGGAGTCGGATATGCCCGAAGCCAGTGACCCAACCTTAGGGAGGGAGCTGTCGAAGGT
GGAGCCGATAACTGGGGTG
##
879
##     GATGAACGCTGGCGGCGTGCTTAACACATGCAAGTCGAGCGAAGCACTTATCATTGACTCTTCGGAAGATTTGATATTTGACTGAGCGGC
GGACGGGTGAGTAACGCGTGGGTAACCTGCCTCATACAGGGGAATAACAGTTAGAAATGGCTGCTAATGCCGCATAAGCGCACAGGACCGCATGGT
CTGGTGTGAAAAACTGAGGTGGTATGAGATGGACCCGCGTCTGATTAGGTAGTTGGTGGGGTAACGGCCTACCAAGCCGACGATCAGTAGCCGGCC
TGAGAGGGTGAACGGCCACATTGGGACTGAGACACGGCCCAGACTCCTACGGGAGGCAGCAGTGGGGAATATTGCACAATGGGGGAAACCCTGATG
CAGCGACGCCGCGTGAAGGAAGAAGTATCTCGGTATGTAAACTTCTATCAGCAGGGAAGAAAATGACGGTACCTGACTAAGAAGCCCCGGCTAACT
ACGTGCCAGCAGCCGCGGTAATACGTAGGGGGCAAGCGTTATCCGGATTTACTGGGTGTAAAGGGAGCGTAGACGGAAGAGCAAGTCTGATGTGAA
AGGCTGGGGCTTAACCCCAGGACTGCATTGGAAACTGTTTTTCTAGAGTGCCGGAGAGGTAAGCGGAATTCCTAGTGTAGCGGTGAAATGCGTAGA
TATTAGGAGGAACACCAGTGGCGAAGGCGGCTTACTGGACGGTAACTGACGTTGAGGCTCGAAAGCGTGGGGAGCAAACAGGATTAGATACCCTGG
TAGTCCACGCCGTAAACGATGAATACTAGGTGTCGGGGTGCAAAGCAGTTCGGTGCCGCAGCAAACGCAATAAGTATTCCACCTGGGGAGTACGTT
CGCAAGAATGAAACTCAAAGGAATTGACGGGGACCCGCACAAGCGGTGGAGCATGTGGTTTAATTCGAAGCAACGCGAAGAACCTTACCAAGTCTT
GACATCTGCCTGACCGTTCCTTAACCGGAGCTTTCCTTCGGGACAGGCAAGACAGGTGGTGCATGGTTGTCGTCAGCTCGTGTCGTGAGATGTTGG
GTTAAGTCCCGCAACGAGCGCAACCCCTATCCTTAGTAGCCAGCAGTACGGCTGGGCACTCTAGGGAGACTGCCGGGGATAACCCGGAGGAAGGCG
GGGACGACGTCAAATCATCATGCCCCTTATGATTTGGGCTACACACGTGCTACAATGGCGTAAACAAAGGGAAGCGAAGCGGTGACGCTTAGCAAA
TCTCAAAAATAACGTCCCAGTTCGGACTGCAGTCTGCAACTCGACTGCACGAAGCTGGAATCGCTAGTAATCGCGAATCAGAATGTCGCGGTGAAT
ACGTTCCCGGGTCTTGTACACACCGCCCGTCACACCATGGGAGTCAGTAACGCCCGAAGTCAGTGACCCAACCTTATGGAGGGAGCTGCCGAAGGC
GGGACCGATAACTGGGGTG
```

```
##
361
##       GATGAACGCTGGCGGCGTGCTTAACACATGCAAGTCGAACGGGGAACATTTTATGGAAGCTTCGGTGGAAATAGCTTGTTCCTAGTGGC
GGACGGGTGAGTAACGCGTGGGTAACCTGCCTCACACTGGGGGATAACAGTCAGAAATGACTGCTAATACCGCATAAGCGCACGGGATTGCATGAT
CCAGTGTGAAAAACTCCGGTGGTGTGAGATGGACCCGCGTTGGATTAGCCAGTTGGCAGGGTAACGGCCTACCAAAGCGACGATCCATAGCCGGCC
TGAGAGGGTGGACGGCCACATTGGGACTGAGACACGGCCCAGACTCCTACGGGAGGCAGCAGTGGGGAATATTGCACAATGGGGGAAACCCTGATG
CAGCGACGCCGCGTGAAGGAAGAAGTATCTCGGTATGTAAACTTCTATCAGCAGGGAAGAAAATGACGGTACCTGACTAAGAAGCCCCGGCTAACT
ACGTGCCAGCAGCCGCGGTAATACGTAGGGGGCAAGCGTTATCCGGATTTACTGGGTGTAAAGGGAGCGTAGACGGAGCAGCAAGTCTGATGTGAA
AGGCAGGGGCTCAACCCCTGGACTGCATTGGAAACTGTTGATCTTGAGTACCGGAGAGGTAAGCGGAATTCCTAGTGTAGCGGTGAAATGCGTAGA
TATTAGGAGGAACACCAGTGGCGAAGGCGGCTTACTGGACGGTAACTGACGTTGAGGCTCGAAAGCGTGGGGAGCAAACAGGATTAGATACCCTGG
TAGTCCACGCCGTAAACGATGAATACTAGGTGTCGGGTGGCAGAGCCATTCGGTGCCGCAGCAAACGCAGTAAGTATTCCACCTGGGGAGTACGTT
CGCAAGAATGAAACTCAAAGGAATTGACGGGGACCCGCACAAGCGGTGGAGCATGTGGTTTAATTCGAAGCAACGCGAAGAACCTTACCAAGTCTT
GACATCCCGATGACCGGCACGTAACGGTGCCTTCTCTTCGGAGCATCGGAGACAGGTGGTGCATGGTTGTCGTCAGCTCGTGTCGTGAGATGTTGG
GTTAAGTCCCGCAACGAGCGCAACCCCTATCCTTAGTAGCCAGCGGTTTGGCCGGGCACTCTGAGGAGACTGCCAGGGATAACCTGGAGGAAGGCG
GGGATGACGTCAAATCATCATGCCCCTTATGATTTGGGCTACACACGTGCTACAATGGCGTAAACAAAGGGAAGCGAGAGTGTGAGCTTAAGCAAA
TCCCAAAAATAACGTCCCAGTTCGGACTGCAGTCTGCAACTCGACTGCACGAAGCTGGAATCGCTAGTAATCGCGGATCAGAATGCCGCGGTGAAT
ACGTTCCCGGGTCTTGTACACACCGCCCGTCACACCATGGGAGTCAGTAACGCCCGAAGTCAGTGACCGAACCGAAAGGACGGAGCTGCCGAAGGC
GGGACGGATGACTGGGGTG
##
357
##       GATGAACGCTGGCGGCGTGCTTAACACATGCAAGTCGAACGGGGAACATTTTATGGAAGCTTCGGTGGAAATAGCTTGTTCCTAGTGGC
GGACGGGTGAGTAACGCGTGGGTAACCTGCCTCACACTGGGGGATAACAGTCAGAAATGACTGCTAATACCGCATAAGCGCACGGGATTGCATGAT
CCAGTGTGAAAAACTCCGGTGGTGTGAGATGGACCCGCGTTGGATTAGCCAGTTGGCAGGGTAACGGCCTACCAAAGCGACGATCCATAGCCGGCC
TGAGAGGGTGGACGGCCACATTGGGACTGAGACACGGCCCAGACTCCTACGGGAGGCAGCAGTGGGGAATATTGCACAATGGGGGAAACCCTGATG
CAGCGACGCCGCGTGAAGGAAGAAGTATCTCGGTATGTAAACTTCTATCAGCAGGGAAGAAAATGACGGTACCTGACTAAGAAGCCCCGGCTAACT
ACGTGCCAGCAGCCGCGGTAATACGTAGGGGGCAAGCGTTATCCGGATTTACTGGGTGTAAAGGGAGCGTAGACGGAGCAGCAAGTCTGATGTGAA
AGGCGGGGGCTCAACCCCCGGACTGCATTGGAAACTGTTGATCTTGAGTACCGGAGAGGTAAGCGGAATTCCTAGTGTAGCGGTGAAATGCGTAGA
TATTAGGAGGAACACCAGTGGCGAAGGCGGCTTACTGGACGGTAACTGACGTTGAGGCTCGAAAGCGTGGGGAGCAAACAGGATTAGATACCCTGG
TAGTCCACGCCGTAAACGATGAATACTAGGTGTCGGGTGGCAGAGCCATTCGGTGCCGCAGCAAACGCAGTAAGTATTCCACCTGGGGAGTACGTT
CGCAAGAATGAAACTCAAAGGAATTGACGGGGACCCGCACAAGCGGTGGAGCATGTGGTTTAATTCGAAGCAACGCGAAGAACCTTACCAAGTCTT
GACATCCCGATGACCGGCACGTAACGGTGCCTTCTCTTCGGAGCATCGGAGACAGGTGGTGCATGGTTGTCGTCAGCTCGTGTCGTGAGATGTTGG
GTTAAGTCCCGCAACGAGCGCAACCCCTATCCTTAGTAGCCAGCGGTTTGGCCGGGCACTCTGAGGAGACTGCCAGGGATAACCTGGAGGAAGGCG
GGGATGACGTCAAATCATCATGCCCCTTATGATTTGGGCTACACACGTGCTACAATGGCGTAAACAAAGGGAAGCGAGAGTGTGAGCTTAAGCAAA
TCCCAAAAATAACGTCCCAGTTCGGACTGCAGTCTGCAACTCGACTGCACGAAGCTGGAATCGCTAGTAATCGCGGATCAGAATGCCGCGGTGAAT
ACGTTCCCGGGTCTTGTACACACCGCCCGTCACACCATGGGAGTCAGTAACGCCCGAAGTCAGTGACCGAACCGAAAGGACGGAGCTGCCGAAGGC
GGGACGGATGACTGGGGTG
##
339
##     GATGAACGCTGGCGGCGTGCTTAACACATGCAAGTCGAGCGAAGCACTTATCATTGACTCTTCGGAAGATTTGATATTTGACTGAGCGGC
GGACGGGTGAGTAACGCGTGGGTAACCTGCCTCATACAGGGGAATAACAGTTAGAAATGGCTGCTAATGCCGCATAAGCGCACAGGACCGCATGGT
TCGGTGTGAAAAACCGAGGTGGTATGAGATGGACCCGCGTCTGATTAGGTAGTTGGTGGGGTAACGGCCTACCAAGCCGACGATCAGTAGCCGGCC
TGAGAGGGTGAACGGCCACATTGGGACTGAGACACGGCCCAGACTCCTACGGGAGGCAGCAGTGGGGAATATTGCACAATGGGGGAAACCCTGATG
CAGCGACGCCGCGTGAAGGAAGAAGTATCTCGGTATGTAAACTTCTATCAGCAGGGAAGAAAATGACGGTACCTGACTAAGAAGCCCCGGCTAACT
ACGTGCCAGCAGCCGCGGTAATACGTAGGGGGCAAGCGTTATCCGGATTTACTGGGTGTAAAGGGAGCGTAGACGGAAGAGCAAGTCTGATGTGAA
AGGCTGGGGCTTAACCCCAGGACTGCATTGGAAACTGTTTTTCTAGAGTGCCGGAGAGGTAAGCGGAATTCCTAGTGTAGCGGTGAAATGCGTAGA
TATTAGGAGGAACACCAGTGGCGAAGGCGGCTTACTGGACGGTAACTGACGTTGAGGCTCGAAAGCGTGGGGAGCAAACAGGATTAGATACCCTGG
TAGTCCACGCCGTAAACGATGAATACTAGGTGTCGGGGTGCAAAGCAGTTCGGTGCCGCAGCAAACGCAATAAGTATTCCACCTGGGGAGTACGTT
CGCAAGAATGAAACTCAAAGGAATTGACGGGGACCCGCACAAGCGGTGGAGCATGTGGTTTAATTCGAAGCAACGCGAAGAACCTTACCAAGTCTT
GACATCTGCCTGACCGTTCCTTAACCGGAGCTTTCCTTCGGGACAGGCAAGACAGGTGGTGCATGGTTGTCGTCAGCTCGTGTCGTGAGATGTTGG
GTTAAGTCCCGCAACGAGCGCAACCCCTATCCTTAGTAGCCAGCAGTACGGCTGGGCACTCTAGGGAGACTGCCGGGGATAACCCGGAGGAAGGCG
GGGACGACGTCAAATCATCATGCCCCTTATGATTTGGGCTACACACGTGCTACAATGGCGTAAACAAAGGGAAGCGAAGCGGTGACGCTTAGCAAA
TCTCAAAAATAACGTCCCAGTTCGGACTGCAGTCTGCAACTCGACTGCACGAAGCTGGAATCGCTAGTAATCGCGAATCAGAATGTCGCGGTGAAT
```

```
ACGTTCCCGGGTCTTGTACACACCGCCCGTCACACCATGGGAGTCAGTAACGCCCGAAGTCAGTGACCCAACCTTATGGAGGGAGCTGCCGAAGGC
GGGACCGATAACTGGGGTG
##
331
```

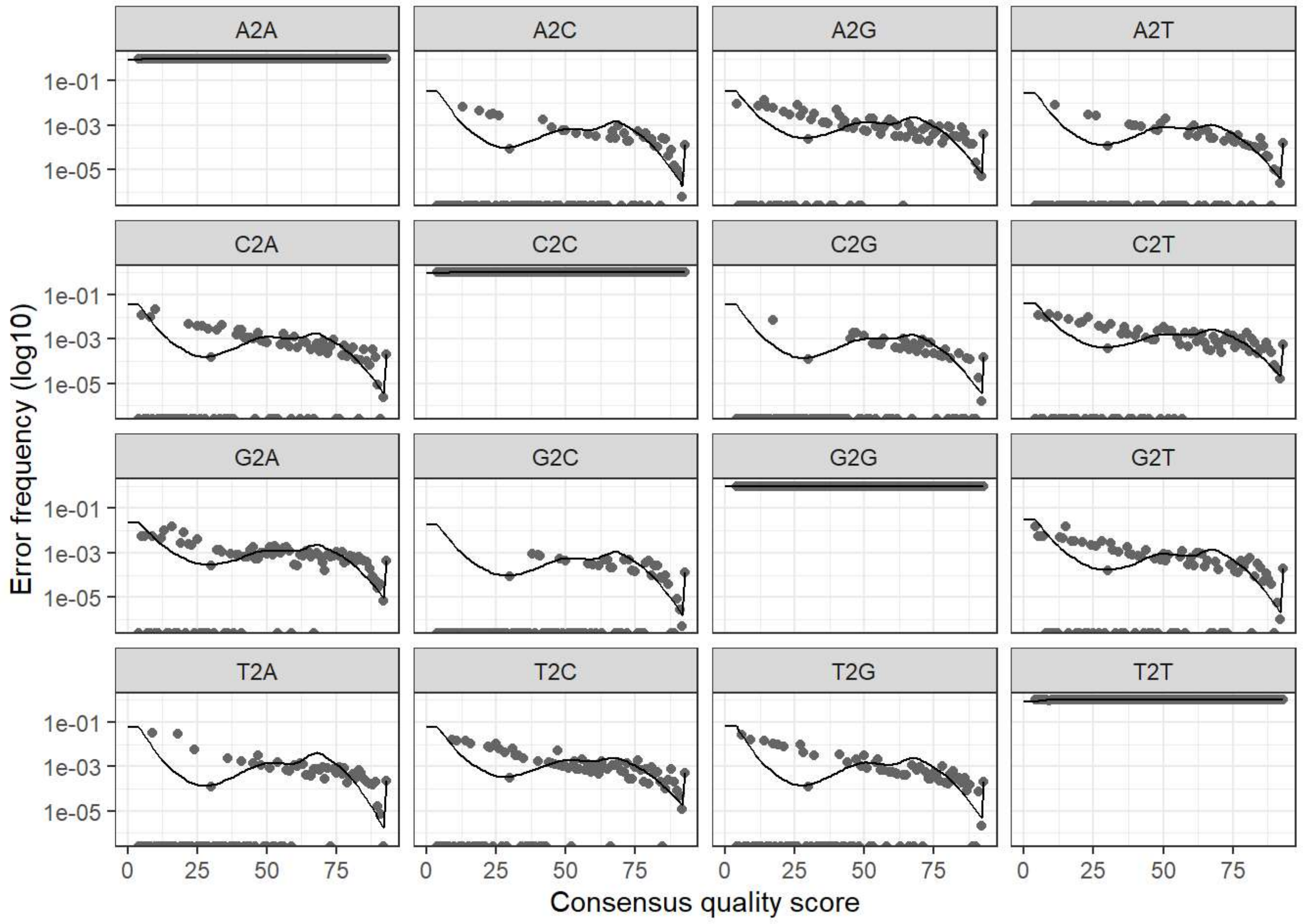
As we can see the list of unique sequences and their counts were generated. We can also see that there is a signifigant relative abundance of these unique sequences. This allows the "learnerrors" module to run approriatly. If there wasn't enough abundance of unique reads than the error model wouldn't succeed. However this doesn't seem to be the case here and we can therefore assume that the dereplicaiton procedure was a success! If you wish to check the other samples you can switch the sample name in the code "head(drp `R11_1_P3C3.fastq.gz` uniques)" to whichever sample you would like to view.

The DADA2 algorithm makes use of a parametric error model (err) and every amplicon dataset has a different set of error rates. The "learnErrors" method learns this error model from the data, by alternating estimation of the error rates and inference of sample composition until they converge on a jointly consistent solution. As in many machine-learning problems, the algorithm must begin with an initial guess, for which the maximum possible error rates in this data are used (the error rates if only the most abundant sequence is correct and all the rest are errors).

```
err <- learnErrors(drp, errorEstimationFunction=PacBioErrfun, BAND_SIZE=32, multithread=TRUE) #
 10s of seconds
```

```
## 123895827 total bases in 84860 reads from 4 samples will be used for learning the error rate
s.
```

```
saveRDS(err, file.path(base_path, "v5_long_read_tutorial/Pacbio/RDS/Fecal_err.rds"))
plotErrors(err)
```



The error rates for each possible transition (A→C, A→G, …) are shown in the plot above. Points are the observed error rates for each consensus quality score. The black line shows the estimated error rates after convergence of the machine-learning algorith. Here the estimated error rates (black line) are a good fit to the observed rates (points), and the error rates drop with increased quality as expected. Everything looks reasonable and we proceed with confidence.

Now that the error rates are learned we will finally run the star of the show dada2; utilizing both of the output files generated from our previous steps. There are a myriad of options included in the dada2 package, the most important of which are included in the step below but in the negative (option is not turned on). To turn on these options follow the directions included in the description of each of the options below.
The first of the afore mentioned options is the “pool” option. In this option you can decide whether or not to pool your samples when running dada2. This allows for sharing information across samples and can increase the specificity of the dada2 algorithm to resolve ASVs that maybe in low abundance. In order to turn on this option you will need to change “pool=FALSE” to “pool=TRUE”. You can also use the option “pool=psuedo”, which inputs all ASVs detected in at least two samples in the first sample processing step as priors to the second step. Another option to mention is the “OMEGA_A” option which sets the threshold for when DADA2 calls unique sequences significantly overabundant, and therefore creates a new partition with that sequence as the center. Default is 1e-40, which is a conservative setting to avoid making false positive inferences, but which comes at the cost of reducing the ability to identify some rare variants. To change this parameter to be less conservative and more sensitive to detecting rare variants change the number in "OMEGA_A=(1*10^-40)" to a larger number. The last option that we will mention here is “DETECT_SINGLETONS”, which allows reporting ASVs which originated from more one read (default is to only report ASVs that originate from 2 or more reads). This will increase the sensitivity of the dada2 algorithm particularly in low abundance environments. To turn this option on change “DETECT_SINGLETONS=FALSE” to “DETECT_SINGLETONS=TRUE”. For more indepth information regarding these options you can read the dada2 documentation found here:

https://www.bioconductor.org/packages/devel/bioc/manuals/dada2/man/dada2.pdf (https://www.bioconductor.org/packages/devel/bioc/manuals/dada2/man/dada2.pdf) For examples of how these options work you can look at the sensitivity dada2 workflow found here: https://benjjneb.github.io/dada2/pseudo.html (https://benjjneb.github.io/dada2/pseudo.html)

```
dd <- dada(drp, err=err, BAND_SIZE=32, multithread=TRUE, pool=FALSE,OMEGA_A=(1*10^-40), DETECT_S
INGLETONS=FALSE) # seconds
```

```
## Sample 1 - 17436 reads in 4669 unique sequences.
## Sample 2 - 26605 reads in 5179 unique sequences.
## Sample 3 - 24081 reads in 5326 unique sequences.
## Sample 4 - 16738 reads in 3816 unique sequences.
## Sample 5 - 28623 reads in 6661 unique sequences.
## Sample 6 - 24831 reads in 5788 unique sequences.
## Sample 7 - 21759 reads in 5748 unique sequences.
## Sample 8 - 18700 reads in 5343 unique sequences.
## Sample 9 - 13258 reads in 3772 unique sequences.
## Sample 10 - 72940 reads in 22309 unique sequences.
```

```
saveRDS(dd, file.path(base_path, "v5_long_read_tutorial/Pacbio/RDS/Fecal_dd.rds"))
cbind(ccs=prim[,1], primers=prim[,2], filtered=track[,2], denoised=sapply(dd, function(x) sum(x
$denoised)))
```

```
##                         ccs primers filtered denoised
## R11_1_P3C3.fastq.gz 19627   17436    17436    17242
## R3_1_P3C3.fastq.gz  30983   26605    26605    26490
## R3_2_P3C3.fastq.gz  27505   24477    24081    23846
## R3_3_P3C3.fastq.gz  18783   16738    16738    16665
## R9_1_P3C3.fastq.gz  32133   28623    28623    28396
## R9_1B_P3C3.fastq.gz 28156   24832    24831    24591
## R9_2_P3C3.fastq.gz  24760   21759    21759    21403
## R9_3_P3C3.fastq.gz  21621   18701    18700    18182
## R9_4_P3C3.fastq.gz  15169   13481    13258    12875
## Zymo_P1C1.fastq.gz  77453   73057    72940    72831
```

```
st <- makeSequenceTable(dd); dim(st)
```

```
## [1]   10 1431
```

As you can see the output table above includes read tracking, such as read numbers for ccs, primer containing, filtered, and denoised reads.

Identifying and Removing chimeras

While the core dada method corrects substitution and indel errors, chimeras still remain. Fortunately, the accuracy of sequence variants after denoising makes identifying chimeric ASVs simpler than when dealing with fuzzy OTUs. First we need to identify how many if any bimeras ther are, as well as the percentage of them in the overall reads:

```
bim <- isBimeraDenovo(st, minFoldParentOverAbundance=3.5)
# Higher MFPOA to avoid flagging intra-genomic variants
table(bim)
```

```
## bim
## FALSE  TRUE
##  1393    38
```

```
sum(st[,bim])/sum(st)
```

```
## [1] 0.004563444
```

As you can see there are 38 bimera present in the dataset, which comes out to a 0.6316% bimera rate in the reads.

Assigning Taxonomy

Now that we have processed our data via dada2, we can plot and pull some interesting conclusions from the data. The first thing to be done is assign taxonomic assignments to the generated ASVs. This is done by utilizing the assignTaxonomy method inate to the dada2 package. This method uses trained data to inform the algorithm which taxa belong to which taxonomic assignments. To accomplish this we need a source taxonomy file which for this sample dadset can be found here: https://zenodo.org/record/801832/files/silva_nr_v128_train_set.fa.gz?download=1 (https://zenodo.org/record/801832/files/silva_nr_v128_train_set.fa.gz?download=1). You will then have to move this file into the tax folder found in the tutorial folder. Then we run the main function and voilà! We have taxonomies!

```
tax_path <- file.path(base_path, "v5_long_read_tutorial/tax/silva_nr_v128_train_set.fa.gz")
tax <- assignTaxonomy(st, tax_path, multithread=TRUE)
tax[,"Genus"] <- gsub("Escherichia/Shigella", "Escherichia", tax[,"Genus"]) # Reformat to be com
patible with other data sources
head(unname(tax))
```

```
##      [,1]       [,2]              [,3]                  [,4]
## [1,] "Bacteria" "Verrucomicrobia" "Verrucomicrobiae"    "Verrucomicrobiales"
## [2,] "Bacteria" "Firmicutes"      "Negativicutes"       "Selenomonadales"
## [3,] "Bacteria" "Proteobacteria"  "Gammaproteobacteria" "Enterobacteriales"
## [4,] "Bacteria" "Firmicutes"      "Bacilli"             "Bacillales"
## [5,] "Bacteria" "Firmicutes"      "Bacilli"             "Bacillales"
## [6,] "Bacteria" "Firmicutes"      "Bacilli"             "Bacillales"
##      [,5]                  [,6]
## [1,] "Verrucomicrobiaceae" "Akkermansia"
## [2,] "Acidaminococcaceae"  "Acidaminococcus"
## [3,] "Enterobacteriaceae"  "Salmonella"
## [4,] "Bacillaceae"         "Bacillus"
## [5,] "Staphylococcaceae"   "Staphylococcus"
## [6,] "Listeriaceae"        "Listeria"
```

Data Management

In this next section we will be extracting the file sample names for downstream visulaizations and then saving the processed data as R objects for future analyis. This will allow easy access for furture workflows.

```
sample.names <- sapply(strsplit(fn_r, "/"), function(x) paste(x[length(x)], sep="_"))
sample.names <- sapply(strsplit(sample.names, "_"), function(x) paste(x[1],x[2], sep="_"))
sample.names<- gsub("_", ".", sample.names)
sample.names<- gsub("^R", "R_", sample.names)
sample.names <- gsub(".fastq.gz", "", sample.names)
rownames(st) <- sample.names
sample.names
```

```
##  [1] "R_11.1"    "R_3.1"     "R_3.2"     "R_3.3"     "R_9.1"     "R_9.1B"
##  [7] "R_9.2"     "R_9.3"     "R_9.4"     "Zymo.P1C1"
```

```
#save R objects
saveRDS(st, file.path(base_path, "v5_long_read_tutorial/Pacbio/RDS/Fecal_st.rds"))
saveRDS(tax, file.path(base_path, "v5_long_read_tutorial/Pacbio/RDS/Fecal_tax_Silva128.rds"))

#relaod R objects
st <- readRDS(file.path(base_path, "v5_long_read_tutorial/Pacbio/RDS/Fecal_st.rds"))
tax <- readRDS(file.path(base_path, "v5_long_read_tutorial/Pacbio/RDS/Fecal_tax_Silva128.rds"))
```

Sample metadata loading

Import the metadata for the samples to facilitate downstream processing and visualizations

```
pac_path_metadata<-file.path(base_path, "v5_long_read_tutorial/fecal_metadata.csv")
ft <- sweep(st, 1, rowSums(st), "/")
df<-read.csv(file = pac_path_metadata)

df$SampleID <- gsub("_", ".", df$X)
df$X<- gsub("_", ".", df$X)
df$SampleID <- gsub("^R", "R_", df$SampleID)
df$X <- gsub("^R", "R_", df$X)
rownames(df) <- df$SampleID
#df <- df[sample.names2,-1]
head(df)
```

```
##             X weight SampleOrder Subject SampleID
## R_3.1   R_3.1 0.2035           1      R3    R_3.1
## R_3.2   R_3.2 0.2834           2      R3    R_3.2
## R_9.1   R_9.1 0.2205           1      R9    R_9.1
## R_11.1 R_11.1 0.1845           1     R11   R_11.1
```

Inspect E.coli

In the mock community datasets, we resolved the full complement of 16S sequence variants in E. coli and used them to precisely classify those E. coli strains as O157:H7 and K-12. Let’s see if we can achieve similar reconstruction of E. coli strains in these real fecal samples. This is done by retreiving the sequences and then checking the sequences against their taxonomic assinments. This is then printed out in a tabular format.

```
sq <- getSequences(st)
is.ecoli <- tax[,"Genus"] %in% "Esherichica/Shigella"
sqec <- sq[is.ecoli]
which(is.ecoli)
```

```
##  [1]    7   18   22   30   34   41   46   68   71   74   75   81   83   84   85
## [16]   90  459  471  497  682 1185
```

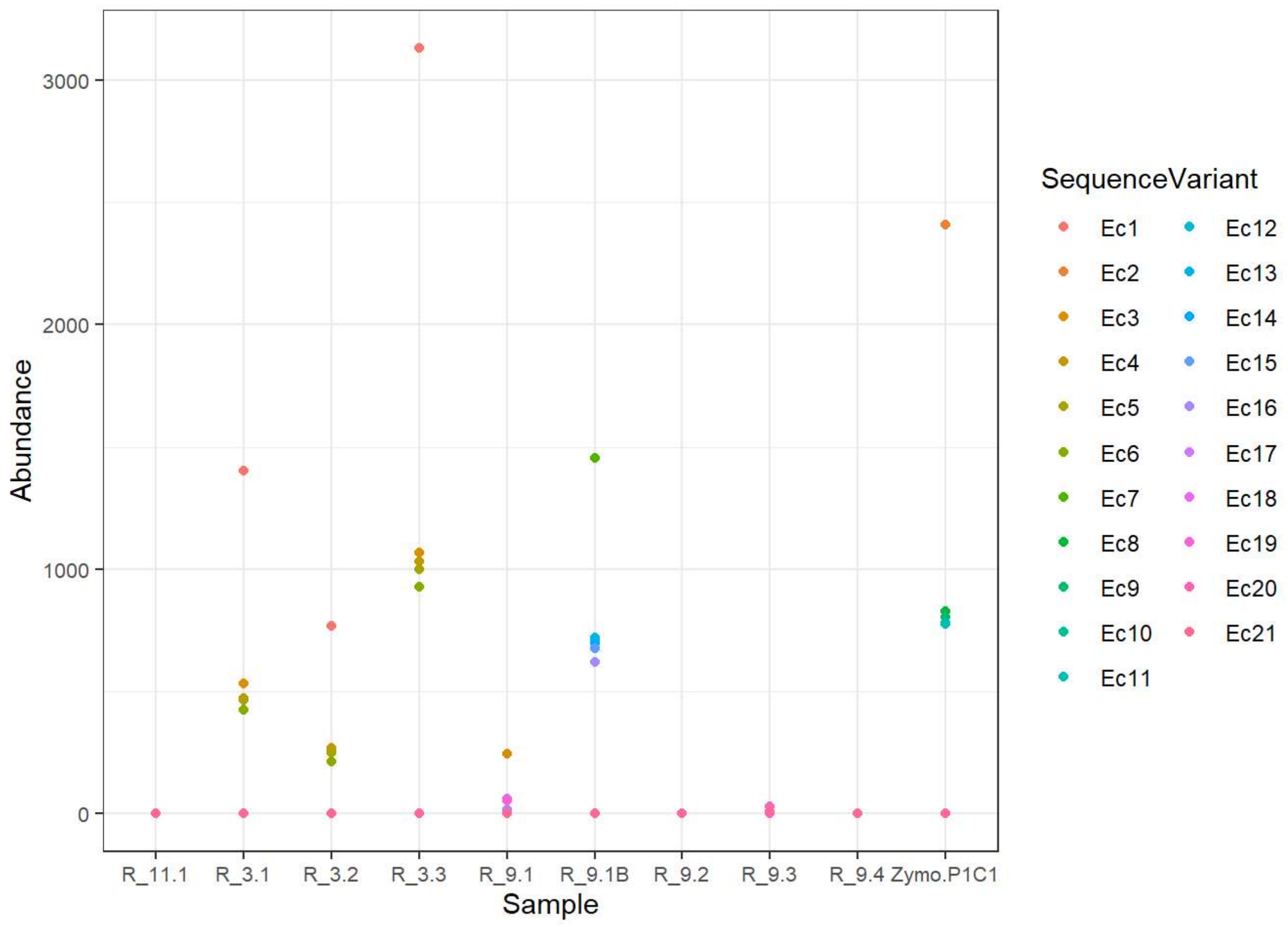
```
rowSums(st[,is.ecoli]>0)
```

```
##    R_11.1     R_3.1     R_3.2     R_3.3     R_9.1    R_9.1B     R_9.2     R_9.3
##         0         5         5         5        10         6         0         2
##     R_9.4 Zymo.P1C1
##         0         5
```

There are several ASVs that are identified as E.coli but since we expect 3-6 unique alleles per strain that probably only represents 3-4 strains.

Now we can vizualize the distribution of the E.coli variants

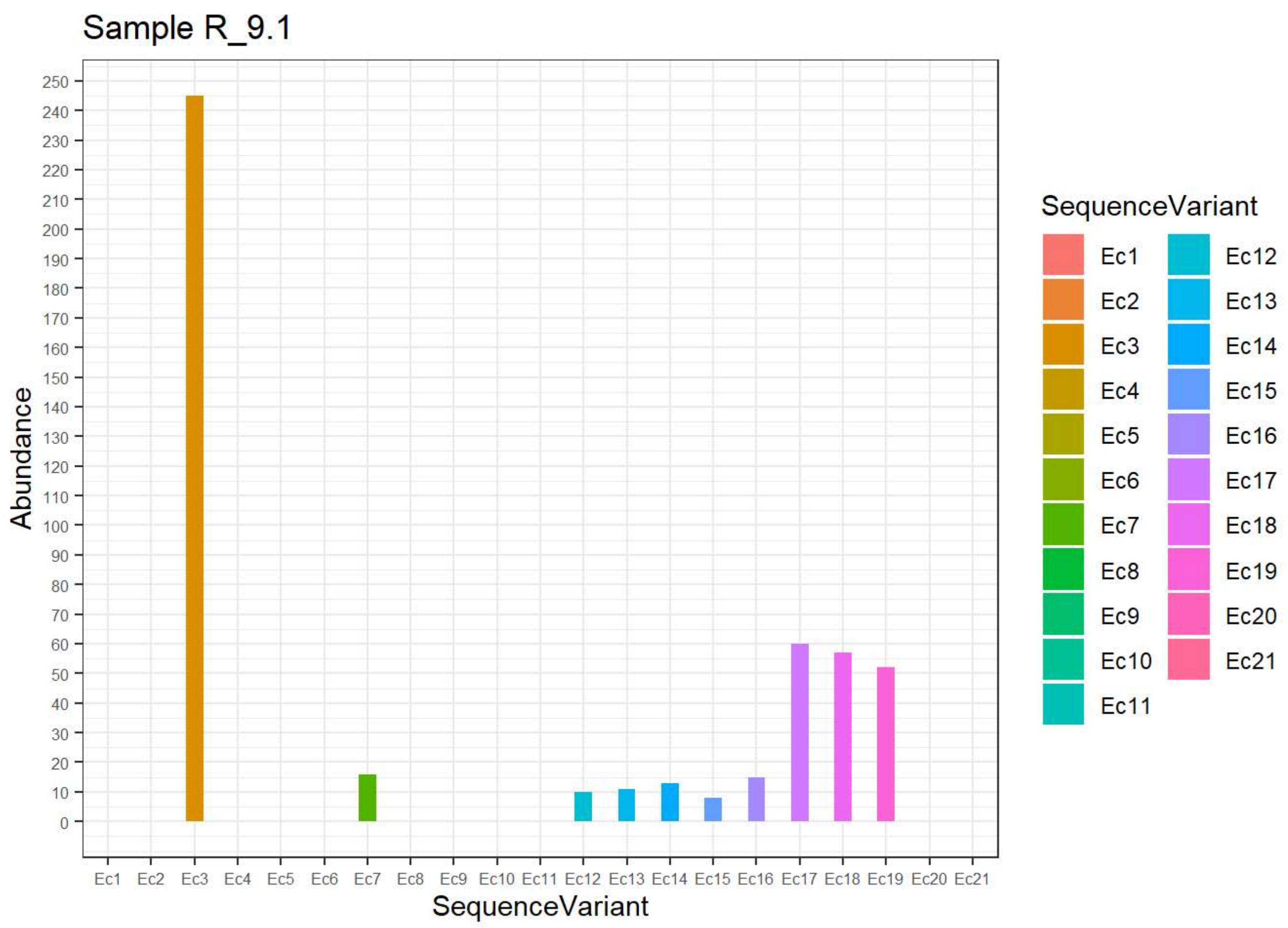
```
ecdf <- data.frame(st[,is.ecoli])
names(sqec) <- paste0("Ec", seq(ncol(ecdf)))
ecnames <- names(sqec); names(ecnames) <- sqec # map to ecnames from sequences
colnames(ecdf) <- names(sqec)
ecdf <- cbind(ecdf, Sample=rownames(st))
ecm <- melt(ecdf, id.vars="Sample", value.name="Abundance", variable.name="SequenceVariant")
plot<-ggplot(data=ecm, aes(x=Sample, y=Abundance, color=SequenceVariant)) + geom_point()
plot+theme(axis.text = element_text(size = 8))
```



Very clear 3:1:1:1:1 full complement signal in R_3, consistent over the time-course from R_3.1 to R_3.2 to R_3.3. R_9.1B also has a clear 2:1:1:1:1:1 full complement. R_9.1 is less clear because fo the lower abundances, but may have 2 distinct strains given the 10 total E. coli variants in that sample (see previous code block). Note that R_9.1 precedes R_9.1B from the same subject.

Lets look a bit closer at R_9.1, shall we?

```
isam <- "R_9.1"
plot<-ggplot(data=ecm[ecm$Sample == isam,], aes(x=SequenceVariant, y=Abundance, fill=SequenceVar
iant)) +
  geom_col(width=0.4) + scale_y_continuous(breaks=seq(0,250,10)) +
  ggtitle(paste("Sample", isam) )
plot + theme(axis.text = element_text(size = 6))
```



With high confidence, this suggest to me a high abundance strain with a 4:1:1:1 full complement of Ec2:Ec13:Ec14. With moderate confidence, this suggests to me a lower abundance strain with a 2:1:1:1:1:1 full complement of Ec6:Ec7:Ec8:Ec9:Ec10:Ec11. And, in fact, that is the strain because that was the strain subject R9 has in the next time-point!

The exact abundance of each strain in the sample can then be vizualized in data form:

```
ecdf["R_9.1",]
```

```
##       Ec1 Ec2 Ec3 Ec4 Ec5 Ec6 Ec7 Ec8 Ec9 Ec10 Ec11 Ec12 Ec13 Ec14 Ec15 Ec16
## R_9.1   0   0 245   0   0   0  16   0   0    0    0   10   11   13    8   15
##       Ec17 Ec18 Ec19 Ec20 Ec21 Sample
## R_9.1   60   57   52    0    0  R_9.1
```

Other detailed plots and vizuations can be seen here https://benjjneb.github.io/LRASManuscript/LRASms_fecal.html (https://benjjneb.github.io/LRASManuscript/LRASms_fecal.html)

This now concludes our tutorial! Throughout this workflow we have taken raw sequencing data, proccessed it using cutting edge methods, and then vizualized that data in a pleasing and nuanced manner. Now you are ready to go out into the world of long read analysis and try this workflow on your own data. Good luck and may the force be with you!