Raw Reads

Raw reads were provided in the format shown below:

@M05468:54:000000000-CRNFB:1:1102:18650:1000 1:N:0:TTAGGTCG+TGACAACC
NTAGTCAGCCGCGGTAATTCCAGCTCCAATAGCGTATATTAAAGTTGTTGCAGTTAAAAAGCTCGTAGTTGAACTTCGGG
GTCTACCCATTGGTCAGGCTTAATTGTCTGCACTGGTGAGATGGATCTCTACCTTCTGAGGAACTTTCATGCCATTTATTT
GGTGTGATGGGAAATCAGGACTGTTACTTTGAAAAAATTAGAGTGTTTAAAGCAGACTATCGTCTGAATACATTAGCATG
GAATAATGAAATAGGACGGACTGATTCTATTTCGTTGGTTTCTAGGATCACCGTAATGAT
+
#8ACCGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGFGGGGGGGGFFGGGGGGGGGGGGGGGG
GGGGGGGGGGGGGGGGGGGGGGGGGGGGGFGGGGGGGGGGGGGGGGGGGGGGGGGGGCFFGGGGGFGGGGGGGG
GGGGGGGGGGGGGGGGGGGGGGFFFEGGGGGGGGGGGGGG7FEFEGGCFGGGGFFGGGCFGDGEFG;DFGGGFGF
*GGGG,@C7@FFGCA@CFCFFFGGFF4*9AFB7>F6FFEF4*)6?DC3A6+:::+<C@9>FFGF355)3+;34A555;<FB01:6)*

The fastq format has read lengths of 301 bp and orients the reads with a heterozygous spacer and biological primers (AML2 or WANDA) at the start of the read. The heterozygous spacer varies from 1-5 bp. As the experiment randomizes R1 and R2, either a forward or reverse primer can be present in a read. Barcodes are also randomized in the index files. In our index files, barcodes contained in I1 were reverse compliments, while I2 were exact matches regardless of read orientation.

Demultiplexing

Reads and barcodes were distributed amongst four different files: R1, R2, I1, and I2. Our demultiplexing script accomplishes four key things; removing and appending the spacer to the header line, placing all sequences containing primer 1 into R1 and all sequences containing primer 2 into R2, removing the biological primers, and separating each read into the appropriate sample. A user adjustable tolerance to base pair mismatches is provided, however it does not account for indels. A user adjustable average base pair quality score per read is also provided.

Our demultiplexing algorithm takes the following parameters:
'-R', '--Read' 'File Name for Read 1 as a gzip file'
'-r', '--read', 'File Name for Read 2 as a gzip file'
'-I', '--Index' 'File Name for index 1 as a gzip file'
'-i', '--index','File Name for index 2 as a gzip file'
'-b', '--map', 'Mapping File as .csv '
'-P', '--forwardPrimer', 'Forward primer sequence'
'-p', '--reversePrimer', 'Reverse Primer Sequence'

'-t', '--tolerance' , 'Amount of mismatched nucleotides allowed in forwardPrimer and reversePrimer, must be less than 5 '
'-q', '--quality' 'quality score threshold'

The heterogeneity spacer sequences for each read are combined into a single UMI sequence. This UMI sequence is appended with an underscore to the first column of the FASTQ header. Since deduplication with UMI-tools requires UMI sequences of uniform length, UMI sequences (lengths ranging from 2 - 12 bp) are altered to a uniform length of 12 bp with a filler character `X`. The filler not being a biological base ensures a short UMI sequence will not be confused with a longer UMI sequence. The following is an example of an UMI appended to a FASTQ header:

```
@M05468:54:000000000-CRNFB:1:1102:21010:1005_CATGCNTCAX
2:N:0:GAAGTACC+CATCTGCT
```

In this case UMI sequence `CATGCNTCA` is 11 bp long, so a single `X` lengthens the tag to 12 bp.

Deduplication

A benefit of the heterozygous spacers is their utility in detecting duplicates. Deduplicating uses both UMI tags appended in the previous demultiplexing step and alignment with the UMI-tools `dedup` command.

First, Bowtie2 is used to align reads to a reference FASTA. The reference FASTA is the same that will be used in the subsequent DADA2 analysis. In our case, we used a MaarjAM reference of the SSU region of AM fungi. After alignment, sam files are converted to sorted and indexed BAM files with samtools. Through whole alignment workflow, the UMI tag is appended to the first field in the SAM format. With correctly tagged BAM files, UMI tools `dedup` is run with the flag --paired and the --method set to `unique`. The unique method groups reads by their exact UMI sequences and filters a single read with the highest alignment score. If multiple reads have the same alignment score, UMI-tools picks the first read encountered. The result is a deduplicated BAM file for each sample.

Deduplication inevitably leads to some instances of singletons, which are not compatible with DADA2. Therefore, reads are sorted by read name with samtools sort and then divided into three separate FASTQ's (forward, reverse, & singelton) with samtools fastq. The singelton reads are discarded and the deduplication step is complete. The reads are ready to be filtered and analyzed in DADA2.

DADA2

For processing in DADA2 the group wrote a script called `dada2pipeline.R`. The script takes a directory of reads through the entire workflow outlined in the DADA2 tutorial version 1.12 (https://benjjneb.github.io/dada2/tutorial.html). After the workflow the generated amplicon sequencing variant (ASV) tables with added taxonomy columns are variance transformed. Every input read should contain a single ASV.

The script requires five inputs, which can be added as flags in bash. First, a directory containing reads of interest. Reads should be separated into files by sample and follow the specifications in DADA2. Second, a directory name where output files can be stored. Third, a taxonomic reference FASTA (ideally the same one used in the deduplication step). Fourth, a metadata file with sample names in the first column followed by variables of interest. Sample names must exactly match the file names in the input reads. Fifth, a formula used by DESeq2 in variance transformation. From our experience, this formula can be just about anything. As long as there is a single discrete variable, the differences between output ASV tables are miniscule.

At the beginning of the pipeline, reads are filtered and trimmed with the `filterAndTrim` function at the following settings. Truncation quality was set to a score of two. This means reads are truncated the moment base falls below the cut-off. Minimum reads length after truncation was set to 50 bp, and the maximum estimated error to a score of 2. Based on an average of ten quality scores, `filterAndTrim` will discount a read when it's average is below the designated max error. After `filterAndTrim` low quality reads are either shortened or removed.

The next step in the pipeline is to create a list of distinct ASV's and count their number per sample. In order to distinguish similar ASV's across, the 'learnErrors' function is used to predict the error rates in the sequencing data. The prediction is then used to judge whether or not an ASV is unique. A second deduplication step is performed without the assistance of UMI's in the function `derepFastq`. The `dada` function is run on forward and reverse reads separately at default parameters. The `dada` forward and reverse ASV's are then merged with `mergePairs`. Since the SSU region is ~550 bp, merged 301 bp ASV's should contain most of the amplicon sequence. Any ASV's that did not overlap at least 20 bp are removed at this step. Lastly chimeras are removed with `removeBimeraDenovo` at default settings. The result is abundance counts for each ASV in a tab-delimited file. This file is called an ASV table.

After ASV abundances are generated and filtered, the pipeline adds taxonomic data with the fasta reference input. `assignTaxonomy` is run with "tryRC" set to True. ASV's and their reverse complements (`tryRC`) are aligned to the fasta reference. The result is an ASV table with counts of sequence abundances per sample and stratified taxonomic assignments.

Variance transformation normalizes count data by size factor. This normalization helps account for bias towards samples introduced by library prep and sequencing across different samples. In the variance stabilization portion of the script, ASV tables are combined with metadata to generate DESeq2 objects. Generating DESeq2 objects, requires an “experimental design” in a formula. The experimental design can contain any metadata variable of interest. After `getVarianceStabalizedData` is run at default, the data is converted back to an ASV table. The only difference between the new ASV table is values in abundance counts.

The `dada2pipeline.R` script generates five files in the output directory. A “track_reads.csv” provides the total number of reads for each sample at each step of DADA2 (input, filtered, denoised, & merged). “ASV_summary.tsv” is a table of the sum ASV counts for each sample. “ASV_table.tsv” is the ASV table combined with taxonomy produced in the DADA2 step. “ASV-VT.tsv” a variance transformed version of “ASV_table.tsv”. Lastly, “physeq_object.RDS” is a phyloseq object of merged variance transformed counts, taxonomy, and the metadata for downstream analysis.