FlowClus User Guide

The following is an exploration of the uses of FlowClus, a program that filters and denoises amplicon reads produced by Roche-454 and Ion Torrent sequencing technologies. We will highlight some of the more commonly used options of FlowClus; a complete description of all the parameters can be found in the README that accompanies the program. We will follow the analysis given in Gaspar and Thomas [1].

I. Input files

FlowClus requires two input files: a flowgram file and a master file.

The flowgram file is the text version of the binary .sff file that is produced by the sequencing machine. It contains the flowgrams, sequences, and quality scores for each read. The binary-text conversion is best performed by the 454 Tools program sffinfo, but we have also been able to use process_sff.py in QIIME [3] for this purpose.

Note that it is critically important that the flowgram file follow the proper format. The data labels must be exact matches, or FlowClus will complain. One can alter the data labels in FlowClus.h and recompile the program, but we strongly recommend that the user obtain a correctly formatted sff.txt file using sffinfo or process_sff.py.

Another cautionary note: when analyzing Ion Torrent reads, FlowClus will sometimes produce no output, even though the flowgram file seems to be properly formatted. This is due to the fact that the "Clip Qual Right" values in the flowgram file are all listed as 0, incorrectly. To remedy this, we have written a script, addCQR.pl, which fixes the flowgram file.

In this workflow, we will be using the sff.txt file from Gaspar and Thomas [2], available at the NCBI Sequence Read Archive, accession SRR653182.

The second input, the master file, lists the primer and mid tag sequences, along with corresponding primer and sample names. The primer sequences can contain IUPAC ambiguous DNA codes. Sample names can be repeated for amplicons sequenced bidirectionally (or multiple amplicons for each sample). The master file for our dataset is given in Appendix 1.

For the convenience of those used to working in QIIME [3], we have written a script (convertMapping.pl) that will produce a FlowClus-compatible master file from one or more QIIME mapping files. Note that, since QIIME mapping files do not contain primer names, the primers are assigned numerical names ("0", "1", "2", etc) by convertMapping.pl. One may wish to edit the resulting master file to specify more informative names.

II. Filtering

The filtering process serves many purposes: selecting reads that have a valid mid tag and primer; truncating the ends of reads (or removing entire reads) that are likely to contain numerous errors; preparing the flowgrams for the denoising process.

With FlowClus, the filtering process begins by matching the 5' sequence of a read to a mid tag and primer in the master file. One may specify a number of mismatches to these sequences to allow (via the -em and -ep parameters), but the default is to require an exact match (-em 0 -ep 0). Once a read has a mid tag - primer match, it is further analyzed according to user-selected filtering parameters. The default is to keep the read as is, with no additional filtering. Therefore, it is entirely up to the user to choose filtering options. Fortunately, FlowClus provides a wide variety of criteria to choose from, based on the read's sequence, quality scores, and flowgram.

For the sample dataset, we can choose parameters similar to those recommended in the QIIME denoising pipeline:

```
$ ./FlowClus -m master.csv -i SRR653182.sff.txt -a
    -l 200 -L 600 -n 0 -r -er 1 -wl 50 -wq 25.0
    -c filterCount.txt -cv filterVerb.txt -st
```

Let us take a closer look at the parameters:

`-a`	Filter only (do not denoise).
`-l 200`	Eliminate a read shorter than 200bp. Also, if any other filtering criterion results in a read shorter than 200bp, eliminate it.
`-L 600`	Eliminate a read longer than 600bp.
`-n 0`	Truncate a read immediately prior to the first N (ambiguous nucleotide) in the sequence. (Eliminate if the resulting read is shorter than 200bp.)

`-r -er 1`	Truncate a read immediately prior to the reverse primer, if it is found. Allow 1 mismatch.
`-wl 50 -wq 25.0`	Truncate a read immediately prior to a window of 50 quality scores whose average is below 25. (Eliminate if the resulting read is shorter than 200bp.)
`-c filterCount.txt`	Provide an accounting of the mid tag - primer matches, and the number of truncations and eliminations, to filterCount.txt.
`-cv filterVerb.txt`	Provide a verbose description of what happened to each read that matched a mid tag - primer during the filtering process, to filterVerb.txt.
`-st`	Print status updates while the program is running. (important for those of us who can't stand staring at a blinking cursor, wondering if anything is actually happening)

For an output, FlowClus produces cleaned flowgram files that are ready for denoising. Since reads originating from different primers cannot be denoised together, separate files are created for each primer analyzed. In our example, that consists of four files: 341F.flow, 926R.flow, 968F.flow, and 1401R.flow.

The format of the cleaned flowgram files (such as 341F.flow) is the following:

```
800 TACGTACG
GZIPSVE01C46MC-sample2 1.08 0.98 0.04 0.00 3.75 0.14 1.93 0.01 0.12 3.70 0.03
GZIPSVE01CTE9O-sample5 1.09 0.96 0.06 0.03 3.66 0.08 1.83 0.03 0.12 1.03 1.00
GZIPSVE01D9776-sample3 1.15 1.10 0.11 0.00 3.99 0.17 1.98 0.06 0.16 4.04 0.14
GZIPSVE01DUBBA-sample6 1.09 1.07 0.09 0.00 4.02 0.11 2.01 0.03 0.13 3.79 0.08
GZIPSVE01A7XBW-sample6 1.03 1.06 1.04 0.04 2.80 0.08 2.03 0.06 0.09 1.02 0.04
GZIPSVE01DHZ3Q-sample13 1.03 0.98 0.07 0.00 0.98 0.04 1.09 0.00 1.96 0.05 1.9
GZIPSVE01EQDPB-sample4 1.09 1.01 0.09 0.02 3.96 0.19 1.89 0.11 0.14 4.04 0.10
GZIPSVE01BBIXS-sample1 1.07 1.05 0.08 0.06 3.97 0.10 1.97 0.00 0.10 1.06 1.15
GZIPSVE01BMUF5-sample2 1.12 1.03 0.10 0.01 4.05 0.18 1.98 0.00 0.15 3.78 0.02
GZIPSVE01C4COV-sample1 1.07 1.04 0.10 0.01 4.01 0.11 1.95 0.04 0.12 1.03 1.20
GZIPSVE01DTZ9F-sample14 1.02 1.04 0.03 0.00 3.71 0.09 1.91 0.00 0.15 0.92 1.1
GZIPSVE01APRYN-sample1 1.09 1.11 0.09 0.00 4.19 0.12 1.98 0.04 0.12 0.99 1.20
GZIPSVE01DGLG7-sample4 1.04 1.07 0.04 0.00 3.92 0.06 1.88 0.00 0.08 1.02 0.86
GZIPSVE01DGZY8-sample1 1.05 1.05 0.11 0.00 4.05 0.09 1.92 0.04 0.11 0.99 1.16
GZIPSVE01CV2AP-sample1 1.20 0.93 0.15 0.01 3.85 0.14 1.91 0.00 0.11 1.12 1.08
GZIPSVE01DI1S1-sample4 1.12 0.92 0.10 0.00 3.64 0.11 1.93 0.00 0.08 1.01 0.89
GZIPSVE01A55PA-sample4 1.11 0.97 0.12 0.00 3.66 0.18 1.83 0.06 0.13 3.50 0.04
GZIPSVE01CNJEH-sample4 1.02 0.97 0.08 0.03 3.93 0.17 1.90 0.09 0.13 4.18 0.05
```

The header line tells FlowClus' denoiser that the reads were derived from a sequencing run consisting of 800 flows (in this case, Roche-454 Titanium), and that the flow order was regular ("TACG", repeated). Each remaining line gives a read's accession, the read's sample name, and the trimmed flowgram. The flows corresponding to the mid tag and primer have been removed from the 5' end of the flowgram, and the flows corresponding to the truncated 3' end, if applicable, have similarly been removed.

Other output files, such as a fasta file, are optional at this stage. In our example, we specified two output files containing information about the filtering process. Let us examine the first file, filterCount.txt.

The top section of filterCount.txt gives counts of the reads that were eliminated or truncated due to each of the specified filtering criteria.

	Min. sequence length (200)	Max. sequence length for elimination (600)	Max. ambiguous bases allowed before truncation (0)	Reverse primer removed	Min. window quality score (length=50, qual=25.0)	Total
Reads eliminated	2226	0	107	n/a	3702	6035
Reads truncated	n/a	n/a	1359	12486	15199	29044

Following this is a section listing, for each primer, how many reads matched a mid tag and primer, and how many reads were ultimately printed (i.e. passed the filtering step).

Reads analyzed	1280719				
	341F	926R	968F	1401R	Total
Mid-primer matches	8087	8877	10962	12701	40627
Reads printed	5988	7922	9090	11592	34592

This is already a wealth of information that should be examined before moving on the the final section of the report. A total of 6,035 reads (14.9% of the 40,627 that had a mid tag - primer match) were removed in the filtering process. A majority of the eliminations were due to a bad quality window that left the read shorter than the 200bp minimum length, and most of the rest were because the read was too short to begin with. Of the 34,592 reads that passed filtering, 29,044 (84.0%) were truncated, mostly due to a bad quality window or the removal of the reverse primer. Now, if, for example, one were uncomfortable with the bad quality window criterion having such a dominant effect on the reads, one would re-run the filtering step and choose different values for this criterion (such as -wl 50 -wq 20.0) or perhaps different criteria altogether. Note that choosing no criteria would result in all 40,627 reads passing this step.

Another thing to notice is that primer 341F had the fewest mid tag - primer matches, yet it had the most reads eliminated of the four primers. This brings us to the

final section of the report, which breaks down the numbers in the first two sections by primer and sample:

341F								
	Min. sequence length (200)	Max. sequence length for elimination (600)	Max. ambiguous bases allowed before truncation (0)	Reverse primer removed	Min. window quality score (length=50, qual=25.0)	Total		
sample1								
Reads eliminated	272	0	2	n/a	627	901	Mid-primer matches	3370
Reads truncated	n/a	n/a	316	0	1395	1711	Reads printed	2469
sample2								
Reads eliminated	17	0	1	n/a	52	70	Mid-primer matches	272
Reads truncated	n/a	n/a	13	0	140	153	Reads printed	202
sample3								
Reads eliminated	27	0	0	n/a	71	98	Mid-primer matches	323
Reads truncated	n/a	n/a	4	0	160	164	Reads printed	225
sample4								
Reads eliminated	86	0	1	n/a	199	286	Mid-primer matches	1521
Reads truncated	n/a	n/a	11	0	871	882	Reads printed	1235
341F Totals								
Reads eliminated	649	0	12	n/a	1438	2099	Mid-primer matches	8087
Reads truncated	n/a	n/a	393	2	3893	4288	Reads printed	5988

It is apparent, when comparing these totals to those of the other primers, that the bad quality window criterion was responsible for the excessive number of 341F reads that were eliminated. Another notable aspect in the report is the fact that the 12,486 reads truncated due to the reverse primer were almost entirely from the 968F and 1401R primers. This is not surprising, though: the 968F-1401R amplicon is much shorter than the 341F-926R one.

One thing lacking in the above report is quantification of read lengths. That is, a read that was truncated of its last 100bp due to a bad quality window counts the same as one that was truncated of just 1bp. Such information can be gleaned from the other report produced, filterVerb.txt. This verbose report lists, for each of the 40,627 reads, how and why it was modified in the filtering step:

Read	Sample	Primer	Outcome	Criterion	Length before	Length after
GZIPSVE01DVSIO	sample9	341F	Eliminated	Min. sequence length (200)	67	
GZIPSVE01A11EA	sample8	1401R	Eliminated	Min. sequence length (200)	92	
GZIPSVE01CPDSL	sample1	1401R	Truncated	Reverse primer removed	479	428
GZIPSVE01B7J27	sample9	341F	Eliminated	Min. sequence length (200)	56	
GZIPSVE01C8IUP	sample1	968F	Passed			478
GZIPSVE01BIW8F	sample5	926R	Eliminated	Min. sequence length (200)	86	
GZIPSVE01EFB9C	sample6	1401R	Truncated	Reverse primer removed	479	427

This can further help to quantify the effects of the selected filtering criteria.

In the filtering step, as reflected in these reports, a given read is either eliminated, truncated, or neither ("Passed"). If multiple criteria would have eliminated or truncated a read, the criterion credited is the first in the categories listed in Appendix 2 (Sequence, Quality Scores, Flowgram). In analyzing the sequence or flowgram of a read, the read must first pass any length restrictions. After this, the sequence or flowgram is examined 5′ to 3′, and the criterion that is violated first is credited with the elimination or truncation.

III. Denoising

FlowClus attempts to correct sequencing errors by analyzing the cleaned flowgram files produced by the filtering step. The most important choice is whether to denoise by clustering or by using a trie data structure.

A. Clustering

This method of denoising involves clustering flowgrams that are identical, or at least so similar that the only differences are due to sequencing noise. FlowClus assigns a flowgram to a cluster if its flow values are the same as those of the cluster, to within some margin of error. The margin of error, referred to as the "denoising distance," is set by the user.

Let us denoise the sample dataset:

```
$ ./FlowClus -m master.csv -b -j 0.50 -ch -x -d misses.csv
    -st
```

This translates into the following:

`-b`	Denoise only (using previous filtering output).
`-j 0.50`	Use a constant denoising distance of 0.50.
`-ch`	Produce files that can be used for de novo chimera checking.
`-x`	Format the output fasta file to be QIIME-compatible.
`-d misses.csv`	Provide an accounting of the denoising "misses" (described below) to misses.csv.
`-st`	Print status updates while the program is running.

The method of denoising was not specified because the default is clustering. The flowgram files were not specified because, by default, FlowClus searches for files named <primer>.flow and will complain if it cannot find them.

Several output files are produced by FlowClus, including the main output fasta file. But before reviewing them, let us explore the denoising distance parameter.

The choice of a denoising distance is extremely important. When FlowClus analyzes a flowgram, it compares the individual flow values to those of a cluster. If any of the pairs of flow values are further away from each other than the denoising distance, the flowgram does not join that cluster. So with our denoising distance of 0.50, for example, query flow values between 0.59 and 1.59 were not considered significantly different from a cluster's flow value of 1.09. This distance was constant for all cluster flow values, so the range for a cluster flow value of 4.13 was from 3.63 to 4.63. Note that a constant value of 0.50 is similar to how the 454 software interprets a flow value, except that 454 calls bases using integers as its reference values, and FlowClus uses the cluster's flow values.

It is well known that longer homopolymers are more prone to noise, so using a constant distance may not be optimal (i.e. it may be better to have a larger range for 4.13 than for 1.09). Balzer *et al.* [4], in building a read simulator, calculated standard deviations that gradually increase for each homopolymer length. We have extended their data to produce stddev.txt, which lists distances for each flow value. With FlowClus, one can denoise using a constant multiple of these distances, specified by the -k parameter. A good starting value would be -k 5.

If that still is not good enough, one can produce a file similar to stddev.txt that contains custom distances for each flow value. One can specify either a single distance, or two distances (comma-separated) if one wants asymmetric intervals (for example, if the distances for the flow value of 0.58 are given as "0.35,0.56", the interval of query flow values will range from 0.23 to 1.14 [assuming -k 1 is specified]). Note that there must be a distance (or distances) given for each possible flow value analyzed; this extends from 0.00 to the maximum flow value specified by the -u parameter (by default, this is 19.99, meaning 2000 distances must be given in the file).

So, one can specify denoising distances in three different ways, but how does one know that a certain distance (or set of distances) is a good choice, or is better than others? We recommend two methods for making such a determination. First, it is important to understand that FlowClus is designed to correct only sequencing errors. It is well established that sequencing errors in Roche-454 and Ion Torrent consist

almost exclusively of insertions and deletions, with substitutions being much rarer. Therefore, one should expect that the process of denoising by FlowClus should cause far more insertions and deletions than substitutions. So, the first method we recommend is to quantify how the reads have been changed by denoising. This is the same method we established for evaluating other denoising pipelines [2]. In fact, the script AlignClusMus.pl published with that paper can be used directly:

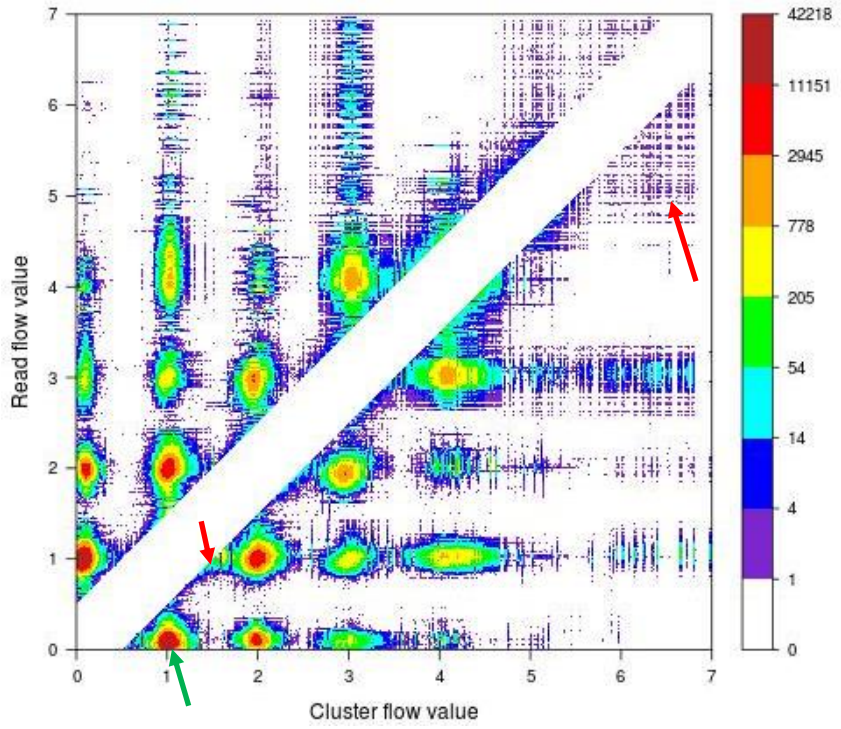
```
$ perl AlignClusMus.pl filter.fasta denoised.fasta
```

This script aligns the reads from filter.fasta (produced by the filtering step of FlowClus, if we had specified -e filter.fasta) to the corresponding reads in denoised.fasta, using an overly complicated method involving ClustalW and MUSCLE. A lengthy report detailing the differences is produced by AlignClusMus.pl, along with some summary statistics:

```
Number of reads analyzed: 34592
3' gap (SD)   Subs    Ins     Del      Total
0.3 (0.5)     46      1740    2601     4387
```

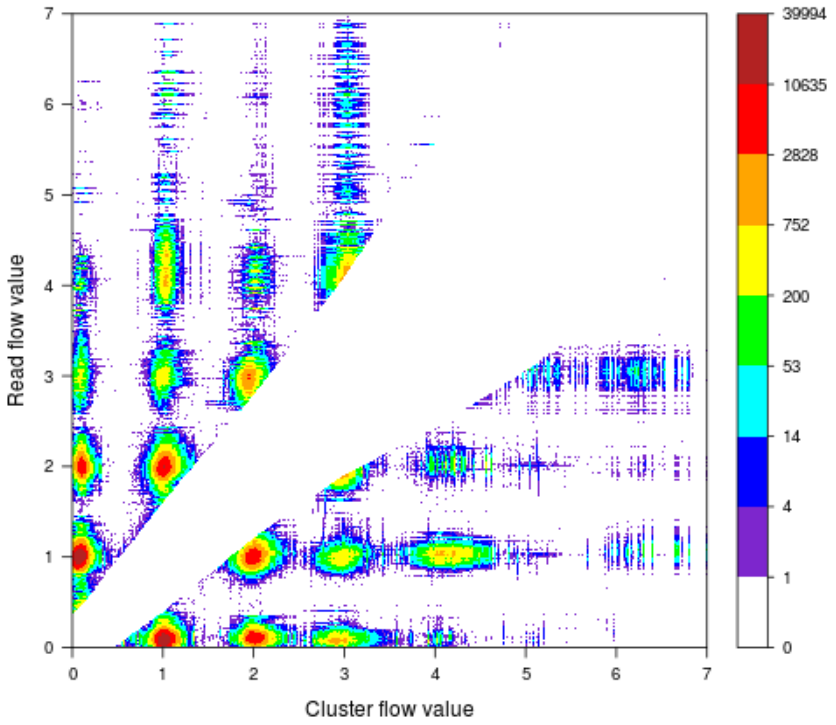
This is good: only 1% of all the changes made by FlowClus were substitutions. (The 3' gap refers to the number of bases removed (or added) to the 3' end of a read; this number may be large and negative when comparing the original reads to those after filtering, but it should always be close to 0 when comparing the filtered reads to those denoised by FlowClus.)

The second method for evaluating the outcome of denoising is to examine the denoising "misses" printed to misses.csv. When FlowClus is clustering the flowgrams, it records the flow values of the cluster and the read whenever they are judged as being significantly different (again, based on the denoising distance). One can use the R script levelplot.r to visualize these misses as -- you guessed it! -- a levelplot:



The white stripe down the middle indicates where there were no misses, because all those flow values were judged as not being different (with a constant denoising value of 0.50). Most of the misses are grouped around local maxima corresponding to integral flow values (e.g. the green arrow, where the cluster had a flow value of 1 and the read had a flow value of 0). This is good; it is likely these flow values are distinct, and thus those reads should not have joined those clusters. On the other hand, the red arrows point to a couple places that indicate that the denoising distance could have been increased. The lower red arrow points to a small region where the cluster flow values were around 1.40 - 1.60 and the read flow values were just below the 0.50 distance, at 0.80 - 1.00. The upper red arrow points to a region of larger flow values, which are known to be inherently more noisy in these sequencing technologies. Although the number of misses in this region is low (in the purple-blue 1-14 range), it is probably not a good idea to prevent a flowgram from joining a cluster just because it contains a flow value of 5.92 at a position where the cluster has a value of 6.76.

We would be remiss in not examining a levelplot made when using the distances taken from Balzer *et al*. In this case, we specified -k 5 (meaning a multiple of 5 of the distances in stddev.txt):



Now the misses at larger flow values have almost entirely disappeared, which is good. However, other potential problems have arisen: the local maximum at (3, 2) has been eroded, and there is a new local maximum in the vicinity of (0, 0.5).

There is much room for exploration with this visualization, but we must move on to other matters, such as the other output files produced by the denoising step.

Unlike the filtering step, an output fasta file (by default, "denoised.fasta") is always produced during the denoising step (otherwise, what would be the point of denoising?). If we had not specified -x, the fasta file would look like this:

```
>GZIPSVE02FWCOH
ATATCGCGAGCCTACGGGAGGCAGCAGTGGGGAATATTGGACAATGGGCGAAAGCCTGATCCA
>GZIPSVE01EET80
ATATCGCGAGCCTACGGGAGGCAGCAGTGGGGAATATTGGACAATGGGCGAAAGCCTGATCCA
>GZIPSVE02FZHQJ
ATATCGCGAGCCTACGGGAGGCAGCAGTGGGGAATATTGGACAATGGGCGAAAGCCTGATCCA
>GZIPSVE02H3HSW
ATATCGCGAGCCTACGGGAGGCAGCAGTGGGGAATATTGGACAATGGGCGAAAGCCTGATCCA
```

But we did use -x, so instead we have this:

```
>sample1_1 GZIPSVE02FWCOH 341F
TGGGGAATATTGGACAATGGGCGAAAGCCTGATCCAGCCATGCCGCGTGTGTGAAGAAGGTCT
>sample1_2 GZIPSVE01EET80 341F
TGGGGAATATTGGACAATGGGCGAAAGCCTGATCCAGCCATGCCGCGTGTGTGAAGAAGGTCT
>sample1_3 GZIPSVE02FZHQJ 341F
TGGGGAATATTGGACAATGGGCGAAAGCCTGATCCAGCCATGCCGCGTGTGTGAAGAAGGTCT
>sample1_4 GZIPSVE02H3HSW 341F
TGGGGAATATTGGACAATGGGCGAAAGCCTGATCCAGCCATGCCGCGTGTGTGAAGAAGGTCT
```

The mid tags and primers have been removed from the sequences, and the headers are of the form normally produced by split_libraries.py in QIIME. This means that the file can be used directly for OTU clustering, such as by pick_otus.py in QIIME. The fasta headers, in addition to containing the sample name, a unique number, and the read's accession, also have the primer name, which is not required by QIIME but may prove useful down the road.

Note that, since the denoising occurred for each primer separately, the fasta file lists all of the 341F reads first, followed in order by the 926R, 968F, and 1401R reads.

Before picking OTUs, one may wish to screen for chimeras first. Performing de novo chimera checking, such as by UCHIME [5], requires a fasta file with abundance information. Such files (one for each primer) are produced by FlowClus when the -ch option is specified.

Here is the beginning of 1401R.chfasta:

```
>GZIPSVE01EWGE6 /ab=1535/
TATTCACCGCGACATTCTGATTCGCGATTACTAGCGATTCCGACTTCACGCAGTCGAGTTGCAGAC
>GZIPSVE01BV7Z6 /ab=793/
TATTCACCGCGGCATTCTGATCCGCGATTACTAGCGATTCCGACTTCATGGAGTCGAGTTGCAGAC
>GZIPSVE01B6UCX /ab=621/
TATTCACCGCAGTATGCTGACCTGCGATTACTAGCGATTCCTCCTTCACGTAGGCGAGTTGCAGCC
>GZIPSVE01BJ2OI /ab=552/
TATTCACCGCGACATTCTGATTCGCGATTACTAGCGATTCCGACTTCACGCAGTCGAGTTGCAGAC
>GZIPSVE01CPDSL /ab=11/
TATTCACCGCGACATTCTGATTCGCGATTACTAGCGATTCCGACTTCACGCAGTCGAGTTGCAGAC
>GZIPSVE01DEGN0 /ab=165/
TATTCACCGCAGTATGCTGACCTGCGATTACTAGCGATTCCTCCTTCACGTAGGCGAGTTGCAGCC
```

To create this file, FlowClus prints the longest read in each cluster (after denoising), and indicates, in the header, the number of reads in the cluster. So, for example, read GZIPSVE01EWGE6 was the longest read in a cluster that contained 1,534 other reads. This file can be given to UCHIME directly.

The other files produced for chimera checking are the mapping files. These files list the representative reads for each cluster, followed by a comma-separated list of all the reads in that cluster. Here is the beginning of 1401R.chmap:

```
GZIPSVE01EWGE6 GZIPSVE02GGGGK,GZIPSVE02FSMS0,GZIPSVE02GZZBN,GZIPSV
GZIPSVE01BV7Z6 GZIPSVE02IS7AN,GZIPSVE01ADU1Q,GZIPSVE02G748U,GZIPSV
GZIPSVE01B6UCX GZIPSVE01E223K,GZIPSVE01C5QJO,GZIPSVE02HYE1V,GZIPSV
GZIPSVE01BJ2OI GZIPSVE02FH4BK,GZIPSVE02G0497,GZIPSVE02HN3QR,GZIPSV
GZIPSVE01CPDSL GZIPSVE02GLBQQ,GZIPSVE01CVEGM,GZIPSVE02FQVPG,GZIPSV
GZIPSVE01DEGN0 GZIPSVE02F8JXF,GZIPSVE01EUCCI,GZIPSVE01BSL0Q,GZIPSV
```

This is an important file, in order to be able to process the output from UCHIME. For example, suppose that read GZIPSVE01CPDSL were declared a chimera by UCHIME. Then it, and the other ten reads in its cluster, should be removed from the denoised fasta file. Those reads can be found by consulting the mapping file: GZIPSVE02GLBQQ, GZIPSVE01CVEGM, GZIPSVE02FQVPG, etc.

B. Trie

Denoising by trie is quite similar to denoising by clustering, except that it is completely different. Instead of comparing a flowgram to a set of clusters, FlowClus places a flowgram into the trie by comparing its flow values to those of the trie's nodes. If the flow values are not significantly different, the flowgram is added to that node; otherwise, it forms a new branch in the trie. In comparing the flow values, the same denoising distance parameters (-j and -k) apply.

The command-line is virtually identical to that specified for clustering, except for the inclusion of -tr (and specifying a different output fasta file):

```
$ ./FlowClus -m master.csv -b -j 0.50 -ch -x -d misses.csv
    -st -tr -o denoisedTrie.fasta
```

The first thing to notice is how much faster this runs than with clustering. It also uses far less memory. These attributes are not particularly important for this small dataset, but they are significant for those who need to denoise very large datasets (say, more than two million reads, which can take a day or more to denoise by clustering). This increased efficiency is achieved at the expense of precision; reads are analyzed a single time when they are placed into the trie (with clustering, every read is analyzed in each of two iterations). Also, the abundance information for chimera checking is not as precise; more on that later.

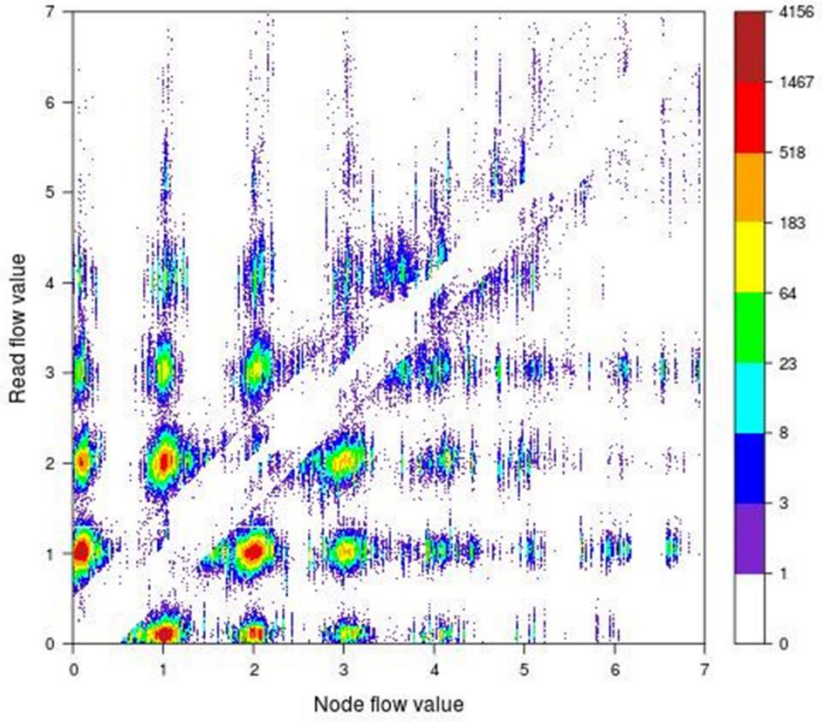
First, let us take a look at the output fasta file, denoisedTrie.fasta:

```
>sample6_1 GZIPSVE02GYH08 341F
GGAATCTTCGGCAATGGGGGGAACCCTGACCGAGCAACGCCGCGTGAGTGAAGAAGGTTTTC
>sample7_2 GZIPSVE02GWB39 341F
GGAGGAATATTGCCCAATGGACGAAAGTCTGAGGCAGCAACACCGCGTGGAGGACGAAGGCC
>sample8_3 GZIPSVE02HO46N 341F
GGAGGAATATTGGTCAATGGGTGCAAGCCTGAACCAGCCATCCCGCGTGAAGGACGACTGCC
>sample6_4 GZIPSVE01BE9X8 341F
GGAGTAATCTTCCACAATGGGCGCAAGCCCGATGGAGCAACGCCGCGTGGAGGATGACGGCC
```

Since we specified -x, the mid tags and primers have been removed from the reads, just like with clustering.

The denoising misses can be visualized the same way as with clustering. However, the process of comparing flow values to those of a series of nodes means that multiple nodes may be within the threshhold for a given flow value. When this is the case, FlowClus places the flowgram with the node to which its value is closest, and all the other nodes are recorded as misses. Therefore, the misses after denoising by trie look considerably different than after clustering.

With a constant denoising distance of 0.50, we do not see the same clean white stripe that we saw previously:



It may be more difficult to tweak the denoising distance on the basis of this visualization. However, one would be wise not to forgo the quantification of read

changes by AlignClusMus.pl, to ensure that the chosen denoising distance does not result in the reads' being altered in a manner inconsistent with noise removal.

The files produced for chimera checking follow the same format as those after clustering. The important difference is that, since clusters are not formed, FlowClus prints one read from each leaf node (a leaf node is one without any children nodes), after denoising, and gives the abundance as the number of reads that map to that node or any of its ancestor nodes. In general, this makes for a longer <primer>.chfasta file than with clustering:

```
>GZIPSVE02I7K6G /ab=1/
GATTCACCGCCGTATTGCTGACCGGCGATTACTAGCGATTCCGACTTCATGCAGGCGAGTTGCAGCCTGCAATC
>GZIPSVE01C0WD5 /ab=1/
GATTCACCGCCGTATTGCTGACCGGCGATTACTAGCGATTCCGACTTCATGCAGGCGAGTTGCAGCCTGCAATC
>GZIPSVE02IO19G /ab=2/
GATTCACCGCGGCATTCTGATCCGCGATTACTAGCGATTCCGACTTCATGGAGTCGAGTTGCAGACTCCAATCC
>GZIPSVE02HEDND /ab=1/
GATTCACCGCTGTATAGCTGACCAGCGATTACTAGCGATTCCATCTTCATGGAGGCGAGTTGCAGCCTTCAATC
>GZIPSVE02G6M33 /ab=4/
GATTCACCGCTGTATAGCTGACCAGCGATTACTAGCGATTCCATCTTCATGAAGGCGAGTTGCAGCCTTCAATC
>GZIPSVE02I7IDQ /ab=4/
GATTCACCGCTGTATAGCTGACCAGCGATTACTAGCGATTCCATCTTCATGAAGGCGAGTTGCAGCCTTCAATC
```

If a read maps to an internal node that has multiple leaf children nodes, it will be counted multiple times in producing the abundance information. It will also show up multiple times in the mapping file, as we can see that in the corresponding section of 1401R.chmap:

```
GZIPSVE02I7K6G GZIPSVE02I7K6G
GZIPSVE01C0WD5 GZIPSVE01C0WD5
GZIPSVE02IO19G GZIPSVE02HO27D,GZIPSVE02IO19G
GZIPSVE02HEDND GZIPSVE02HEDND
GZIPSVE02G6M33 GZIPSVE02FI30X,GZIPSVE01DSA5N,GZIPSVE01C0396,GZIPSVE02G6M33
GZIPSVE02I7IDQ GZIPSVE02FI30X,GZIPSVE01DSA5N,GZIPSVE01C0396,GZIPSVE02I7IDQ
```

Three reads (GZIPSVE02FI30X, GZIPSVE01DSA5N, and GZIPSVE01C0396) mapped to ancestor nodes of two leaf nodes (where reads GZIPSVE02G6M33 and GZIPSVE02I7IDQ mapped). The leaf nodes are both given abundances of four in 1401R.chfasta (above). UCHIME does not mind this abundance accounting, since it only cares about relative abundance. But, the question arises, what should happen to the three reads if only one of the two is declared a chimera by UCHIME? We have no specific recommendations about this situation, except to say that the correct approach would likely take into account the reads' lengths and the output from the --uchimealns option of UCHIME.

IV. Conclusion

We hope you have enjoyed this journey through (some of) the features of FlowClus, and that the program helps to service your denoising needs. We would like to conclude by pointing out that the filtering and denoising commands could have been performed in a single step:

```
$ ./FlowClus -m master.csv -i SRR653182.sff.txt -ab
    -l 200 -L 600 -n 0 -r -er 1 -wl 50 -wq 25.0
    -c filterCount.txt -cv filterVerb.txt -j 0.50 -ch -x
    -d misses.csv -st
```

The main virtue of running the steps separately is that it allows one to optimize the filtering step, before moving on to the (generally longer) denoising step.

But we leave it up to you to decide how to analyze your dataset -- this has always been the guiding principle in designing this software.

-- John M. Gaspar (jsh58@wildcats.unh.edu)

V. Bonus! Denoising multiple sequencing runs

Because of its limited computational requirements in terms of time and memory, FlowClus is uniquely suited to be able to denoise very large datasets, including reads from multiple sequencing runs. In general, it is conceptually better to denoise data together as much as possible, allowing for error correction of reads derived from rare taxa.

However, as we have already seen, denoising by FlowClus can occur only for reads sequenced from the same (or nearly identical) primer. FlowClus also requires that the sequencing runs were performed using the same flow order (otherwise, the flowgrams could not be compared in any meaningful fashion). Another caveat is that FlowClus does not take multiple sff.txt files as an input. If there is no overlap of mid tags or sample names for the different sequencing runs, one can use the 454 Tools program sfffile to combine the different .sff files, before producing the sff.txt file that can be analyzed by FlowClus in a straightforward manner.

Assuming none of those issues applies, FlowClus can perform the analysis using the following workflow: 1) filter each of the sequencing runs separately; 2) combine the filtered flowgrams; 3) denoise together. We will follow this workflow to analyze the dataset from Krych *et al.* [6].

First, we retrieved the raw data and mid tag - primer sequences from the NCBI Sequence Read Archive. We combined the four .sff files within each of the three datasets (baseline, synbiotic, and placebo) using the 454 Tools program sfffile. Since some of the mid tags were reused among the three, we did not combine them further; instead, we produced three sff.txt files (using sffinfo) and corresponding master files. We then filtered the three datasets separately using FlowClus:

```
$ ./FlowClus -m baseline.master.csv -i baseline.sff.txt -a
    -em 1 -ep 2 -r -er 1 -n 0 -l 150 -L 600 -wq 25.0 -wl 50
    -f flowBase -st

$ ./FlowClus -m synbiotic.master.csv -i synbiotic.sff.txt
    -a -em 1 -ep 2 -r -er 1 -n 0 -l 150 -L 600 -wq 25.0
    -wl 50 -f flowSyn -st

$ ./FlowClus -m placebo.master.csv -i placebo.sff.txt -a
    -em 1 -ep 2 -r -er 1 -n 0 -l 150 -L 600 -wq 25.0 -wl 50
    -f flowPlac -st
```

All of the sequencing runs consisted of a single amplicon sequenced from one primer (341F), so the three output files produced by these commands were 341F.flowBase, 341F.flowSyn, and 341F.flowPlac.

Before combining cleaned flowgram files, it is important to examine their headers. As stated above, if the flow orders are different, the flowgram files cannot be denoised together. If the number of flows is different, then it is important to use the largest number as the header of the combined file.

In our case, each of the three cleaned flowgram files had the same header ("800 TACGTACG"). We used a text editor to remove this line from 341F.flowSyn and 341F.flowPlac, and then concatenated the files:

```
$ cat 341F.flowBase 341F.flowSyn 341F.flowPlac > 341F.flow
```

The resulting file, 341F.flow, contained all of the cleaned flowgrams, with a single header line.

We made a combined master file for all of the samples, and then denoised:

```
$ ./FlowClus -m master.csv -b -j 0.5 -ch -x -st
```

For large samples such as this, denoising by trie (-tr) is significantly faster than clustering and uses less memory. In this case, denoising by clustering required about 12 hours, whereas the trie took less than 100 seconds and 1.2GB of memory.

VI. References

[1] Gaspar JM, Thomas WK: FlowClus: efficiently filtering and denoising pyrosequenced amplicons. *BMC Bioinformatics*, in press, 2015.

[2] Gaspar JM, Thomas WK: Assessing the consequences of denoising marker-based metagenomic data. *PLoS One* 2013, 8: e60458.

[3] Caporaso JG, *et al.*: QIIME allows analysis of high-throughput community sequencing data. *Nat Methods* 2010, 7: 335-6.

[4] Balzer S, *et al.*: Characteristics of 454 pyrosequencing data--enabling realistic simulation with flowsim. *Bioinformatics* 2010, 26: 420-5.

[5] Edgar RC, *et al.*: UCHIME improves sensitivity and speed of chimera detection. *Bioinformatics* 2011, 27: 2194-200.

[6] Krych L, *et al.*: Quantitatively different, yet qualitatively alike: a meta-analysis of the mouse core gut microbiome with a view towards the human gut microbiome. *PLoS One* 2013, 8: e62578.

Appendix 1 -- Sample master file

```
primer,341F,CCTACGGGAGGCAGCAG
reverse,CCGTCAATTCMTTTGAGTTT
midtag,sample1,ATATCGCGAG
midtag,sample2,AGACTCGACGT
midtag,sample3,AGTACGAGAGT
midtag,sample4,AGTACTACTAT
midtag,sample5,AGTAGACGTCT
midtag,sample6,AGTCGTACACT
midtag,sample7,AGTGTAGTAGT
midtag,sample8,ATAGTATACGT
midtag,sample9,CAGTACGTACT
midtag,sample10,CGACGACGCGT
midtag,sample11,CGACGAGTACT
midtag,sample12,CGATACTACGT
midtag,sample13,CGTACGTCGAT
midtag,sample14,CTACTCGTAGT
primer,926R,CCGTCAATTCMTTTGAGTTT
reverse,CCTACGGGAGGCAGCAG
midtag,sample1,AGCACTGTAG
midtag,sample2,ACGTCGGGTCT
midtag,sample3,ACTCTCGGACT
midtag,sample4,ATAGTAGGACT
midtag,sample5,AGACGTCGACT
midtag,sample6,AGTGTAGGACT
midtag,sample7,ACTACTAGACT
midtag,sample8,ACGTATAGTAT
midtag,sample9,AGTACGTGCTG
midtag,sample10,ACGCGTGGTCG
midtag,sample11,AGTACTGGTCG
midtag,sample12,ACGTAGTGTCG
midtag,sample13,ATCGACGGACG
midtag,sample14,ACTACGGGTAG
primer,968F,AACGCGAAGAACCTTAC
reverse,CGGTGTGTACAAGGCCCGGGAACG
midtag,sample1,ACGAGTGCGT
midtag,sample2,AGACTCGACGT
midtag,sample3,AGTACGAGAGT
midtag,sample4,AGTACTACTAT
midtag,sample5,AGTAGACGTCT
midtag,sample6,AGTCGTACACT
midtag,sample7,AGTGTAGTAGT
midtag,sample8,ATAGTATACGT
midtag,sample9,CAGTACGTACT
midtag,sample10,CGACGACGCGT
midtag,sample11,CGACGAGTACT
midtag,sample12,CGATACTACGT
midtag,sample13,CGTACGTCGAT
midtag,sample14,CTACTCGTAGT
primer,1401R,CGGTGTGTACAAGGCCCGGGAACG
reverse,AACGCGAAGAACCTTAC
midtag,sample1,ACGAGTGCGT
midtag,sample2,ACGTCGGGTCT
midtag,sample3,ACTCTCGGACT
midtag,sample4,ATAGTAGGACT
midtag,sample5,AGACGTCGACT
midtag,sample6,AGTGTAGGACT
midtag,sample7,ACTACTAGACT
midtag,sample8,ACGTATAGTAT
midtag,sample9,AGTACGTGCTG
midtag,sample10,ACGCGTGGTCG
midtag,sample11,AGTACTGGTCG
midtag,sample12,ACGTAGTGTCG
midtag,sample13,ATCGACGGACG
midtag,sample14,ACTACGGGTAG
```

Appendix 2 -- Filtering order of operations

Sequence

- Minimum sequence length
- Maximum sequence length
- Maximum sequence length for truncation
- Maximum ambiguous bases
- Maximum ambiguous bases for truncation
- Maximum homopolymer length
- Maximum homopolymer length for truncation
- Remove reverse primer
- Require reverse primer (and remove)

Quality scores

- Average quality score
- Sliding window of quality scores (option to eliminate)

Flowgram

- Minimum flowgram length
- Maximum flowgram length (for truncation)
- Noisy interval of flow values
- Maximum flow value
- No signal for 4 flows

During filtering by FlowClus, a given read is either eliminated, truncated, or neither. If multiple criteria would have eliminated or truncated a read, the criterion credited is the first in the categories listed above (Sequence, Quality Scores, Flowgram). In analyzing the sequence or flowgram of a read, the read must first pass

any length restrictions. After this, the sequence or flowgram is examined 5′ to 3′, and the criterion that is violated first is credited with the elimination or truncation.

If a user selects the "Maximum ambiguous bases" criterion with a value of 2, any read that contains more than 2 ambiguous bases (Ns) will be eliminated. If the user selects the similar "Maximum ambiguous bases for truncation" criterion with a value of 2, a read will be truncated immediately prior to the third N in its sequence. This may still result in the read's elimination, if the resulting sequence falls below the minimum specified by the "Minimum sequence length" or "Minimum flowgram length" criteria.

A complete description of these criteria, and the other parameters of FlowClus, is given in the README that accompanies the program.