trim.seqs

commands

The **trim.seqs** command provides the preprocessing features needed to screen and sort pyrosequences. Similar analyses are provided by the rdp; here we give you added flexibility and speed. The command will enable you to trim off primer sequences and barcodes, use the barcode information to generate a group file and split a fasta file into sub-files, screen sequences based on the qual file that comes from 454 sequencers, cull sequences based on sequence length and the presence of ambiguous bases and get the reverse complement of your sequences. While this analysis is clearly geared towards pyrosequencing collections, it can also be used with traditional Sanger sequences. Here we use the raw data used by Sahl in his example analysis of deep anoxic cenotes.

Default settings

The **trim.seqs** command requires that the user provide a fasta formatted sequence file and at least one of the options listed below. For example:

```
mothur > trim.seqs(fasta=sahl09.fna, oligos=sahl09.oligos)
```

This command will generate three files - sahl09.trim.fasta, sahl09.scrap.fasta, and sahl09.groups. The *trim.fasta file is a fasta-formatted sequence file that contains all of the sequences that passed the user-defined quality control requirements. The outputted sequences will be trimmed to remove the user-provided primers, barcodes, and sequences that drop below a quality threshold. The groups file is a two column list with the first column indicating the sequence names of those sequences in the *trim.fasta file and the second column the group that it came from. The *scrap.fasta file is a fasta-formatted sequence file with the original untrimmed sequences. In addition, following the sequence name there is a pipe (i.e. “|”) followed by a one-letter code for why the sequence failed. For example:

```
>001030_1690_3908|bf
...
>001060_1651_0162|f
```

In the case of the first sequence it failed because it was not possible to find the barcode (b), which probably kept it from finding the forward primer (f) as well. In the second sequence, it was able to find the barcode, but not the forward primer. Other abbreviations are described below.

For purposes of comparison, using summary.seqs on the input data generates the following:

```
mothur > summary.seqs(fasta=Sahl09.fna)

       Start   End NBases  Ambigs  Polymer
Minimum:   1   35  35  0   2
2.5%-tile: 1   79  79  0   3
25%-tile:  1   254 254 0   4
Median:    1   264 264 0   5
75%-tile:  1   271 271 0   5
97.5%-tile:    1   288 288 1   6
Maximum:   1   397 397 16  31
# of Seqs: 124480
```

Options

oligos

The oligos option takes a file that can contain the sequences of the forward and reverse primers and barcodes and their sample identifier. Each line of the oligos file can start with the key words “forward”, “reverse”, and “barcode” or it can start with a “#” to tell mothur to ignore that line of the oligos file. For example, consider a trimmed version of sahl09.oligos:

```
forward    CATGCTGCCTCCCGTAGGAGT
#reverse   TCAGAGTTTGATCCTGGCTCAG
barcode    AACCAACC    ALP50M
barcode    AACCAAGG    AZAC1
barcode    AACCATCG    ALP2B
barcode    AACCATGC    ALP1B
barcode    AACCGCAT    ALP80M
barcode    AACCGCTA    ALPG2
barcode    AACCGGAA    AZ273
...
```

The forward primer is best thought of as the forward sequencing primer. So if you are using the 16S rRNA primers 27f and 338r to generate sequencing substrate, but you are sequencing off of the 338r end of the fragment, you would list 338r as the forward primer and 27f as the reverse. Here we are using a “#” for the reverse primer to indicate that we don’t want mothur to screen for the reverse primer because the GSFLX sequencing platform cannot generate sequences that are >250 bp long. If the “#” were removed, all of the sequences would wind up in the scrap file. The lines starting with barcode follow the format of “barcode” - tab - barcode sequence - tab - sample identifier - line break. There is no limit to the number of primers or barcodes that mothur can handle. The forward and reverse primers can also be degenerate using standard IUPAC nomenclature. You can enter your oligos as upper or lowercase letters. It has been shown that sequencing errors in the PCR primer region of a sequence correlate highly with poor sequence quality. Therefore, it is suggested that an exact match to the primer or barcode sequences be required.

Note: trim.seqs will not find the oligos anywhere in the sequences to be processed; the forward primer sequence must come at the start of the read, and the reverse primer sequence (if used) must come at the very end of the read.

Running **trim.seqs** with the sahl.oligos file and using summary.seqs to analyze the output files generates the following:

```
mothur > trim.seqs(fasta=sahl09.fna, oligos=sahl09.oligos)

mothur > summary.seqs(fasta=sahl09.trim.fasta)

       Start   End NBases  Ambigs  Polymer
Minimum:   1   14  14  0   1
2.5%-tile: 1   109 109 0   3
25%-tile:  1   226 226 0   4
Median:    1   235 235 0   5
75%-tile:  1   242 242 0   5
97.5%-tile:    1   259 259 1   6
Maximum:   1   368 368 12  31
# of Seqs: 117491

mothur > summary.seqs(fasta=sahl09.scrap.fasta)

       Start   End NBases  Ambigs  Polymer
Minimum:   1   35  35  0   2
2.5%-tile: 1   53  53  0   3
25%-tile:  1   247 247 0   4
Median:    1   261 261 0   4
75%-tile:  1   270 270 0   5
```

```
97.5%-tile:    1   287 287 2   6
Maximum:   1   317 317 16  8
# of Seqs: 6989
```

Notice that 94% of the sequences passed this quality check. Also, recall that the *trim.fasta file sequences have been trimmed whereas the *scrap.fasta sequences have not. Some people may be interested in have a non-exact screen to trim the reverse primer because a portion of the primer may remain. Although this is not done in trim.seqs, once your sequences are aligned, you can use the screen.seqs or filter.seqs command to remove the region that corresponds to your primer.

name

A name file contains two columns. The first column contains the name of a reference sequence that is in a fasta file and the second column contains the names of the sequences (separated by commas) that the reference sequence represents. The list of names in the second column should always contain at least the reference sequence name.

count

The count file is similar to the name file in that it is used to represent the number of duplicate sequences for a given representative sequence. If you run **trim.seqs** with an oligos file that contains group labels, **trim.seqs** will create a new *.trim.count_table with the group information included.

qfile

In addition to a fna file, the 454 platform generates a qual file which mirrors the fna file. The differences is that in place of letters, the qual file has numbers indicating the quality of each base. For example, an “N” will be represented by a “0” because it is a poor base call. If a qfile is provided, you must provide either the qaverage or qthreshold options as well.

qaverage

The qaverage option tells mothur to calculate the average quality score for each sequence and to remove those sequences that have an average below the value provided to the option. For example:

```
mothur > trim.seqs(fasta=sahl09.fna, qfile=sahl09.qual, qaverage=25)

mothur > summary.seqs(fasta=sahl09.trim.fasta)

        Start   End NBases  Ambigs  Polymer
Minimum:    1   43  43  0   2
2.5%-tile: 1   83  83  0   3
25%-tile:  1   254 254 0   4
Median:    1   264 264 0   5
75%-tile:  1   271 271 0   5
97.5%-tile:    1   288 288 1   6
Maximum:   1   397 397 11  31
# of Seqs: 124000
```

Notice that more than 99% of the sequences passed this quality check. If you look at those sequences relegated to the scrap.fasta file you will find a “q” code to indicate that they failed this test. Incidentally, Huse and colleagues found that sequences generated with a GS20 and had an average score below 25 had more errors than those with higher averages. Since it is unclear how this will translate between platforms, you are encouraged to experiment.

qwindowaverage

The qwindowaverage parameter allows you to set a minimum average quality score allowed over a window. Reads (and corresponding quality files) are truncated at the end of the last window before the average quality score falls below the threshold, even if downstream windows would again rise above the average quality score threshold.

qwindowsize

The qwindowsize parameter allows you to set a number of bases in a window. Default=50.

rollaverage

The rollaverage parameter allows you to set a minimum rolling average quality score allowed over a window.

qstepsize

The qstepsize parameter allows you to set a number of bases to move the window over. Default=1.

qthreshold

The qthreshold option is similar to the qaverage option, except that if at any point a base call in a sequence has a quality score below the value provided to the option, the sequence is terminated. If any other options are provided, the they are applied to the quality-trimmed sequences. This may be an overly conservative criterion because any region with homopolymer will tend to have a lower quality score. Feel free to experiment:

```
mothur > trim.seqs(fasta=sahl09.fna, qfile=sahl09.qual, qthreshold=25)

mothur > summary.seqs(fasta=sahl09.trim.fasta)

       Start   End NBases  Ambigs  Polymer
Minimum:   1   0   0   0   1
2.5%-tile: 1   18  18  0   2
25%-tile:  1   158 158 0   3
Median:    1   216 216 0   4
75%-tile:  1   245 245 0   5
97.5%-tile:    1   267 267 0   6
Maximum:   1   302 302 0   8
# of Seqs: 124480
```

If you counted the number of bases in sahl09.fna and sahl09.trim.fna you would see that they dropped from 32,024,203 to 23,869,236 for a 25% drop.

qtrim

By default qtrim is true, meaning if a sequence falls below the qthreshold it will be trimmed and put in the trim file. If qtrim is set to false and a sequence falls below the qthreshold, it will be put into the scrap file.

flip

One of the popular approaches to generating pyrosequences from the 16S rRNA gene is to sequence off of the end of the fragments with the reverse PCR primer. Obviously you would then want to get the reverse complement of the sequences. Because the keyword “reverse” is already taken and “reversecomplement” is too long, we use the highly scientific “flip” option to calculate the reverse complement of the sequence. Also be aware of the reverse.seqs command, which does the same thing:

```
mothur > trim.seqs(fasta=sahl09.fna, flip=T)
```

checkorient

If you are running the **trim.seqs** command with paired barcodes or primers, you can use the checkorient parameter. When checkorient=t and mothur can't find the barcodes and primers, it will search the reverse compliment.

maxambig

Huse and colleagues found that the presence of any ambiguous base calls in GS20 pyrosequences was a harbinger for overall poor sequence quality. Within the **trim.seqs** command you can cull those sequences that have ambiguous bases:

```
mothur > trim.seqs(fasta=sahl09.fna, maxambig=0)
mothur > summary.seqs(fasta=sahl09.trim.fasta)

       Start   End NBases  Ambigs  Polymer
Minimum:   1   43  43  0   2
2.5%-tile: 1   156 156 0   3
25%-tile:  1   255 255 0   4
Median:    1   264 264 0   5
75%-tile:  1   271 271 0   5
97.5%-tile:    1   288 288 0   6
Maximum:   1   397 397 0   31
# of Seqs: 119753
```

This removed about 4% of the sequences. Those sequences that wind up in scrap.fasta for failing this option will be denoted with the "n" code. If you like to play with fire (and didn't your mothur warn you about such things?) you can change the value of maxambig to any positive integer.

maxhomop

Looking at the summary.seqs output for this dataset you may have noticed that the longest homopolymer in the dataset is 31 bases long. This is highly suspect as it is well-established that 454 technology struggles with homopolymers. To cap the homopolymer length you use the maxhomop option:

```
mothur > trim.seqs(fasta=sahl09.fna, maxhomop=10)
mothur > summary.seqs(fasta=sahl09.trim.fasta)

       Start   End NBases  Ambigs  Polymer
Minimum:   1   35  35  0   2
2.5%-tile: 1   79  79  0   3
25%-tile:  1   254 254 0   4
Median:    1   264 264 0   5
75%-tile:  1   271 271 0   5
97.5%-tile:    1   288 288 1   6
Maximum:   1   397 397 16  10
# of Seqs: 124473
```

This removed 7 sequences from the original collection and they are coded with an "h" in the scrap.fasta file. You should investigate your region to determine an appropriate homopolymer length.

minlength & maxlength

Huse and colleagues also found that large variations in sequence length was an indicator of poor sequence quality. Of course, they were working with sequences that they knew the correct sequence lengths, so you should be cautious about screening based on length. Looking at these data, you might predict that sequences shorter than 200 bp or longer than 300 bp are probably bad, but it is a judgement call:

```
mothur > trim.seqs(fasta=sahl09.fna, minlength=200)
mothur > summary.seqs(fasta=sahl09.trim.fasta)

       Start   End NBases  Ambigs  Polymer
Minimum:   1   200 200 0   3
2.5%-tile: 1   242 242 0   3
25%-tile:  1   256 256 0   4
Median:    1   264 264 0   5
75%-tile:  1   271 271 0   5
97.5%-tile:    1   289 289 1   6
Maximum:   1   397 397 16  31
# of Seqs: 119573

mothur > trim.seqs(fasta=sahl09.fna, maxlength=300)
mothur > summary.seqs(fasta=sahl09.trim.fasta)

       Start   End NBases  Ambigs  Polymer
Minimum:   1   35  35  0   2
2.5%-tile: 1   79  79  0   3
25%-tile:  1   254 254 0   4
Median:    1   264 264 0   5
75%-tile:  1   271 271 0   5
97.5%-tile:    1   288 288 1   6
Maximum:   1   300 300 16  31
# of Seqs: 124078

mothur > trim.seqs(fasta=sahl09.fna, minlength=200, maxlength=300)
mothur > summary.seqs(fasta=sahl09.trim.fasta)

       Start   End NBases  Ambigs  Polymer
Minimum:   1   200 200 0   3
2.5%-tile: 1   242 242 0   3
25%-tile:  1   256 256 0   4
Median:    1   264 264 0   5
75%-tile:  1   271 271 0   5
97.5%-tile:    1   288 288 1   6
Maximum:   1   300 300 16  31
# of Seqs: 119171
```

With both the minlength and maxlength criteria set we lose about 4% of the sequences. These will be denoted the an “l” code in the scrap.fasta file.

bdiffs & pdiffs & ldiffs & sdiffs & tdiffs

These parameters are used to allow differences in the barcode, primers, linkers and spacers. pdiffs is maximum number of differences to the primer sequence, default=0. bdiffs is maximum number of differences to the barcode sequence, default=0. ldiffs is maximum number of differences to the linker sequence, default=0. sdiffs is maximum number of differences to the spacer sequence, default=0. tdiffs is maximum total number of differences to the barcode, primer, linker and spacer (default to pdiffs + bdiffs + ldiffs + sdiffs).

```
mothur > trim.seqs(fasta=sahl09.fna, oligos=sahl09.oligos, bdiffs=1, pdiffs=1, tdiffs=1)
mothur > summary.seqs(fasta=sahl09.trim.fasta)


       Start   End NBases  Ambigs  Polymer
Minimum:   1   14  14  0   1
2.5%-tile: 1   96  96  0   3
25%-tile:  1   226 226 0   4
Median:    1   235 235 0   5
75%-tile:  1   242 242 0   5
97.5%-tile:    1   260 260 1   6
Maximum:   1   368 368 12  31
# of Seqs: 121622
```

keepfirst & removelast

The keepfirst parameter trims the sequence to the first keepfirst number of bases after the barcode or primers are removed, before the sequence is checked to see if it meets the other requirements. The removelast removes the last removelast number of bases after the barcode or primers are removed, before the sequence is checked to see if it meets the other requirements.

allfiles

With the barcodes it is possible to sequence many different samples in a single 454 run. It may happen that some of these samples should not be analyzed together for your analysis (e.g. comparing fecal to rhizosphere microbial communities). To parse apart the sequences that belong to each sample, use the indivfiles option. This will generate a fasta and groups file for each barcode defined in your oligos file:

```
mothur > trim.seqs(fasta=sahl09.fna, oligos=sahl09.fna, allfiles=T)
```

For this oligos file, mothur will generate 144 fasta and 144 groups files. One set of files corresponding to group ABORUPB has no sequences in the original fna file and so the trimmed fasta and qual files are empty. You can use mothur to merge fasta and groups files using the merge.files command.

keepforward

The keepforward parameter allows you indicate you want to keep the primer. Default=F.

processors

The processors parameter allows you to run the command with multiple processors. Default processors=Autodetect number of available processors and use all available.

```
mothur > trim.seqs(fasta=sahl09.fna, oligos=sahl09.fna, processors=2)
```

logtransform

The logtransform parameter allows you to indicate you want the averages for the qwindowaverage, rollaverage and qaverage to be calculated using a logtransform. Default=F.

Putting it together

Any of the above options can be combined to suit your taste. If I were doing the analysis, I might do something that looks like this:

```
mothur > trim.seqs(fasta=sahl09.fna, oligos=sahl09.oligos, minlength=200, maxlength=300, m
mothur > summary.seqs(fasta=sahl09.trim.fasta)

       Start   End NBases  Ambigs  Polymer
Minimum:   1   200 200 0   3
2.5%-tile: 1   215 215 0   3
25%-tile:  1   227 227 0   4
Median:    1   235 235 0   5
75%-tile:  1   242 242 0   5
97.5%-tile:    1   260 260 0   6
Maximum:   1   300 300 0   10
# of Seqs: 110758
```

This represents approximately 89% of the original sequences.

Revisions

- 1.24.0 - added linker and spacer options to the oligos file, as well as ldiffs and sdiffs parameters.
- 1.24.0 - added keepforward parameter.
- 1.24.0 - paralellized for Windows.
- 1.25.0 - allow for characters other than ATGC in reverse primers.
- 1.25.0 - fixed bug with Windows processors=2
- 1.25.0 - fixed bug that caused seqs to go to scrap if no oligos file was given, introduced in 1.24 when we added linkers and spacers.
- 1.28.0 - added count parameter
- 1.30.0 - added checkorient parameter.
- 1.30.0 - **trim.seqs** can now use paired barcodes and primers in the oligos file.
- 1.32.0 - Bug Fix: if primer length + barcode length + pdiffs + bdiffs > sequence length, mothur crashed. Sequence should be scrapped.
- 1.33.0 - added log transform parameter.
- 1.33.0 - reorient parameter no longer requires paired barcodes and primers.
- 1.40.0 - Rewrite of threaded code. Default processors=Autodetect number of available processors and use all available.
- 1.40.4 - Bug Fix: Trim.seqs multiple matches error. #427
- 1.40.5 - Bug Fix: Trim.seqs “could not open” error. #444
- 1.43.0 - Bug Fix: Trim.seqs not removing paired primers. #667
- 1.44.0 - Bug Fix: Fixes bug with **trim.seqs** if no name file is given and allfiles=T. #696