16S rRNA Gene Sequence Curation Pipeline
Department of Microbiology & Immunology, The University of Michigan

Author: Patrick D. Schloss
Version: 1.0c
Effective Date: April 12, 2011

1 Abstract

This SOP describes the two pipelines that were utilized to process the 16S rRNA gene sequences generated during the pre-Production (PPS) and Production (PP) phases. The goals were to (i) generate tables of counts of the number of 16S rRNA genes that affiliated with different operational taxonomic units (OTUs) and (ii) phylotypes within each sample and to create a phylogenetic tree representing the relationship between all of the sequences in the dataset.

2 Introduction

In addition to creating the final deliverables, the overarching design criteria in developing this SOP was to reduce the error rates associated with PCR and sequencing. Although it was not possible to remove every errors, we wanted to have a quantitative assessment of the error rates. With this in mind, the two pipelines that we generated were vetted using the CeFOS mock community datasets in consultation with the needs of the Data Analysis Working Group, as described in the reference cited in Section 7, below. The SOP utilizes the mothur software package and the commands listed below work with mothur v.1.18.

3 Requirements

3.1 16S rRNA gene sequences

Standard flowgram files (SFF files) were obtained from the Short Read Archive (SRA) and deposited in the DACC FTP site. When these files were submitted to the SRA they were passed through a filter to remove any sequences suspected of being human DNA. The SFF files represented sequences obtained from the V13, V35, and V69 regions; the most common region sequenced was the V35 region. In addition, sequencing metadata files containing the PCR primer and barcode sequence and sample identification tag were obtained from the DACC and were originally supplied by the sequencing centers. These metadata were converted into an oligos file (http://www.mothur.org/wiki/Trim.seqs#oligos).

3.2 Reference files

3.2.1 Alignment references

A customized collection of 16S rRNA gene sequences that were compatible with the 50,000 column SILVA-based alignment was used as the basis for this analysis. To simplify the computational requirements, we generated three reference alignments corresponding to the

V13, V35, and V69 regions. These files are available from the DACC ftp server in the folder /16S/Production/Analysis/PPS-and-SRP002395-1.0/Schloss_Lab-2.0/mothur.analysis/ref:

- silva.v19.align.bz2 – The full 50,000 column reference alignment with 14,956 sequences
- silva.v13.align.bz2 – The V13 region of the alignment (positions 1044-11892)
- silva.v35.align.bz2 – The V35 region of the alignment (positions 6426-27654)
- silva.v69.align.bz2 – The V35 region of the alignment (positions 31189-43116)

3.2.2 "Gold" reference

The gold reference set created for chimera checking with the mothur version of ChimeraSlayer was realigned to the alignment references above and corresponded to the same regions. The original version of the gold reference is available at http://microbiomeutil.sourceforge.net. In addition, because the gold references did not necessarily cover the full 16S rRNA gene, we culled any sequences that partially covered the region of interest. These files are available from the DACC ftp server in the folder /16S/Production/Analysis/PPS-and-SRP002395-1.0/Schloss_Lab-2.0/mothur.analysis/ref:

- rRNA16S.gold.v19.align.bz2
- rRNA16S.gold.v13.align.bz2
- rRNA16S.gold.v35.align.bz2
- rRNA16S.gold.v69.align.bz2

3.2.3 Classification training set

A re-formatted version of Training Set 6 was created to be compatible with the mothur implementation of the Bayesian classifier. The original training set can be found at http://sourceforge.net/projects/rdp-classifier. The converted files are available from the DACC ftp server in the folder /16S/Production/Analysis/PPS-and-SRP002395-1.0/Schloss_Lab-2.0/mothur.analysis/ref:

- trainset6_032010.fa.bz2
- trainset6_032010.tax.bz2

3.3 Software requirements

mothur is a stand-alone software package with no external dependencies. The SOP description that follows is compatible with version 1.18 of the software available at http://www.mothur.org.

3.4 Hardware requirements

All steps in the SOP were implemented on a collection of nodes each containing two quad cores and access to 48 GB of RAM. This amount of RAM was excessive and was probably only needed for the cluster command. Although there is an MPI version of mothur, it was not used in this version because the hardware did not have infiniband connections between nodes and the parallelization available within mothur was sufficient without having to use MPI.

4 Procedure

4.1 High quality trimming pipeline

The high quality trimming pipeline results in a neighbor joining tree and tables describing the number of sequences that affiliated with each OTU or phylotype for each sample. Based on analysis of mock community data, the expected error rate is approximately 0.02%.

Command	Comment
`sffinfo(sff=<sff file tag>.sff)`	Generate the FASTA and quality score file for each sff file
`trim.seqs(fasta=<sff file tag>.fasta, oligos=<sff file tag>.oligos, qfile=<sff file tag>.qual, maxambig=0, maxhomop=8, flip=T, bdiffs=1, pdiffs=2, qwindowaverage=35, qwindowsize=50, processors=8)`	Use a 50-bp sliding window and when the average quality score drops below 35, the sequence is trimmed. If any sequence has an ambiguous base call, a homopolymer longer than 8 bp, more than 1 mismatch to the barcode, or more than 2 mismatches to the primer, the sequence is trimmed. This step creates a trimmed fasta file and a group file that indicates which sample each sequence came from. Because the fragments were sequenced from the 3' to 5' end of the 16S rRNA gene, we then obtain the reverse complement for each sequence.
`system(cat *.<region>.*fasta > may1.<region>.fasta);`	Split the V13, V35, and V69 sequence files are split into separate folders and processed in parallel. All of the trimmed fasta and group files are concatenated to create a single fasta and group file for each region.
`system(cat *.<region>.*groups > may1.<region>.groups);`	
`unique.seqs(fasta=current)`	To reduce computational complexity we identify the unique sequences, but keep track of the redundant sequences using a names file.
`align.seqs(fasta=current, reference=silva.<region>.align, processors=8)`	The sequences are then aligned to the appropriate SILVA-based reference alignment.
`screen.seqs(fasta=current, name=current,`	We remove any sequence that does not end at or

Author: Patrick D. Schloss

Version: 1.0c
Effective Date: April 12, 2011

`group=current, minlength=200, end=<region position>, processors=8)`	after a specified position in the alignment that varies by region of the 16S rRNA gene.
`chimera.slayer(fasta=current, reference=rRNA16S.gold.<region>.align, processors=8)`	Chimeras are identified using a mothur-based implementation of the chimera.slayer using a region-specific version of the gold database. This turns out to be the slowest step in the entire pipeline.
`remove.seqs(fasta=current, name=current, group=current, accnos=current, dups=T)`	Chimeric sequences are removed along with their records in the name and group file.
`filter.seqs(fasta=current, vertical=T, trump=., processors=8)`	Aligned sequences are filtered to remove columns that contain only gap positions and any columns where there is missing data. This has the effect of trimming the sequences to only overlap over the same region and producing sequences that are all approximately 200 bp long.
`unique.seqs(fasta=current, name=current)`	Again, unique sequences are identified following trimming to simplify downstream processing effort.
`pre.cluster(fasta=current, name=current, diffs=2)`	An alignment-based pre-clustering step is used where rarer sequences are clustered with more abundant sequences if they differ by 2 or fewer nt.
`classify.seqs(cutoff=80, fasta=current, template=trainset6_032010.fa, taxonomy=trainset6_032010.tax, name=current, processors=8)`	All unique sequences are classified using the mothur-based implementation of the Bayesian classifier and trained on the RDP's Training Set v.6. An 80% pseudobootstrap cutoff is used for assigning a taxonomic classification.
`dist.seqs(fasta=current, processors=8, output=lt)`	Pairwise distances were calculated between all unique sequences and stored as a PHYLIP-formatted distance matrix.
`cluster(phylip=current, name=current)`	Average neighbor clustering is performed and the redundant names are brought back in so that all high quality sequences are incorporated in the analysis.
`classify.otu(taxonomy=current, list=current, name=current, cutoff=51, label=0.03)`	Using a 0.03 OTU definition, a majority consensus taxonomy is obtained for each OTU using the previously identified taxonomies based on a 80%

`make.shared(list=current, group=current, label=0.03)`	The OTU assignment data and sequence to sample mapping data are used to generate the OTU-based table to count the number of sequences per OTU per sample.
`phylotype(taxonomy=current, name=current)`	Based on the classification data from above, sequences are binned together into phylotypes that have the same taxonomy (including the redundant sequence names).
`classify.otu(taxonomy=current, list=current, name=current, cutoff=51, label=1)`	Using phylotypes at a bin of 1 (i.e. genus), we get the taxonomic name for each phylotype.
`make.shared(list=current, group=current, label=1)`	The phylotype assignment data and sequence to sample mapping data are used to generate the phylotype-based table to count the number of sequences per phylotype per sample.
`clearcut(phylip=current, neighbor=T)`	We use the deterministic neighbor-joining clustering algorithm implemented in clearcut (but run through mothur) to generate a phylogenetic tree.

4.2 Low quality trimming pipeline

The low quality trimming pipeline results in a table describing the number of sequences that affiliated with each phylotype for each sample. Based on analysis of mock community data, the expected error rate is approximately 0.40%.

`sffinfo(sff=<sff file tag>.sff)`	Generate the FASTA and quality score file for each sff file
`trim.seqs(fasta=<sff file tag>.fasta, oligos=<sff file tag>.oligos, qfile=<sff file tag>.qual, maxambig=0, maxhomop=8, flip=T, bdiffs=1, pdiffs=2, rollaverage=35, allfiles=T, processors=8)`	Calculate the average quality score for the sequence starting at the frist base. When that average drops below 35, the sequence is trimmed. If any sequence has an ambiguous base call, a homopolymer longer than 8 bp, more than 1 mismatch to the barcode, or more than 2 mismatches to the primer, the sequence is trimmed. This step creates a trimmed fasta file and a group file that indicates which sample each sequence came from. Because the fragments were sequenced from the 3' to 5' end of the 16S rRNA gene, we then obtain the reverse complement for each sequence.

16S rRNA Gene Sequence Curation Pipeline
Department of Microbiology & Immunology, The University of Michigan

Author: Patrick D. Schloss
Version: 1.0c
Effective Date: April 12, 2011

`system(cat *.<region>.*fasta > may1.<region>.fasta);`	Split the V13, V35, and V69 sequence files are split into separate folders and processed in parallel. All of the trimmed fasta and group files are concatenated to create a single fasta and group file for each region.
`system(cat *.<region>.*groups > may1.<region>.groups);`	
`unique.seqs(fasta=may1.<region>.fasta)`	To reduce computational complexity we identify the unique sequences, but keep track of the redundant sequences using a names file.
`align.seqs(candidate=current, template=silva.<region>.align, processors=8)`	The sequences are then aligned to the appropriate SILVA-based reference alignment.
`screen.seqs(fasta=current, name=current, group=current, end=<12082>, minlength=200, processors=8)`	We remove any sequence that does not end at or after a specified position in the alignment that varies by region of the 16S rRNA gene.
`chimera.slayer(fasta=current, template=rRNA16S.gold.<region>.align, processors=8)`	Chimeras are identified using a mothur-based implementation of the chimera.slayer using a region-specific version of the gold database. This turns out to be the slowest step in the entire pipeline.
`remove.seqs(fasta=current, name=current, group=current, accnos=current, dups=T)`	Chimeric sequences are removed along with their records in the name and group file.
`classify.seqs(cutoff=80, fasta=current, template=trainset6_032010.fa, taxonomy=trainset6_032010.tax, processors=8)`	All unique sequences are classified using the mothur-based implementation of the Bayesian classifier and trained on the RDP's Training Set v.6. An 80% pseudobootstrap cutoff is used for assigning a taxonomic classification.
`phylotype(taxonomy=current, name=current)`	Based on the classification data from above, sequences are binned together into phylotypes that have the same taxonomy (including the redundant sequence names).
`classify.otu(taxonomy=current, list=current, name=current, cutoff=51, label=1)`	Using phylotypes at a bin of 1 (i.e. genus), we get the taxonomic name for each phylotype.
`make.shared(list=current, group=current, label=1)`	The phylotype assignment data and sequence to sample mapping data are used to generate the phylotype-based table to count the number of sequences per phylotype per sample.

Author: Patrick D. Schloss
Version: 1.0c
Effective Date: April 12, 2011

5 Implementation

6 Discussion

7 Related Documents & References

Schloss, PD, Gevers, D and Westcott, SL. 2011. Reducing the Effects of PCR Amplification and Sequencing Artifacts on 16S rRNA-Based Studies. PLoS ONE 6(12): e27310.

8 Revision History

Version	Author/Reviewer	Date	Change Made
1.00	Patrick D. Schloss	04/12/2011	Establish SOP
1.0c		09/20/2011	Converted to standard template