**Bacterial 16S rRNA gene Amplicon Analysis Workflow**

This document outlines the workflow required to analyse 16S rRNA amplicon sequences for Bacteria (27f – 519r) to produce Amplicon Sequence Variant (ZOTU) information for the Australian Microbiome database

ZOTU analysis is performed using a combination of per sample and per sequencing run (sequencing plate) basis. The workflow allows for the inclusion of known “prior” sequences to be added following denoising steps. The workflow consists of the following stages:

**A] Sequence preparation and merging**  
 1. Merge paired end reads (non-merged reads are discarded)  
 2. Convert fastq file format to fasta file format  
 3. Add sampleID, runID and "sample=" information to the sequence headers

4. Generate abundance table of unique sequences per sample  
**B] Sample-wise ZOTU analysis**  
 1. Quality screening, ZOTU calling on individual samples on a plate

2. Concatenate all sample-wise ZOTU in the sequencing run into a single file

**C] Plate-wise ZOTU analysis**  
 1. Concatenate all sequences per sequencing run into a single file

2. Quality screening, ZOTU calling of concatenated samples on the plate

3. Concatenate plate-wise, sample-wise and prior sequences into a single file  
 4. Remove duplicated ZOTU sequences and sequence mapping  
 5. Classify and remove sequences in the wrong orientation  
 6. Replace arbitrary ZOTU ID's with the sequence itself in the table index  
**D] Following analysis, all data is combined to give a single dataset, by the following steps:**  
 1. Merge tables into a single table  
 2. Remove controls from the abundance tables, to create separate sample and control datasets  
 3. Make a fasta from the abundance table  
 4. Classify Sequences

**Software used**

The following software is used in the steps below:

1. FLASH2 (Magoc and Salzberg, 2011)  
 2. Mothur (Schloss, et al., 2009)  
 3. USEARCH (Edgar 2010)  
 4. SeqTk (<https://github.com/lh3/seqtk>)  
 5. FASTX (<http://hannonlab.cshl.edu/fastx_toolkit/>)  
 6. QIIME2 (Bolyen et al., 2019)  
 6. Python3

**A] Sequence preparation and merging**

**Merge the paired end reads**

Paired end reads are merged using **FLASH2** (Magoc and Salzberg, 2011). **FLASH2** is run with the following arguments:

--min-overlap=30 --max-overlap=250

Following merging, the merge quality is manually checked by examining the **FLASH2** log file for the percentage of reads that were merged. Plates with low merge rates (< 70%) are manually checked to see if the alignments can be improved.  
Unmerged reads are discarded.

**File naming**

For sequence files with AM sampleIDs, file names are changed to the following format:

sampleID\_plateID.fasta

For sequence files downloaded from NCBI, file names are changed to a similar format using the NCBI biosampleID and BioProjectID, as below.

biosampleID\_BioProjectID.fasta

The sampleID of the mock communities and negative controls on each plate are standardised to the following naming convention (all listed for completeness):

- 16S Bacteria mock communities: **BACMOCK**  
- A16 Archaeal mock communities: **ARCMOCK**  
- 18Sv4 Eukaryote mock communities: **EUKMOCKV4**  
- 18Sv9 Eukaryote mock communities: **EUKMOCKV9**  
- ITS Fungal mock communities: **FUNMOCK**  
- Negative control: **NEG**  
- STAN: **STAN**  
- MSA-1002 (ATCC) 20 Strain even mix genomic material: **ATCC1002MOCK**

**Fastq** files are then converted to **fasta format** using **seqTk**

**Add sample and file name to the sequence headers**

Sample identifiers are added to the header of each sequence for downstream processing. As each fasta is now named **sampleID\_plateID.fasta** we simply add the file name with the extension to the sequence header. At this stage we also add any other information and delimiters that downstream programs will likely require. For **USEARCH** we add “sample=” and “;”, for **QIIME** we add “\_”. A bash script to perform this can be found at <https://raw.githubusercontent.com/AusMicrobiome/misc_tools/master/add_sample_name.sh>

**Identify unique sequences per sample**

An abundance table of all unique sequences in each sample on the plate is prepared. Unique sequences are identified using **FASTX** with the following **USEARCH** command:

usearch -fastx\_uniques SampleID\_plateID.fasta -fastaout SampleID\_plateID\_uniques.fasta -sizeout

**Unique sequences are then converted into a 3 column tab separated abundance table containing the columns: seq\tSample\tAbundance. Unique sequences per sample (non-Denoised or quality filtered) are provided as a data output and are available by request through the Australian Microbiome Website.**

**B] Sample-wise ZOTU analysis**

**Quality screening, ZOTU calling and sequence mapping**

The first step removes sequences that have ambiguous bases or have more than 12 homopolymers.

mothur "#set.dir(modifynames=F); screen.seqs(fasta= sampleID\_plateID.fasta, maxambig=0, maxhomop=12, processors=10);"

Reads are dereplicated.

usearch -fastx\_uniques sampleID\_plateID.good.fasta -fastaout sampleID\_plateID.good\_uniques.fasta -sizeout

Sort unique reads by abundance.

usearch -sortbysize sampleID\_plateID.good\_uniques.fasta -fastaout sampleID\_plateID.good\_sorted\_uniques.fasta -sizeout

ZOTUs are called by **UNOISE3**, from sequences that have => 8 representatives.

usearch -unoise3 sampleID\_plateID.good\_sorted\_uniques.fasta -ZOTUs sampleID\_plateID.good\_sorted\_uniques\_ZOTUs.fasta -minsize 8

**Concatenate all sample-wise ZOTUs into a single file**

All ZOTU files for each sample are concatenated into a single file, this file is later combined with ZOTUs obtained from the per-plate analysis. The resulting file name is standardised to the format: **plateID\_all\_SW\_ZOTUs.fasta**

**C] Plate-wise ZOTU analysis**

**Concatenate all sequences per sequencing run into a single file**

All sample fasta files for each plate are concatenated into a single file for plate-wise ZOTU calling. The resulting file name is standardised to the format: **plateID\_all\_16S.fasta**

**Quality screening, ZOTU calling and sequence mapping**

The first step removes sequences that have ambiguous bases or have more than 12 homopolymers.

mothur "#set.dir(modifynames=F);summary.seqs(fasta=plateID\_all\_16S.fasta, processors=10); screen.seqs(fasta=current, maxambig=0, maxhomop=12, processors=10); summary.seqs()"

Reads are dereplicated.

usearch -fastx\_uniques plateID\_all\_16S.good.fasta -fastaout plateID\_all\_16S.good\_uniques.fasta -sizeout

Sort unique reads by abundance.

usearch -sortbysize plateID\_all\_16S.good\_uniques.fasta -fastaout plateID\_all\_16S.good\_sorted\_uniques.fasta -sizeout

ZOTUs are called by **UNOISE3**, from sequences that have => 8 representatives.

usearch -unoise3 plateID\_all\_16S.good\_sorted\_uniques.fasta -ZOTUs plateID\_all\_16S.good\_sorted\_uniques\_ZOTUs.fasta -ampout plateID\_all\_16S.good\_sorted\_uniques\_ampout.fasta -tabbedout plateID\_all\_16S.good\_sorted\_uniques\_unoise3.txt -minsize 8

Combine the ZOTUs obtained from sample-wise and plate-wise analysis with user submitted prior sequences into a single fasta file. The resulting file name is standardised to the format: **plateID\_all\_16S.good\_sorted\_uniques\_ZOTUs\_renamed\_SWpriors.fasta**

Dereplicate the combined ZOTU file.

usearch -fastx\_uniques **plateID\_all\_16S.good\_sorted\_uniques\_ZOTUs\_renamed\_SWpriors.fasta** -fastaout **plateID\_all\_16S.good\_sorted\_uniques\_ZOTUs\_renamed\_SWpriors**\_uniq.fasta

Map reads to ZOTUs to generate abundances.

Reads are mapped against the ZOTUs using **USEARCH**. Note the termination conditions on the mapping run (**-maxaccepts 0**), this seems to be required to ensure the best match is found and to produce consistent results when adding multiple plates together, as we do later.

usearch -otutab plateID\_all\_16S.fasta -ZOTUs plateID\_all\_16S.good\_sorted\_uniques\_ZOTUs\_SWpriors\_uniq.fasta -otutabout plateID\_all\_16S.good\_sorted\_uniques\_PWSW\_ZOTUtab\_MA0.txt -mapout plateID\_all\_16S.good\_sorted\_uniques\_PWSW\_zmap\_MA0.txt -maxaccepts 0 -threads X

**Replace ZOTU table indexes with ZOTU sequence**

After denoising and mapping the ZOTU tables have an arbitrary ZOTU number as the index, we replace this with the sequence that the arbitrary number represents. These sequences are unique strings and allow tables to be merged etc. downstream easily. They also negate the need to maintain a dictionary of ZOTUs and the sequences they represent.

**Classify and remove flipped sequences**

A final QC step is performed to remove likely erroneous sequences. The ZOTUs are classified, with those that do not align to the 16S database in the correct orientation being removed. Those that need to be “flipped” to a new orientation are likely errors, since we know the reads should be in 27f – 519r orientation. This step typically removes < 10 ZOTUs from the database.

**Classify the seqs against 16S database**

"#set.dir(modifynames=F); classify.seqs(fasta=plateID\_all\_16S.good\_sorted\_uniques\_ZOTUtab\_relabelled\_MA0.fasta, reference=gg\_13\_8\_99.fasta, taxonomy=gg\_13\_8\_99.gg.tax, cutoff=60, probs=FALSE)"

**Use the \*acnos.flip list to remove flipped sequences from the table**

The above produces the final abundance table, with ZOTU sequences as index for each sequencing run. These tables are then combined as below to produce a single dataset for the Australian Microbiome.

**D] Prepare the single dataset**

Now we have a ZOTU abundance table for each plate, with ZOTU’s as row and sampleID\_plateID as column headers. To prepare this data for ingest into the AM database the following steps are carried out:

1. Each table is converted from short to long format (from rectangular to 3 column, with the following columns: ZOTU, sampleID, Abundance)  
2. All of these 3 column tables are concatenated into a single table  
3. Controls and samples are split into separate tables  
4. Sequencing run ID's are removed from the column headers and any sample sequenced more than once has the OTUs grouped and abundances summed to give a single abundance per sample  
5. A fasta file of unique ZOTU's is created from all ZOTU's in this final table  
6. Sequences are classified.

**Classify the sequences**

Sequences are classified to provide taxonomies relative to the Silva 138 database (Quast et al., 2013; Yilmaz et al 2014; Glöckner 2017). Sequence classification is performed using **QIIME2** sklearn Bayesian classifier with the following arguments:

--p-confidence 0.6 --p-read-orientation same

**References**

Magoc, T. and Salzberg, S. (2011). FLASH: Fast length adjustment of short reads to improve genome assemblies. Bioinformatics 27(21): 2957-2963.  
Schloss, P.D., et al., (2009). Introducing mothur: Open-source, platform-independent, community-supported software for describing and comparing microbial communities. Appl Environ Microbiol 75(23):7537-7541.  
Edgar, R.C., (2010), Search and clustering orders of magnitude faster than BLAST, Bioinformatics 26(19): 2460-2461.  
Seqtk available at: <https://github.com/lh3/seqtk> (last accessed 23 Jan 2019).  
FASTX available at: (<http://hannonlab.cshl.edu/fastx_toolkit/>) (last accessed 23 Jan 2019).  
DeSantis T.Z., Hugenholtz P., Larsen N., Rojas M., Brodie E.L., Keller K., Huber T., Dalevi D., Hu P., Andersen G.L.(2006) Greengenes, a Chimera-Checked 16S rRNA Gene Database and Workbench Compatible with ARB. Appl. Environ. Microbiol. 72(7): 5069-5072; DOI: 10.1128/AEM.03006-05  
Quast C, Pruesse E, Yilmaz P, Gerken J, Schweer T, Yarza P, Peplies J, Glöckner FO (2013) The SILVA ribosomal RNA gene database project: improved data processing and web-based tools. Nucl. Acids Res. 41 (D1): D590-D596.  
Yilmaz P, Parfrey LW, Yarza P, Gerken J, Pruesse E, Quast C, Schweer T, Peplies J, Ludwig W, Glöckner FO (2014) The SILVA and “All-species Living Tree Project (LTP)” taxonomic frameworks. Nucl. Acids Res. 42:D643-D648  
Glöckner FO, Yilmaz P, Quast C, Gerken J, Beccati A, Ciuprina A, Bruns G, Yarza P, Peplies J, Westram R, Ludwig W (2017) 25 years of serving the community with ribosomal RNA gene reference databases and tools. J. Biotechnol.

Bolyen E, Rideout JR, Dillon MR, Bokulich NA, Abnet CC, Al-Ghalith GA, Alexander H, Alm EJ, Arumugam M, Asnicar F, Bai Y, Bisanz JE, Bittinger K, Brejnrod A, Brislawn CJ, Brown CT, Callahan BJ, Caraballo-Rodríguez AM, Chase J, Cope EK, Da Silva R, Diener C, Dorrestein PC, Douglas GM, Durall DM, Duvallet C, Edwardson CF, Ernst M, Estaki M, Fouquier J, Gauglitz JM, Gibbons SM, Gibson DL, Gonzalez A, Gorlick K, Guo J, Hillmann B, Holmes S, Holste H, Huttenhower C, Huttley GA, Janssen S, Jarmusch AK, Jiang L, Kaehler BD, Kang KB, Keefe CR, Keim P, Kelley ST, Knights D, Koester I, Kosciolek T, Kreps J, Langille MGI, Lee J, Ley R, Liu YX, Loftfield E, Lozupone C, Maher M, Marotz C, Martin BD, McDonald D, McIver LJ, Melnik AV, Metcalf JL, Morgan SC, Morton JT, Naimey AT, Navas-Molina JA, Nothias LF, Orchanian SB, Pearson T, Peoples SL, Petras D, Preuss ML, Pruesse E, Rasmussen LB, Rivers A, Robeson MS, Rosenthal P, Segata N, Shaffer M, Shiffer A, Sinha R, Song SJ, Spear JR, Swafford AD, Thompson LR, Torres PJ, Trinh P, Tripathi A, Turnbaugh PJ, Ul-Hasan S, van der Hooft JJJ, Vargas F, Vázquez-Baeza Y, Vogtmann E, von Hippel M, Walters W, Wan Y, Wang M, Warren J, Weber KC, Williamson CHD, Willis AD, Xu ZZ, Zaneveld JR, Zhang Y, Zhu Q, Knight R, and Caporaso JG. (2019) Reproducible, interactive, scalable and extensible microbiome data science using QIIME 2. Nature Biotechnology 37: 852–857.<https://doi.org/10.1038/s41587-019-0209-9>**Archaeal 16S rRNA gene Amplicon Analysis Workflow**

This document outlines the workflow required to analyse 16S rRNA amplicon sequences for Archaea (A2f-519r) to produce Amplicon Sequence Variant (ZOTU) information for the Australian Microbiome database

The ZOTU analysis is performed using a combination of per sample and per sequencing run (sequencing plate) basis. The workflow allows for the inclusion of known “prior” sequences to be added following denoising steps. The workflow consists of the following stages:

**A] Sequence preparation and merging**  
 1. Merge paired end reads (non-merged reads are discarded)  
 2. Convert fastq file format to fasta file format  
 3. Add sampleID, plateID and "sample=" information to the sequence headers

4. Trim forward and reverse primers from merged reads

5. Generate abundance table of unique sequences per sample  
**B] Sample-wise ZOTU analysis**

1. Quality screening, ZOTU calling on individual samples on a plate

2. Concatenate all sample-wise ZOTU in the sequencing run into a single file

**C] Plate-wise ZOTU analysis**  
 1. Concatenate all sequences per sequencing run into a single file

2. Quality screening, ZOTU calling of concatenated samples on the plate

3. Concatenate plate-wise, sample-wise and prior sequences into a single file  
 4. Remove duplicated ZOTU sequences and sequence mapping  
 5. Classify and remove flipped sequences  
 6. Replace arbitrary ZOTU ID's with the sequence itself in the table index  
**D] Following analysis, all data is combined to give a single dataset, by the following steps:**  
 1. Merge tables into a single table  
 2. Remove controls from the abundance tables, to create separate sample and control datasets  
 3. Make a fasta from the abundance table  
 4. Classify Sequences

**Software used**

The following software is used in the steps below:

1. FLASH2 (Magoc and Salzberg, 2011)  
 2. Mothur (Schloss, et al., 2009)  
 3. USEARCH (Edgar, 2010)  
 4. SeqTk (<https://github.com/lh3/seqtk>)  
 5. FASTX (<http://hannonlab.cshl.edu/fastx_toolkit/>)  
 6. Cutadapt (Martin, [DOI:10.14806/ej.17.1.200](http://dx.doi.org/10.14806/ej.17.1.200))  
 7. QIIME2 (Bolyen et al., 2019)  
 8. Python3

**A] Sequence preparation and merging**

**Merge the paired end reads**

Paired end reads are merged using **FLASH2** (Magoc and Salzberg, 2011). **FLASH2** is run with the following arguments:

--min-overlap=30 --max-overlap=250

Following merging, the merge quality is manually checked by examining the **FLASH2** log file for the percentage of reads that were merged. Plates with low merge rates (< 70%) are manually checked to see if the alignments can be improved.  
Unmerged reads are discarded.

**File naming**

For sequence files with AM sampleIDs, file names are changed to the following format:

sampleID\_plateID.fasta

For sequence files downloaded from NCBI, file names are changed to a similar format using the NCBI biosampleID and BioProjectID, as below.

biosampleID\_BioProjectID.fasta

The sampleID of the mock communities and negative controls on each plate are standardised to the following naming convention (all listed for completeness):

- 16S Bacteria mock communities: **BACMOCK**  
- A16 Archaeal mock communities: **ARCMOCK**  
- 18Sv4 Eukaryote mock communities: **EUKMOCKV4**  
- 18Sv9 Eukaryote mock communities: **EUKMOCKV9**  
- ITS Fungal mock communities: **FUNMOCK**  
- Negative control: **NEG**  
- STAN: **STAN**- MSA-1002 (ATCC) 20 Strain even mix genomic material: **ATCC1002MOCK**

**Fastq** files are then converted to **fasta format** using **seqTk**

**Add sample and file name to the sequence headers**

Sample identifiers are added to the header of each sequence for downstream processing. As each fasta is now named **sampleID\_plateID.fasta** we simply add the file name with the extension to the sequence header. At this stage we also add any other information and delimiters that downstream programs will likely require. For **USEARCH** we add “sample=” and “;”, for **QIIME** we add “\_”. A bash script to perform this can be found at [https://raw.githubusercontent.com/AusMicrobiome/misc\_tools/master/add\_sample\_name.sh](https://github.com/AusMicrobiome/misc_tools)

**Remove forward and reverse primers from the merged reads**

Primers are removed from the merged reads using **Cutadapt** v2.8 with the following arguments:

**Reverse primer removal:**

**--cores=0 -a AGCMGCCGCGGTAATWCX -O 17 -e 0.01**

**Forward primer removal:**

**--minimum-length 1 --cores=0 -g XTTCCGGTTGATCCYGCCGGA -O 20 -e 0.01**

**Identify unique sequences per sample**

An abundance table of all unique sequences in each sample on the plate is prepared. Unique sequences are identified using **FASTX** with the following **USEARCH** command:

usearch -fastx\_uniques SampleID\_plateID.trimmed.fasta -fastaout SampleID\_plateID\_uniques.fasta -sizeout

**Unique sequences are then converted into a 3 column tab separated abundance table containing the columns: seq\tSample\tAbundance. Unique sequences per sample (non-Denoised or quality filtered) are provided as a data output and are available by request through the Australian Microbiome Website.**

**B] Sample-wise ZOTU analysis**

**Quality screening, ZOTU calling and sequence mapping**

The first step removes sequences that have ambiguous bases or have more than 12 homopolymers.

mothur "#set.dir(modifynames=F); screen.seqs(fasta= sampleID\_plateID.trimmed.fasta, minlength=380, maxlength=520, maxambig=0, maxhomop=12, processors=10);"

Reads are dereplicated

usearch -fastx\_uniques sampleID\_plateID.good.fasta -fastaout sampleID\_plateID.trimmed.good\_uniques.fasta -sizeout

Sort unique reads by abundance

usearch -sortbysize sampleID\_plateID.trimmed.good\_uniques.fasta -fastaout sampleID\_plateID.trimmed.good\_sorted\_uniques.fasta

ZOTUs are called by **UNOISE3**, from sequences that have => 8 representatives

usearch -unoise3 sampleID\_plateID.trimmed.good\_sorted\_uniques.fasta -ZOTUs sampleID\_plateID.trimmed.good\_sorted\_uniques\_ZOTUs.fasta -minsize 8

**Concatenate all sample-wise ZOTUs into a single file**

All ZOTU files for each sample are concatenated into a single file, this file is later combined with ZOTUs obtained from the per-plate analysis. The resulting file name is standardised to the format: **plateID\_all\_SW\_ZOTUs.fasta**

**C] Plate-wise ZOTU analysis**

**Concatenate all sequences per sequencing run into a single file**

All sample fasta files for each plate are concatenated into a single file for ZOTU calling. The resulting file name is standardised to the format: **plateID\_all\_A16S.fasta**

**Quality screening, ZOTU calling and sequence mapping**

The first step removes sequences that have ambiguous bases or have more than 12 homopolymers.

mothur "#set.dir(modifynames=F);summary.seqs(fasta=plateID\_all\_A16S.trimmed.fasta, processors=10); screen.seqs(fasta=current, minlength=380, maxlength=520, maxambig=0, maxhomop=12, processors=10); summary.seqs()"

Reads are dereplicated

usearch -fastx\_uniques plateID\_all\_A16S.trimmed.good.fasta -fastaout plateID\_all\_A16S.trimmed.good\_uniques.fasta -sizeout

Sort unique reads by abundance

usearch -sortbysize plateID\_all\_A16S.trimmed.good\_uniques.fasta -fastaout plateID\_all\_A16S.trimmed.good\_sorted\_uniques.fasta -sizeout

ZOTUs are called by **UNOISE3**, from sequences that have => 8 representatives

usearch -unoise3 plateID\_all\_A16S.trimmed.good\_sorted\_uniques.fasta -ZOTUs plateID\_all\_A16S.trimmed.good\_sorted\_uniques\_ZOTUs.fasta -ampout plateID\_all\_A16S.trimmed.good\_sorted\_uniques\_ampout.fasta -tabbedout plateID\_all\_A16S.trimmed.good\_sorted\_uniques\_unoise3.txt -minsize 8

Combine the ZOTUs obtained from sample-wise and plate-wise analysis with a defined set of “prior” sequences into a single fasta file. The resulting file name is standardised to the format: **plateID\_all\_A16S.trimmed.good\_sorted\_uniques\_ZOTUs\_renamed\_SWpriors.fasta**

Dereplicate the combined ZOTU file

usearch -fastx\_uniques plateID\_all\_A16S.trimmed.good\_sorted\_uniques\_ZOTUs\_renamed\_SWpriors.fasta -fastaout plateID\_all\_A16S.trimmed.good\_sorted\_uniques\_ZOTUs\_renamed\_SWpriors\_uniq.fasta

Map reads to ZOTUs to generate abundances

Reads are mapped against the ZOTUs using **USEARCH**. Note the termination conditions on the mapping run (**-maxaccepts 0**), this seems to be required to ensure the best match is found and to produce consistent results when adding multiple plates together, as we do later.

usearch -otutab plateID\_all\_16S.fasta -ZOTUs plateID\_all\_A16S.trimmed.good\_sorted\_uniques\_ZOTUs\_SWpriors\_uniq.fasta -otutabout plateID\_all\_A16S.trimmed.good\_sorted\_uniques\_PWSW\_ZOTUtab\_MA0.txt -mapout plateID\_all\_A16S.trimmed.good\_sorted\_uniques\_PWSW\_zmap\_MA0.txt -maxaccepts 0 -threads X

**Replace ZOTU table indexes with ZOTU sequence**

After denoising and mapping the ZOTU tables have an arbitrary ZOTU number as the index, we replace this with the sequence that the arbitrary number represents. These sequences are unique strings and allow tables to be merged etc. downstream easily. They also negate the need to maintain a dictionary of ZOTUs and the sequences they represent.

**Classify and remove flipped sequences**

A final QC step is performed to remove likely erroneous sequences. The ZOTUs are classified, with those that do not align to the 16S database in the correct orientation being removed. Those that need to be “flipped” to a new orientation are likely errors, since we know the reads should be in A2f – 519r orientation. This step typically removes < 10 ZOTUs from the database.

**Classify the seqs against 16S database**

"#set.dir(modifynames=F); classify.seqs(fasta=plateID\_all\_A16S.trimmed.good\_sorted\_uniques\_ZOTUtab\_relabelled\_MA0.fasta, reference=gg\_13\_8\_99.fasta, taxonomy=gg\_13\_8\_99.gg.tax, cutoff=60, probs=FALSE)"

**Use the \*acnos.flip list to remove flipped sequences from the table**

The above produces the final abundance table, with ZOTU sequences as index for each sequencing run. These tables are then combined as below to produce a single dataset for the Australian Microbiome.

**D] Prepare the single dataset**

Now we have a ZOTU abundance table for each plate, with ZOTU’s as row and sampleID\_plateID as column headers. To prepare this data for ingest into the AM database the following steps are carried out:

1. Each table is converted from short to long format (from rectangular to 3 column, with the following columns: ZOTU, sampleID, Abundance)  
2. All of these 3 column tables are concatenated into a single table  
3. Controls and samples are split into separate tables  
4. Sequencing run ID's are removed from the column headers and any sample sequenced more than once has the OTUs grouped and abundances summed to give a single abundance per sample  
5. A fasta file of unique ZOTU's is created from all ZOTU's in this final table  
6. Sequences are classified.

**Classify the sequences**

Sequences are classified to provide taxonomies relative to the Silva 138 database (Quast et al., 2013; Yilmaz et al 2014; Glöckner 2017). Sequence classification is performed using **QIIME2** sklearn Bayesian classifier with the following arguments:

--p-confidence 0.6 --p-read-orientation same

**References**

Magoc, T. and Salzberg, S. (2011). FLASH: Fast length adjustment of short reads to improve genome assemblies. Bioinformatics 27(21): 2957-2963.  
Schloss, P.D., et al., (2009). Introducing mothur: Open-source, platform-independent, community-supported software for describing and comparing microbial communities. Appl Environ Microbiol 75(23):7537-7541.  
Edgar, R.C., (2010), Search and clustering orders of magnitude faster than BLAST, Bioinformatics 26(19): 2460-2461.  
Seqtk available at: <https://github.com/lh3/seqtk> (last accessed 23 Jan 2019).  
FASTX available at: (<http://hannonlab.cshl.edu/fastx_toolkit/>) (last accessed 23 Jan 2019).  
DeSantis T.Z., Hugenholtz P., Larsen N., Rojas M., Brodie E.L., Keller K., Huber T., Dalevi D., Hu P., Andersen G.L.(2006) Greengenes, a Chimera-Checked 16S rRNA Gene Database and Workbench Compatible with ARB. Appl. Environ. Microbiol. 72(7): 5069-5072; DOI: 10.1128/AEM.03006-05  
Quast C, Pruesse E, Yilmaz P, Gerken J, Schweer T, Yarza P, Peplies J, Glöckner FO (2013) The SILVA ribosomal RNA gene database project: improved data processing and web-based tools. Nucl. Acids Res. 41 (D1): D590-D596.  
Yilmaz P, Parfrey LW, Yarza P, Gerken J, Pruesse E, Quast C, Schweer T, Peplies J, Ludwig W, Glöckner FO (2014) The SILVA and “All-species Living Tree Project (LTP)” taxonomic frameworks. Nucl. Acids Res. 42:D643-D648  
Glöckner FO, Yilmaz P, Quast C, Gerken J, Beccati A, Ciuprina A, Bruns G, Yarza P, Peplies J, Westram R, Ludwig W (2017) 25 years of serving the community with ribosomal RNA gene reference databases and tools. J. Biotechnol.

Martin, M. Cutadapt removes adapter sequences from high-throughput sequencing reads [DOI:10.14806/ej.17.1.200](http://dx.doi.org/10.14806/ej.17.1.200)

Bolyen E, Rideout JR, Dillon MR, Bokulich NA, Abnet CC, Al-Ghalith GA, Alexander H, Alm EJ, Arumugam M, Asnicar F, Bai Y, Bisanz JE, Bittinger K, Brejnrod A, Brislawn CJ, Brown CT, Callahan BJ, Caraballo-Rodríguez AM, Chase J, Cope EK, Da Silva R, Diener C, Dorrestein PC, Douglas GM, Durall DM, Duvallet C, Edwardson CF, Ernst M, Estaki M, Fouquier J, Gauglitz JM, Gibbons SM, Gibson DL, Gonzalez A, Gorlick K, Guo J, Hillmann B, Holmes S, Holste H, Huttenhower C, Huttley GA, Janssen S, Jarmusch AK, Jiang L, Kaehler BD, Kang KB, Keefe CR, Keim P, Kelley ST, Knights D, Koester I, Kosciolek T, Kreps J, Langille MGI, Lee J, Ley R, Liu YX, Loftfield E, Lozupone C, Maher M, Marotz C, Martin BD, McDonald D, McIver LJ, Melnik AV, Metcalf JL, Morgan SC, Morton JT, Naimey AT, Navas-Molina JA, Nothias LF, Orchanian SB, Pearson T, Peoples SL, Petras D, Preuss ML, Pruesse E, Rasmussen LB, Rivers A, Robeson MS, Rosenthal P, Segata N, Shaffer M, Shiffer A, Sinha R, Song SJ, Spear JR, Swafford AD, Thompson LR, Torres PJ, Trinh P, Tripathi A, Turnbaugh PJ, Ul-Hasan S, van der Hooft JJJ, Vargas F, Vázquez-Baeza Y, Vogtmann E, von Hippel M, Walters W, Wan Y, Wang M, Warren J, Weber KC, Williamson CHD, Willis AD, Xu ZZ, Zaneveld JR, Zhang Y, Zhu Q, Knight R, and Caporaso JG. (2019) Reproducible, interactive, scalable and extensible microbiome data science using QIIME 2. Nature Biotechnology 37: 852–857.<https://doi.org/10.1038/s41587-019-0209-9>

# Eukaryotic 18S rRNA gene Amplicon Analysis Workflow

This section outlines the workflow required to analyse 18S rRNA gene amplicon sequences to produce Amplicon Sequence Variant information for the Australian microbiome database.

This workflow covers both 18S variable region 4 (18Sv4) amplified by the 18S\_V4f/18S\_V4r primer set and 18S variable region 9 (18Sv9) amplified by the ILM\_Euk\_1391f/ILM\_EukBr primer set.

The ZOTU analysis is performed using a combination of per sample and per sequencing run (sequencing plate) basis. The workflow allows for the inclusion of known “prior” sequences to be added following denoising steps. The workflow consists of the following stages:

**A] Sequence preparation and merging**  
 1. Trim paired end reads (18Sv4 only)

2. Merge paired end reads (non-merged reads are discarded)  
 3. Convert fastq file format to fasta file format  
 4. Add sampleID, runID and "sample=" information to the sequence headers

5. Generate abundance table of unique sequences per sample  
**B] Sample-wise ZOTU analysis**

1. Quality screening, ZOTU calling on individual samples on a plate

2. Concatenate all sample-wise ZOTU in the sequencing run into a single file

**C] Plate-wise ZOTU analysis**  
 1. Concatenate all sequences per sequencing run into a single file

2. Quality screening, ZOTU calling of concatenated samples on the plate

3. Concatenate plate-wise, sample-wise and prior sequences into a single file

4. Remove duplicated ZOTU sequences and sequence mapping  
 5. Classify and remove flipped sequences  
 6. Replace arbitrary ZOTU ID's with the sequence itself in the table index  
**D] Following analysis, all data is combined to give a single dataset, by the following steps:**  
 1. Merge tables into a single table  
 2. Remove controls from the abundance tables, to create separate sample and control datasets  
 3. Make a fasta from the abundance table  
 4. Classify Sequences

**Software used**

The following software is used in the steps below:

1. FLASH2 (Magoc and Salzberg, 2011)(for 18Sv9) or FLASH2 (Magoc and Salzberg, 2011)(for 18Sv4)   
 2. Mothur (Schloss, et al., 2009)  
 3. USEARCH (Edgar 2010)  
 4. Seqtk (<https://github.com/lh3/seqtk>)  
 5. FASTX (<http://hannonlab.cshl.edu/fastx_toolkit/>)

6. QIIME2 (Bolyen et al., 2019)  
 7. Python3

**A] Sequence preparation and merging**

**Merge the paired end reads**

For 18Sv4 amplicons, primer removal is performed using **seqTk** by hard trimming 20 nucleotides from the 5′ end of R1 sequences and 21 nucleotides from the 5′ end of its respective R2 paired end read. Sequences are merged using **FLASH2** (Magoc and Salzberg, 2011). **FLASH2** is run with the following arguments:

--min-overlap=50 --max-overlap=160 --allow-outies

18Sv9 paired end reads are merged using FLASH2 with the following arguments:

--min-overlap=50 --max-overlap=120 --allow-outies

Following merging, the merge quality is manually checked by examining the FLASH log file for the percentage of reads that were merged. Plates with low merge rates (< 70%) are manually checked to see if the alignments can be improved.  
Unmerged reads are discarded.

**File naming**

For sequence files with AM sampleIDs, file names are changed to the following format:

sampleID\_plateID.fasta

For sequence files downloaded from NCBI, file names are changed to a similar format using the NCBI biosampleID and BioProjectID, as below.

biosampleID\_BioProjectID.fasta

The sampleID of the mock communities and negative controls on each plate are standardised to the following naming convention (all listed for completeness):

- 16S Bacteria mock communities: **BACMOCK**  
- A16 Archaeal mock communities: **ARCMOCK**  
- 18Sv4 Eukaryote mock communities: **EUKMOCKV4**  
- 18Sv9 Eukaryote mock communities: **EUKMOCKV9**  
- ITS Fungal mock communities: **FUNMOCK**  
- Negative control: **NEG**  
- STAN: **STAN**  
- MSA-1002 (ATCC) 20 Strain even mix genomic material: **ATCC1002MOCK**

**Fastq** files are then converted to **fasta** using **seqTk**

**Add sample and file name to the sequence headers**

Sample identifiers are added to the header of each sequence for downstream processing. As each fasta is now named **sampleID\_plateID.fasta** we simply add the file name with the extension to the sequence header. At this stage we also add any other information and delimiters that downstream programs will likely require. For **USEARCH** we add “sample=” and “;”, for **QIIME** we add “\_”. A bash script to perform this can be found at [https://raw.githubusercontent.com/AusMicrobiome/misc\_tools/master/add\_sample\_name.sh](https://github.com/AusMicrobiome/misc_tools)

**Identify unique sequences per sample**

An abundance table of all unique sequences in each sample on the plate is prepared. Unique sequences are identified using **FASTX** with the following **USEARCH** command:

usearch -fastx\_uniques SampleID\_plateID.trimmed.fasta -fastaout SampleID\_plateID\_uniques.fasta -sizeout

**Unique sequences are then converted into a 3 column tab separated abundance table containing the columns: seq\tSample\tAbundance. Unique sequences per sample (non-Denoised or quality filtered) are provided as a data output and are available by request through the Australian Microbiome Website.**

**B] Sample-wise ZOTU analysis**

**Quality screening, ZOTU calling and sequence mapping**

The first step performs some quality control on the sequences, with each amplicon region having different parameters.

For 18Sv4, the quality screen removes sequences that are too short, have ambiguous bases, or have more than 12 homopolymers.

mothur "#set.dir(modifynames=F); screen.seqs(fasta= sampleID\_plateID.fasta, minlength=300, maxambig=0, maxhomop=12, processors=10);"

For 18Sv9, the quality screen removes sequences that have ambiguous bases, or have more than 12 homopolymers.

mothur "#set.dir(modifynames=F); screen.seqs(fasta= sampleID\_plateID.fasta, maxambig=0, maxhomop=12, processors=10);"

Following quality screening, the remainder of the ZOTU workflow follows the same parameters for both amplicon types

Reads are dereplicated

usearch -fastx\_uniques sampleID\_plateID.good.fasta -fastaout sampleID\_plateID.good\_uniques.fasta -sizeout

Sort unique reads by abundance

usearch -sortbysize sampleID\_plateID.good\_uniques.fasta -fastaout sampleID\_plateID.good\_sorted\_uniques.fasta -sizeout

ZOTUs are called by **UNOISE3**, from sequences that have => 8 representatives

usearch -unoise3 sampleID\_plateID.good\_sorted\_uniques.fasta -ZOTUs sampleID\_plateID.good\_sorted\_uniques\_ZOTUs.fasta -minsize 8

**Concatenate all sample-wise ZOTUs into a single file**

All ZOTU files for each sample are concatenated into a single file, this file is later combined with ZOTUs obtained from the per-plate analysis. The resulting file name is standardised to the format: **plateID\_all\_SW\_ZOTUs.fasta**

**C] Plate-wise ZOTU analysis**

**Concatenate all sequences per sequencing run into a single file**

All sample fasta files for each plate are concatenated into a single file ZOTU calling. The resulting file name is standardised to the format: **plateID\_all\_18Svn.fasta**.  
Where 18Svn represents the 18S variable region being analysed (18Sv4 or 18Sv9)

**Quality screening, ZOTU calling and sequence mapping**

The first step performs some quality control on the sequences, with each amplicon region having different parameters:  
For 18Sv4, quality screening removes sequences that are too short, have ambiguous bases, or have more than 12 homopolymers.

mothur "#set.dir(modifynames=F);summary.seqs(fasta=plateID\_all\_18Sv4.fasta, processors=10); screen.seqs(fasta=current, minlength=300, maxambig=0, maxhomop=12, processors=10); summary.seqs()"

For 18Sv9 quality screening removes sequences have ambiguous bases, or have more than 12 homopolymers

mothur "#set.dir(modifynames=F);summary.seqs(fasta=plateID\_all\_18Sv9.fasta, processors=10); screen.seqs(fasta=current, maxambig=0, maxhomop=12, processors=10); summary.seqs()"

Following quality screening, the remainder of the ZOTU workflow follows the same parameters for both amplicon types.

Next reads are dereplicated

usearch -fastx\_uniques plateID\_all\_18Svn.good.fasta -fastaout plateID\_all\_18SVn.good\_uniques.fasta -sizeout

Sort unique reads by abundance

usearch -sortbysize plateID\_all\_18Svn.good\_uniques.fasta -fastaout plateID\_all\_16S.good\_sorted\_uniques.fasta -sizeout

ZOTUs are called by **UNOISE3**, from sequences that have => 8 representatives

usearch -unoise3 plateID\_all\_18Svn.good\_sorted\_uniques.fasta -ZOTUs plateID\_all\_18Svn.good\_sorted\_uniques\_ZOTUs.fasta -ampout plateID\_all\_18Svn.good\_sorted\_uniques\_ampout.fasta -tabbedout plateID\_all\_18Svn.good\_sorted\_uniques\_unoise3.txt -minsize 8

Combine the ZOTUs obtained from sample-wise and plate-wise analysis with a defined set of “prior” sequences into a single fasta file. The resulting file name is standardised to the format: **plateID\_all\_18Svn.good\_sorted\_uniques\_ZOTUs\_renamed\_SWpriors.fasta**

Dereplicate the combined ZOTU file

usearch -fastx\_uniques plateID\_all\_18Svn.good\_sorted\_uniques\_ZOTUs\_renamed\_SWpriors.fasta -fastaout plateID\_all\_18Svn.good\_sorted\_uniques\_ZOTUs\_renamed\_SWpriors\_uniq.fasta

Map reads to ZOTUs to generate abundances

Reads are mapped against the ZOTUs using **USEARCH**. Note the termination conditions on the mapping run (**-maxaccepts 0**), this seems to be required to ensure the best match is found and to produce consistent results when adding multiple plates together, as we do later.

usearch -otutab plateID\_all\_18Svn.fasta -ZOTUs plateID\_all\_18Svn.good\_sorted\_uniques\_ZOTUs\_renamed\_SWpriors\_uniq.fasta -otutabout plateID\_all\_18Svn.good\_sorted\_uniques\_PWSW\_ZOTUtab\_MA0.txt -mapout plateID\_all\_18Svn.good\_sorted\_uniques\_PWSW\_zmap\_MA0.txt -maxaccepts 0 -threads 10

**Classify and remove flipped sequences**

A final QC step is performed to remove likely erroneous sequences. The ZOTUs are classified, with those that do not align to the 18S Silva database (Quast et al., 2013; Yilmaz et al 2014; Glöckner 2017) in the correct orientation being removed. Those that need to be “flipped” to a new orientation are likely errors, since we know the reads should be in correct orientation for their respective primer sets 18S\_V4f/18S\_V4r or ILM\_Euk\_1391f/ILM\_EukBr. This step typically removes < 10 ZOTUs from the database.

**1. classify the seqs against 18S database**

"#set.dir(modifynames=F); classify.seqs(fasta=plateID\_all\_18Svn.good\_sorted\_uniques\_PWSW\_zotutab\_relabelled\_MA0.fasta, reference=silva.nr\_v132.align, taxonomy=silva.nr\_v132.tax, cutoff=60, probs=FALSE, processors=5)"

**2. use the \*acnos.flip list to remove sequences from the table**

The above produces the final abundance table, with sequences as index for each sequencing run. These tables are then combined as below to produce a single dataset.

**Replace table indexes with ZOTU sequence**

Currently the tables have an arbitrary OTU number as the index, replace this with the sequence that the arbitrary number represents. These sequences are unique strings and allow tables to be merged etc. downstream easily. They also negate the need to maintain a dictionary of ZOTUs and the sequences they represent.

**D] Prepare the single dataset**

Now we have a ZOTU abundance table for each plate, with ZOTU’s as row and SampleID\_plateID as column headers. To prepare this data for ingest into the AM database the following steps are carried out:

1. Each table is converted from short to long format (from rectangular to 3 column, with columns ZOTU, sampleID, Abundance)  
 2. All of these 3 column tables are concatenated into a single table  
 3. Controls and samples are split into separate tables  
 4. Sequencing run ID's are removed from the column headers and any sample sequenced more than once has the OTUs grouped and abundances summed to give a single abundance per sample  
 5. A fasta file of unique ZOTU's is created from all ZOTU's in this final table  
 6. Sequences are classified.

**Classify the sequences**

Sequences are classified to provide taxonomies relative to the Silva 138 database (Quast et al., 2013; Yilmaz et al 2014; Glöckner 2017). Sequence classification is performed using QIIME2 sklearn Bayesian classifier with the following arguments:

--p-confidence 0.6 --p-read-orientation same

**References**

Magoc, T. and Salzberg, S. (2011). FLASH: Fast length adjustment of short reads to improve genome assemblies. Bioinformatics 27(21): 2957-2963.  
Schloss, P.D., et al., (2009). Introducing mothur: Open-source, platform-independent, community-supported software for describing and comparing microbial communities. Appl Environ Microbiol 75(23):7537-7541.  
Edgar, R.C., (2010), Search and clustering orders of magnitude faster than BLAST, Bioinformatics 26(19): 2460-2461.  
Seqtk available at: <https://github.com/lh3/seqtk> (last accessed 23 Jan 2019).  
FASTX available at: (<http://hannonlab.cshl.edu/fastx_toolkit/>) (last accessed 23 Jan 2019).  
DeSantis T.Z., Hugenholtz P., Larsen N., Rojas M., Brodie E.L., Keller K., Huber T., Dalevi D., Hu P., Andersen G.L.(2006) Greengenes, a Chimera-Checked 16S rRNA Gene Database and Workbench Compatible with ARB. Appl. Environ. Microbiol. 72(7): 5069-5072; DOI: 10.1128/AEM.03006-05  
Quast C, Pruesse E, Yilmaz P, Gerken J, Schweer T, Yarza P, Peplies J, Glöckner FO (2013) The SILVA ribosomal RNA gene database project: improved data processing and web-based tools. Nucl. Acids Res. 41 (D1): D590-D596.  
Yilmaz P, Parfrey LW, Yarza P, Gerken J, Pruesse E, Quast C, Schweer T, Peplies J, Ludwig W, Glöckner FO (2014) The SILVA and “All-species Living Tree Project (LTP)” taxonomic frameworks. Nucl. Acids Res. 42:D643-D648  
Glöckner FO, Yilmaz P, Quast C, Gerken J, Beccati A, Ciuprina A, Bruns G, Yarza P, Peplies J, Westram R, Ludwig W (2017) 25 years of serving the community with ribosomal RNA gene reference databases and tools. J. Biotechnol.

Bolyen E, Rideout JR, Dillon MR, Bokulich NA, Abnet CC, Al-Ghalith GA, Alexander H, Alm EJ, Arumugam M, Asnicar F, Bai Y, Bisanz JE, Bittinger K, Brejnrod A, Brislawn CJ, Brown CT, Callahan BJ, Caraballo-Rodríguez AM, Chase J, Cope EK, Da Silva R, Diener C, Dorrestein PC, Douglas GM, Durall DM, Duvallet C, Edwardson CF, Ernst M, Estaki M, Fouquier J, Gauglitz JM, Gibbons SM, Gibson DL, Gonzalez A, Gorlick K, Guo J, Hillmann B, Holmes S, Holste H, Huttenhower C, Huttley GA, Janssen S, Jarmusch AK, Jiang L, Kaehler BD, Kang KB, Keefe CR, Keim P, Kelley ST, Knights D, Koester I, Kosciolek T, Kreps J, Langille MGI, Lee J, Ley R, Liu YX, Loftfield E, Lozupone C, Maher M, Marotz C, Martin BD, McDonald D, McIver LJ, Melnik AV, Metcalf JL, Morgan SC, Morton JT, Naimey AT, Navas-Molina JA, Nothias LF, Orchanian SB, Pearson T, Peoples SL, Petras D, Preuss ML, Pruesse E, Rasmussen LB, Rivers A, Robeson MS, Rosenthal P, Segata N, Shaffer M, Shiffer A, Sinha R, Song SJ, Spear JR, Swafford AD, Thompson LR, Torres PJ, Trinh P, Tripathi A, Turnbaugh PJ, Ul-Hasan S, van der Hooft JJJ, Vargas F, Vázquez-Baeza Y, Vogtmann E, von Hippel M, Walters W, Wan Y, Wang M, Warren J, Weber KC, Williamson CHD, Willis AD, Xu ZZ, Zaneveld JR, Zhang Y, Zhu Q, Knight R, and Caporaso JG. (2019) Reproducible, interactive, scalable and extensible microbiome data science using QIIME 2. Nature Biotechnology 37: 852–857.<https://doi.org/10.1038/s41587-019-0209-9>

# Fungal ITS Amplicon Analysis Workflow

This section outlines the workflow required to analyse amplicon sequences of the internal transcribed spacer (ITS) region located between the small and large rRNA subunits to produce Amplicon Sequence Variant (ZOTU) information for the Australian microbiome Initiative. Amplicons are derived from primers targeting the fungal ITS1 and ITS4 regions (ITS1F and ITS4).  
the ZOTU analysis is performed both on a per sample and per sequencing run (sequencing plate) basis. The workflow allows for the inclusion of known “prior” sequences to be added following denoising steps. The workflow consists of the following stages:

**A] Sequence preparation and merging**  
 1. Convert R1 and R2 fastq files to fasta file format

2. Generate the reverse complement of R2 reads  
 3. Identify and isolate putative fungal ITS1 and ITS2 regions from R1 and R2 reads  
 4. Add sampleID, runID and "sample=" information to the sequence headers

5. Generate abundance table of unique sequences per sample  
**B] Sample-wise ZOTU analysis**

1. Quality screening, ZOTU calling on individual samples on a plate

2. Concatenate all sample-wise ZOTU in the sequencing run into a single file

**C] Plate-wise ZOTU analysis**  
 1. Concatenate all sequences per sequencing run into a single file

2. Quality screening, ZOTU calling of concatenated samples on the plate

3. Concatenate plate-wise, sample-wise and prior sequences into a single file

4. Remove duplicated ZOTU sequences and sequence mapping  
 5. Classify and remove flipped sequences  
 6. Replace arbitrary ZOTU ID's with the sequence itself in the table index  
**D] Following analysis, all data is combined to give a single dataset, by the following steps:**  
 1. Merge tables into a single table  
 2. Remove controls from the abundance tables, to create separate samples and control datasets  
 3. Make a fasta from the abundance table  
 4. Classify Sequences

**Software used**

1. ITSx (Bengtsson-Palme et al., 2013)  
 2. Mothur (Schloss, et al., 2009)  
 3. USEARCH (Edgar 2010)  
 4. Seqtk (<https://github.com/lh3/seqtk>)  
 5. FASTX (<http://hannonlab.cshl.edu/fastx_toolkit/>)  
 6. Python3.X

**A] Sequence preparation and merging**

**Identify and isolate ITS1 and ITS2 regions**

Illumina fastq R1 and R2 files are first converted to fasta file format using SeqTk. In addition, SeqTk is used to generate the reverse complement of R2 reads.

ITSx (Bengtsson-Palme et al., 2013) is the used to identify and isolate fungal ITS1 and ITS2 regions from neighbouring ribosomal genes (SSU, 5S and LSU rRNA sequences). Arguments used for ITSx are as follows:

-t F --complement F --preserve T --partial 100 --save\_regions ITSn --detailed\_results T

R1 and R2 reads not identified as ITS by ITSx are discarded

**Rename files**

For sequence files with AM sampleIDs, file names are changed to the following format:

sampleID\_plateID.fasta

For sequence files downloaded from NCBI, file names are changed to a similar format using the NCBI biosampleID and BioProjectID, as below.

biosampleID\_BioProjectID.fasta

The sampleID of the mock communities and negative controls on each plate are standardised to the following naming convention (all listed for completeness):

- 16S Bacteria mock communities: **BACMOCK**  
- A16 Archaeal mock communities: **ARCMOCK**  
- 18Sv4 Eukaryote mock communities: **EUKMOCKV4**  
- 18Sv9 Eukaryote mock communities: **EUKMOCKV9**  
- ITS Fungal mock communities: **FUNMOCK**  
- Negative control: **NEG**  
- STAN: **STAN**  
- MSA-1002 (ATCC) 20 Strain even mix genomic material: **ATCC1002MOCK**

**Add sample and file name to the sequence headers**

Sample identifiers are added to the header of each sequence for downstream processing. As each fasta is now named **SampleID\_plateID.fasta** we simply add the file name with the extension to the sequence header. At this stage we also add any other information and delimiters that downstream programs will likely require. For **USEARCH** we add “sample=” and “;”, for **QIIME** we add “\_”. A bash script to perform this can be found at [https://raw.githubusercontent.com/AusMicrobiome/misc\_tools/master/add\_sample\_name.sh](https://github.com/AusMicrobiome/misc_tools)

**unique sequences per sample**

An abundance table of all unique sequences in each sample on the plate is prepared. Unique sequences are identified using **FASTX** with the following **USEARCH** command:

usearch -fastx\_uniques SampleID\_plateID.fasta -fastaout SampleID\_plateID\_uniques.fasta -sizeout

**Unique sequences are then converted into a 3 column tab separated abundance table containing the columns: seq\tSample\tAbundance. Unique sequences per sample (non-Denoised or quality filtered) are provided as a data output and are available by request through the Australian Microbiome Website.**

**B] Sample-wise ZOTU analysis**

**Quality screening, ZOTU calling and sequence mapping**

The first step performs some quality control on the sequences.

mothur "#set.dir(modifynames=F); screen.seqs(fasta= sampleID\_plateID.fasta, maxambig=0, maxhomop=12, processors=10);"

Reads are dereplicated

usearch -fastx\_uniques sampleID\_plateID.good.fasta -fastaout sampleID\_plateID.good\_uniques.fasta -sizeout

Sort unique reads by abundance

usearch -sortbysize sampleID\_plateID.good\_uniques.fasta -fastaout sampleID\_plateID.good\_sorted\_uniques.fasta -sizeout

ZOTUs are called by **UNOISE3**, from sequences that have => 8 representatives

usearch -unoise3 sampleID\_plateID.good\_sorted\_uniques.fasta -ZOTUs sampleID\_plateID.good\_sorted\_uniques\_ZOTUs.fasta -minsize 8

**Concatenate all sample-wise ZOTUs into a single file**

All ZOTU files for each sample are concatenated into a single file, this file is later combined with ZOTUs obtained from the per-plate analysis. The resulting file name is standardised to the format: **plateID\_all\_SW\_ZOTUs.fasta**

**C] Plate-wise ZOTU analysis**

**Concatenate all sequences into a single file**

All sample fasta files for each plate are concatenated into a single file ZOTU. The resulting file name is standardised to the format: **plateID\_all\_ITSn.fasta**.  
Where ITSn represents the ITS region being analysed (e.g., ITS1 or ITS2)

**Quality screening, ZOTU calling and sequence mapping**

The first step removes sequences that have ambiguous bases, or have more than 12 homopolymers.

mothur "#set.dir(modifynames=F);summary.seqs(fasta=plateID\_all\_ITSn.fasta, processors=10); screen.seqs(fasta=current, maxambig=0, maxhomop=12, processors=10); summary.seqs()"

Next reads are dereplicated

usearch -fastx\_uniques plateID\_all\_ITSn.good.fasta -fastaout plateID\_all\_ITSn.good\_uniques.fasta -sizeout

Sort unique reads by abundance

usearch -sortbysize plateID\_all\_ITSn.good\_uniques.fasta -fastaout plateID\_all\_ITSn.good\_sorted\_uniques.fasta -sizeout

ZOTUs are called by **UNOISE3**, from sequences that have => 8 representatives

usearch -unoise3 plateID\_all\_ITSn.good\_sorted\_uniques.fasta -zotus plateID\_all\_ITSn.good\_sorted\_uniques\_zotus.fasta -ampout plateID\_all\_ITSn.good\_sorted\_uniques\_ampout.fasta -tabbedout plateID\_all\_ITSn.good\_sorted\_uniques\_unoise3.txt -minsize 8

Combine the ZOTUs obtained from sample-wise and plate-wise analysis with a defined set of “prior” sequences into a single fasta file. The resulting file name is standardised to the format: **plateID\_all\_ITSn.good\_sorted\_uniques\_ZOTUs\_renamed\_SWpriors.fasta**

Dereplicate the combined ZOTU file

usearch -fastx\_uniques plateID\_all\_ITSn.good\_sorted\_uniques\_ZOTUs\_renamed\_SWpriors.fasta -fastaout plateID\_all\_ITSn.good\_sorted\_uniques\_ZOTUs\_renamed\_SWpriors\_uniq.fasta

Map reads to ZOTUs to generate abundances

Reads are mapped against the ZOTUs using **USEARCH**. Note the termination conditions on the mapping run (**-maxaccepts 0**), this seems to be required to ensure the best match is found and to produce consistent results when adding multiple plates together, as we do later.

usearch -otutab plateID\_all\_ITSn.fasta -ZOTUs plateID\_all\_ITSn.good\_sorted\_uniques\_ZOTUs\_renamed\_SWpriors\_uniq.fasta -otutabout plateID\_all\_ITSn.good\_sorted\_uniques\_PWSW\_ZOTUtab\_MA0.txt -mapout plateID\_all\_ITSn.good\_sorted\_uniques\_PWSW\_zmap\_MA0.txt -maxaccepts 0 -threads 10

**Replace ZOTU table indexes with ZOTU sequence**

Currently the ZOTU tables have an arbitrary ZOTU number as the index, replace this with the sequence that the arbitrary number represents. These sequences are unique strings and allow tables to be merged etc. downstream easily. They also negate the need to maintain a dictionary of ZOTUs and the sequences they represent.

**Classify and remove flipped sequences**

A final QC step is performed to remove likely erroneous sequences. The ZOTUs are classified, with those that do not align to the UNITE SH ITS database in the correct orientation being removed. As we know that the R1 sequences are correctly orientated and the reverse complement of the R2 also puts it into the correct orientation, sequences that need to be “flipped” to a new orientation to obtain the best alignment to the database are most likely errors. This step typically removes < 10 ZOTUs from the database.

**Classify the seqs against UNITE ITS database**

classify.seqs(fasta=${plateID}\_all\_ITSn.good\_sorted\_uniques\_PWSW\_zotutab\_relabelled\_MA0.fasta, reference=UNITEv7\_sh\_dynamic\_s.fasta, taxonomy=UNITEv8\_sh\_dynamic\_s.tax, cutoff=60, probs=T)

**Use the \*acnos.flip list to remove flipped sequences from the table**

The above produces the final abundance table, with sequences as index for each sequencing run. These tables are then combined as below to produce a single dataset.

**D] Prepare the single dataset**

Now we have a ZOTU abundance table for each plate, with ZOTU’s as row and sampleID\_plateID as column headers. To prepare this data for ingest into the AM database the following steps are carried out:

1. Each table is converted from short to long format (from rectangular to 3 column, with columns ZOTU, sampleID, Abundance)  
 2. All of these 3 column tables are concatenated into a single table  
 3. Controls and samples are split into separate tables  
 4. Sequencing run ID's are removed from the column headers and any sample sequenced more than once has the OTUs grouped and abundances summed to give a single abundance per sample  
 5. A fasta file is created from all ZOTU's in this final table  
 6. Sequences are classified.

**Classify the sequences**

Sequences are classified to provide taxonomies relative to the UNITE v8 SH ITS database as below:

mothur "#classify.seqs(fasta=seqs\_listSET.fasta, reference=UNITEv8\_sh\_dynamic\_s.fasta, taxonomy=UNITEv8\_sh\_dynamic\_s.tax, cutoff=60, probs=FALSE)"

**References**

Bengtsson-Palme, J., Veldre V., Ryberg M., Hartmann M., Branco S., Wang Z., Godhe A., Bertrand Y., De Wit P., Sanchez M., Ebersberger I., Sanli K., de Souza F., Kristiansson E., Abarenkov K., Eriksson K.M, Nilsson R.H.(2013). ITSx: Improved software detection and extraction of ITS1 and ITS2 from ribosomal ITS sequences of fungi and other eukaryotes for use in environmental sequencing. Methods in Ecology and Evolution, 4: 914-919.  
Schloss, P.D., et al., (2009). Introducing mothur: Open-source, platform-independent, community-supported software for describing and comparing microbial communities. Appl Environ Microbiol 75(23):7537-7541.  
Edgar, R.C., (2010), Search and clustering orders of magnitude faster than BLAST, Bioinformatics 26(19): 2460-2461.  
Seqtk available at: <https://github.com/lh3/seqtk> (last accessed 23 Jan 2019).  
FASTX available at: (<http://hannonlab.cshl.edu/fastx_toolkit/>) (last accessed 23 Jan 2019).  
Nilsson RH, Larsson K-H, Taylor AFS, Bengtsson-Palme J, Jeppesen TS, Schigel D, Kennedy P, Picard K, Glöckner FO, Tedersoo L, Saar I, Kõljalg U, Abarenkov K. (2018). The UNITE database for molecular identification of fungi: handling dark taxa and parallel taxonomic classifications. Nucleic Acids Research, DOI: 10.1093/nar/gky1022

# Metaxa Identification of Small Subuint Ribosomal RNA in Metagenomes

This document outlines the workflow required to identify small subunit ribosomal (SSU rRNA) from metagenomic studies for inclusion in the Australian microbiome database.

Analysis is completed on a per sequencing run (sequencing plate) basis. The workflow consists of the following stages:

**Sequence Analysis**

**A] Sequence analysis and reporting**

1. Identify paired end reads with identity to SSU rRNA using Metaxa2 (v2.2.3)

2. Summarise taxonomic assignments

**B] Preparation of the single dataset**

1. Merge tables into a single table

### **Software used**

1. Metax2 v2.2.3 (Bengtsson-Palme et al., 2015)  
   Note: Metaxa2 requires dependencies to be installed. The following are used:
   * HMMER version 3.1b2 (<http://hmmer.janelia.org/software>)
   * BLAST+ (v2.7.1) (<https://ftp.ncbi.nlm.nih.gov/blast/executables/blast>+/)
   * BLAST-legacy (v2.2.26) (<https://ftp.ncbi.nlm.nih.gov/blast/executables/legacy.NOTSUPPORTED/>)
   * MAFFT v7.301 (<http://mafft.cbrc.jp/alignment/software/>)
2. Python3.x

## **A] Sequence analysis and reporting**

## **Taxonomic Analysis of Sequences**

Metaxa2 is run with the following arguments

metaxa2 -1 SampleID\_R1.fastq.gz -2 SampleID\_R2.fastq.gz -o SampleID\_out -f q --date T --cpu 20 --table T

## **Reporting Taxonomy Predictions**

The Metaxa2 Taxonomy Traversal tool is used to report the taxonomic predictions at specific cutoffs that approximately correspond to different nodes of the taxonomic tree (e.g., kingdoms, phyla, classes, orders, families, genera, species, subspecies). The output of this analysis also includes the total abundance of each unique taxa observed.

The input for the taxonomy traversal tool is the SampleID\_taxonomy.txt file obtained during Metaxa2 analysis, and is invoked using the following command:

metaxa2\_ttt -i SampleID\_taxonomy.txt -o SampleID\_taxonomy.txt

**B] Prepare the single dataset**

Now we have a taxonomy/abundance table for each sample read pair, with the taxonomy and abundance represented as tab separated columns. To prepare this data for ingest into the AM database the following steps are carried out using python3:

1. Using the level 7 taxonomy file(s), sampleID’s are added to each row as a tab separated column  
 2. All 3 column tables are concatenated into a single table  
 3. Reads classified as unknown, mitochondria or chloroplast are removed  
 4. Any sample sequenced more than once are grouped by sampleID and taxonomy string with abundances summed to give a single abundance