# Bacterial and Archaeal 16S rRNA gene Amplicon Analysis Workflow

This section outlines the workflow required to analyse 16S rRNA amplicon sequences for Bacteria (27f – 519r) and Archaea (A2f-519r), to produce Amplicon Sequence Variant (ASV) information for the Australian Microbiome database  
Analysis is completed on a per sequencing run (sequencing plate) basis. 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. Concatenate all sequences per sequencing run into a single file  
 **B] Sequence analysis**  
 1. Quality screening, zotu/ASV calling and sequence mapping  
 2. Classify and remove flipped sequences  
 3. Replace arbitrary ASV ID's with the sequence itself in the table index  
 **C] 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 Required**

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. Python3

**A] Sequence preparation and merging**

**Merge the paired end reads**

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

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

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**

File names are formatted to the following format:

sampleID\_plateID.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**

**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 “\_”.

#!/bin/bash  
#this initiates the loop for all files with extension .fasta in the list generated by ls -l  
 **for file in `ls -1 \*.fasta`  
 do**  
#tell me which file you're working on  
 **echo working on $file**  
#cut the appropriate string from the name of the file.  
 **VAR1=`echo $file | cut -d "." -f 1`**  
#tell me what you've cut and called VAR1  
 **echo VAR1 is $VAR1**  
#use VAR1 to add the sample name to the sequence id. Substitute > for >VAR1\_ in all files. The variable (VAR1) needs to be in quotation marks to be passed from bash to perl, the perl script needs to be embedded in the bash script with the ` ` marks  
 **`perl -pi -e 's/\>/\>sample='$VAR1';\_16S\_/' $file;`  
 done**

**Identify unique sequences per sample**

An abundance table of all unique sequences in each sample on the plate is then prepared. Unique sequences are first identified using fastx using the following Usearch command:

usearch64 -fastx\_uniques SampleID\_plateID.fasta -fastaout SampleID\_plateID\_uniques.fasta -sizeout

**Unique sequences per sample (non-Denoised or quality filtered) are provided as a data output and are available from the Australian Microbiome Website.**

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

All of the files for each plate are concatenated into a single file for zotu/ASV calling. The resulting file name is standardised to the format: **plateID\_all\_16S.fasta**

**B] Sequence analysis**

**Quality screening, zotu/ASV calling and sequence mapping**

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

mothur "#set.dir(modifynames=F);summary.seqs(fasta=plateID\_all.fasta, processors=10); screen.seqs(fasta=current, minlength=432, maxlength=520, maxambig=0, maxhomop=12, processors=10); summary.seqs()"

Reads are dereplicated

usearch64 -fastx\_uniques plateID\_all\_16S.good.fasta -fastaout plateID\_all\_16S.good\_uniques.fasta -sizeout

Sort unique reads by abundance

usearch64 -sortbysize plateID\_all\_16S.good\_uniques.fasta -fastaout plateID\_all\_16S.good\_sorted\_uniques.fasta -sizeout

ASV/Zotus are called by **UNOISE3**, from sequences that have => 4 representatives

usearch64 -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 4

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.

usearch64 -otutab plateID\_all\_16S.fasta -zotus plateID\_all\_16S.good\_sorted\_uniques\_zotus\_renamed.fasta -otutabout plateID\_all\_16S.good\_sorted\_uniques\_zotutab\_MA0.txt -mapout plateID\_all\_16S.good\_sorted\_uniques\_zmap\_MA0.txt -maxaccepts 0 -threads X

**Classify and remove flipped sequences**

A final QC step is performed to remove likely erroneous sequences. The ASVs 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.

**1. 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)"

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

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

**Replace table indexes with ASV sequence**

After denoising and mapping the ASV 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 ASVs and the sequences they represent.

**C] Prepare the single dataset**

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

1. Each table is converted from short to long format (from rectangular to 3 column, with columns ASV, 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 abundances from these runs summed to give a single abundance per sample  
5. A fasta file of unique ASV's is created from all ASV's in this final table  
6. Sequences are classified.

**Classify the sequences**

Sequences are classified to provide taxonnomies relative to the Silva database (Quast et al., 2013; Yilmaz et al 2014; Glöckner 2017) as below:

mothur "#classify.seqs(fasta=seqs.fasta, reference=SILVA132.ng.fasta, taxonomy=SILVA132.tax, cutoff=60, probs=FALSE, processors=X)"

**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.

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

**A] Sequence preparation and merging**  
 1. Trim and 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. Concatenate all sequences per sequencing run into a single file  
 **B] Sequence analysis**  
 1. Quality screening, zotu/ASV calling and sequence mapping  
 2. Classify and remove flipped sequences  
 3. Replace arbitrary ASV ID's with the sequence itself in the table index  
 **C] 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 Required**

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. Python3

**A] Sequence preparation and merging**

**Merge the paired end reads**

For 18Sv4 amplicons, primer removal is performed using seqTk by hard triming 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**

File names are formatted to the following format:

sampleID\_plateID.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 Archael mock communities: **ARCMOCK**  
- 18Sv4 Eukaryote mock communities: **EUKMOCKv4**  
- 18Sv9 Eukaryote mock communities: **EUKMOCKv9**  
- ITS Fungal mock communities: **FUNMOCK**  
- Negative control: **NEG**  
- STAN: **STAN**

**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 “\_”.

#!/bin/bash  
#this initiates the loop for all files with extension .fasta in the list generated by ls -l  
 **for file in `ls -1 \*.fasta`  
 do**  
#tell me which file you're working on  
 **echo working on $file**  
#cut the appropriate string from the name of the file.  
 **VAR1=`echo $file | cut -d "." -f 1`**  
#tell me what you've cut and called VAR1  
 **echo VAR1 is $VAR1**  
#use VAR1 to add the sample name to the sequence id. Substitute > for >VAR1\_ in all files. The variable (VAR1) needs to be in quotation marks to be passed from bash to perl, the perl script needs to be embedded in the bash script with the ` ` marks  
 **`perl -pi -e 's/\>/\>sample='$VAR1';\_18S\_/' $file;`  
 done**

**Concatenate all sequences into a single file**

All of the files for each plate are concatenated into a single file zotu/ASV 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)

**B] Sequence analysis**

**Quality screening, zotu/ASV calling and sequence mapping**

The first step performs some quality control on the sequences, with each ampplicon having different parameters:  
For 18SV4, the first step 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\_18SVn.fasta, processors=10); screen.seqs(fasta=current, minlength=300, maxambig=0, maxhomop=12, processors=10); summary.seqs()"

For 18SV9 the first step removes sequences have ambiguous bases, or have more than 12 homopolymers

mothur "#set.dir(modifynames=F);summary.seqs(fasta=plateID\_all\_18SVn.fasta, processors=10); screen.seqs(fasta=current, maxambig=0, maxhomop=12, processors=10); summary.seqs()"

Next reads are dereplicated

usearch64 -fastx\_uniques plateID\_all\_16S.good.fasta -fastaout plateID\_all\_18SVn.good\_uniques.fasta -sizeout

Sort unique reads by abundance

usearch64 -sortbysize plateID\_all\_18SVn.good\_uniques.fasta -fastaout plateID\_all\_16S.good\_sorted\_uniques.fasta -sizeout

ASV/zotus are called by **UNOISE3**, from sequences that have => 4 representatives

usearch64 -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 4

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.

usearch64 -otutab plateID\_all\_18SVn.fasta -zotus plateID\_all\_18SVn.good\_sorted\_uniques\_zotus\_renamed.fasta -otutabout plateID\_all\_18SVn.good\_sorted\_uniques\_zotutab\_MA0.txt -mapout plateID\_all\_18SVn.good\_sorted\_uniques\_zmap\_MA0.txt -maxaccepts 0 -threads X

**Classify and remove flipped sequences**

A final QC step is perfomed to remove likely erroneous sequences. The ASVs 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.

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

"#set.dir(modifynames=F); classify.seqs(fasta=plateID\_all\_18Svn.good\_sorted\_uniques\_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 ASV sequence**

Currently the tables have an arbitary 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 ASVs and the sequences they represent.

**C] Prepare the single dataset**

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

1. Each table is converted from short to long format (from rectangular to 3 column, with columns ASV, 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 abundances from these runs summed to give a single abundance per sample  
 5. A fasta file of unique ASV's is created from all ASV's in this final table  
 6. Sequences are classified.

**Classify the sequences**

Sequences are classified to provide taxonomies relative to the Silva database (Quast et al., 2013; Yilmaz et al 2014; Glöckner 2017) as below:

mothur "#classify.seqs(fasta=seqs.fasta, reference=SILVA132.ng.fasta, taxonomy=SILVA132.tax, cutoff=60, probs=FALSE, processors=X)"

**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.

.

# 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 (ASV) information for the Australian microbiome Initiative. Amplicons are derived from primers targeting the fungal ITS1 and ITS4 regions (ITS1F and ITS4).  
Analysis is completed on a per sequencing run (sequencing plate) basis. The workflow consists of the following stages:

**A] Sequence preparation and merging**  
 1. Convert fastq file format to fasta file format   
 2. Identify and isolate putative fungal ITS1 and ITS2 regions from R1 and R2 reads  
 3. Add sampleID, runID and "sample=" information to the sequence headers  
 4. Concatenate all sequences per sequencing run into a single file  
 **B] Sequence analysis**  
 1. Quality screening, zotu/ASV calling and sequence mapping  
 2. Classify and remove flipped sequences  
 3. Replace arbitrary ASV ID's with the sequence itself in the table index  
 **C] 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 Required**

The following software is used in the steps below:

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**

File names are formatted to the following format:

sampleID\_plateID.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**

**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 “\_”.

#!/bin/bash  
#this initiates the loop for all files with extension .fasta in the list generated by ls -l  
 **for file in `ls -1 \*.fasta`  
 do**  
#tell me which file you're working on  
 **echo working on $file**  
#cut the appropriate string from the name of the file.  
 **VAR1=`echo $file | cut -d "." -f 1`**  
#tell me what you've cut and called VAR1  
 **echo VAR1 is $VAR1**  
#use VAR1 to add the sample name to the sequence id. Substitute > for >VAR1\_ in all files. The variable (VAR1) needs to be in quotation marks to be passed from bash to perl, the perl script needs to be embedded in the bash script with the ` ` marks  
 **`perl -pi -e 's/\>/\>sample='$VAR1';\_ITS\_/' $file;`  
 done**

**Concatenate all sequences into a single file**

All of the files for each plate are concatenated into a single file zotu/ASV calling. The resulting file name is standardised to the format: **plateID\_all\_ITSn.fasta**.  
Where ITSn represents the ITS region being analysed (ITS1 or ITS2)

**B] Sequence analysis**

**Quality screening, zotu/ASV 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

usearch64 -fastx\_uniques plateID\_all\_ITSn.good.fasta -fastaout plateID\_all\_ITSn.good\_uniques.fasta -sizeout

Sort unique reads by abundance

usearch64 -sortbysize plateID\_all\_ITSn.good\_uniques.fasta -fastaout plateID\_all\_ITSn.good\_sorted\_uniques.fasta -sizeout

ASV/Zotus are called by **UNOISE3**, from sequences that have => 4 representatives

usearch64 -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 4

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.

usearch64 -otutab plateID\_all\_ITSn.fasta -zotus plateID\_all\_ITSn.good\_sorted\_uniques\_zotus\_renamed.fasta -otutabout plateID\_all\_ITSn.good\_sorted\_uniques\_zotutab\_MA0.txt -mapout plateID\_all\_ITSn.good\_sorted\_uniques\_zmap\_MA0.txt -maxaccepts 0 -threads X

**Classify and remove flipped sequences**

A final QC step is performed to remove likely erroneous sequences. The ASVs are classified, with those that do not align to the UNITE SH ITS database in the correct orientation being removed. As we know the that R1 sequences 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.

**1. classify the seqs against UNITE ITS database**

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

**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 ASV sequence**

Currently the tables have an arbitary 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 ASVs and the sequences they represent.

**C] Prepare the single dataset**

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

1. Each table is converted from short to long format (from rectangular to 3 column, with columns ASV, 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 abundances from these runs summed to give a single abundance per sample  
 5. A fasta file is created from all ASV's in this final table  
 6. Sequences are classified.

**Classify the sequences**

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

mothur "#classify.seqs(fasta=seqs\_listSET.fasta, reference=UNITEv7\_sh\_dynamic\_s.fasta, taxonomy=UNITEv7\_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 (v2.1.3)

2. Summarise taxonomic assignments

**B] Preparation of the single dataset**

1. Merge tables into a single table

### Software required

1. Metax2 v2.1.3 (Bengtsson-Palme et al., 2015)  
   Note: Metaxa2 requires dependencies to be installed. The following were 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

Input files for Metaxa2 analysis are paired end R1 and R2 fastq files. The first step of the analysis is to unzip compressed R1 and R2 fastq.gz files using gunzip.

Metaxa2 is run on the unzipped files with the following arguments

metaxa2 -1 SampleID\_R1.fastq -2 SampleID\_R2.fastq -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**

## Summarising Taxonomy Predictions

The Metaxa2 Data Collector is used to generate a combined abundance table from multiple taxonomy traversal outputs. It performs this by merging files at each specified taxonomic level (e.g., SampleID.level\_X.txt) into a square abundance table.

The following script will run the Metaxa2 Data Collector for each taxonomic level (1-9) and output a corresponding rectangular abundance matrix.

for i in 1 2 3 4 5 6 7 8 9;

do

echo $i;

metaxa2\_dc -o level\_${i}\_collected -r \_.\*$ \*level\_${i}.txt;

done

For ingestion into the Australian Microbiome the rectangular abundance matrix output from the Metaxa2 Data Collector is converted to a three column abundance matrix.