otupipe

http://drive5.com/otupipe
robert@drive5.com

Version 1.1
August 19, 2011

Introduction

The otupipe script creates OTUs from next-generation sequence reads for single-region experiments such as 16S and ITS. Otupipe is a bash script, so you must have a bash shell installed. If you're using Windows, you can run otupipe under Cygwin.

Designed for 454 fixed-region sequencing

Version 1.1 is designed for reads generated by Roche 454 sequencing of a region of a gene extracted from a fixed pair of primers. Typically the gene is 16S, but this script should also work well for other regions such as ITS.

Using otupipe with Illumina

Experiments using Illumina sequencing on the 16S gene appear to show that error rates are much higher than the rates found in genome resequencing. The reasons for this are not understood, and to the best of my knowledge no effective denoising / error-correction solution for Illumina has been published for this type of experiment. Otupipe probably works at least as well as any other solution for generating OTUs from Illumina reads. I suggest setting the MINSIZE option to a much larger value, perhaps something like 100 to 1000, and keeping only the largest OTUs, perhaps with a minimum size of 5000 to 10000.

USEARCH

Otupipe requires USEARCH, so you should install this first if you don't already have it.

Installation

Otupipe is distributed as a tarball named something like otupipe1.1.123.tz, where 1.1.123 is the version number. To extract the files, use:

```
tar -zxvf otupipe1.1.123.tz
```

This creates a sub-directory otupipe1.0.123 with the following files:

```
otupipe.bash
fasta_number.py
```

Before running otupipe.bash, you should set up your environment as explained in the following table.

Action	Description
Install USEARCH.	The otupipe script requires USEARCH. To install, download the binary file and install in a directory that will be accessible from your bash $PATH when you run otupipe.
fasta_number.py in $PATH.	The fasta_number.py script must be accessible through your bash $PATH when you run otupipe.bash. You should copy fasta_number.py to a directory that is in your $PATH variable, or update your $PATH to include the otupipe directory.
Set $u.	The $u environment variable must be set to the name or path of your USEARCH binary. To verify that it is set correctly, try this command: `$u --version`
Set $UCHIME_REFDB.	The $UCHIME_REFDB database must be set to the path name of the reference database to use for UCHIME in reference database mode. You don't need to install UCHIME separately; USEARCH has its own implementation of the UCHIME algorithm. For a reference database, I generally recommend the 'gold' set distributed with ChimeraSlayer. For convenience, a copy of the gold set is available for download at http://drive5.com/otupipe, but I recommend checking the ChimeraSlayer page to see if a more recent version is available. Example: `export UCHIME_REFDB=/public/otupipe/gold.fa`

Basic usage

The otupipe script requires two arguments: an input file in FASTA format containing quality-filtered reads and the name of a directory to contain the output files; this directory will be created if it does not already exist. For example:

```
otupipe.bash reads.fasta outdir [existing_otus.fasta]
```

Output files

Three output files are created, as described in the following table.

Filename	Description
otus.fa	FASTA file containing one representative sequence for each OTU. Sequences are named OTU_1, OTU_2... up to the number of OTUs.
chimeras.fa	FASTA file containing representative sequences for chimeras found in the reads. This file is clustered at 97% by default to reduce redundancy in a similar way to OTUs. Sequences are named CHIMERA_1, CHIMERA_2 etc.
readmap.uc	File in UCLUST format classifying reads as OTU, chimera or unknown.

The readmap.uc file

The readmap.uc file is generated by using the input file (reads) as a query against a database containing sequences from the otus.fa and chimeras.fa files. The UCLUST format is described in the USEARCH manual.

A UCLUST file is a tab-separated text file with one record per line. The readmap.uc file has one line for each read; either a hit (H) or no match (N) record. Most reads will match either an OTU or a chimera (H records); a small fraction of the reads will fail to match (N records). Reads that do not match are outliers that cannot be reliably classified. They might be rare biological sequences, or more likely they have errors due to PCR amplification or sequencing error.

Example readmap.uc records

UCLUST files have ten fields. Here are some example records.

Type 1	Cluster 2	Length 3	PctId 4	Strand 5	Lo 6	Hi 7	Alignment 8	Label 9	Target 10
H	97	224	97.3	+	0	0	224M5I	FV9NWLF01B7XBX	CHIMERA_21
H	0	254	100.0	+	0	0	254M	FV9NWLF01ENNCH	OTU_1
H	180	251	97.6	+	0	0	18MI233M	FV9NWLF01EU5SK	CHIMERA_104
H	12	264	99.6	+	0	0	264MI	FV9NWLF01DYX6E	OTU_13
H	169	250	100.0	+	0	0	250M	FV9NWLF01BUPFU	CHIMERA_93
H	1	242	100.0	+	0	0	242M2I	FV9NWLF01C3TV5	OTU_2
N	*	245	*	*	*	*	*	FV9NWLF01DT3FI	*

The most relevant fields are field 1 which is H (hit) or N (no match), field 4, which gives the identity of the match (≥ the $PCTID_BIN parameter, which defaults to 97), field 9 which is the read label, and field 10 which is the label of the closest representative sequence (OTU_n or CHIMERA_n).

Typically you will need to convert the readmap.uc file to a different format for downstream tools such as QIIME or mothur that perform phylotyping and population analysis, e.g., compute alpha and beta diversities. The UCLUST format is designed to be easy to parse, so it should be straightforward to write a file format conversion script in Perl, Python etc. If you need help with this, by all means contact me.

User-settable parameters

Parameters can be changed by setting environment variables before running otupipe.bash. Supported parameters are described in the following table.

Parameter	Default	Description
MINSIZE	4	Discard clusters < MINSIZE in error correction round. Small clusters are more likely to be noise. You can reduce this value if it is important not to lose rare biological sequences, but this will typically produce more spurious OTUs.
PCTID_ERR	97	Identity threshold for error correction.
PCTID_OTU	97	Identity threshold for OTU clustering.
PCTID_BIN	97	Identity threshold for read classification step. Reducing this value will cause more reads to be classified (OTU or chimera).
ABSKEW	2	Abundance skew for *de novo* chimera filtering. See the UCHIME paper and supplementary material for a discussion of abundance skew.
OTUTMPDIR	./otutmp.$$.$RANDOM	Name for directory to store temporary files. This directory is deleted (unless the script fails).
PREFIX	OTU_	Prefix for OTU labels.
CHIMPREFIX	CHIMERA_	Prefix for chimera labels.

For example,

```
export MINSIZE=2
export ABSKEW=4
otupipe reads.fasta out_size2_skew4
```

Using an existing OTU file

You can optionally use an existing OTU file as follows:

```
otupipe.bash reads.fasta outdir existing_otus.fasta
```

Here, existing_otus.fasta is a FASTA file containing representative sequences for precomputed OTUs. For example, existing_otus.fasta could be centroids created by otupipe.bash from some other set of reads. The existing_otus.fasta file *must* use a different label convention to avoid collisions. For example, if you use the default labels OTU_1, OTU_2... for new OTUs, then existing_otus.fasta could use labels X_1, X_2 and so on.

If you specify an existing_otus.fasta file, then:

- The otus.fa file will contain only new OTUs. New OTUs are created from reads that do not match the existing_otus.fasta file at the PCTID_OTU threshold, and will have identity < PCTID_OTU with all sequences in existing_otus.fa. New OTUs will be numbered 1, 2 etc.

- The readmap.uc file will map reads to both existing and new OTUs.