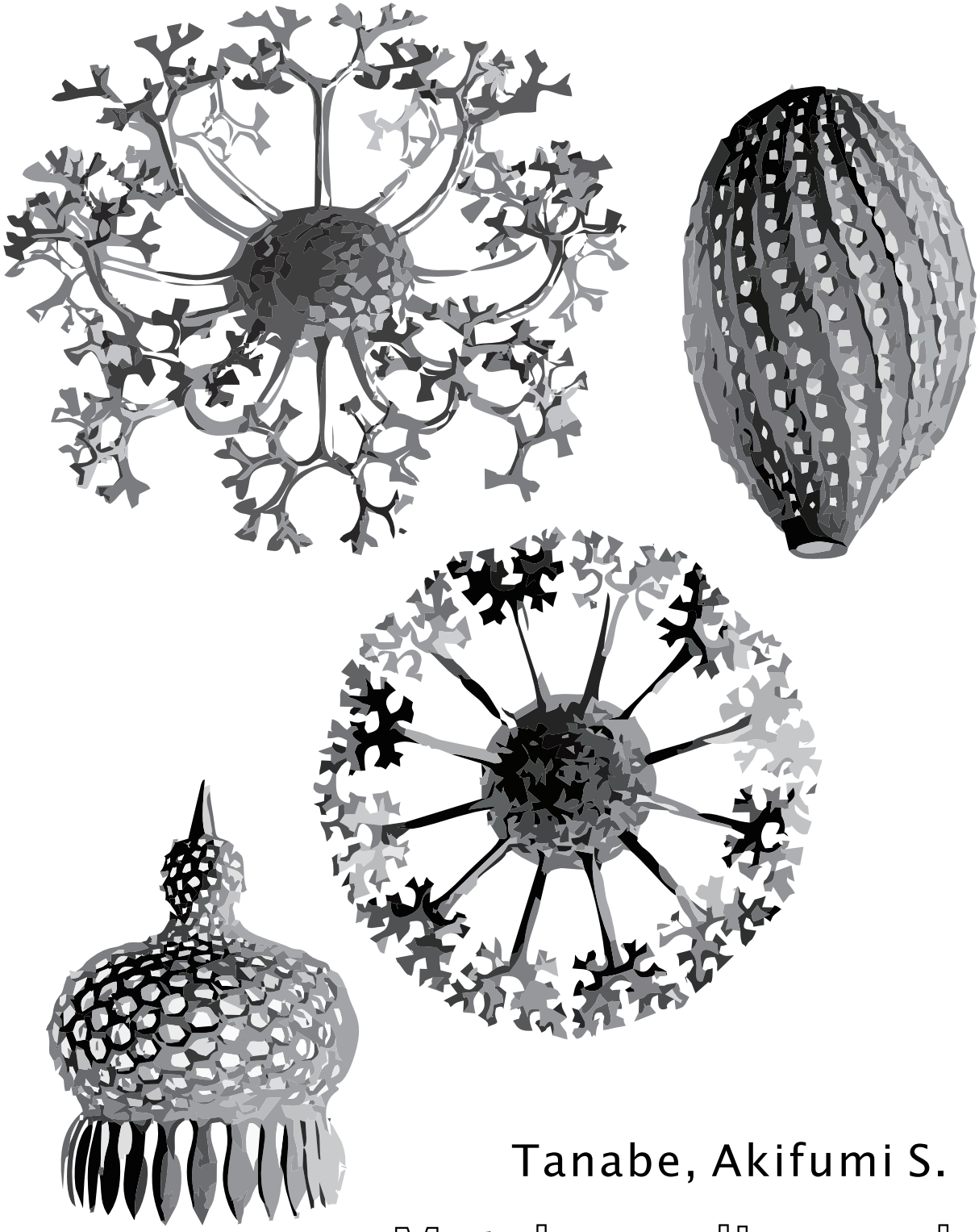

Tanabe, Akifumi S.

Metabarcoding and DNA barcoding for Ecologists

Metabarcoding and DNA barcoding for Ecologists

Akifumi S. Tanabe

September 4, 2017

Table of Contents

Preface

I started the writing of this text to promote amplicon sequence assember "Assams" and utility programs package "Claident" for sequence clustering and identification. In this text, I explain not only usage of these programs but also the methods and procedures of data collection and taxonomic identification by DNA barcoding.

Metabarcoding has been already widely used in bacterial reseaches, but it's utility is not limited to bacteria. Metabarcoding is also applicable to soil fungi, fungi inhabiting the bodies of animals and plants, aquatic planktons, and environmental DNA emitted from macro-organisms. I explain the methods of amplification of barcoding locus from environmental DNA or metagenomes, sequencing by high-throughput sequencers (a.k.a next-generation sequencers), taxonomic assignment of nucleotide sequences (a.k.a. DNA barcoding), and observation of presence/absence of operational taxonomic units.

I hope that this text helps you. I am grateful to Dr. Hirokazu Toju (Graduate Shool of Human and Environmental Sciences, Kyoto University), Dr. Satoshi Nagai (National Research Institute of Fisheries Science, Japan Fisheries Research and Education Agency), and you.

Legends

In this text, the input commands to terminals and display outputs are described as below.

```
# comments
> command argument1 \
argument2 \
argument3 ↓
output of command
> command argument1 argument2 argument3 ↓
output of command
```

In the above example, the same commands `command argument1 argument2 argument3` were executed twice. The outputs `output of command` were displayed after execution. The characters between # and line feed were comments and needless to input. > and space of line head indicate the prompt of terminal. Do not type these characters. ↓ means the end of input commands and arguments and needless to input, but you need to type Enter key to input line feed. I use line feed within commands or arguments for viewability. Such line feed is led by \. Therefore, the line feeds led by \ do not mean the end of commands or arguments, or designation to input Enter key. Involuntary line feeds may be generated by word wrap function depending on your read environment, but do not mean the end of commands or arguments, or designation to input Enter key.

The file content is shown as below in this text.

```
| The content of first line
| The content of second line
```

| and space of line head indicate the line head in the file, do not exist in the file and needless to input these characters. This code is written to help you to distinguish true line feeds and involuntary line feeds.

Chapter 0

Installing softwares and preparing analysis environment

In this text, I assume Debian GNU/Linux jessie (hereafter Debian) or Ubuntu Linux 14.04 LTS (hereafter Ubuntu) as operating system. If you use Windows PC, please install Debian or Ubuntu. Cygwin or Windows Subsystem for Linux provided for Windows 10 build 14393 or later can be used for the following analysis, but the programs run much more slowly. You can use CD, DVD or USB memory to boot installer of Linux. If your PC has only one storage device, you need to reduce Windows partition by using partition resizer software such as EaseUS Partition Master or using a partition resize function contained in the installer. You can also use newly added internal storage devices or external storage devices connected by USB. There are several variations of Ubuntu, and I recommend Xubuntu rather than normal Ubuntu.

Debian and Ubuntu can be installed to Mac. If there is no enough space, you need to resize OSX partition with the aid of Disk Utility or add storage device. The rEFIt or rEFInd boot selecter may be required to boot Debian, Ubuntu or the installer of them on Mac. If you install rEFIt or rEFInd to your Mac, you can boot the installer of Debian or Ubuntu from CD, DVD or USB memory. Do not delete existing partition of OSX. If you have enough free space, you don't need to use Disk Utility to resize existing partition. You can install Debian or Ubuntu to external storage devices on Mac.

I assume Intel64/AMD64 (x86_64) CPU machine as analysis environment. The other CPU machine can be used for analysis, but you need to solve problems by yourself. The 64 bits version of Debian or Ubuntu is also required because 32 bits version cannot use large memory.

0.1 Installation of Claident, Assams, databases, and the other required programs

Run the following commands in terminal or console as the user that can use `sudo`. Then, all of the required softwares will be installed. The installer will ask password to you when `sudo` is used.

```
> mkdir -p ~/workingdirectory ↓
> cd ~/workingdirectory ↓
> wget https://www.claident.org/installClaident_Debian.sh ↓
> sh installClaident_Debian.sh ↓
> wget https://www.claident.org/installOptions_Debian.sh ↓
> sh installOptions_Debian.sh ↓
> wget https://www.claident.org/installDB_Debian.sh ↓
> sh installDB_Debian.sh ↓
> wget https://www.claident.org/installUCHIMEDB_Debian.sh ↓
> sh installUCHIMEDB_Debian.sh ↓
> cd .. ↓
> rm -r workingdirectory ↓
```

By default, the softwares will be installed to `/usr/local`. In the installation, you will see `Permission denied` error and the installer ask password to you. If the installer continue after password input, you don't need to care about the error. The installer try to install without `sudo` at first and the installation output the above error. Then, the installer try to install using `sudo`.

If you need proxy to connect the internet, execute the following commands to set environment variables before execution of the installer.

```
> export http_proxy=http://server.address:portnumber ↓
> export https_proxy=$http_proxy ↓
> export ftp_proxy=$http_proxy ↓
> export all_proxy=$http_proxy ↓
```

If the proxy requires username and password, execute the following commands instead of the above commands.

```
> export http_proxy=http://username:password@server.address:portnumber ↓
> export https_proxy=$http_proxy ↓
> export ftp_proxy=$http_proxy ↓
> export all_proxy=$http_proxy ↓
```

0.1.1 Upgrading to new version

If you want to upgrade all of the softwares and the databases, run the same commands as initial installation. By this procedure, Assams, Claident, PEAR, VSEARCH, Metaxa and ITSx will be installed to `/usr/local`, and NCBI BLAST+, BLAST databases for molecular identification, taxonomy databases and the other required programs will be installed to `/usr/local/share/claident`. NCBI BLAST+ and BLAST databases used by Claident can co-exist system wide installation of NCBI BLAST+ and BLAST databases.

You can disable a part of upgrade like below.

```
> mkdir -p ~/workingdirectory ↓
> cd ~/workingdirectory ↓
# disable upgrade of Assams
> touch .assams ↓
```

```
# disable upgrade of Claident
> touch .claident ↓
# disable upgrade of PEAR
> touch .pear ↓
# disable upgrade of VSEARCH
> touch .vsearch ↓
# disable upgrade of NCBI BLAST+
> touch .blast ↓
# execute upgrade
> wget https://www.claident.org/installClaident_Debian.sh ↓
> sh installClaident_Debian.sh ↓
# disable upgrade of sff_extract
> touch .sffextract ↓
# disable upgrade of HMMer
> touch .hmmer ↓
# disable upgrade of MAFFT
> touch .mafft ↓
# disable upgrade of Metaxa
> touch .metaxa ↓
# disable upgrade of ITSx
> touch .itsx ↓
# execute upgrade
> wget https://www.claident.org/installOptions_Debian.sh ↓
> sh installOptions_Debian.sh ↓
# disable upgrade of ‘‘overall’’ BLAST and taxonomy databases
> touch .overall ↓
# execute upgrade
> wget https://www.claident.org/installDB_Debian.sh ↓
> sh installDB_Debian.sh ↓
# disable upgrade of ‘‘Claident Databases for UCHIME’’
> touch .cdu ↓
# disable upgrade of ‘‘rdp’’ reference database for chimera detection
> touch .rdp ↓
# disable upgrade of ‘‘silva’’ reference databases for chimera detection
> touch .silva ↓
# disable upgrade of ‘‘unite’’ reference databases for chimera detection
> touch .unite ↓
# execute upgrade
> wget https://www.claident.org/installUCHIMEDB_Debian.sh ↓
> sh installUCHIMEDB_Debian.sh ↓
> cd .. ↓
> rm -r workingdirectory ↓
```

0.1.2 Installing to non-default path

If you install the softwares based on the above procedure, the softwares will be installed to `/usr/local`. The executable commands will be installed to `/usr/local/bin`. You can change these install path for coexistence with the other programs such as older versions like below.

```
> mkdir -p ~/workingdirectory ↓
> cd ~/workingdirectory ↓
> export PREFIX=install_path ↓
> wget https://www.claident.org/installClaident_Debian.sh ↓
```

```
> sh installClaident_Debian.sh ↓
> wget https://www.claident.org/installOptions_Debian.sh ↓
> sh installOptions_Debian.sh ↓
> wget https://www.claident.org/installDB_Debian.sh ↓
> sh installDB_Debian.sh ↓
> wget https://www.claident.org/installUCHIMEDB_Debian.sh ↓
> sh installUCHIMEDB_Debian.sh ↓
> cd .. ↓
> rm -r workingdirectory ↓
```

In this case, the following commands need to be executed before analysis.

```
> export PATH=install_path/bin:$PATH ↓
```

You can omit above command if the above command is added to `~/.bash_profile` or `~/.bashrc`.

0.1.3 How to install multiple versions in a computer

If you install Claident and the other softwares to default install path of a computer to which Claident was already installed, all softwares will be overwritten. As noted above, multiple versions of Claident can coexist if you install Claident to non-default path. Note that a configuration file `.claident` placed at a home directory of login user (`/home/username`) or `/etc/claident` cannot coexist at the same path. You need to replace this file before changing the version of Claident. The configuration file at the home directory of login user will be used preferentially. To use multiple version, I recommend to make user account for each version and to install Claident to the home directory of each user. Then, the version of Claident can be switched by switching login user.

Chapter 1

Sequencing of multiple samples by next-generation sequencers

In this chaper, I explain brief overview of tagged multiplex sequencing method by Roche GS series sequencers, Ion PGM and Illumina MiSeq. These sequencers can read over-400bp contiguously and are suitable for metabarcoding and DNA barcoding. Note that MiSeq requires concatenation of paired-end reads. Therefore, PCR amplicons should be 500bp or shorter (400bp is recommended) in order to concatenate paired-end reads. Forward and reverse reads can be analyzed separetely, but I cannot recommend such analysis because reverse reads are usually low quality.

The next-generation sequencers output extremely large amount of nucleotide sequences in single run. Running costs of single run is much higher than Sanger method-based sequencers. To use such sequencers efficiently, multiplex sequencing method was developed. Multiplex identifier tag sequences are added to target sequences to identify the sample of origin, and the multiple tagged samples are mixed and sequenced in single run in this method. This method can extremely reduce per-sample sequencing costs. Multiplex identifier tag is also called as “barcode”. However, nucleotide sequence for DNA barcoding is called as “barcode sequence”. This is very confusing and “multiplex identifier tag” is too long. Thus, I call multiplex identifier tag sequence as just “tag” in this text. Please notice that tag is often called as “index”.

In the following analysis, chimera sequences constructed in PCR and erroneous sequences potentially causes misinterpretation of analysis results. If multiple PCR replicates are prepared, tagged and sequenced separately, shared sequences among all replicates can be considered as nonchimeric and less erroneous. This is because there are huge number of sequence combinations and joint points but no error sequence pattern is only one for one true sequence and nonchimeric and no error sequences likely to be observed at all replicates. Program cannot remove chimeras and errors enough but we can expect that the combination of PCR replicates and program improves removal efficiency of chimeras and errors. After removal of chimeras and errors, the number of sequences of PCR replicates can be summed up and used in subsequent analysis.

1.1 PCR using tag- and adapter-jointed-primers

In order to add tag to amplicon, PCR using tag-jointed primer is the easiest way. This method requires a set of tag-jointed primers. In addition, library preparation kits for next-generation sequencers usually presume that the adapter sequences specified by manufacturers are added to the both end of target sequences. Thus, the following tag- and adapter-jointed primer is used for PCR.

```
5' — [adapter] — [tag] — [specific primer] — 3'
```

If this kind of primers are used for the both forward and reverse primers, the following amplicon sequences will be constructed.

```
5' — [adapter-F] — [tag-F] — [specific primer-F] — [target sequence] — [specific primer-R (reverse
complement)] — [tag-R (reverse complement)] — [adapter-R (reverse complement)] — 3'
```

In the case of single-end read, tag-F leads specific primer-F and target sequence in the sequence data.

The supplement of Hamady *et al.* (2008) may be useful for picking tag sequences. In the case of single-end sequencing, 3'-side tag is not required, and tagless primer can be used for PCR. In the case of paired-end sequencing, single index (tag) can be applied, but dual index (tag) is recommended for detecting unlikely tag combinations which means that forward and reverse sequences are mispaired.

Using above primer sets for PCR, primers anneal to templates in "Y"-formation, and the amplicon sequences which have tags and adapters for both ends will be constructed. Then, the amplicon solutions are mixed in the same concentration and sequenced based on manufacturer's protocol. Spectrophotometer (including Nanodrop) is inappropreate for the measurement of the concentration of solution because measurement of dsDNA using spectrophotometer is likely to be affected by the other contaminants. I recommend Qubit (ThermoFisher) for measurement of dsDNA concentration. Quantitative PCR-based method can also be recommended but it's expensive and more time-consuming.

Primer annealing position sequence can also be used for recognizing the sample of origin. Therefore, the sequences of multiple loci, for example plant *rbcL* and *matK*, from same sample set tagged by same tag set can be multiplexed and sequenced. Of course, the sequences of multiple loci can also be recognized by themselves. Smaller number of cycles and longer extension time were recommended for PCR. Because the required amount of DNA for sequence sample preparation is not so high, the larger number of cycles of PCR amplification is not needed. The larger number of cycles and shorter extension time generates more incompletely extended amplicon sequences and the incompletely extended amplicon sequences are re-extend using different template sequences in next cycle. Such sequences are called as "chimeric DNA". Chimeric DNAs causes a discovery of non-existent novel species or a overestimation of species diversity. To reduce chimeric DNA construction, using high-fidelity DNA polymerase such as Phusion (Finnzymes) or KOD (TOYOBO) is effective. Stevens *et al.* (2013) reported that slowing cooling-down from denaturation temperature to annealing temperature reduced chimeric DNA construction. If your thermal cycler can change cooling speed, slowing cooling-down from denaturation temperature to annealing temperature can be recommended. Chimeric DNA sequences can also be eliminated by computer programs after sequencing. Because chimera removal by programs is incomplete and the nonchimeric

sequences shrink, we cannot do better than reduce chimeric DNA construction.

In the case of hardly amplifiable templates, using Ampdirect Plus (Shimadzu) for PCR buffer or crushing by homogenizer or beads before DNA extraction is recommended. Deep freezing before crushing can also be recommended. Removal of polyphenols or polysaccharides might be required if your sample contain those chemicals. If PCR amplification using tag- and adapter-jointed-primers fail, try two-step PCR that consist of primary PCR (20–30 cycles) using primers without tags and adapters, purification of amplicons by ExoSAP-IT, and secondary PCR (5–10cycles) using amplicons of primary PCR as templates and tag- and adapter-jointed-primers.

1.1.1 Decreasing costs by interim adapters

Tag- and adaper-jointed-primers are very long and expensive. In addition, we need to buy tag- and adaper-jointed-primers for each locus. To reduce cost of tag- and adaper-jointed-primers, interim adapter-jointed primers and two-step PCR is useful. The following primer set is used in primary PCR.

```
5' — [interim adapter] — [specific primer] — 3'
```

This PCR product have interim adapter sequences at the both ends. This PCR product is used as template in secondary PCR after purification. The following primer set is used in secondary PCR.

```
5' — [adapter specified by manufacturer] — [tag] — [interim adapter] — 3'
```

This two-step PCR enables us to reuse secondary PCR primers. However, this two-step PCR may increase PCR errors and PCR amplification biases, and decrease target sequence lengths. Note that final PCR product is constructed as the following style.

```
5' — [adapter-F specified by manufacturer] — [tag-F] — [interim adapter-F] — [specific primer-F] — [target
sequence] — [specific primer-R (reverse complement)] — [interim adapter-R (reverse complement)] — [tag-R
(reverse complement)] — [adapter-R (reverse complement) specified by manufacturer] — 3'
```

Illumina's multiplex sequencing method (Illumina corporation, 2013) using Nextera XT Index Kit is same as the above method. In the dual-index paired-end sequencing based on this method, the first read start from behind of interim adapter-F (i.e. head of specific primer-F) to target sequence. The second read start from behind of interim adapter-R and contains tag-R (index1) sequence. The third read start from behind of adapter-F and contains tag-F (index2) sequence. The last read start from ahead of interim adapter-R (i.e. tail of specific primer-R) to target sequence. The first, second, third and last reads are saved as `*_R1_*.fastq.gz`, `*_R2_*.fastq.gz`, `*_R3_*.fastq.gz` and `*_R4_*.fastq.gz`, respectively. The first, second and third reads are same strand, but last read is reverse strand. Because the sequencing primers for the first and the last reads are targeting interim adapter-F and interim adapter-R, respectively, the first and the last reads contains the sequences of specific primer-F and specific primer-R, respectively. Thus, the target sequences contained in the first and the last reads are shrinked. If the length of the target sequence is 500 bp or longer, there might be no overlap and paired-end reads cannot be concatenate. If specific primer-F and specific primer-R are used as sequencing primers for the first and the last reads, you can exclude the sequences of specific primer-F and specific primer-R from the first and the last reads. However, the following quality improvement method by insertion of N cannot be applied in such case.

1.1.2 Quality improvement by insertion of N

On the Illumina platform, luminescence of syntheses of DNA on a flowcell is detected by optical sensor. PCR amplicons of metagenomes are single locus and much more homogeneous than genome shotgun or RNA-seq library sequences. In such case, neighboring sequences on a flowcell is difficult to distinguish one from the other. In addition, if the nucleotide of the most sequences (especially first 12 nucleotides) are the same and nonluminescence, the Illumina platform sequencer will determined as failure and crash. To avoid this problem, insertion of `NNNNNN` between specific primer and interim adapter is effective. `NNNNNN` of the head of sequences enables sequencers to distinguish neighboring sequences and prevent black out, and the sequencing quality therefore will be improved (Nelson *et al.*, 2014). The varied length of `NNNNNN` causes artificial frameshift and also effective (Fadrosh *et al.*, 2014). PhiX control can be reduced by using the above methods, and the application sequences will increase.

Chapter 2

Preprocessing of nucleotide sequence data

Roche GS series sequencers and Ion PGM output raw sequencing data as `*.sff`. Illumina platform sequencers output `*.fastq` files. In this chapter, the procedures of demultiplexing, quality-trimming and quality-filtering. The `clsplitseq` command of Claident is recommended for demultiplexing because the programs provided by manufacturer ignores the quality of tag positions. The following commands should be executed in the terminal or console. Fundamental knowledge of terminal operations is required. If you are unfamiliar with terminal operations, you need to become understandable about the contents of appendix B.

2.1 Importing sequence data deposited to SRA/DRA/ERA or demultiplexed FASTQ

Claident assumes `SequenceID__RunID__TagID__PrimerID` for definition lines of sequences, and `RunID__TagID__PrimerID` for file names (without extension). Therefore, the sequence data deposited to SRA/DRA/ERA or demultiplexed FASTQ cannot be used as is. The `climportfastq` of Claident can convert such data. If your data is paired-end, you need to concatenate and filter the sequences before conversion (see section 2.3.3). The following plain text file is required for conversion.

```
| SequenceFileName1 RunID__TagID__PrimerID
| SequenceFileName2 RunID__TagID__PrimerID
| SequenceFileName3 RunID__TagID__PrimerID
```

Dummy RunID and PrimerID is acceptable. PrimerID need to be the same among the sample used the same primer set. TagID need to be different among the different sample files. TagID can be the same as the sequence file name.

After the above file was prepared, execute `climportfastq` like the following and the above file should be given as an input file.

```
> climportfastq \
--numthreads=NumberOfCPUs \
inputfile \
outputfolder ↓
```

Then, you can find converted files in the output folder. If your sequence data is single-end, quality filtering explained in section 2.2.3 is recommended.

2.2 For Roche GS series sequencers and Ion PGM

2.2.1 Converting SFF to FASTQ

First of all, conversion of raw SFF format file to FASTQ file is needed like the following.

```
> sff_extract -c inputfile(SFF) ↓
```

`-c` argument enables trimming of `TCAG` at the head of sequences. If you add `TCAG` to the head of tag sequences, do not use this argument. Assuming your SFF file name is `HOGEHOGE.sff`, `HOGEHOGE.fastq` will be saved as FASTQ file. `HOGEHOGE.xml` will also be generated, but this is not required. The output sequences have tag sequences at the beginning, followed by primer-F and target sequences, and primer-R (reverse complement) at the end. Note that all sequences are not completely read from the beginning to the end, the incomplete sequences are included. The `sff_extract` command is used in this book, but any other programs which can clip `TCAG` at the beginning can be used. If the SFF to FASTQ converter program cannot clip `TCAG` at the beginning, adding `TCAG` to the beginning of tag sequences to give to `clsplitseq` also works well, but the quality values will be strictly checked.

2.2.2 Demultiplexing of sequences

The FASTQ file that contain the sequences from multiple samples need to be demultiplexed based on tag sequences and primer sequences before the subsequent analysis. To do this process, a FASTA file which contain tag sequences and another FASTA file which contain primer-F sequences are required.

```
| >TagID
| [tag sequence]
| >examplesample1
| ACGTACGT
```

```
| >PrimerID
| [primer sequence]
| >exampleprimer1
| ACGTACGTACGTACGTACGT
```

Degenerate codes of nucleotides are not allowed for tag sequences, but those are allowed for primer sequences. Both of tag and primer FASTQ files can contain multiple sequences. If you use interim adapter explained in section 1.1.1, primer sequences should be written like the following.

```
| >PrimerID
| [interim adapter][primer sequence]
| >exampleprimer1
| TGATACTCGATACGTACGTACGTACGTACGTACGT
```

Thus, the sequences between tag and target sequences should be written in primer FASTA file.

All the above files are prepared, the following command can demultiplex nucleotide sequences to each sample FASTQ file.

```
> clsplitseq \
--runname=RunID \
--tagfile=TagSequenceFile \
--primerfile=PrimerSequenceFile \
--minqualtag=27 \
--numthreads=NumberOfCPUs \
inputfile \
outputfolder ↓
```

RunID must differ among different sequencing runs. RunID is given by sequencer in many cases, you can use such sequencer generated RunID. RunID is usually contained in sequence file name or sequence name in sequence file, but the naming rules are different among sequencing platforms. Therefore, `clsplitseq` requires RunID given by user. `--minqualtag` is an argument that specifies minimum quality threshold of tag position sequences. If 1 or more lower quality nucleotide than this threshold value is contained by a sequence, such sequence will be omitted from output sequences. 27 for minimum quality threshold is proposed by Kunin *et al.* (2010) for 3'-tail trimming of the sequences of Roche GS series sequencers. The different value might be more suitable for the other sequencers. In many cases, 30 is used for minimum quality threshold and can be recommended.

If multiplex sequencing technique is not used, `--tagfile` argument can be omitted. However, just omit of `--tagfile` generates incompatible FASTQ files for Claident. In such case, you should add identifier (dummy is acceptable) of tag sequences using `--indexname=TagID` argument.

The tag and primer position sequences are trimmed from the output sequences. Tag position sequence match is evaluated exactly and strictly. There are no arguments to tolerate a mismatch. Primer position sequence is aligned based on Needleman-Wunsch algorithm and evaluated allowing 14% of mismatches (the threshold can be changed). The output files are named as `RunID__TagID__PrimerID.fastq.gz` and saved in the output folder. `clsplitseq` can use multiple CPUs for faster processing. If your computer have 4 CPU cores, 4 should be speficied for `--numthreads` argument. Note that operating system and/or writing speed of storage devices might limit processing speed. By default, the output files are compressed by GZIP. Therefore, decompression is required to read/write by incompatible programs with gzipped FASTQ files. The commands of Claident used below can treat gzipped FASTQ files.

Before submission of manuscripts, sequence data need to be deposited to public database such as DDBJ Sequence Read Archive (DRA). Gzipped FASTQ files in this step can be used for the data deposition.

If you sequenced a number of samples by multiple sequencing runs

Multiple demultiplexing by `clsplitseq` are required. However, `clsplitseq` cannot write already existing folder by default. The secondary run of `clsplitseq` requires `--append` argument like below.

```
> clsplitseq \
--runname=RunID \
--tagfile=TagSequenceFile \
--primerfile=PrimerSequenceFile \
--minqualtag=27 \
--numthreads=NumberOfCPUs \
inputfile1 \
outputfolder ↓
> clsplitseq \
--runname=RunID \
--tagfile=TagSequenceFile \
--primerfile=PrimerSequenceFile \
--minqualtag=27 \
--numthreads=NumberOfCPUs \
--append \
inputfile2 \
outputfolder ↓
```

If your tag sequence lengths are unequal

`clsplitseq` assumes that all tag sequence lengths are equal for faster processing. The unequal length tags must be splitted to multiple tag sequence files and multiple demultiplexing runs of `clcplitseq` are required as the following.

```
> clsplitseq \
--runname=RunID \
--tagfile=TagSequenceFile1 \
--primerfile=PrimerSequenceFile \
--minqualtag=27 \
--numthreads=NumberOfCPUs \
inputfile \
outputfolder ↓
> clsplitseq \
--runname=RunID \
--tagfile=TagSequenceFile2 \
--primerfile=PrimerSequenceFile \
--minqualtag=27 \
--numthreads=NumberOfCPUs \
--append \
inputfile \
outputfolder ↓
```

Recognition and elimination of reverse primer positions

In the above procedure, reverse primer position and subsequent sequences are not eliminated. Reverse primer position and subsequent sequences are artificial and should be eliminated if possible. To do so, reverse primer sequence file like the following is required.

```
| >PrimerID
| [primer sequence]
| >exampleprimer1
| TCAGTCAGTCAGTCAGTCAG
```

Multiple reverse primers can written in this file. Note that the N-th reverse primer sequence is assumed to associate with the N-th forward primer sequence. Therefore, the different number of primer sequences between forward and reverse primer sequence files causes an error. If there are the samples whose forward or reverse primer sequence is same but the other primer sequence is different, both combinations of forward and reverse primer sequences need to be written as different primers in the files.

After the preparation of the above file, perform `clsplitseq` as the following.

```
> clsplitseq \
--runname=RunID \
--tagfile=TagSequenceFile \
--primerfile=ForwardPrimerSequenceFile \
--reverseprimerfile=ReversePrimerSequenceFile \
--reversecomplement \
--minqualtag=27 \
--numthreads=NumberOfCPUs \
inputfile \
outputfolder ↓
```

In this processing, reverse-complement sequence of reverse primer is searched based on Needleman-Wunsch algorithm allowing 15% (this value can be changed) of mismatches and reverse primer position and subsequent sequence is eliminated in addition to the above process. If reverse-complement sequence of reverse primer is not found and the other requirement is fullfilled, the sequence will be saved to output file by default. The `--needreverseprimer` argument is required to filter out the sequence which does not contain reverse-complement sequence of reverse primer.

2.2.3 Trimming low quality tail and filtering low quality sequences

FASTQ sequences have read quality information. The low quality 3'-tail can be trimmed and the low quality sequences can be filtered out based on the quality values. The `clfilterseq` command can perform such processing as the following.

```
> clfilterseq \
--minqual=27 \
--minquallen=3 \
--minlen=350 \
--maxlen=400 \
--maxplowqual=0.1 \
--numthreads=NumberOfCPUs \
inputfile \
outputfile ↓
```

The values of `--minqual` and `--minquallen` indicate the minimum threshold of read quality value and size of sliding window, respectively. The above command trims 3'-tail positions until 3 bp long sequence whose read quality is 27 or higher in all 3 positions are observed. In addition, trimmed sequences shorter than `--minlen` will be filtered out and trimmed sequences longer than `--maxlen` will be trimmed to `--maxlen`. The remaining sequences containing `--maxplowqual` or more rate of lower quality positions than `--minqual` will also be filtered out. The output is a file by default, but can be saved to the file in the new folder using `--output=folder` argument. The output file name is same as the input file name in this case. If you want to save the output files to the existing folder, add `--append` argument.

If you want to apply `clfilterseq` to the all files in the output folder of `clsplitseq`, run the following command.

```
> for f in OutputFolderOfclsplitseq/*.fastq.gz ↓
do clfilterseq \
--output=folder \
--append \
--minqual=27 \
--minquallen=3 \
--minlen=350 \
--maxlen=400 \
--maxplowqual=0.1 \
--numthreads=NumberOfCPUs \
$f \
outputfolder ↓
done ↓
```

2.3 For Illumina platform sequences

2.3.1 Converting from BCL to FASTQ

The analysis software of Illumina platform sequences can demultiplex sequencing reads, but ignores read quality of tag positions. Therefore, the sequences have low quality tag positions possibly saved to demultiplexed FASTQ. To filering out such sequences, pre-demultiplexed FASTQ files are required and can be converted from BCL files with the aid of bcl2fastq. There are 1.x and 2.x series of bcl2fastq and both series can be used for Claident. However, the sequencers may be compatible to either 1.x or 2.x, you need to select proper version. Pre-demultiplexed FASTQ can be demultiplexed by `clsplitseq` in Claident. See appendix to install bcl2fastq.

To convert BCL to FASTQ, run data folder (superjacent folder of BaseCalls folder) need to be copied to the PC

installed bcl2fastq. If there is `SampleSheet.csv` in run data folder, this file must be renamed or deleted.

In the case of bcl2fastq 1.x, the following commands make FASTQ files from BCL files of 8 bp dual indexed 300PE sequencing data.

```
> cd RunDataFolder ↓
> configureBclToFastq.pl \
--fastq-cluster-count 0 \
--use-bases-mask Y300n,Y8,Y8,Y300n \
--input-dir BaseCalls \
--output-dir outputfolder ↓
> make -j4 ↓
```

The `--fastq-cluster-count 0` argument disable large output file splitting. The `--use-bases-mask Y300n,Y8,Y8,Y300n` is an argument to save forward 300 bp read (last base is trimmed), 8 bp index 1 (reverse-complement of tag-R), 8 bp index 2 (tag-F) and reverse 300 bp read (last base is trimmed) to `*_R1_001.fastq.gz`, `*_R2_001.fastq.gz`, `*_R3_001.fastq.gz` and `*_R4_001.fastq.gz`, respectively. The value of `--use-bases-mask` argument need to be changed for the other sequencing settings. For 6 bp single indexed 250SE and 8 bp dual indexed 300SE sequencing data, `--use-bases-mask Y250n,Y6` and `--use-bases-mask Y300n,Y8,Y8` should be suitable, respectively. `make -j4` executes the conversion using 4 CPUs. The output files will be compressed by GZIP. The extension `.gz` of output files indicates that the file is compressed by GZIP. Claident is compliant with gzipped FASTQ files and decompression is not required.

In the case of bcl2fastq 2.x, perform the following command.

```
> bcl2fastq \
--processing-threads NumberOfCPUs \
--use-bases-mask Y300n,Y8,Y8,Y300n \
--runfolder-dir RunDataFolder \
--output-dir outputfolder ↓
```

The `--processing-threads`, `--use-bases-mask` and `--runfolder-dir` indicate the number of processor used in conversion, masking option in common with 1.x and run data folder, respectively.

2.3.2 Demultiplexing of sequences

FASTA files containing tag (index) sequences and primer sequences like the following are needed for demultiplexing. FASTA files containing secondary tag (index) sequences and reverse primer sequences are also required for paired-end sequencing data.

```
| >TagID
| [tag sequence]
| >examplesample1
| ACGTACGT
```

```
| >PrimerID
| [primer sequence]
| >exampleprimer1
| ACGTACGTACGTACGTACGT
```

Degenerate code is not allowed for tag sequences, but can be used in primer sequences. Multiple tags and primers can be written in the files, but the N-th reverse tag/primer sequence is assumed to associate with the N-th forward tag/primer sequence. Therefore, the different number of tag/primer sequences between forward and reverse tag/primer sequence files causes an error. If there are the samples whose forward or reverse tag/primer sequence is same but the other tag/primer sequence is different, both combinations of forward and reverse tag/primer sequences need to be written as different tags/primers in the files. If you added `N` in front of primer, `N` need to be added in primer sequence. If your `N` length is unequal, only the longest `N` should be written in the file.

All the required files prepared, the following command demultiplex sequences to each sample file.

```
> clsplitseq \
--runname=RunID \
--index1file=Index1Sequence(tag-Rrevcomp)File \
--index2file=Index2Sequence(tag-F)File \
--primerfile=ForwardPrimerSequenceFile \
--reverseprimerfile=ReversePrimerSequenceFile \
--minqualtag=30 \
--numthreads=NumberofCPUs \
inputfile1 \
inputfile2 \
inputfile3 \
inputfile4 \
outputfolder ↓
```

The input files should be specified in the order of `*_R1_001.fastq.gz`, `*_R2_001.fastq.gz`, `*_R3_001.fastq.gz` and `*_R4_001.fastq.gz`. The `--index1file` and `--index2file` arguments requires the FASTA sequence files of index 1 (reverse-complement of tag-R) and index 2 (tag-F), respectively. By default, the acceptable mismatches are 14% and 15% for forward and reverse primers, respectively. If you added `N` in front of primer, the `--truncateN=enable` argument need to be given. This argument enables exclusion of `N` of primer and matched positions of sequences in calculation of the rate of mismatches. Therefore, only the longest `N` is required to find `N`-added primer even if the length of `N` is unequal. After the processing, the number of sequences in demultiplexed files should be compared with those in demultiplexed files generated by Illumina softwares. Correctly demultiplexed files should contain fewer sequences than demultiplexed files generated by Illumina softwares. If you used specific primers for sequencing primers, forward and reverse sequences do not contain specific primer positions. In such cases, `--primerfile` and `--reverseprimerfile` arguments are not required, but `--primername=PrimerID` argument need to be given for converting sequence names as compliant with Claident. Dummy PrimerID is acceptable but no PrimerID is not.

If you do not perform multiplex sequencing using tag/index, `--index1file` and `--index2file` arguments are unneeded, but `--indexname=TagID` argument must be given for converting sequence names as compliant with Claident. Dummy TagID is acceptable but no TagID is not.

After demultiplexing, `RunID__TagID__PrimerID.forward.fastq.gz` and `RunID__TagID__PrimerID.reverse.fastq.gz` will be generated. These gzipped FASTQ files can be used for data deposition to sequence read archive sites such as DDBJ Sequence Read Archive (DRA). In deposition process to DRA, it is required that the sequence lengths are equal or not. Because primer position sequences that can be unequal lengths even if only one primer set was used are eliminated from demultiplexed sequence files, do not specify that the sequence lengths are equal.

2.3.3 Concatenating forward and reverse sequences

In the case of overlapped paired-end

The `clconcatpair` command in Claident can be used for concatenating overlapped paired-end sequence data. The `clconcatpair` concatenate forward and reverse sequences based on overlap positions using VSEARCH by the following command.

```
> clconcatpair \
--mode=OVL \
--numthreads=NumberOfCPUs \
inputfolder \
outputfolder ↓
```

This command finds `*.forward.fastq` and `*.reverse.fastq` in inputfolder, and concatenate the pairs automatically. Gzipped `.gz` and/or bzip2ed `.bz2` files are also be found and concatenated. Concatenated sequence files will be generated as `*.fastq.gz` in outputfolder.

If input file names are not compliant with `*.forward.fastq` and `*.reverse.fastq`, the following command can be used for concatenating a pair of files.

```
> clconcatpair \
--mode=OVL \
--numthreads=NumberOfCPUs \
inputfile1 \
inputfile2 \
outputfile ↓
```

The forward and reverse sequence FASTQ files should be given as inputfile1 and inputfile2, respectively. Addition of `.gz` or `.bz2` is required for output file compression.

In the case of non-overlapped paired-end

If there are no overlaps between forward and reverse sequences, quality-trimming and quality-filtering using `clfilterseq` like the following should be performed at first.

```
> clfilterseq \
--minqual=30 \
--minquallen=3 \
--minlen=100 \
--maxplowqual=0.1 \
--numthreads=NumberOfCPUs \
inputfile1 \
inputfile2 \
outputfolder ↓
```

The values of `--minqual` and `--minquallen` indicate the minimum threshold of read quality value and size of sliding window, respectively. The above command trims 3'-tail positions until 3 bp long sequence whose read quality is 30 or higher in all 3 positions are observed. In addition, trimmed sequences shorter than `--minlen` will be filtered out. The remaining sequences containing `--maxplowqual` or more rate of lower quality positions than `--minqual` will also be filtered out. In this process, filtering out one of the sequence of a pair, the other sequence of the pair will also be filtered out. The output will be generated as the same name files in outputfolder. If you want to output to existing folder, you need to add `--append` argument. To apply the above command to all the pairs of `*.forward.fastq` and `*.reverse.fastq` in the current folder, execute the following commands.

```
> for f in `ls *.forward.fastq.gz | grep -P -o '^[^\.]+'`↓
do clfilterseq \
--minqual=30 \
--minquallen=3 \
--minlen=100 \
--maxplowqual=0.1 \
--numthreads=NumberOfCPUs \
$f.forward.fastq.gz \
$f.reverse.fastq.gz \
outputfolder ↓
done ↓
```

After the quality-trimming and quality-filtering like above, perform sequence concatenation with the aid of `clconcatpair` like below.

```
> clconcatpair \
--mode=NON \
--numthreads=NumberOfCPUs \
inputfolder \
outputfolder ↓
```

In this process, the forward and reverse sequences like the following are assumed as input.

```
5' — forward sequence — 3'
5' — reverse sequence — 3'
```

The `clconcatpair --mode=NON` command will concatenate these sequence pairs and make sequences like the following.

```
5' — reverse sequence (reverse-complement) — ACGTACGTACGTACGT — forward sequence — 3'
```

Conduct removal of noisy and/or chimeric sequences in the same way as concatenated overlapped paired-end

sequence data. In the sequence clustering by `clclassseqv` and the raw reads mapping to centroid sequences by `clrecoverseqv`, add `--paddinglen=16` argument. The concatenated sequences like above causes overvaluation of sequence similarity because of artificial padding sequence `ACGTACGTACGTACGT`. The `--paddinglen=16` argument will offset such overvaluation by exclusion of `ACGTACGTACGTACGT` from sequence similarity calculation and cluster concatenated sequences based on correct sequence similarity.

In the estimation of host organism, split concatenated sequences based on `ACGTACGTACGTACGT`, assign taxonomy to forward and reverse sequences separately. Then, merge 2 taxonomy (see section 7.3). Generally speaking, forward sequences shows higher quality than reverse sequences, prefering forward sequence taxonomy is recommended if there is no *a priori* infomations about identification power and variability of forward and reverse sequences. Sequence division based on `ACGTACGTACGTACGT` can be applied to the sequences by the following command.

```
>cldivseq \
--query=ACGTACGTACGTACGT \
inputfile \
outputfile1 \
outputfile2
```

The outputfile1 and outputfile2 contain reverse-complement of reverse sequences and forward sequences, respectively.

Concatenating overlapping paired-end sequences using PEAR

PEAR (Zhang *et al.*, 2014) can also be used for concatenation of overlapped paired-end sequences. The following command will concatenate the pairs of sequences.

```
> pear \
-p 0.0001 \
-u 0 \
-j NumberOfCPUs \
-f RunID__TagID__PrimerID.forward.fastq.gz \
-r RunID__TagID__PrimerID.reverse.fastq.gz \
-o RunID__TagID__PrimerID ↓
```

If the processes correctly finished, the following files will be generated.

RunID__TagID__PrimerID.assembled.fastq Concatenated sequences.
RunID__TagID__PrimerID.unassembled.forward.fastq Unconcatenated forward sequences.
RunID__TagID__PrimerID.unassembled.reverse.fastq Unconcatenated reverse sequences.
RunID__TagID__PrimerID.discarded.fastq Discarded sequences by statistical test.

Only `RunID__TagID__PrimerID.assembled.fastq` is required in subsequent procedures. These output files are not compressed and consume large amount of storages, compression by GZIP or BZIP2 is recommended.

To apply concatenation to all the pairs of forward and reverse sequence files by PEAR, execute the following command.

```
> for f in `ls *.forward.fastq.gz | grep -P -o '^[^\.]+'`↓
do pear \
-p 0.0001 \
-u 0 \
-j NumberOfCPUs \
-f $f.forward.fastq.gz \
-r $f.reverse.fastq.gz \
-o $f ↓
done ↓
```

Concatenating overlapping paired-end sequences using VSEARCH

VSEARCH can also be used directly for concatenation of overlapped paired-end sequences. The following command will concatenate the pairs of sequences.

```
> vsearch \
--threads NumberOfCPUs \
--fastq_mergepairs RunID_TagID_PrimerID.forward.fastq.gz \
--reverse RunID_TagID_PrimerID.reverse.fastq.gz \
--fastq_allowmergestagger \
--fastqout RunID_TagID_PrimerID.assembled.fastq ↓
```

If the amplicon sequences are shorter than read length, read tail of forward sequence possibly exceeds read head of reverse sequence or read tail of reverse sequence possibly exceeds read head of forward sequence. VSEARCH does not concatenate such sequences by default. The `--fastq_allowmergestagger` argument enables such sequence concatenation. The overhang positions which exceeded read head of the other sequence will be eliminated because such positions are artificial. The `--fastq_allowmergestagger` argument is not required if there is no such sequences. Using PEAR, the same processing as VSEARCH with `--fastq_allowmergestagger` will be performed. The unconcatenated forward and reverse sequences can be obtained by `--fastqout_notmerged_fwd outputfile` and `--fastqout_notmerged_rev outputfile` arguments, respectively. The minimum overlap length, the minimum length of concatenated sequence, the maximum length of concatenated sequence, the maximum number of allowed mismatches and the maximum allowed expected errors in concatenated sequence can be specified by `--fastq_minovlen`, `--fastq_minmergelen`, `--fastq_maxmergelen`, `--fastq_maxdiffs` and `--fastq_maxee`, respectively. In the concatenation of overlapped paired-end sequences by `clconcatpair`, `--fastq_minovlen 20 --fastq_maxdiffs 20` is used by default, but `--fastq_minovlen 10 --fastq_maxdiffs 5` is used by default of VSEARCH.

To apply concatenation by VSEARCH to all the files in current folder, execute the following command.

```
> for f in `ls *.forward.fastq.gz | grep -P -o '^[^\.]+'`↓
do vsearch \
--threads NumberOfCPUs \
--fastq_mergepairs $f.forward.fastq.gz \
--reverse $f.reverse.fastq.gz \
--fastq_allowmergestagger \
--fastqout $f.assembled.fastq ↓
done ↓
```

2.3.4 Filtering potentially erroneous sequences

There are read quality values in the FASTQ files. Therefore, we can filter out potentially erroneous sequences using these quality values. To do so, the following command can conduct such processing.

```
> clfilterseq \
--minqual=30 \
--maxplowqual=0.1 \
--numthreads=NumberOfCPUs \
inputfile \
outputfile ↓
```

The sequences containing `--maxplowqual` or more rate of lower quality positions than `--minqual` will also be filtered out by the above command. The output is a file by default, but adding `--output=folder` argument changes to save output as the same name file in the outputfolder. If you want to save output file to existing folder, `--append` argument is needed. In the case of concatenated sequences of overlapped paired-end sequences generated by Illumina platform sequencers, positions close to the both end is usually high quality and overlapped positions is also high quality if the same positions of forward and reverse sequences are matched. Therefore, trimming low quality positions close to the both end is needless. Filtering out sequences containing low quality positions is recommended for concatenated overlapped paired-end sequences. The existing sequence filtering programs such as FastQC (Andrews, 2010) or PRINSEQ (Schmieder & Edwards, 2011) are also recommended.

To apply the same processing to the concatenated sequences by PEAR, execute the following command.

```
> for f in *.assembled.fastq ↓
do clfilterseq \
--output=folder \
--append \
--minqual=30 \
--maxplowqual=0.1 \
--numthreads=NumberOfCPUs \
$f \
outputfolder ↓
done ↓
```

Using quality values, we can calculate expected number of read errors. Quality-filtering based on the maximum allowed expected errors in input sequences can also be applied using VSEARCH. The following command can apply such quality-filtering to the input sequences.

```
> vsearch \
--threads NumberOfCPUs \
--fastq_filter inputfile \
--fastq_maxee 1.0 \
--fastqout outputfile ↓
```

For the single-end sequence data and unconcatenated sequence data, same quality-trimming and quality-filtering

as Roche GS series sequencers and Ion PGM can be recommended (see section 2.2.3). Note that thresholds for quality values and read lengths should be changed.

2.4 If you sequenced same PCR amplicons multiply or replicated PCR amplicons of same templates

Because chimeric sequences are constructed in each PCR tube, whether the sequences are came from same tube or not should be given to analysis programs. Therefore, several procedure is required to give such information to the programs in the cases of replicated sequencing of same PCR amplicons and sequencing of replicated PCR amplicons of same templates.

2.4.1 In the case of sequencing of replicated PCR amplicons of same templates using same tags in the same run

In this case, chimera removal based on replicates of PCR cannot be applied. This case can treat as same as unreplicated PCR.

2.4.2 In the case of sequencing of replicated PCR amplicons of same templates using different tags in the same run

In Claident, RunID__TagID__PrimerID is used as sample IDs. Therefore, there are multiple samples from the same templates in this case. It may be good idea that common sequences among replicated samples are treated as noiseless and nonchimeric, and uncommon sequences are treated as noisy and/or chimeric sequences if noise occurrence and chimera formation can be assumed as random. However, noise occurrence and chimera formation likely to be nonrandom. Noisy and/or chimeric sequences might be occurred across all replicates. The effectiveness of this method does not confirmed enough, and combination of noisy and/or chimeric sequence removal based on PCR replicates and algorithms can be recommended (see also Lange *et al.* (2015)). Such combination is supported in Claident, it is explained later. One of the final output of Claident is a sample x OTU table containing the number of sequences in every cell. The table modification command `clfiltersum` can be used for integration of multiple samples to one, and cell numbers will be summed up.

2.4.3 In the case of sequencing of same PCR amplicons using same tags in different runs

In Claident, RunID__TagID__PrimerID is used as sample IDs. Therefore, there are multiple samples from the same templates in this case. Such samples can be treated as separate or integrated to one sample. It is recommended that separate samples have been used subsequent analysis without any change, all filtered and denoised sequences of multiple runs are given to clustering commands, and such samples are finally integrated in a sample x OTU table. The table generation command `clsumclass` with `--runname` argument or table modification command

`clfiltersum` with `--runname` argument can be used to replace RunIDs and to integrate multiple samples to one, and cell numbers will be summed up.

If you want to integrate multiple samples from the same templates at this time, execute the following commands.

```
> clsplitseq \
snip* \
--runname=F00 \
inputfile1 \
outputfolder ↓
> clsplitseq \
snip* \
--runname=F00 \
--append \
inputfile2 \
outputfolder ↓
```

The inputfile1 and inputfile2 are the FASTQ files of first and second run, respectively. The `--runname` argument replaces RunID of the sequence names to `F00`. Thus, all the sequences of both input files will be saved to `F00__TagID__PrimerID.fastq.gz` in the output folder.

In Claident, the sequences whose names contain the same RunID, TagID and PrimerID are treated as the sequences from the same samples. If tag and primer sequence files are the same, TagID and PrimerID of output sequences are the same. Replacing RunID to `F00`, all of RunID, TagID and PrimerID become the same in the sequences from the same samples.

2.4.4 In the case of sequencing of replicated PCR amplicons of same templates using same tags in different runs

You should have multiple FASTQ files. Run demultiplexing by `clsplitseq` and quality-trimming and quality-filtering by `clfilterseq` separately, and save processed sequences to different folders. After separate processing of noisy and/or chimeric sequence removal, give all processed sequence files to clustering programs at once. One of the final output of Claident is a sample x OTU table containing the number of sequences in every cell. The table modification command `clfiltersum` can be used for integration of multiple samples to one, and cell numbers will be summed up.

2.4.5 In the case of sequencing of replicated PCR amplicons of same templates using different tags in different runs

In Claident, RunID__TagID__PrimerID is used as sample IDs. Therefore, there are multiple replicated samples from the same templates in this case. Such samples can be treated as separate or integrated to one sample. It is recommended that separate samples have been used subsequent analysis without any change and such samples are finally integrated in a sample x OTU table. It may be good idea that common sequences among replicated samples are treated as noiseless and nonchimeric, and uncommon sequences are treated as noisy

and/or chimeric sequences if noise occurrence and chimera formation can be assumed as random. Noisy and/or chimeric sequences might be occurred across all replicates. The effectiveness of this method does not confirmed enough, and combination of noisy and/or chimeric sequence removal based on PCR replicates and algorithms can be recommended (see also Lange *et al.* (2015)). Such combination is supported in Claident, it is explained later. One of the final output of Claident is a sample x OTU table containing the number of sequences in every cell. The table modification command `clfiltersum` can be used for integration of multiple samples to one, and cell numbers will be summed up.

In the demultiplexing using `clsplitseq`, process multiple FASTQ files like the following.

```
> clsplitseq \
snip* \
--tagfile=TagSequenceFile1 \
--primerfile=PrimerSequenceFile \
inputfile1 \
outputfolder ↓
> clsplitseq \
snip* \
--tagfile=TagSequenceFile2 \
--primerfile=PrimerSequenceFile \
--append \
inputfile2 \
outputfolder ↓
```

The sequence file of primer and the output folder should be the same. The different RunID should be given for `--runname` argument. The different tag sequence files should be prepared and given. Note that the same TagID should be specified for the replicated samples from the same templates even if tag/index sequence is different. In addition, different TagID should be specified for the samples from different templates even if tag/index sequence is the same. In the case of the following tag sequence files, the sequences of sample1 were added `ACGTACGT` as a tag in the first run, and `ATGCATGC` in the second run.

```
| >sample1
| ACGTACGT
| >sample2
| ATGCATGC
```

```
| >sample1
| ATGCATGC
| >sample2
| ACGTACGT
```

The sequences of sample2 were added `ATGCATGC` as a tag in the first run, and `ACGTACGT` in the second run. The subsequent analysis is explained later.

Chapter 3

Noisy and/or chimeric sequence removal

Claident can detect noisy sequences containing read errors and/or copy errors based on sequence abandance. This method is similar to the method implemented in CD-HIT-OTU (Li *et al.*, 2012). In the old pipeline using Assams for dereplication and clustering, chimera removal based on UCHIME (Edgar *et al.*, 2011) algorithm can be applied. Chimera removal based on UCHIME (Edgar *et al.*, 2011) algorithm is applied after OTU picking in the new pipeline using VSEARCH for dereplication and clustering.

Run the following command to perform noisy sequence detection and removal. Note that multiple input files can be given, and all sequence files of the same run should be given at once because sequencing quality varied among different runs. If there are too many sequences and the processing requires long time, give one sequence file and run the following command several times.

```
> clcleanseqv \
--derepmode=PREFIX \
--primarymaxnmismatch=0 \
--secondarymaxnmismatch=1 \
--pnoisycluster=0.5 \
--numthreads=NumberOfCPUs \
inputfile1 \
snip* \
inputfileN \
outputfolder ↓
```

Whether full-length perfect matching (`FULLLENGTH`) or prefix search (`PREFIX`) applied in dereplication is given for the `--derepmode` argument. `FULLLENGTH` is recommended for concatenated sequences of overlapped paired-end sequencing. `PREFIX` is recommended for single-end or concatenated sequences of non-overlapped paired-end sequencing. The `--primarymaxnmismatch` argument indicates the number of mismatches in primary clustering and 0 is recommended for the most cases. The `--secondarymaxnmismatch` argument describes the number of mismatches in secondary clustering and 1 is recommended for the most cases. For the noisy data, use `--primarymaxnmismatch=1 --secondarymaxnmismatch=3` or `--primarymaxnmismatch=2 --secondarymaxnmismatch=5`. Twice as `--primarymaxnmismatch` plus one should be specified for `--secondarymaxnmismatch`. The `--pnoisycluster` argument determines sensitivity of noise detection. Decimal value larger than 0 and smaller than 1 must be specified for this argument. The larger value acquire higher sensitivity. If you use 97% identity cutoff in clustering, 0.5 is recommended and this is the default value.

If you use 99% or larger identity cutoff in clustering, 0.9 or larger value might be more suitable. The larger value causes exclusion of more low abundance sequences, more sequences should be obtained per sample.

The following files should be saved in the output folder.

parameter.txt The minimum size of primary clusters remained
primarycluster.denoised.fasta.gz Representative sequences of primary clusters determined as non-noisy
primarycluster.fasta.gz Representative sequences of primary clusters
secondarycluster.fasta.gz Representative sequences of secondary clusters
RunID__TagID__PrimerID.noisyreads.txt.gz List of sequences determined as noisy
RunID__TagID__PrimerID.singletons.txt.gz List of singletons after primary clustering

Many other files might be generated and do not delete such files because those might be required in subsequent analysis.

3.1 Noisy and/or chimeric sequence removal based on PCR replicates

To detect and remove noisy and/or chimeric sequences using PCR replicates, which samples are from the same templates need to be provided as a text file like below.

```
| sample1  sample2  sample3
| sample4  sample5
| sample6  sample7
```

Tab-delimited sample list in a line indicates samples from the same templates. Samples from different templates must be placed in different lines. 3 or more replicates are allowed. The number of replicates can vary among templates.

The above file are prepared, the following command removes uncommon primary cluster as noisy and/or chimeric.

```
> clcleanseqv \
--replicatelist=ListOfPCRreplicates \
--derepmode=PREFIX \
--primarymaxnmismatch=0 \
--secondarymaxnmismatch=1 \
--pnoisycluster=0.5 \
--numthreads=NumberOfCPUs \
inputfile1 \
snip* \
inputfileN \
outputfolder ↓
```

If 3 or more replicates are available, only common sequences among all replicates are determined as non-noisy and nonchimeric. If a primary cluster occured among multiple templates and the cluster determined as noisy or chimeric in a template, the cluster determined as noisy or chimeric in the other templates and will be excluded from all the samples by default. However, the following arguments can change this decision method.

`--minnreplicate` Specified as integer larger than 1. If the number of replicates occured is equal to or larger than this value, such primary cluster will be determined as non-noisy and nonchimeric. The default value is `2`.

`--minpreplicate` Specified as decimal larger than 0. If the proportion of replicates occured (number of replicates occured / total number of replicates of the same template) is equal to or larger than this value, such primary cluster will be determined as non-noisy and nonchimeric. The default value is `1`.

`--minnpositive` Specified as integer larger than 0. If the number of sequences determined as noisy or chimeric of a primary cluster is equal to or larger than this value, the cluster determined as noisy or chimeric in all the samples. The default value is `1`.

`--minppositive` Specified as decimal equal to or larger than 0. If the proportion of sequences determined as noisy or chimeric of a primary cluster (number of sequences determined as noisy or chimeric of a primary cluster / total number of sequences of a primary cluster) is equal to or larger than this value, the cluster determined as noisy or chimeric in all the samples. The default value is `0`.

`--runname` Specify RunID. RunIDs in all sample names will be replaced to this RunID. If multiple sample names become to the same, such samples will be integrated.

The `--minnreplicate` and `--minpreplicate` are arguments about intrasample judgement. The `--minnpositive` and `--minppositive` are arguments about intersample judgement. If both of `--minnreplicate` and `--minpreplicate` are fullfilled, the primary cluster will be determined as non-noisy and nonchimeric. If both of `--minnpositive` and `--minppositive` are fullfilled, the primary cluster will be determined as noisy or chimeric. The decision is common among all samples and different decision among samples are not allowed. If there are samples that is not written in replicate list file, such sample is not used in this decision.

If `clcleanseqv` is executed like above, the following files will be saved additionally.

primarycluster.chimeraremoved.fasta.gz Representative sequences determined as nonchimeric.
primarycluster.cleaned.fasta.gz Representative sequences determined as non-noisy and nonchimeric
RunID__TagID__PrimerID.chimericreads.txt.gz List of sequences determined as chimeric

Chapter 4

OTU picking based on nucleotide sequence clustering

4.1 Inter-sample clustering

To pick OTUs by clustering, run the following command.

```
> clclassseqv \
--minident=0.97 \
--numthreads=NumberOfCPUs \
inputfile1 \
snip* \
inputfileN \
outputfolder ↓
```

Give `primarycluster.cleaned.fasta.gz` (if you applied noisy and/or chimeric sequence removal using PCR replicates) or `primarycluster.denoised.fasta.gz` (if you did not apply noisy and/or chimeric sequence removal using PCR replicates) which is generated by `clcleanseqv` as input files. If you are using non-overlapped paired-end sequence data concatenated by `clconcatpair`, give `--paddinglen=16` argument additionally.

In the output folder, the following files should be saved.

clustered.otu.gz Compressed file which records affiliation of raw sequences to OTUs
clustered.fasta Representative sequences of OTUs

4.2 Mapping raw sequencing reads to representative sequences of the clusters

If there are the raw sequences determined as noisy or chimeric which are as similar or more similar to representative sequences than specified identity threshold, such raw sequences can be recovered in this step. This

process can decrease excluded sequences.

To perform this process, run the following command.

```
> clrecoverseqv \
--minident=0.97 \
--centroid=RepresentativeSequenceFile \
--numthreads=NumberOfCPUs \
inputfile1 \
snip* \
inputfileN \
outputfolder ↓
```

Specify `clustered.fasta` generated by `clclassseqv` and `primarycluster.fasta.gz` generated by `clcleanseqv` as representative sequence file and input file, respectively. If you are using non-overlapped paired-end sequence data concatenated by `clconcatpair`, give `--paddinglen=16` argument additionally. In the output folder, the output files same as `clclassseqv` will be generated.

Chapter 5

Summarizing and post-processing of OTU picking results

5.1 Making summary table

The following command generates a sample x OTU table containing the number of sequences in every cell from OTU picking results.

```
> clsumclass \
--output=Matrix \
inputfile \
outputfile ↓
```

Give `clustered.otu.gz` generated by `clclassseqv` or `clrecoverseqv` as input file. The output file is tab-delimited text file like Table 5.1 and can be edit by spreadsheet softwares such as Microsoft Excel. Note that spreadsheet software can not read too large table. This file can be used for community ecological analysis in R.

samplename	OTU1	OTU2	OTU3	OTU4	OTU5	OTU6
sampleA	2371	0	0	12	3	0
sampleB	0	1518	0	25	0	1
sampleC	1398	0	0	8	77	6
sampleD	0	1436	0	10	0	0
sampleE	0	0	1360	0	15	3
sampleF	0	0	977	55	6	8

Table 5.1 An example of summary table — Numbers in cells indicate the observed number of raw sequences.

5.2 Excluding specified OTUs and/or samples from summary table

The `clsumclass` command output all OTUs and samples to summary table. The following command can filter samples and/or OTUs which matches several conditions.

```
> clfiltersum \
arguments \
inputfile \
outputfile ↓
```

Both input file and output file are tab-delimited text files of summary tables. Acceptable arguments are listed below.

`--minnseqotu` Specified as integer as large as or larger than 0. OTUs whose number of raw sequencing reads of every sample is lower than this value will be excluded.
`--minpseqotu` Specified as decimal ranging from 0 to 1. OTUs whose proportion of raw sequencing reads (number of raw reads / total number of raw reads of sample) of every sample is lower than this value will be excluded.
`--minntotalseqotu` Specified as integer as large as or larger than 0. OTUs whose total number of raw sequencing reads is lower than this value will be excluded.
`--minnseqsample` Specified as integer as large as or larger than 0. Samples whose number of raw sequencing reads of every OTU is lower than this value will be excluded.
`--minpseqsample` Specified as decimal ranging from 0 to 1. Samples whose proportion of raw sequencing reads (number of raw reads / total number of raw reads of OTU) of every OTU is lower than this value will be excluded.
`--minntotalseqsample` Specified as integer as large as or larger than 0. Samples whose total number of raw sequencing reads is lower than this value will be excluded.
`--otu` Specify OTUs as comma-delimited names you want to keep.
`--negativeotu` Specify OTUs as comma-delimited names you want to eliminate.
`--otulist` Specify a file name. In the file, write an OTU name per line you want to keep.
`--negativeotulist` Specify a file name. In the file, write an OTU name per line you want to eliminate.
`--otuseq` Specify a FASTA sequence file name. In the file, write OTU names you want to keep. Sequences will be ignored.
`--negativeotuseq` Specify a FASTA sequence file name. In the file, write OTU names you want to eliminate. Sequences will be ignored.
`--sample` Specify samples as comma-delimited names you want to keep.
`--negativesample` Specify samples as comma-delimited names you want to eliminate.
`--samplelist` Specify a file name. In the file, write a sample name per line you want to keep.
`--negativesamplelist` Specify a file name. In the file, write a sample name per line you want to eliminate.
`--replicatelist` Specify a file name. In the file, write PCR replicates as tab-delimited sample names in a line. PCR replicates will be integrated in output file. The number of raw reads will be summed up.
`--runname` Specify RunID. RunIDs in all sample names will be replaced this RunID. If samples whose names are completely matched are occur, the samples will be integrated in output file. The number of raw reads will be summed up.

Applying post OTU picking chimeric sequence removal or nontarget sequence removal explained later, OTU filtering using remaining sequences and `--otuseq` argument can be applied to summary table. If you want to apply additional filtering to summary table, this OTU filtering should be applied at first.

5.3 Chimera removal based on UCHIME algorithm

Post OTU picking chimera removal based on UCHIME algorithm with and without (*de novo*) reference sequences can be applied additionally. If you want to apply both with and without reference chimera removal, without reference (*de novo*) chimera removal should be applied at first, and then reference-based chimera removal should be applied. To perform these chimera removal, use `clrunuchime` command. In this command, UCHIME algorithm implemented in VSEARCH is used.

5.3.1 *De novo* chimera removal

Execute `clrunuchime` like the following.

```
> clrunuchime \
--otufile=*.otu.gz \
inputfile \
outputfolder ↓
```

The output files in output folder is explained later.

5.3.2 Reference-based chimera removal

Execute `clrunuchime` like the following.

```
> clrunuchime \
--referencedb=ReferenceDatabase \
inputfile \
outputfolder ↓
```

If you already applied *de novo* chimera removal, there is `nonchimeras.fasta` in the output folder of *de novo* chimera removal. This file is recommended for input file of reference-based chimera removal. The output files in output folder is explained later.

The following ready-made reference databases are provided and installed.

cdu12s Claident Database for UCHIME for animal 12S ver.20170513
cducox1 Claident Database for UCHIME for animal COX1(COI) ver.20170513
cducytb Claident Database for UCHIME for animal Cyt-b ver.20170513
cdumatk Claident Database for UCHIME for plant matK ver.20170513
cdurbcl Claident Database for UCHIME for plant rbcL ver.20170513
cdutrnhpsba Claident Database for UCHIME for plant trnH-psbA ver.20170513
rdpgoldv9 RDP Gold v9 for prokaryotic 16S
silva128LSUref SILVA Release 128 for LSU rRNA
silva128SSUrefnr99 SILVA Release 128 Nr99 for SSU rRNA
unite20161201 UNITE ver.20161201 for fungal ITS
unite20161201untrim UNITE ver.20161201 without trimming for fungal ITS
unite20161201its1 UNITE ver.20161201 for fungal ITS1
unite20161201its2 UNITE ver.20161201 for fungal ITS2

5.3.3 About contents of output folder

In the output folder of `clrunuchime`, the following files will be saved.

chimeras.fasta Sequences determined as chimeric
nonchimeras.fasta Sequences determined as nonchimeric

uchimealns.txt Alignment used in chimera detection
uchimeout.txt Detected parent sequences, chimera scores and the other information

To know meaning of each element in `uchimeout.txt`, see the following URL.
http://drive5.com/usearch/manual/uchimeout.html

5.4 Excluding low-abundance OTUs from OTU sequences

In the output folder of OTU picking, `clustered.fasta` is saved as representative sequence file. This file can be used for taxonomic assignment. However, the number of OTUs is sometimes too large to assign taxonomy if rare OTUs are kept. In such cases, extracting more abundant OTUs than specified value is useful. The following command exclude OTUs observed less than 5.

```
> clfilterseq \
--otufile=*.otu.gz \
--minnseq=5 \
inputfile \
outputfile ↓
```

5.5 Sequence splitting based on conservative motif recognition

If your data sequences contain multiple loci (e.g. ITS1–5.8S rRNA–ITS2), splitting loci might cause better taxonomic assignment. If conservative motif exists at the border of loci or close to the border, such motif can be used for splitting. Universal primer annealing positions are recommended for such conservative motif. The following command can divide sequences to anterior and posterior of matched positions.

```
> cldivseq \
--query=Sequence \
--border=start \
inputfile \
outputfile1 \
outputfile2 ↓
```

By default, given query sequence will be searched from the sequences contained in inputfile based on Needleman-Wunsch algorithm allowing 15% mismatches. If matched positions are found, sequences will be divided on head of matched positions to anterior and posterior sequences to outputfile1 and outputfile2, respectively. Unmatched sequences will be output to outputfile1. Given query sequence must be same strand as target sequences. To use different strand query, add `--reversecomplement` argument.

If `--border=end` is specified, sequences will be divided on tail of matched positions. If `--border=both` is given (this is default), anterior sequences of head of matched positions and posterior sequences of tail of matched positions will be saved, and matched positions will be excluded from both output. If query is unmatched, undivided sequences will be saved to outputfile1 and outputfile2 will lack the sequences. Specifying the

`--makedummy` argument, dummy sequence `A` will be saved to outputfile2. This argument is useful for merging multiple taxonomic assignment results of multiple loci. Repeat `cldivseq` execution to divide sequences to 3 or more subsequences.

5.6 ITS, SSU rRNA or LSU rRNA sequence extraction using ITSx or Metaxa

ITSx can detect ITS1 and ITS2, and extract those positions only (Bengtsson-Palme *et al.*, 2013). In many taxa, ITS is highly variable but SSU, 5.8S and LSU rRNA is much more conservative. Such conservative positions cause misidentifications or unidentified results because conservative position of distant taxa will match to query sequences. Extracting ITS positions by ITSx might solve this problem.

Metaxa can detect SSU (12S/16S/18S) rRNA and LSU (26S/28S) rRNA, and extract those positions only (Bengtsson *et al.*, 2011). In the case of SSU rRNA, Metaxa can also distinguish among eukaryotic nuclear, mitochondrial, chloroplast and prokaryotic. SSU rRNA is widely used for DNA barcoding and metabarcoding of eukaryotes, but is contained not only by nuclear of eukaryotes but also by mitochondria, chloroplast and contaminated prokaryotes. Therefore, nontarget SSU rRNA frequently contaminate the data. Such nontarget SSU should be deleted for community ecological analysis.

After the filtering sequences by ITSx or Metaxa, `clfiltersum` with `--otuseq` argument can be used to justify summary table to the sequence file (see section 5.2).

5.7 Searching and excluding nontarget sequences

ITSx and Metaxa can apply to ITS and SSU/LSU rRNA only and cannot apply to the other loci. For the other loci, gene prediction and annotation programs and multiple alignment programs can be used for finding nontarget sequences. In ClustalW2, ClustalX2 (Larkin *et al.*, 2007) and MAFFT (Katoh & Standley, 2013), sorting function based on phylogenetic relatedness is available. Aligning sequences applying this sorting function, you can easily find some types of nontarget sequences by eyes with the aid of multiple alignment viewer. After the filtering sequences by this method, `clfiltersum` with `--otuseq` argument can be used to justify summary table to the sequence file (see section 5.2).

Chapter 6

Data deposition to DRA

Representative sequences can be deposited to DDBJ, EMBL or GenBank if required, but raw sequencing reads should be deposited to Sequence Read Archive such as DDBJ Sequence Read Archive (DRA). If you sequenced multiple samples with tag sequences in a NGS run like above, demultiplexed (tag and primer positions are trimmed) sequence files should be deposited. FASTQ files made by `clsplitseq` command can be used for deposition.

XML files recoding metadata about sample information need to be created. DRA provides XML creation support tool, but such tools is not easy-to-use in a case of many samples. To reduce time and effort to make XML, `clmaketsv` and `clmakexml` can be used.

To deposit raw sequences to DRA, user account of D-way of DDBJ and public key registration are required. Read DRA Handbook
http://trace.ddbj.nig.ac.jp/dra/submission_e.shtml
to know detailed procedures. In the deposition process, concept of Submission, Study, Experiment, Sample and Run, and association of those are important and need to be understood.

In DRA, Study need to be registered to BioProject database which is a research project database, and to be referred to BioProject ID (accession number). Read BioProject Handbook
http://trace.ddbj.nig.ac.jp/bioproject/submission_e.html
and register Study to BioProject before data deposition to DRA.

Sample also need to be registered to BioSample database which is a biological specimen database, and to be referred to BioSample ID (accession number). You need to read BioSample Handbook
http://trace.ddbj.nig.ac.jp/biosample/submission_e.html
and to register Sample to BioSample before data deposition to DRA. In the cases of metabarcoding using amplicon sequences amplified by universal primer set, MIMarks-Survey should be specified as type of MIxS. In BioSample database, sampling locality, elevation, depth, temperature, humidity, pH etc. can be added as metadata. Sample information should be added as many as possible for future generations and yourself. Meaning of each item is explained in checklist
http://wiki.gensc.org/index.php?title=MIMARKS
provided by Genomic Standards Consortium. If you cannot understand meaning of each item, ask to DDBJ. In the cases of underground samples and underseafloor samples, hight or depth of sampling point from mean

sea level, hight or depth of sampling point from ground/seafloor surface and hight or depth of ground/seafloor surface from mean sea level need to be distinguished. Because it is very confusing, please be careful.

6.1 Preparing tab-delimited text file for XML generation

In the XML files, contained sequence infomation need to be provided for each FASTQ files. Because this is very costful, `clmaketsv` generates simple tab-delimited text file, and `clmakexml` creates XML files from edited tab-delimited text file. The `clmaketsv` command can run like below.

```
> clmaketsv \
inputfile1 \
snip* \
inputfileN \
outputfile ↓
```

FASTQ files to deposit to DRA should be specified as input files. Wild cards can be used in input file name. Once tab-delimited text generated, use spreadsheet softwares such as Microsoft Excel to edit this file, and fill in the blank cells. Note that Microsoft Excel automatically convert several types of characters and this function cannot be disabled. If `[Foo,Bar]` is found in a cell, select `Foo` or `Bar` and delete bracket, comma and alternatives. If you find `<Fill in this cell>` in a cell, fill this cell according to written instruction in <> and delete <>. You might do something with the other cells. If you do not have BioProject ID and/or BioSample ID because of assignment delay, you can use Submission ID with prefix PSUB for BioProject and SSUB for BioSample.

6.2 XML generation from tab-delimited text

After editing tab-delimited text, run `clmakexml` like below.

```
> clmakexml \
Tab-DelimitedTextFile \
Submission-ID ↓
```

Then, XML files required to deposit FASTQ files will be created. Multiple tab-delimited text files can be given to `clmakexml`. In such cases, the contents of first 3 lines in tab-delimited text of secondary or n-ary files will be ignored.

Submission-ID is assigned by DRA when [Create new submission(s)] button is pushed. Submission-ID is compliant with `UserID-0001` and should be given for last argument of `clmakexml`. The execution of `clmakexml` is completed, 3 `Submission-ID.*.xml` files will be generated. These files can be upload to DRA using XML Upload. All processes were completed, DRA accession number is assigned and is sent from DRA to your Email address. Note that BioProject ID (accession number) was written in manuscript in most cases.

Chapter 7

Estimation of host organisms of nucleotide sequences (a.k.a. DNA barcoding)

DNA barcoding is a taxonomic identification method of biological specimens using nucleotide sequences and becoming widely applied to broad area. However, our reference sequence database is not enough for species level identification and algorithm that is suitable for incomplete reference database is lacking. To solve this problem, I proposed a new criterion that sequence distance between query and nearest-neighbor must be smaller than maximum distance within resulting taxon, developed QCauto algorithm which fullfills this criterion (Tanabe & Toju, 2013) and implemented in Claident. Note that we can expect that host organism of query is same as nearest-neighbor in the case that potentially observable species are completely described and barcode DNA sequences are completely sequenced and registered to reference database. For such case, 1-nearest-neighbor (closest match) method is also implemented in Claident.

7.1 Retrieval of neighborhood sequences based on BLAST search

The GenBank IDs of neighborhood sequences required to fullfill "sequence distance between query and nearest-neighbor must be smaller than maximum distance within resulting taxon" can be retrieved by the following command.

```
> clidentseq \
--blastdb=overall_genus \
--numthreads=NumberOfCPUs \
inputfile \
outputfile ↓
```

Give FASTA formatted nucleotide sequence file as an input file. Specify reference sequence database for BLAST search to `--blastdb` argument. The following databases should be installed with Claident.

animals_COX1_genus Animal (Metazoa) *COX1* sequences which have genus or lower level taxonomic information
animals_COX1_species Animal (Metazoa) *COX1* sequences which have species or lower level taxonomic information
animals_mt_genus Animal (Metazoa) mitochondrial sequences which have genus or lower level taxonomic information
animals_mt_species Animal (Metazoa) mitochondrial sequences which have species or lower level taxonomic information

eukaryota_LSU_genus Eukaryotic LSU (28S) rRNA sequences which have genus or lower level taxonomic information
eukaryota_LSU_species Eukaryotic LSU (28S) rRNA sequences which have species or lower level taxonomic information
eukaryota_SSU_genus Eukaryotic SSU (18S) rRNA sequences which have genus or lower level taxonomic information
eukaryota_SSU_species Eukaryotic SSU (18S) rRNA sequences which have species or lower level taxonomic information
fungi_ITS_genus Fungal ITS sequences which have genus or lower level taxonomic information
fungi_ITS_species Fungal ITS sequences which have species or lower level taxonomic information
overall_class NCBI nt sequences which have class or lower level taxonomic information
overall_order NCBI nt sequences which have order or lower level taxonomic information
overall_family NCBI nt sequences which have family or lower level taxonomic information
overall_genus NCBI nt sequences which have genus or lower level taxonomic information
overall_species NCBI nt sequences which have species or lower level taxonomic information
plants_matK_genus Plant *matK* sequences which have genus or lower level taxonomic information
plants_matK_species Plant *matK* sequences which have species or lower level taxonomic information
plants_rbcL_genus Plant *rbcL* sequences which have genus or lower level taxonomic information
plants_rbcL_species Plant *rbcL* sequences which have species or lower level taxonomic information
plants_trnH-psbA_genus Plant *trnH–psbA* sequences which have genus or lower level taxonomic information
plants_trnH-psbA_species Plant *trnH–psbA* sequences which have species or lower level taxonomic information
prokaryota_16S_genus Prokaryotic (Bacterial and Archaeal) 16S rRNA sequences which have genus or lower level taxonomic information
prokaryota_16S_species Prokaryotic (Bacterial and Archaeal) 16S rRNA sequences which have species or lower level taxonomic information
prokaryota_all_genus Prokaryotic (Bacterial and Archaeal) sequences which have genus or lower level taxonomic information
prokaryota_all_species Prokaryotic (Bacterial and Archaeal) sequences which have species or lower level taxonomic information
semiall_class Reduced database excluding all sequences of vertebrates, *Caenorhabditis* and *Drosophila* from `overall_class`
semiall_order Reduced database excluding all sequences of vertebrates, *Caenorhabditis* and *Drosophila* from `overall_order`
semiall_family Reduced database excluding all sequences of vertebrates, *Caenorhabditis* and *Drosophila* from `overall_family`
semiall_genus Reduced database excluding all sequences of vertebrates, *Caenorhabditis* and *Drosophila* from `overall_genus`
semiall_species Reduced database excluding all sequences of vertebrates, *Caenorhabditis* and *Drosophila* from `overall_species`

The `overall_*` databases are very large and require large memory but are versatile and robust to contamination of sequences of unexpected taxa or loci. Thus, the `overall_*` are strongly recommended in most cases. In the `overall_genus` database, the sequences which have genus or lower level taxonomic information are kept and the other sequences are excluded and nearest-neighbor often cannot be found in minor taxa. The `overall_class` database is recommended for such cases. The other databases are much smaller than `overall_*` and require much smaller memory.

7.1.1 Accelerating BLAST search using cache databases

The `clidentseq` command runs BLAST search multiple times and requires long time. To reduce runtime of `clidentseq`, cache sequence database containing top-10,000 high score sequences can be constructed for each query sequence. Processing speed of `clidentseq` can be extremely accelerated by using cache database and strongly recommended. To construct cache database, run `clmakecachedb` like the following.

```
> clmakecachedb \
--blastdb=overall_genus \
--numthreads=NumberOfCPUs \
inputfile \
outputfolder ↓
```

In the execution of `clidentseq`, give the output folder of `clmakecachedb` to `--blastdb` argument, then cache database will be used. Note that the input files of `clmakecachedb` and `clidentseq` must be tha same file.

Required amount of memory is also extremely reduced.

7.1.2 In the case of perfect reference database

If potentially observable species are completely described and barcode DNA sequences are completely sequenced and registered to reference database, retrieving top-1 and tie sequences whose percent-identity to query is 99% or more is recommended and it can be performed by the following command.

```
> clidentseq \
blastn -task megablast -word_size 16 end \
--method=1,99% \
--blastdb=overall_genus \
--numthreads=NumberOfCPUs \
inputfile \
outputfile ↓
```

In search of top-1 and tie sequences whose percent-identity to query is 99% or more, we can expect that overleaping hardly occur even if `-task megablast -word_size 16` is given as BLAST search argument, thus this argument is specified.

7.2 Taxonomic assignment based on neighborhood sequences

The following command assigns taxonomy to query ascending taxonomic level until taxonomic information of neighborhood sequences are completely matched. This is called as lowest common ancestor (LCA) algorithm (Huson *et al.*, 2007).

```
> classigntax \
--taxdb=overall_genus \
inputfile \
outputfile ↓
```

Give the output file of `clidentseq` as input file. The `--taxdb` argument is to give taxonomy database of reference sequences. The same name taxonomy databases as reference sequence databases should be installed and specified.

The `classigntax` requires 2 or more neighborhood sequences by default. If `--method=1,99%` was given to `clidentseq` like section 7.1.2, taxonomic assignment of all sequences must be failed. In such case, reduce required minimum number of neighborhood sequences like below.

```
> classigntax \
--taxdb=overall_genus \
--minnsupporter=1 \
inputfile \
outputfile ↓
```

The output file of `classigntax` is tab-delimited text file like Table 7.1.

query	phylum	genus	species
seqA	Ascomycota	*Chloridium*	*Chloridium virescens*
seqB	Ascomycota	*Chloridium*	*Chloridium virescens*
seqC	Ascomycota	*Chloridium*	*Chloridium virescens*
seqD	Basidiomycota	*Amanita*	*Amanita fuliginea*
seqE	Basidiomycota	*Coltriciella*	*Coltriciella dependens*
seqF	Basidiomycota	*Filobasidium*	*Filobasidium uniguttulatum*
seqG	Basidiomycota	*Laccaria*	*Laccaria bicolor*
seqH	Basidiomycota	*Lactarius*	*Lactarius quietus*
seqI	Basidiomycota	*Russula*	*Russula densifolia*
seqJ	Basidiomycota	*Russula*	*Russula densifolia*
seqK	Basidiomycota	*Russula*	*Russula densifolia*
seqL	Basidiomycota	*Russula*	*Russula vesca*
seqM	Basidiomycota	*Agaricus*	
seqN	Basidiomycota	*Amanita*	
seqO	Basidiomycota	*Amanita*	
seqP	Ascomycota	*Bisporella*	
seqQ	Ascomycota	*Capronia*	
seqR	Ascomycota	*Capronia*	
seqS	Ascomycota	*Cenococcum*	

Table 7.1 Examples of taxonomic assignment — Blank cel means unidentified. Several taxonomic levels are omitted for reduce width.

By default, the `classigntax` command assigns taxonomy to query ascending taxonomic level until taxonomic information of neighborhood sequences are completely matched. If reference database is contaminated by misidentified sequence and such sequence is contain in neighborhoods, lower level taxonomy likely to be unidentified. This "strict consensus" method is robust but too conservative in some cases. The following command allows 5% of neighborhood sequences have different taxonomic information to resulting taxonomy.

```
> classigntax \
--taxdb=overall_genus \
--maxpopposer=0.05 \
--minsoratio=19 \
inputfile \
outputfile ↓
```

The `classigntax` treat neighborhood sequences which have same taxonomic information to resulting taxonomy as "supporter", neighborhood sequences which have different taxonomic information to resulting taxonomy as "opposer". The `--maxpopposer` argument is to specify maximum allowing proportion of opposer sequences. The `--minsoratio` argument is to set minimum required ratio of "supporter / opposer". Because there are possibly unidentified sequences which do not have taxonomic information and such sequences are neither supporter nor opposer, the above 2 arguments are required. In the case of above command, 5% of opposers are tolerated and 19 times as many supporters as opposers are required.

7.3 Merging multiple taxonomic assignments based on consensus

In plant chloroplast loci, single locus query sequences cannot be identified at the lower taxonomic level in many cases because of lack of variations. Merging multiple results based on `overall_genus` (more reliable database) and `overall_class` (less reliable database) are often useful. Merging multiple results based on "strict consensus" LCA (strict LCA) and "95%-majority rule consensus" LCA (relaxed LCA) is also useful in some cases. Such merging multiple assignment results can be performed with the aid of `clmergeassign`.

Multiple assignment results of *rbcL* and *matK* can be merged prefering lower level identification by the following command.

```
> clmergeassign \
--priority=equal \
--preferlower \
ResultOfrbcL \
ResultOfmatK \
outputfile ↓
```

The following command merges multiple results accepting that query identified as same taxon in both results and that query identified in one result and unidentified in the other result. Note that mismatch at the higher level taxonomy is not allowed.

```
> clmergeassign \
--priority=equal \
ResultOfrbcL \
ResultOfmatK \
outputfile ↓
```

Three assignment results of *rbcL*, *matK* and *trnH–psbA* can be merged prefering lower level identification by the following command.

```
> clmergeassign \
--priority=equal \
--preferlower \
ResultOfrbcL \
ResultOfmatK \
ResultOftrnH-psbA \
outputfile ↓
```

The following command merges two results prefering result of `overall_genus`.

```
> clmergeassign \
--priority=descend \
ResultOfoverall_genus \
ResultOfoverall_class \
outputfile ↓
```

In this case, result of `overall_class` is accepted if matched to result of `overall_genus` and identified at the lower level than result of `overall_genus`.

The following command merges two results prefering result of strict LCA.

```
> clmergeassign \
--priority=descend \
ResultOfStrictLCA \
ResultOfRelaxedLCA \
outputfile ↓
```

In this case, result of relaxed LCA is accepted if matched to result of strict LCA and identified at the lower level than result of strict LCA.

References

Andrews, S. (2010) Software distributed by the author at http://www.bioinformatics.babraham.ac.uk/projects/fastqc/.

Bengtsson, J., Eriksson, K. M., Hartmann, M., Wang, Z., Shenoy, B. D., Grelet, G.-A., Abarenkov, K., Petri, A., Rosenblad, M. A., Nilsson, R. H. (2011) Metaxa: a software tool for automated detection and discrimination among ribosomal small subunit (12S/16S/18S) sequences of archaea, bacteria, eukaryotes, mitochondria, and chloroplasts in metagenomes and environmental sequencing datasets. *Antonie Van Leeuwenhoek*, **100**, 471–475.

Bengtsson-Palme, J., Ryberg, M., Hartmann, M., Branco, S., Wang, Z., Godhe, A., De Wit, P., Sánchez-García, M., Ebersberger, I., de Sousa, F., Amend, A., Jumpponen, A., Unterseher, M., Kristiansson, E., Abarenkov, K., Bertrand, Y. J. K., Sanli, K., Eriksson, K. M., Vik, U., Veldre, V., Nilsson, R. H. (2013) Improved software detection and extraction of ITS1 and ITS2 from ribosomal ITS sequences of fungi and other eukaryotes for analysis of environmental sequencing data *Methods in Ecology and Evolution*, **4**, 914–919.

Edgar, R. C., Haas, B. J., Clemente, J. C., Quince, C., Knight, R. (2011) UCHIME improves sensitivity and speed of chimera detection. *Bioinformatics*, **27**, 2194–2200.

Fadrosh, D. W., Ma, B., Gajer, P., Sengamalay, N., Ott, S., Brotman, R. M., Ravel, J. (2014) An improved dual-indexing approach for multiplexed 16S rRNA gene sequencing on the Illumina MiSeq platform. *Microbiome*, **2**, 6.

Hamady, M., Walker, J. J., Harris, J. K., Gold, N. J., Knight, R. (2008) Error-correcting barcoded primers for pyrosequencing hundreds of samples in multiplex. *Nature Methods*, **5**, 235–237.

Huson, D. H., Auch, A. F., Qi, J., Schuster, S. C. (2007) MEGAN analysis of metagenomic data. *Genome Research*, **17**, 377–386.

Illumina corporation (2013) 16S metagenomic sequencing library preparation.

Katoh, K., Standley, D. M. (2013) MAFFT multiple sequence alignment software version 7: improvements in performance and usability. *Molecular Biology and Evolution*, **30**, 772–780.

Kunin, V., Engelbrektson, A., Ochman, H., Hugenholtz, P. (2010) Wrinkles in the rare biosphere: pyrosequencing errors can lead to artificial inflation of diversity estimates. *Environmental Microbiology*, **12**, 118–123.

Lange, A., Jost, S., Heider, D., Bock, C., Budeus, B., Schilling, E., Strittmatter, A., Boenigk, J., Hoffmann, D. (2015) AmpliconDuo: A Split-Sample Filtering Protocol for High-Throughput Amplicon Sequencing of Microbial Communities. *PLoS One*, **10**, e0141590.

Larkin, M. A., Blackshields, G., Brown, N. P., Chenna, R., McGettigan, P. A., McWilliam, H., Valentin, F., Wallace, I. M., Wilm, A., Lopez, R., Thompson, J. D., Gibson, T. J., Higgins, D. G. (2007) Clustal W and Clustal X version 2.0. *Bioinformatics*, **23**, 2947–2948.

Li, W., Fu, L., Niu, B., Wu, S., Wooley, J. (2012) Ultrafast clustering algorithms for metagenomic sequence analysis. *Briefings in Bioinformatics*, **13**, 656–668.

Nelson, M. C., Morrison, H. G., Benjamino, J., Grim, S. L., Graf, J. (2014) Analysis, optimization and verification of Illumina-generated 16S rRNA gene amplicon surveys. *PLoS One*, **9**, e94249.

Schmieder, R., Edwards, R. (2011) Quality control and preprocessing of metagenomic datasets. *Bioinformatics*,

27, 863–864.

Stevens, J. L., Jackson, R. L., Olson, J. B. (2013) Slowing PCR ramp speed reduces chimera formation from environmental samples. *Journal of Microbiological Methods*, **93**, 203–205.

Tanabe, A. S., Toju, H. (2013) Two new computational methods for universal DNA barcoding: a benchmark using barcode sequences of bacteria, archaea, animals, fungi, and land plants. *PLoS One*, **8**, e76910.

Zhang, J., Kobert, K., Flouri, T., Stamatakis, A. (2014) PEAR: a fast and accurate Illumina Paired-End reAd mergeR. *Bioinformatics*, **30**, 614–620.

Appendix A

Instllation of optional programs

A.1 Installation of bcl2fastq

Bcl2fastq is a program to convert BCL formatted basecall data to FASTQ provided by Illumina. Bcl2fastq is available at
http://support.illumina.com/downloads/bcl2fastq_conversion_software.html

Bcl2fastq v1.8.4 for Illumina sequencing systems using versions of RTA earlier than RTA version 1.18.54 can be installed like below.

```
# Install alien
$ sudo apt-get install alien
# Download bcl2fastq v1.8.4 from Illumina
$ wget \
ftp://webdata:webdata@ussd-ftp.illumina.com/Downloads/Software/bcl2fastq/bcl2fastq-1.8.4-Linux-x86_64.rpm
# Convert .rpm to .deb and install .deb
$ sudo alien -i \
bcl2fastq-1.8.4-Linux-x86_64.rpm
# Install required packages
$ sudo apt-get install libxml-simple-perl xsltproc
```

In newer version of Ubuntu, incompatible Perl to bcl2fastq is installed. To solve this problem, run the following commands.

```
$ sudo perl -i -npe \
's/qw\(ELAND_FASTQ_FILES_PER_PROCESS\)/\("ELAND_FASTQ_FILES_PER_PROCESS"\)/' \
/usr/local/lib/bcl2fastq-1.8.4/perl/Casava/Alignment/Config.pm
$ sudo perl -i -npe \
's/qw\(ELAND_GENOME\)/\("ELAND_GENOME"\)/' \
/usr/local/lib/bcl2fastq-1.8.4/perl/Casava/Alignment/Config.pm
```

You can also use text editor program to edit line 747 and line 751 of
`/usr/local/lib/bcl2fastq-1.8.4/perl/Casava/Alignment/Config.pm`
and replace `qw(FOO)` to `("FOO")`.

To install bcl2fastq2 v2.18.0.12 for all Illumina sequencing systems running RTA version 1.18.54 and above, run the following commands.

```
# Install rpm2cpio and cpio
> sudo apt-get install rpm2cpio cpio ↓
# Download bcl2fastq2 v2.18.0.12 from Illumina
> wget -O bcl2fastq2-v2-18-0-12-linux-x86-64.zip http://bit.ly/2drPH3W ↓
# Decompress zip file
> unzip -qq bcl2fastq2-v2-18-0-12-linux-x86-64.zip ↓
# Extract executable command from .rpm
> rpm2cpio \
bcl2fastq2-v2.18.0.12-Linux-x86_64.rpm | cpio -id ↓
# Install executable command
> sudo mv usr/local/bin/bcl2fastq /usr/local/bin/ ↓
# Install the other resources
> sudo cp -R usr/local/share/css /usr/local/share/ ↓
> sudo cp -R usr/local/share/xsl /usr/local/share/ ↓
```

Appendix B

Terminal command examples

B.1 Counting sequences in a file

```
# Count seqs in FASTQ
> grep -P -c '^\+\r?\n?$' inputfile ↓
# Count seqs in gzipped FASTQ
> gzip -dc inputfile | grep -P -c '^\+\r?\n?$' ↓
# Count seqs in bzip2ed FASTQ
> bzip2 -dc inputfile | grep -P -c '^\+\r?\n?$' ↓
# Count seqs in xzed FASTQ
> xz -dc inputfile | grep -P -c '^\+\r?\n?$' ↓
# Count seqs in FASTA
> grep -P -c '^>' inputfile ↓
# Count seqs in gzipped FASTA
> gzip -dc inputfile | grep -P -c '^>' ↓
# Count seqs in bzip2ed FASTA
> bzip2 -dc inputfile | grep -P -c '^>' ↓
# Count seqs in xzed FASTA
> xz -dc inputfile | grep -P -c '^>' ↓
```

B.2 Viewing sequence files

```
# Output contents of uncompressed file to screen
> cat inputfile ↓
# View contents of uncompressed file
> less inputfile ↓
# Output contents of gzipped file to screen
> gzip -dc inputfile ↓
# View contents of gzipped file
> gzip -dc inputfile | less ↓
# Extract first seq from FASTQ and output to screen
> head -n 4 inputfile ↓
# Extract first seq from gzipped FASTQ and output to screen
> gzip -dc inputfile | head -n 4 ↓
# Extract matched seqs to regular expression in seq name from FASTQ and output to screen
> grep -P -A 3 '^\@RegularExpressionOfSequenceName' inputfile ↓
# Extract matched seqs to regular expression in seq name from gzipped FASTQ and output to screen
> gzip -dc inputfile | grep -P -A 3 '^\@RegularExpressionOfSequenceName' ↓
```

B.3 Compression and decompression

```
# Decompress all .fastq.gz files in current folder
> for f in *.fastq.gz ↓
do gzip -d $f ↓
done ↓
# Decompress all .fastq.gz files in current folder and subfolders
> for f in `find .  -name *.fastq.gz` ↓
do gzip -d $f ↓
done ↓
# Decompress all .fastq.gz files in current folder using 4 CPUs
> ls *.fastq.gz | xargs -L 1 -P 4 gzip -d ↓
# Decompress all .fastq.gz files in current folder and subfolders using 4 CPUs
> find .  -name *.fastq.gz | xargs -L 1 -P 4 gzip -d ↓
# Compress all .fastq files in current folder
> for f in *.fastq ↓
do gzip $f ↓
done ↓
# Compress all .fastq files in current folder and subfolders
> for f in `find .  -name *.fastq` ↓
do gzip $f ↓
done ↓
# Compress all .fastq files in current folder using 4 CPUs
> ls *.fastq | xargs -L 1 -P 4 gzip ↓
# Compress all .fastq files in current folder and subfolders using 4 CPUs
> find .  -name *.fastq | xargs -L 1 -P 4 gzip ↓
```

B.4 Extraction and output of sequences

```
# Extract first 10,000 seqs from FASTQ and save those seqs to a file
> head -n 40000 inputfile > outputfile ↓
# Extract last 10,000 seqs from FASTQ and save those seqs to a file
> tail -n 40000 inputfile > outputfile ↓
# Extract first 10,000 seqs from gzipped FASTQ and save those seqs to gzipped FASTQ
> gzip -dc inputfile | head -n 40000 | gzip -c > outputfile ↓
# Extract last 10,000 seqs from gzipped FASTQ and save those seqs to gzipped FASTQ
```

```
> gzip -dc inputfile | tail -n 40000 | gzip -c > outputfile ↓
# Extract matched seqs to regular expression in seq name from gzipped FASTQ and save to gzipped FASTQ
> gzip -dc inputfile | grep -P -A 3 '^\@RegularExpressionOfSequenceName' | gzip -c > outputfile ↓
```