micca Documentation

Release 1.7.0

micca development team

Feb 14, 2019

Getting Started

1 Install 3

2 Supported databases 7

3 Paired-end sequencing - 97% OTU 9

4 Denoising (Illumina only) 15

5 Single-end sequencing 17

6 An introduction to the downstream analysis with R and phyloseq 21

7 Compute basic statistics, rarefy and summarize OTU/SV tables using micca 27

8 Picking OTUs for use in PICRUSt 29

9 OTU picking and Denoising 31

10 Supported file formats 37

11 Changes 39

12 classify 43

13 convert 47

14 filter 49

15 filterstats 51

16 merge 53

17 mergepairs 55

18 msa 57

19 otu 59

20 root 63

21 split 65

22 stats 67

23 tablebar 69

24 tablerare 71

25 tablestats 73

26 tabletotax 75

27 tobiom 77

28 tree 79

29 trim 81

micca (MICrobial Community Analysis) is a software pipeline for the processing of amplicon sequencing data, **from raw sequences** to **OTU tables**, **taxonomy classification** and **phylogenetic tree** inference. The pipeline can be applied to a range of highly conserved genes/spacers, such as **16S rRNA gene**, **Internal Transcribed Spacer (ITS) 18S** and **28S rRNA**. micca is an open-source, GPLv3-licensed software.

- Homepage
- Documentation (latest)
- Issues
- Github page

Key features:

- supports **single-end** (Roche 454, Illumina MiSeq/HiSeq ,Ion Torrent) and **overlapping paired-end** reads (Illumina MiSeq/HiSeq);
- **multithread** de novo greedy, closed-reference, open-reference and swarm OTU picking protocols;
- **denoising** of Illumina reads;
- **state-of-the-art taxonomic classification** algorithms (RDP and consensus-based classifier);
- fast and and memory efficient **NAST** multiple sequence alignment (MSA);
- filters low quality sequences according to the maximum allowed **expected error (EE) rate** %;
- runs on **Linux**, **Mac OS X** and **MS Windows** (through Docker containers)
- **simple, easy to use**.

Docker images are available (compmetagen/micca) starting from version 1.2.2, see the documentation (>=1.3.0) to learn how to use them. Docker hub page.

How to cite: Davide Albanese, Paolo Fontana, Carlotta De Filippo, Duccio Cavalieri and Claudio Donati. **MICCA: a complete and accurate software for taxonomic profiling of metagenomic data**. Scientific Reports 5, Article number: 9743 (2015), doi:10.1038/srep09743, Link. Dataset download: ftp://ftp.fmach.it/metagenomics/micca/scirep/.

micca wraps third party software packages and these **should be cited** if they are used:

- VSEARCH (doi: 10.7717/peerj.2584) used in `classify`, `filter`, `mergepairs`, `otu` and `msa` commands
- MUSCLE (doi: 10.1093/nar/gkh340) used in `msa` and `tree` commands
- FastTree (doi: 10.1371/journal.pone.0009490) used in the `tree` command
- Cutadapt (doi: 10.14806/ej.17.1.200) used in the `trim` command
- RDP classifier (doi: 10.1128/AEM.00062-07) used in the `classify` command
- swarm (doi: 10.7717/peerj.1420) used in the `otu` command

CHAPTER 1

Install

1.1 Using Docker (that is, on MS Windows, Mac OS X and Linux!)

The easiest way to run micca is through Docker. Docker works similarly to a virtual machine image, providing a container in which all the software has already been installed, configured and tested.

1. Install Docker for Linux, Mac OS X or Windows.
2. Run the `Docker Quickstart Terminal` (Mac OS X, Windows) or the `docker` daemon (Linux, `sudo service docker start`).
3. Download the latest version:

   ```
   docker pull compmetagen/micca
   ```

4. Run an instance of the image, mounting the host working directory (e.g. `/Users/davide/micca`) on to the container working directory `/micca`:

   ```
   docker run --rm -t -i -v /Users/davide/micca:/micca -w /micca compmetagen/micca /
   ↪bin/bash
   ```

   You need to write something like `-v //c/Users/davide/micca:/micca` if you are in Windows or `-v /home/davide/micca:/micca` in Linux. The `--rm` option automatically removes the container when it exits.

5. Now you can use micca:

   ```
   root@68f6784e1101:/micca# micca -h
   ```

Note: The RDP classifier is preinstalled in the Docker image, so you can check the software version by typing `echo $RDPPATH`

1.2 Using pip

At the moment, only Python 2.7 is supported.

1.2.1 On Ubuntu >= 12.04 and Debian >=7

We suggest to install the following packages through the package manager:

```
sudo apt-get update
sudo apt-get install build-essential python-numpy gcc gfortran python-dev libblas-dev␣
↪liblapack-dev cython pkg-config libfreetype6 libfreetype6-dev libpng-dev
```

Then, upgrade pip and install setuptools:

```
pip install --upgrade pip
pip install 'setuptools >=14.0'
```

Finally, install micca:

```
sudo pip install micca
```

1.2.2 On Mac OS X

In Mac OS X, we recommend to install Python from Homebrew:

1. Install Xcode;
2. Install Homebrew;
3. Make sure the environment variable `PATH` is properly setted in your `~/.bash_profile` or `~/.bashrc`:

   ```
   .. code-block:: sh
   ```

   export PATH=/usr/local/bin:$PATH

4. Install Python:

   ```
   brew update
   brew install python
   ```

Install the GNU Fortran and the NumPy package:

```
brew install gcc
pip install numpy
```

Finally, install micca:

```
sudo pip install micca
```

1.2.3 Installation problems

- BIOM fatal error: 'numpy/arrayobject.h'. If the installation process returns a message like this:

```
biom/_filter.c:258:10: fatal error: 'numpy/arrayobject.h' file not found
#include "numpy/arrayobject.h"
         ^
1 error generated.
error: command 'clang' failed with exit status 1
```

then you need to run:

```
pip install --global-option=build_ext --global-option="-I/usr/local/lib/python2.7/
↪site-packages/numpy/core/include/" biom-format
```

After that you can install the micca package.

1.3 Install micca from source

In order to install micca from sources (with the standard procedure `python setup.py install`), in addition to Python (>=2.7) and NumPy (>=1.8.0), the following Python packages must be installed:

- SciPy >=0.13.0
- Pandas >=0.17.0
- matplotlib >=1.3.0
- Biopython >=1.50
- cutadapt >=1.9
- biom-format >=1.3.1

The easiest way to install these packages is to is using pip:

```
sudo pip install 'scipy >=0.13.0' 'pandas >=0.17.0' 'matplotlib >=1.3.0' 'biopython >
↪= 1.50' 'cutadapt >=1.9' 'biom-format >=1.3.1'
```

Download the latest version from https://github.com/compmetagen/micca/releases and complete the installation:

```
tar -zxvf micca-X.Y.Z.tar.gz
sudo python setup.py install
```

1.3.1 If you don't have root access

Install micca in a local directory by specifying the `--prefix` argument. Then you need to set the environment variable `PYTHONPATH`:

```
python setup.py install --prefix=/path/to/modules
export PYTHONPATH=$PYTHONPATH:/path/to/modules/lib/python{version}/site-packages
```

Note: In order to export the variable permanently add the command at the bottom of your `~/.bash_profile` or `~/.bashrc` file.

1.4 Testing the installation

```
micca -h
```

1.5 Install RDP classifier (optional)

The RDP Classifier is a naive bayesian classifier for taxonomic assignments (http://sourceforge.net/projects/rdp-classifier/). The RDP classifier can be used in the *classify* command (option `-m/--method rdp`).

Warning: Only RDP Classifier version >2.8 is supported. Install the standard Java or Java compatible runtime (`sudo apt-get install default-jre` in Ubuntu/Debian or go to the Oracle Java homepage for OS X)

Download and unzip the file (RDP classifier 2.11 2015-09-14):

```
wget https://sourceforge.net/projects/rdp-classifier/files/rdp-classifier/rdp_
↪classifier_2.11.zip
unzip rdp_classifier_2.11.zip
```

Now you must set the environment variable `RDPPATH` by typing:

```
$ export RDPPATH=/path-to-rdp-classifier/rdp_classifier_2.11/
```

e.g. `export RDPPATH=/Users/David/rdp_classifier_2.11`.

Note: In order to export the variable permanently add the latest command at the bottom of your `.bashrc` file.

CHAPTER 2

Supported databases

2.1 16S rRNA

- Greengenes ftp://greengenes.microbio.me/greengenes_release/
- Greengenes *Core Set* (useful for NAST) http://greengenes.lbl.gov/Download/Sequence_Data/Fasta_data_files/core_set_aligned.fasta.imputed
- QIIME-formatted SILVA https://www.arb-silva.de/download/archive/qiime/

2.2 ITS

- UNITE https://unite.ut.ee/repository.php (QIIME releases)

Note: In the QIIME releases of the UNITE database the `_s` in the filename (e.g. `sh_qiime_release_s_DD.MM.YYY.zip`) specifies that the database includes singletons.

CHAPTER 3

Paired-end sequencing - 97% OTU

This tutorial describes a standard micca pipeline for the analysis of overlapping paired-end sequences. The main products of this pipeline are:

- the Operational Taxonomic Units (OTUs), defined clustering the processed sequences at a similarity threshold of 97%;
- an OTU table, containining the number of times each OTU is observed in each sample;
- a taxonomic classification for each OTU;
- an OTU phylogenetic tree.

This pipeline is intended for different platforms, such as **Illumina MiSeq** and **Illumina HiSeq**. Although this tutorial explains how to apply the pipeline to **16S rRNA** amplicons, it can be adapted to others markers gene/spacers, e.g. **Internal Transcribed Spacer (ITS)**, **18S** or **28S**.

Table of Contents

- *Dataset download*
- *Merge paired-end sequences*
  - *Compute reads statistics*
- *Primer trimming*
- *Quality filtering*
  - *Choosing parameters for filtering*
  - *Filter sequences*
- *OTU picking*
- *Assign taxonomy*
- *Infer the phylogenetic tree*

  - *Multiple Sequence Alignment (MSA)*
  - *Build the phylogenetic tree*
  - *Midpoint rooting*
- *Build the BIOM file*
- *Further steps*

3.1 Dataset download

The following paired-end 16S rRNA dataset contains **34 samples** from in FASTQ format (V3-V4 region, 341F 5'-CCTACGGGNGGCWGCAG-3', 806Rmod 5'-GACTACNVGGGTWTCTAATCC-3').

Samples comes from the paper "Diversity and Cyclical Seasonal Transitions in the Bacterial Community in a Large and Deep Perialpine Lake" were seasonal variations in the bacterioplankton community composition in the lake Garda were analized. Sampling was carried out at monthly intervals in three layers representative of the epilimnetic and euphotic zones of the lake, 1, 10, and 20 m. The dataset contains only a subset of the entire study (2014 samples only) and raw data were randomly subsampled at 3000 sequences per sample.

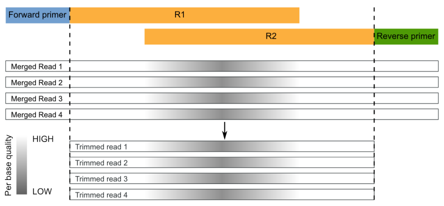
The 2x300-bp paired-end sequencing was carried out on an Illumina MiSeq.

Open a terminal, download the data and prepare the working directory:

```
wget ftp://ftp.fmach.it/metagenomics/micca/examples/garda.tar.gz
tar -zxvf garda.tar.gz
cd garda
```

3.2 Merge paired-end sequences

Overlapping paired-end sequences must be merged to obtain consensus sequences (sometimes called *assembly*). This operation can be performed with the *mergepairs* command.



Moreover, the command merges the samples in a single file where sample names are appended to the sequence identifier, as in *merge* and *split* commands.

Since the sequenced region is about of 465-bp (806-341) and the reads are of 300-bp, the overlap region is quite large ((2x300)-465=135 bp), as rule of thumb we set a minimum overlap length of 100 and maximum number of allowed mismatches of about 1/3, say 30:

```
micca mergepairs -i fastq/*_R1*.fastq -o merged.fastq -l 100 -d 30
```

Note: Starting from micca 1.6.0 staggered read pairs (staggered pairs are pairs where the 3' end of the reverse read has an overhang to the left of the 5' end of the forward read) will be merged by default. To override this feature (and therefore to discard staggered alignments) set the `-n/--nostagger` option.

Note: *mergepairs* works with FASTQ files only.

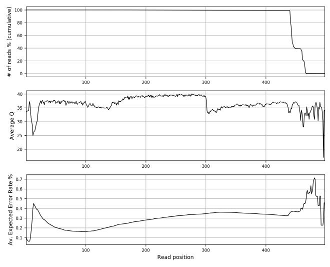
Note: Reverse file names will be constructed by replacing the string `_R1` in the forward file name with `_R2` (typical in Illumina file names, see options `-p/--pattern` and `-e/--repl`).

3.2.1 Compute reads statistics

We can report sequences statistics computed on the file `merged.fastq`. Run the command *stats*:

```
micca stats -i merged.fastq -o stats_merged
```

The command reports in 3 text files and in the relative plots (in PNG format) the length distribution, the Q score distribution and a quality summary. The quality summary plot (*stats_merged/stats_qualsumm_plot.png*) is reported below:



3.3 Primer trimming

Segments which match PCR primers should be now removed. For paired-end (already merged) reads, we recommend to **trim both forward and reverse primers and discard reads that do not contain the forward OR the reverse primer**.

These operations can be performed with the *trim* command:

```
micca trim -i merged.fastq -o trimmed.fastq -w CCTACGGGNGGCWGCAG -r␣
↪GACTACNVGGGTWTCTAATCC -W -R -c
```

The option `-W/--duforward` and `-R/--dureverse` ensures that reads that do not contain the forward or the reverse primer will be discarded. With the option `-c/--searchrc` the command searches reverse complement primers too.

3.4 Quality filtering

Producing high-quality OTUs requires high-quality reads. *filter* filters sequences according to the maximum allowed expected error (EE) rate %. We recommend values <=1%.

For paired-end reads, we recommend to merge pairs first, then quality filter using a maximum EE threshold with **no length truncation**.

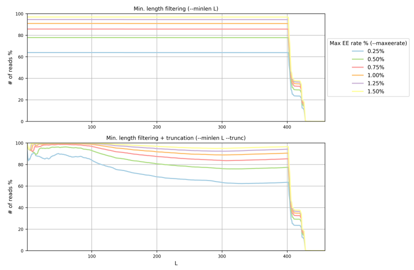
Warning: Parameters for the *filter* command should be chosen using the tool *filterstats*.

3.4.1 Choosing parameters for filtering

The command *filterstats* reports the fraction of reads that would pass for each specified maximum expected error (EE) rate %:

```
micca filterstats -i trimmed.fastq -o filterstats
```

Open the PNG file `filterstats/stats_plot.png`:



In this case (overlapping paired paired-end reads) we are interested in the plot on top (minimum length filtering only). A minimum read length (L) of **400** and a maximum error rate of **0.75%** seems to be a good compromise between the expected error rate and the number of reads remaining. Inspecting the file `filterstats/minlen_stats.txt`, you can see that more than **85%** reads will pass the filter:

```
L   0.25    0.5     0.75    1.0     1.25    1.5
...
399 63.856  77.766  85.664  90.844  94.484  96.853
400 63.856  77.765  85.661  90.842  94.481  96.850
401 63.842  77.747  85.643  90.822  94.459  96.827
...
```

Note: To obtain general sequencing statistics, run the micca command *stats* on the file `trimmed.fastq`.

3.4.2 Filter sequences

Now we can run the *filter* command with the selected parameters:

```
micca filter -i trimmed.fastq -o filtered.fasta -e 0.75 -m 400
```

Note: The maximum number of allowed Ns after truncation can be also specified in *filterstats* and in *filter*.

3.5 OTU picking

To characterize the taxonomic structure of the samples, the sequences are now organized into Operational Taxonomic Units (OTUs) at varying levels of identity. An identity of **97%** represent the common working definition of bacterial species. The *otu* command implements several state-of-the-art approaches for OTU clustering, but in this tutorial we will focus on the **de novo greedy clustering** (see *OTU picking and Denoising*):

```
micca otu -m denovo_greedy -i filtered.fasta -o denovo_greedy_otus -t 4 -c
```

The *otu* command returns several files in the output directory, including the **SV table** (`otutable.txt`) and a FASTA file containing the **representative sequences** (`otus.fasta`).

Note: See *OTU picking and Denoising* to see how to apply the **de novo swarm**, **closed-reference** and the **open-reference** OTU picking strategies to these data.

3.6 Assign taxonomy

Now we can assign taxonomy to each representative sequence using the *classify* command. In this tutorial we use the RDP (https://doi.org/10.1128/AEM.00062-07) classifier.

Note: See *Install* on how to install the RDP classifier on your system.

```
micca classify -m rdp -i denovo_greedy_otus/otus.fasta -o denovo_greedy_otus/taxa.txt
```

classify returns a taxonomy file like this:

```
DENOVO1     Bacteria;Cyanobacteria/Chloroplast;Cyanobacteria
DENOVO2     Bacteria;Cyanobacteria/Chloroplast;Cyanobacteria;Family II;Family II;GpIIa
DENOVO3     Bacteria;Chloroflexi;Anaerolineae;Anaerolineales;Anaerolineaceae
DENOVO4     Bacteria;Proteobacteria;Betaproteobacteria;Burkholderiales;Comamonadaceae;
↪Limnohabitans
...
```

3.7 Infer the phylogenetic tree

These steps are necessary if you want to use phylogenetic-based metrics such as the UniFrac distance (https://doi.org/10.1128/AEM.01996-06) in the downstream analysis.

3.7.1 Multiple Sequence Alignment (MSA)

The *msa* command provides two approaches for MSA: MUSCLE (https://doi.org/10.1093/nar/gkh340) (de novo alignment) and Nearest Alignment Space Termination (NAST) (https://doi.org/10.1093/nar/gkl244) (which uses a template alignment). In this tutorial we will use the NAST alignment method. For 16S rRNA sequences, a good template alignment is the Greengenes Core Set:

```
wget ftp://ftp.fmach.it/metagenomics/micca/dbs/core_set.tar.gz
tar -zxvf core_set.tar.gz
```

At this point we can run the *msa* command:

```
micca msa -m nast -i denovo_greedy_otus/otus.fasta -o denovo_greedy_otus/msa.fasta \
    --nast-template core_set_aligned.fasta.imputed --nast-threads 4
```

3.7.2 Build the phylogenetic tree

At this point we can build the phylogenetic tree from the MSA using *tree*:

```
micca tree -i denovo_greedy_otus/msa.fasta -o denovo_greedy_otus/tree.tree
```

Note: The output tree is in Newick format.

3.7.3 Midpoint rooting

UniFrac metrics require phylogenetic trees to be rooted. The tree can be rooted (in this case at midpoint between the two most distant tips of the tree) using the *root* command:

```
micca root -i denovo_greedy_otus/tree.tree -o denovo_greedy_otus/tree_rooted.tree
```

Note: Tree can also be rooted with the outgroup clade containing selected targets, see *root*.

3.8 Build the BIOM file

The Biological Observation Matrix (BIOM) is a common format for representing OTU tables and metadata and is the core data type for downstream analyses in QIIME and in phyloseq. *tobiom* converts the OTU table and the taxonomy table produced by the previous steps to the BIOM format. In addition, the *Sample data* can be added:

```
micca tobiom -i denovo_greedy_otus/otutable.txt -o denovo_greedy_otus/tables.biom \
    -t denovo_greedy_otus/taxa.txt -s sampledata.txt
```

3.9 Further steps

- *An introduction to the downstream analysis with R and phyloseq*
- *Compute basic statistics, rarefy and summarize OTU/SV tables using micca*

CHAPTER 4

Denoising (Illumina only)

Usually, amplicon sequences are clustered into **Operational Taxonomic Units** (OTUs) using a similarity threshold of 97%, which represents the common working definition of bacterial species.

Another approach consists to identify the **Sequence Variants** (SVs, see *OTU picking and Denoising* for details). This approach avoids clustering sequences at a predefined similarity threshold and usually includes a denoising algorithm in order to identify SVs.

In this tutorial we show how to perform the denoising of Illumina overlapping paired-end sequences in order to detect the SVs. Athough this tutorial explains how to apply the pipeline to 16S paired-end Illumina reads, it can be adapted to Illumina single-end sequening or to others markers gene/spacers, e.g. **Internal Transcribed Spacer (ITS)**, **18S** or **28S**.

Table of Contents

- *Data download and preprocessing*
- *Denoising - Sequence Variants identification*
- *Further steps*

4.1 Data download and preprocessing

In this tutorial we analyze the same dataset used in *Paired-end sequencing - 97% OTU*. Reads merging, primer trimming and quality filtering are the same as in *Paired-end sequencing - 97% OTU*:

```
wget ftp://ftp.fmach.it/metagenomics/micca/examples/garda.tar.gz
tar -zxvf garda.tar.gz
cd garda

micca mergepairs -i fastq/*_R1*.fastq -o merged.fastq -l 100 -d 30
```

(continues on next page)

(continued from previous page)

```
micca trim -i merged.fastq -o trimmed.fastq -w CCTACGGGNGGCWGCAG -r␣
↪GACTACNVGGGTWTCTAATCC -W -R -c
micca filter -i trimmed.fastq -o filtered.fasta -e 0.75 -m 400
```

4.2 Denoising - Sequence Variants identification

The *otu* command implements the UNOISE3 protocol (`denovo_unoise`) which includes dereplication, denoising and chimera filtering:

```
micca otu -m denovo_unoise -i filtered.fasta -o denovo_unoise_otus -t 4 -c
```

The *otu* command returns several files in the output directory, including the **SV table** (`otutable.txt`) and a FASTA file containing the **representative sequences** (`otus.fasta`).

Note: See *OTU picking and Denoising* to see how to apply the **de novo swarm**, **closed-reference** and the **open-reference** OTU picking strategies to these data.

4.3 Further steps

- *Assign taxonomy*
- *Infer the phylogenetic tree*
- *Build the BIOM file*
- *An introduction to the downstream analysis with R and phyloseq*
- *Compute basic statistics, rarefy and summarize OTU/SV tables using micca*

CHAPTER 5

Single-end sequencing

This tutorial describes a standard micca pipeline for the analysis of **single-end** amplicon data. This pipeline is intended for different platforms, such as **Roche 454**, **Illumina MiSeq/HiSeq** and **Ion Torrent**. Although this tutorial explains how to apply the pipeline to **16S rRNA** amplicons, it can be adapted to others markers gene/spacers, e.g. **Internal Transcribed Spacer (ITS)**, **18S** or **28S**.

Table of Contents

- *Dataset download*
- *Merge files*
- *Primer trimming*
- *Quality filtering*
  - *Choosing parameters for filtering*
  - *Filter sequences*
- *OTU picking*
- *Further steps*

5.1 Dataset download

The dataset used in this tutorial is taken from the Barelli et al. paper *Habitat fragmentation is associated to gut microbiota diversity of an endangered primate: implications for conservation* (https://doi.org/doi:10.1038/srep14862). The dataset contains only a subset of the entire study (Mwanihana samples only) for a total of **15 samples** (in FASTQ format) and **235179 16S rRNA amplicon reads** (V1-V3 hypervariable regions, 27F 5'-AGAGTTTGATCMTGGCTCAG, 533R 5'-TTACCGCGGCTGCTGGCAC). The 454 pyrosequencing was carried out on the GS FLX+ system using the XL+ chemistry.

Open a terminal, download the data and prepare the working directory:

```
wget ftp://ftp.fmach.it/metagenomics/micca/examples/mwanihana.tar.gz
tar -zxvf mwanihana.tar.gz
cd mwanihana
```

5.2 Merge files

Now the FASTQ files must be merged in a single file. This operation can be performed with the *merge* command. Sample names will be included into the sequence indentifiers.

```
micca merge -i fastq/*.fastq -o merged.fastq
```

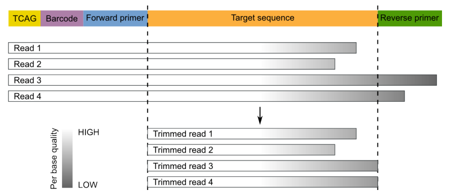
Note: The *merge* command works with FASTQ or FASTA files. If your sequences are in a different format (e.g. SFF or FASTA+QUAL) use *convert* to convert them.

Warning: In the case of multiplexed reads (with 5' barcode sequences) use *split* instead of *merge*. This command will perform demultiplexing and merging at the same time.

Note: In the case of overlapping paired-end reads go to *Paired-end sequencing - 97% OTU* or *Denoising (Illumina only)*.

5.3 Primer trimming

Segments which match PCR primers should be now removed. Typical Roche 454 reads start with a sequence key (e.g. TCAG) followed by the barcode (if it was not previously removed) and the forward primer. For these types of data (and in general, for single-end sequencing) we recommend to **trim both forward reverse primers and discard reads that do not contain the forward primer**. Moreover, sequence preceding (for the forward) or succeding (for the reverse, if found) primers should be removed:



These operations can be performed with the *trim* command:

```
micca trim -i merged.fastq -o trimmed.fastq -w AGAGTTTGATCMTGGCTCAG -r␣
↪GTGCCAGCAGCCGCGGTAA -W
```

The option `-W/--duforward` ensures that reads that do not contain the forward primer will be discarded.

Warning: Do not use the `-R/--dureverse` with single-end reads.

Note: The *trim* command supports IUPAC nucleotide codes and multiple primers. With the option `-c/--searchrc` the command searches reverse complement primers too. *trim* works with FASTQ or FASTA files.

5.4 Quality filtering

Producing high-quality OTUs requires high-quality reads. The *filter* command filters sequences according to the maximum allowed expected error (EE) rate %. We recommend values <=1%. Moreover, to obtain good results in clustering (see *otu*), reads should be **truncated at the same length** when they cover partial amplicons or if quality deteriorates towards the end (common when you have long amplicons in 454 or Illumina single-end sequencing).

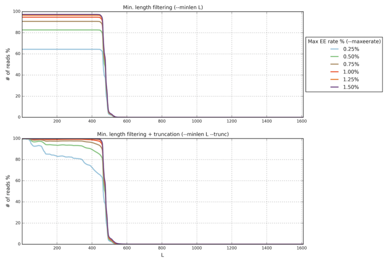
Warning: Parameters for the *filter* command should be chosen using the command *filterstats*.

5.4.1 Choosing parameters for filtering

The command *filterstats* reports the fraction of reads that would pass for each specified maximum expected error (EE) rate %:

```
micca filterstats -i trimmed.fastq -o filterstats
```

Open the PNG file `filterstats/stats_plot.png`:



In this case we are interested in the plot below (minimum length filtering + truncation). A truncation length of **350** and a maximum error rate of **0.5%** seems to be a good compromise between read read length, expected error rate and number of reads remaining. Inspecting the file `filterstats/trunclen_stats.txt`, you can see that more than **92%** reads will pass the filter:

```
L       0.25    0.5     0.75    1.0     1.25    1.5
...
349     78.905  92.472  97.425  99.135  99.705  99.897
350     78.639  92.385  97.389  99.126  99.704  99.896
351     78.369  92.300  97.357  99.116  99.700  99.892
...
```

Note: To obtain general sequencing statistics, run *stats*.

5.4.2 Filter sequences

Now we can run the *filter* command with the selected parameters:

```
micca filter -i trimmed.fastq -o filtered.fasta -e 0.5 -m 350 -t
```

Note: The maximum number of allowed Ns after truncation can be also specified in *filterstats* and in *filter*.

5.5 OTU picking

To characterize the taxonomic structure of the samples, the sequences are now organized into Operational Taxonomic Units (OTUs) at varying levels of identity. An identity of **97%** represent the common working definition of bacterial species. The *otu* command implements several state-of-the-art approaches for OTU clustering, but in this tutorial we will focus on the **de novo greedy clustering** (see *OTU picking and Denoising*):

```
micca otu -i filtered.fasta -o denovo_greedy_otus -d 0.97 -c -t 4
```

The *otu* command returns several files in the output directory, including the **OTU table** (`otutable.txt`) and a FASTA file containing the **representative sequences** (`otus.fasta`).

5.6 Further steps

- *Assign taxonomy*
- *Infer the phylogenetic tree*
- *Build the BIOM file*
- *An introduction to the downstream analysis with R and phyloseq*
- *Compute basic statistics, rarefy and summarize OTU/SV tables using micca*

CHAPTER 6

An introduction to the downstream analysis with R and phyloseq

Note: This tutorial requires *Paired-end sequencing - 97% OTU* to be done.

Note: This tutorial requires R, phyloseq ggplot2 and vegan (tested on R v3.4 and phyloseq v1.22.3) to be installed in your system.

6.1 Import data and preparation

We can import the micca processed data (the BIOM file, the phylogenetic tree and the representative sequences) into the R environment using the phyloseq library.

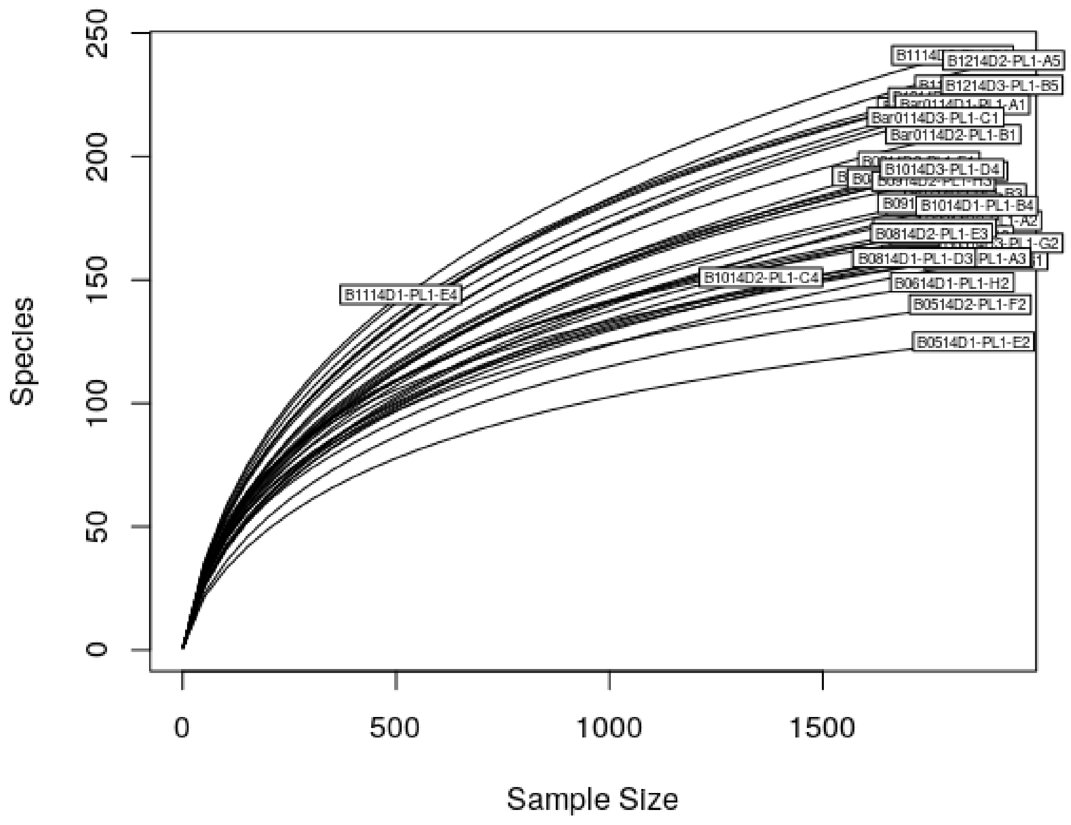
The `import_biom()` function allows to simultaneously import the BIOM file and an associated phylogenetic tree file and reference sequence file.

```
> library("phyloseq")
> library("ggplot2")
> library("vegan")
> setwd("denovo_greedy_otus") # set the working directory
> ps = import_biom("tables.biom", treefilename="tree_rooted.tree", refseqfilename=
↪"otus.fasta")
> sample_data(ps)$Month <- as.numeric(sample_data(ps)$Month)
> ps
phyloseq-class experiment-level object
otu_table()   OTU Table:         [ 529 taxa and 34 samples ]
sample_data() Sample Data:       [ 34 samples by 4 sample variables ]
tax_table()   Taxonomy Table:    [ 529 taxa by 6 taxonomic ranks ]
phy_tree()    Phylogenetic Tree: [ 529 tips and 528 internal nodes ]
refseq()      DNAStringSet:      [ 529 reference sequences ]
```

In this case, the phyloseq object includes the OTU table (which contains the OTU counts for each sample), the sample data matrix (containing the sample metadata), the taxonomy table (the predicted taxonomy for each OTU), the phylogenetic tree, and the OTU representative sequences.

Now this point we can plot the rarefaction curves using vegan:

```
> rarecurve(t(otu_table(ps)), step=50, cex=0.5)
```



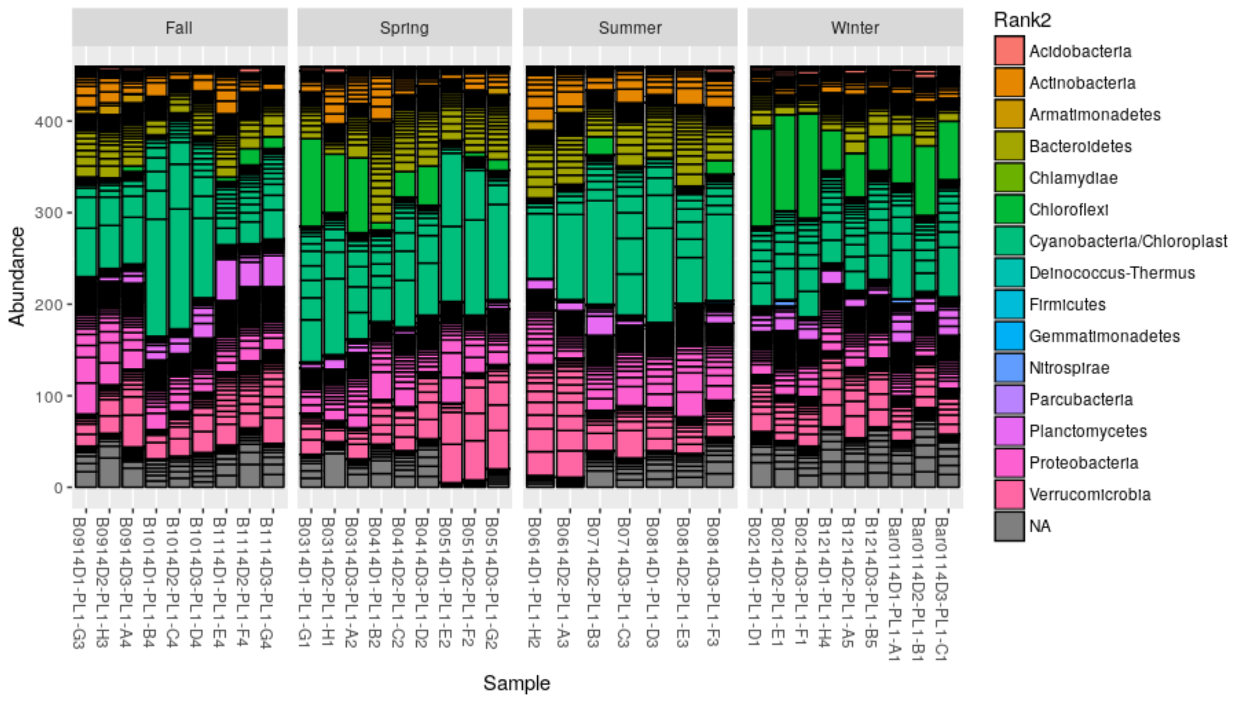
Now, we can rarefy in order to bring the samples to the same depth (in this case is the 90% of the abundance of the sample with less reads):

```
> # rarefy without replacement
> ps.rarefied = rarefy_even_depth(ps, rngseed=1, sample.size=0.9*min(sample_sums(ps)),
↪ replace=F)
```

6.2 Plot abundances

Plot the abundances and color each OTU according its classified phylum (Rank2):

```
> plot_bar(ps.rarefied, fill="Rank2") + facet_wrap(~Season, scales = "free_x", nrow =␣
↪1)
```



Alternatively, we can merge the OTU at the phylum level and build a new phyloseq object:

```
> ps.phylum = tax_glom(ps.rarefied, taxrank = "Rank2", NArm = F)
> ps.phylum
phyloseq-class experiment-level object
otu_table()    OTU Table:          [ 35 taxa and 34 samples ]
sample_data()  Sample Data:        [ 34 samples by 4 sample variables ]
tax_table()    Taxonomy Table:     [ 35 taxa by 6 taxonomic ranks ]
phy_tree()     Phylogenetic Tree:  [ 35 tips and 34 internal nodes ]
refseq()       DNAStringSet:       [ 35 reference sequences ]
```

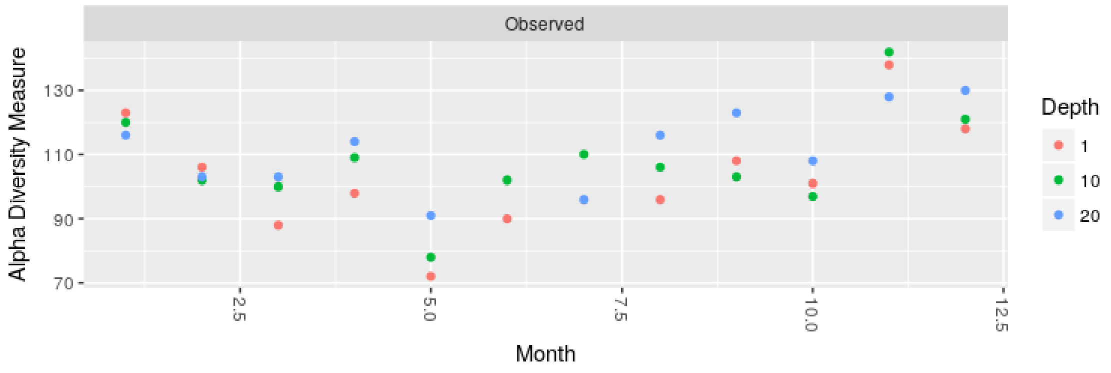
Now we can make the new bar plot at the class level:

```
> plot_bar(ps.phylum, fill="Rank2") + facet_wrap(~Season, scales = "free_x", nrow = 1)
```

6.3 Alpha diversity

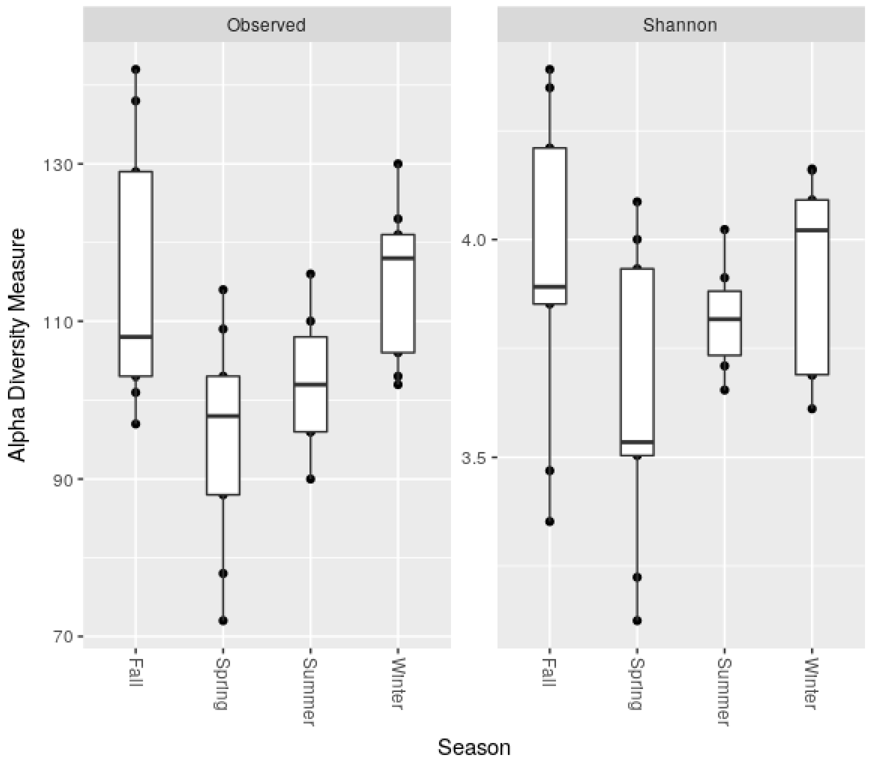
Now we can plot the number of observed OTUs in each month, coloring the values according to the sampling depth:

```
> plot_richness(ps.rarefied, x="Month", color="Depth", measures=c("Observed"))
```



Moreover, we can make a boxplot of the number of OTUs and the Shannon entropy grouping the different months by season:

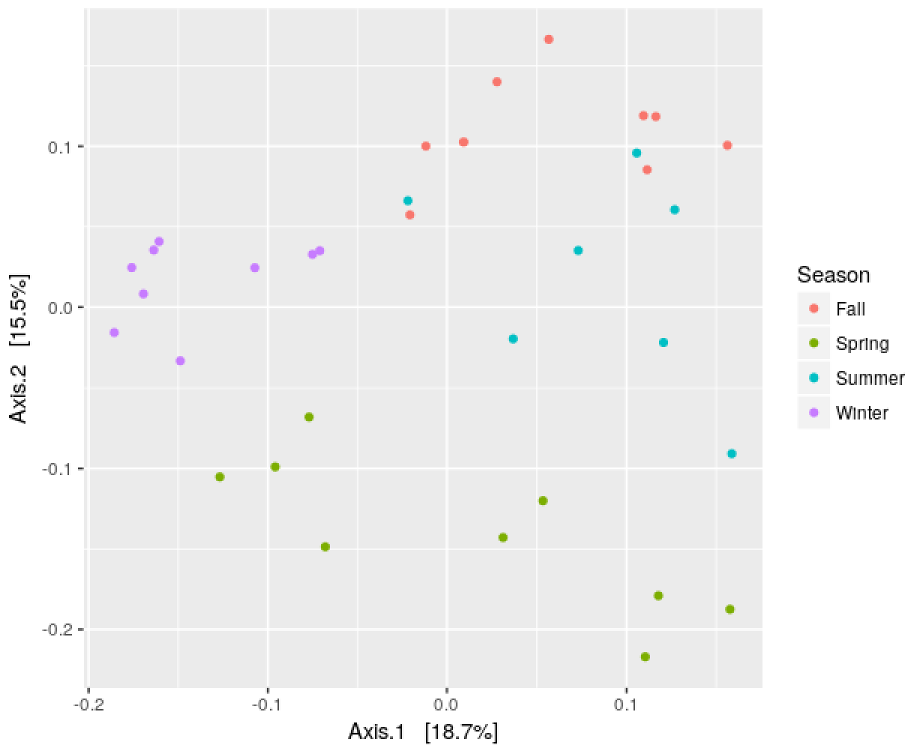
```
> plot_richness(ps.rarefied, x="Season", measures=c("Observed", "Shannon")) + geom_
↪boxplot()
```



6.4 Beta diversity

Now, we can plot the PCoA using the unweighted UniFrac as distance:

```
> # PCoA plot using the unweighted UniFrac as distance
> wunifrac_dist = distance(ps.rarefied, method="unifrac", weighted=F)
> ordination = ordinate(ps.rarefied, method="PCoA", distance=wunifrac_dist)
> plot_ordination(ps.rarefied, ordination, color="Season") + theme(aspect.ratio=1)
```



At this point, we test whether the seasons differ significantly from each other using the permutational ANOVA (PERMANOVA) analysis:

```
> adonis(wunifrac_dist~sample_data(ps.rarefied)$Season)

Call:
adonis(formula = wunifrac_dist ~ sample_data(ps.rarefied)$Season)

Permutation: free
Number of permutations: 999

Terms added sequentially (first to last)

                                Df SumsOfSqs  MeanSqs F.Model      R2 Pr(>F)
sample_data(ps.rarefied)$Season  3    0.6833 0.227765  4.3451 0.3029  0.001 ***
Residuals                       30    1.5726 0.052419         0.6971
Total                           33    2.2559                  1.0000
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

CHAPTER 7

Compute basic statistics, rarefy and summarize OTU/SV tables using micca

Note: This tutorial requires *Paired-end sequencing - 97% OTU* to be done.

The command *tablestats* reports a sample summary, an OTU summary and the rarefaction curves for the input OTU/SV table:

```
micca tablestats -i denovo_greedy_otus/otutable.txt -o tablestats
```

Inspecting the file `tablestats/tablestats_samplesumm.txt` you can see that the less abundant sample contains 512 reads:

```
Sample          Depth   NOTU    NSingle
B1114D1-PL1-E4  512     145     68
B1014D2-PL1-C4  1356    152     57
B0214D3-PL1-F1  1665    192     74
...             ...     ...     ...
```

Note: Rarefaction curves can be inspected through `tablestats/tablestats_rarecurve.txt` and `tablestats/tablestats_rarecurve_plot.png`.

To compare different samples, the OTU/SV table must be subsampled (rarefied) using the command *tablerare*. In this case we are interested in rarefy the table with the depth of the less abundant sample (`B1114D1-PL1-E4`):

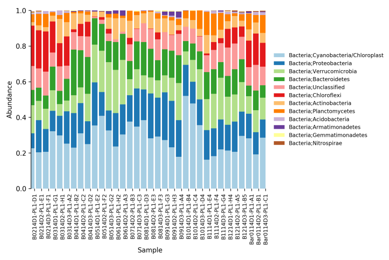
```
micca tablerare -i denovo_greedy_otus/otutable.txt -o denovo_greedy_otus/otutable_
↪rare.txt -d 500
```

Now we can summarize communities by their taxonomic composition. The *tabletotax* creates in the output directory a table for each taxonomic level (`taxtable1.txt`, ..., `taxtableN.txt`). OTU counts are summed together if they have the same taxonomy at the considered level.

```
micca tabletotax -i denovo_greedy_otus/otutable_rare.txt -t denovo_greedy_otus/taxa.
↪txt -o taxtables
```

Finally, we can generate a relative abundance bar plot from generated taxa tables, using the command *tablebar*. In this case only the bar plot relative to the taxonomy level 2 (phylum) will be generated:

```
micca tablebar -i taxtables/taxtable2.txt -o taxtables/taxtable2.png
```



CHAPTER 8

Picking OTUs for use in PICRUSt

PICRUSt (doi: 10.1038/nbt.2676) is a software designed to predict metagenome functional content from marker gene (e.g., 16S rRNA) surveys and full genomes. This tutorial covers how to pick OTUs from 16S rRNA sequences data to use with PICRUSt.

Note: Requires *Quality filtering* in *Single-end sequencing* to be done and the PICRUSt software to be installed in your system. Warning: PICRUSt 1.0.0 requires the biom-format package v1.3.1 to be installed in your system (from the command line run: `pip install biom-format==1.3.1`, for more information see http://biom-format.org/).

PICRUSt requires an *Closed-reference* OTU table computed against the Greengenes reference (clustered at 97% identity). Download the reference database (Greengenes, version 2013/05), clustered at 97% identity:

```
wget ftp://ftp.fmach.it/metagenomics/micca/dbs/gg_2013_05.tar.gz
tar -zxvf gg_2013_05.tar.gz
```

Run the micca closed-reference protocol:

```
micca otu -m closed_ref -i filtered.fasta -o closed_ref_otus -r 97_otus.fasta -d 0.97␣
↪-t 4
cd closed_ref_otus
```

Report the sample summary:

```
micca tablestats -i otutable.txt -o tablestats
head tablestats/tablestats_samplesumm.txt

Sample        Depth    NOTU    NSingle
Mw_03         1084     132     39
Mw_06         1387     122     27
Mw_11         1485     155     44
Mw_07         1528     150     36
Mw_01         1537     143     35
Mw_15         1565     144     35
```

(continues on next page)

(continued from previous page)

```
Mw_14          1610     149     42
Mw_02          1670     143     43
Mw_12          1710     153     54
```

Rarefy the OTU table for the PICRUSt analysis is always a good idea (see https://groups.google.com/forum/#!topic/picrust-users/ev5uZGUIPrQ), so we will rarefy the table at 1084 sequences per sample using *tablerare*:

```
micca tablerare -i otutable.txt -o otutable_rare.txt -d 1084
```

Convert the rarefied OTU table into the BIOM format replacing the OTU IDs with the original sequence IDs using the *tobiom* command:

```
micca tobiom -i otutable_rare.txt -o tables.biom -u otuids.txt
```

Normalize the OTU table by dividing each OTU by the known/predicted 16S copy number abundance using the PICRUSt script `normalize_by_copy_number.py`:

```
normalize_by_copy_number.py -i tables.biom -o normalized_otus.biom
```

Create the final metagenome functional predictions using the PICRUSt script `predict_metagenomes.py`:

```
predict_metagenomes.py -i normalized_otus.biom -o metagenome_predictions.biom
```

Now you can analyze the PICRUSt predicted metagenome as described in http://picrust.github.io/picrust/tutorials/downstream_analysis.html#downstream-analysis-guide.

CHAPTER 9

OTU picking and Denoising

Usually, amplicon sequences are clustered into Operational Taxonomic Units (OTUs) using a similarity threshold of **97%**, which represents the common working definition of bacterial species.

Another approach consists to define the so-called **Sequence Variants** (SVs, a.k.a Amplicon Sequence Variants - ASVs, Exact Sequence Variants ESVs, zero-radius OTUs - ZOTU, unique sequence variants or sub-OTUs). This approach avoids clustering sequences at a predefined similarity threshold and usually includes a denoising algorithm in order to identify SVs (see UNOISE, DADA2, Deblur, oligotyping and swarm).

The *otu* command assigns similar sequences (marker genes such as 16S rRNA and the fungal ITS region) to operational taxonomic units or sequence variants (OTUs or SVs).

- *Methods*
  - *De novo greedy*
  - *De novo UNOISE*
  - *Closed-reference*
  - *Open-reference*
  - *De novo swarm*
- *Definition of identity*
- *Output files*

9.1 Methods

9.1.1 De novo greedy

In denovo greedy clustering (parameter `--method denovo_greedy`), sequences are clustered without relying on an external reference database, using an approach similar to the UPARSE pipeline (https://doi.org/10.1038/nmeth.

2604) and tested in https://doi.org/10.7287/peerj.preprints.1466v1. This protocol includes in a single command dereplication, clustering and chimera filtering:

1. Dereplication. Predict sequence abundances of each sequence by dereplication, order by abundance and discard sequences with abundance value smaller than MINSIZE (option `-s/--minsize`, default value 2);
2. Greedy clustering. Distance (DGC) and abundance-based (AGC) strategies are supported (option `--greedy`, see https://doi.org/10.1186/s40168-015-0081-x and https://doi.org/10.7287/peerj.preprints.1466v1). Therefore, the candidate representative sequences are obtained;
3. Chimera filtering (optional). Remove chimeric sequences from the representatives performing a de novo chimera detection (option `--rmchim`);
4. Map sequences. Map sequences to the representatives.

Example:

```
micca otu -m denovo_greedy -i filtered.fasta -o denovo_greedy_otus -d 0.97 -c -t 4
```

9.1.2 De novo UNOISE

Denoise amplicon sequences using the UNOISE3 protocol. The method is designed for Illumina (paired or unpaired) reads. This protocol includes in a single command dereplication, denoising and chimera filtering:

1. Dereplication; Predict sequence abundances of each sequence by dereplication, order by abundance and discard sequences with abundance value smaller than MINSIZE (option `-s/--minsize`, default value 8);
2. Denoising;
3. Chimera filtering (optional);
4. Map sequences. Map sequences to the representatives.

Example:

```
micca otu -m denovo_unoise -i filtered.fasta -o denovo_unoise_otus -c -t 4
```

9.1.3 Closed-reference

Sequences are clustered against an external reference database and reads that could not be matched are discarded (method `closed_ref`). Example:

Download the reference database (Greengenes), clustered at 97% identity:

```
wget ftp://ftp.fmach.it/metagenomics/micca/dbs/gg_2013_05.tar.gz
tar -zxvf gg_2013_05.tar.gz
```

Run the closed-reference protocol:

```
micca otu -m closed_ref -i filtered.fasta -o closed_ref_otus -r 97_otus.fasta -d 0.97␣
↪-t 4
```

Simply perform a sequence ID matching with the reference taxonomy file (see *classify*):

```
cd closed_ref_otus
micca classify -m otuid -i otuids.txt -o taxa.txt -x ../97_otu_taxonomy.txt
```

9.1.4 Open-reference

Open-reference clustering (`open_ref`): sequences are clustered against an external reference database (as in *Closed-reference*) and reads that could not be matched are clustered with the *De novo greedy* protocol. Example:

Download the reference database (Greengenes), clustered at 97% identity:

```
wget ftp://ftp.fmach.it/metagenomics/micca/dbs/gg_2013_05.tar.gz
tar -zxvf gg_2013_05.tar.gz
```

Run the open-reference protocol:

```
micca otu -m open_ref -i filtered.fasta -o open_ref_otus -r 97_otus.fasta -d 0.97 -t␣
↪4 -c
```

Run the VSEARCH-based consensus classifier or the RDP classifier (see *classify*):

```
cd open_ref_otus
micca classify -m cons -i otus.fasta -o taxa.txt -r ../97_otus.fasta -x ../97_otu_
↪taxonomy.txt -t 4
```

9.1.5 De novo swarm

In denovo swarm clustering (doi: 10.7717/peerj.593, doi: 10.7717/peerj.1420, https://github.com/torognes/swarm, parameter `--method denovo_swarm`), sequences are clustered without relying on an external reference database. From https://github.com/torognes/swarm:

> The purpose of swarm is to provide a novel clustering algorithm that handles massive sets of amplicons. Results of traditional clustering algorithms are strongly input-order dependent, and rely on an arbitrary global clustering threshold. swarm results are resilient to input-order changes and rely on a small local linking threshold d, representing the maximum number of differences between two amplicons. swarm forms stable, high-resolution clusters, with a high yield of biological information.

otu includes in a single command dereplication, clustering and de novo chimera filtering:

1. Dereplication. Predict sequence abundances of each sequence by dereplication, order by abundance and discard sequences with abundance value smaller than MINSIZE (option `--minsize` default value is 1, i.e. no filtering);
2. Swarm clustering. Fastidious option is recommended (`--swarm-fastidious`);
3. Chimera filtering (optional).

Warning: Removing ambiguous nucleotides (`N`) (with the option `--maxns 0` in *filter*) is mandatory if you use the de novo swarm clustering method.

Example:

```
micca filter -i trimmed.fastq -o filtered.fasta -e 0.5 -m 350 -t --maxns 0
micca otu -m denovo_swarm -i filtered.fasta -o otus_denovo_swarm -c --minsize 1 --
↪swarm-fastidious -t 4
```

9.2 Definition of identity

In micca, the pairwise identity (except for 'de novo swarm' and 'denovo unoise') is defined as the edit distance excluding terminal gaps (same as in USEARCH and BLAST):

$$\frac{\#\text{ matching columns}}{\text{alignment length} - \text{terminal gaps}}$$

9.3 Output files

The *otu* command returns in a single directory 5 files:

otutable.txt TAB-delimited file, containing the number of times an OTU is found in each sample (OTU x sample, see *Supported file formats*):

```
OTU      Mw_01 Mw_02 Mw_03 ...
DENOVO1 151    178    177    ...
DENOVO2 339    181    142    ...
DENOVO3 533    305    63     ...
...      ...    ...    ...    ...
```

otus.fasta FASTA containing the representative sequences (OTUs):

```
>DENOVO1
GACGAACGCTGGCGGCGTGCCTAACACATGCAAGTCGAACGGGG...
>DENOVO2
GATGAACGCTAGCTACAGGCTTAACACATGCAAGTCGAGGGGCA...
>DENOVO3
AGTGAACGCTGGCGACGTGGTTAAGACATGCAAGTCGAGCGGTA...
...
```

otuids.txt TAB-delimited file which maps the OTU ids to original sequence ids:

```
DENOVO1 IS0AYJS04JQKIS;sample=Mw_01
DENOVO2 IS0AYJS04JL6RS;sample=Mw_01
DENOVO3 IS0AYJS04H4XNN;sample=Mw_01
...
```

hits.txt TAB-separated file, three-columns, where each column contains: the matching sequence, the representative (seed) and the identity (if available, see *Definition of identity*):

```
IS0AYJS04JE658;sample=Mw_01; IS0AYJS04I4XYN;sample=Mw_01 99.4
IS0AYJS04JPH34;sample=Mw_01; IS0AYJS04JVUBC;sample=Mw_01 98.0
IS0AYJS04I67XN;sample=Mw_01; IS0AYJS04JVUBC;sample=Mw_01 99.7
...
```

otuschim.fasta (only for 'denovo_greedy', 'denovo_swarm' and 'open_ref' mathods, when `-c/--rmchim` is specified) FASTA file containing the chimeric otus.

Warning: Trimming the sequences to a fixed position before clustering is strongly recommended when they cover partial amplicons or if quality deteriorates towards the end (common when you have long amplicons and single-end sequencing), see *Quality filtering*.

Note: De novo OTUs are renamed to `DENOVO[N]` and reference OTUs to `REF[N]`.

CHAPTER 10

Supported file formats

10.1 Sequence files

FASTA and FASTQ Sanger/Illumina 1.8+ format (phred+33) formats are supported. micca provides the *convert* command to convert between sequence file formats.

10.2 Taxonomy files

Taxonomy files map sequence IDs to taxonomy. Input taxonomy files must be TAB-delimited files where rows are either in the form:

1. `SEQID[TAB]k__Bacteria;p__Firmicutes;c__Clostridia;o__Clostridiales;f__;g__;`
2. `SEQID[TAB]Bacteria;Firmicutes;Clostridia;Clostridiales;;;`
3. `SEQID[TAB]Bacteria;Firmicutes;Clostridia;Clostridiales`
4. `SEQID[TAB]D_0__Bacteria;D_1__Firmicutes;D_2__Clostridia;D_3__Clostridiales;D_4__;D_5__;`

Compatible taxonomy files are:

- Greengenes (http://greengenes.secondgenome.com/downloads);
- QIIME-formatted SILVA (https://www.arb-silva.de/download/archive/qiime/);
- UNITE (https://unite.ut.ee/repository.php);
- Human Oral Microbiome Database (HOMD) (http://www.homd.org/).

The output taxonomy file returned by *classify* is a TAB-delimited file where each row is in the format:

```
SEQID[TAB]Bacteria;Firmicutes;Clostridia;Clostridiales
```

10.3 OTU/SV tables and taxonomy tables

The OTU table returned by *otu* is an OTU x sample, TAB-delimited text file, containing the number of times an OTU is found in each sample:

```
OTU      Mw_01 Mw_02 Mw_03 ...
DENOVO1 151   178   177   ...
DENOVO2 339   181   142   ...
DENOVO3 533   305   63    ...
DENOVO4 166   299   115   ...
...     ...   ...   ...   ...
```

The *tabletotax* command returns the "taxonomy tables" for each taxonomic level, e.g.:

```
OTU                                   Mw_01 Mw_02 Mw_03 ...
Bacteria;Bacteroidetes                1363  1543  1168  ...
Bacteria;Cyanobacteria/Chloroplast 0     0     0     ...
Bacteria;Firmicutes                   6257  5780  6761  ...
Bacteria;Lentisphaerae                0     1     0     ...
...                                   ...   ...   ...   ...
```

10.4 Sample data

The sample data file contains all of the information about the samples. In QIIME this file is called Mapping File. In micca, the sample data file must be a TAB-delimited text file (a row for each sample). The first column must be the sample identifier (assigned in *merge*, *split* or *mergepairs*):

```
ID    Group Altitude
Mw_01 Mw1   492
Mw_02 Mw1   492
Mw_09 Mw1   492
Mw_12 Mw1   492
...   ...   ...
```

10.5 Phylogenetic tree

Only the Newick format is supported.

10.6 BIOM file

The *tobiom* command generates OTU/SV tables in the biom version 1.0 JSON file format (http://biom-format.org/documentation/format_versions/biom-1.0.html).

CHAPTER 11

Changes

11.1 Version 1.7.0

- VSEARCH updated to version 2.7.1;
- Denoising protocol added to the otu command for Illumina reads (UNOISE);
- -S/–chim-abskew option added to micca otu command;
- Documentation updated;
- multithread option added to the mergepairs command.

11.2 Version 1.6.2 (bug fix release)

- Definitely fix the “new-line error” in classify;
- Fix bar plots in “stats”.

11.3 Version 1.6.1 (bug fix release)

- Fix the “new-line error” in classify.

11.4 Version 1.6.0

- swarm updated to version 2.1.12;
- Now the mergepairs command allows merging staggered reads by default. With the new option `-n/--nostagger` the command produces the same results of the previous versions (<=1.5);
- classify and tabletotax commands now strip the `D_X__` prefix from the Silva taxonomy files;

- Documentation updated;
- Fix: remove duplicate file closing in micca.api.merge().

11.5 Version 1.5.0

- Now the NAST algorithm trims candidate sequences to that which is bound by the beginning and end points of the alignment span; with the new option `--nast-notrim` in `micca msa` produces the same results of the previous version (<=1.4.0);
- Y-scale in `micca tablebar` when `--raw` fixed;
- "text file busy error" in `micca msa` (nast) fixed;
- pandas deprecation warnings fixed;
- documentation updated.

11.6 Version 1.4.0

- No-filter option added to `micca.api.msa.nast()` (do not remove positions which are gaps in every sequenceces) and to the `msa` command (`--nast-nofilter` option);
- documentation improved.

11.7 Version 1.3.0

- Swarm clustering algorithm added to `micca otu`;
- `micca.api.otu.denovo_swarm()` function added;
- micca v1.3.0 includes: VSEARCH v1.9.5, MUSCLE v3.8.31, FastTree v2.1.8, swarm v2.1.8;
- documentation updated.

11.8 Version 1.2.2

- Mow micca can generate plots with matplotlib when the `DISPLAY` environment variable is undefined;
- `MANIFEST.in`, Dockerfile updated.

11.9 Version 1.2.1

- Dockerfile added;
- Documentation improved.

11.10 Version 1.2.0

- Hits output file option added to `micca.api.msa.nast()` (`hits_fn` option) and to the `msa` command (`--nast-hits` option);
- setup.py improved.

11.11 Version 1.1.0

- Strand option added in `classify` (consensus-based classifier), `msa` (NAST) and `otu` (closed-reference and open reference OTU picking protolcols) commands. Now these commands search both strand (default) instead the plus strand only.

11.12 Version 1.0.0

- micca 1.0.0 includes: VSEARCH v1.9.5, MUSCLE v3.8.31, FastTree v2.1.8.

CHAPTER 12

classify

```
usage: micca classify [-h] -i FILE -o FILE [-m {cons,rdp,otuid}] [-r FILE]
                      [-x FILE] [--cons-id CONS_ID]
                      [--cons-maxhits CONS_MAXHITS]
                      [--cons-minfrac CONS_MINFRAC]
                      [--cons-mincov CONS_MINCOV] [--cons-strand {both,plus}]
                      [--cons-threads THREADS]
                      [--rdp-gene {16srrna,fungallsu,fungalits_warcup,fungalits_unite}]
                      [--rdp-maxmem GB] [--rdp-minconf RDP_MINCONF]

micca classify assigns taxonomy for each sequence in the input file
and provides three methods for classification:

* VSEARCH-based consensus classifier (cons): input sequences are
searched in the reference database with VSEARCH
(https://github.com/torognes/vsearch). For each query sequence the
method retrives up to 'cons-maxhits' hits (i.e. identity >=
'cons-id'). Then, the most specific taxonomic label that is
associated with at least 'cons-minfrac' of the hits is
assigned. The method is similar to the UCLUST-based consensus
taxonomy assigner presented in doi: 10.7287/peerj.preprints.934v2
and available in QIIME.

* RDP classifier (rdp): only RDP classifier version >= 2.8 is
supported (doi:10.1128/AEM.00062-07). In order to use this
classifier RDP must be installed (download at
http://sourceforge.net/projects/rdp-classifier/files/rdp-classifier/)
and the RDPPATH environment variable setted. The available
databases (--rdp-gene) are:

- 16S (16srrna)
- Fungal LSU (28S) (fungallsu)
- Warcup ITS (fungalits_warcup, doi: 10.3852/14-293)
- UNITE ITS (fungalits_unite)
```

(continues on next page)

(continued from previous page)

```
For more information about the RDP classifier go to
http://rdp.cme.msu.edu/classifier/classifier.jsp

* OTU ID classifier (otuid): simply perform a sequence ID matching
with the reference taxonomy file. Recommended strategy when the
closed reference clustering (--method closedref in micca-otu) was
performed. OTU ID classifier requires a tab-delimited file where
the first column contains the current OTU ids and the second column
the reference taxonomy ids (see otuids.txt in micca-otu), e.g.:

REF1[TAB]1110191
REF2[TAB]1104777
REF3[TAB]1078527
...

The input reference taxonomy file (--ref-tax) should be a
tab-delimited file where rows are either in the form:

1. SEQID[TAB]k__Bacteria;p__Firmicutes;c__Clostridia;o__Clostridiales;f__;g__;
2. SEQID[TAB]Bacteria;Firmicutes;Clostridia;Clostridiales;;;
3. SEQID[TAB]Bacteria;Firmicutes;Clostridia;Clostridiales
4. SEQID[TAB]D_0__Bacteria;D_1__Firmicutes;D_2__Clostridia;D_3__Clostridiales;D_4__;D_
↪5__;

Compatible reference database are Greengenes
(http://greengenes.secondgenome.com/downloads), QIIME-formatted SILVA
(https://www.arb-silva.de/download/archive/qiime/) and UNITE
(https://unite.ut.ee/repository.php).

The output file is a tab-delimited file where each row is in the
format:

SEQID[TAB]Bacteria;Firmicutes;Clostridia;Clostridiales

optional arguments:
-h, --help            show this help message and exit

arguments:
-i FILE, --input FILE
                        input FASTA file (for 'cons' and 'rdp') or a tab-
                        delimited OTU ids file (for 'otuid') (required).
-o FILE, --output FILE
                        output taxonomy file (required).
-m {cons,rdp,otuid}, --method {cons,rdp,otuid}
                        classification method (default cons)
-r FILE, --ref FILE   reference sequences in FASTA format, required for
                        'cons' classifier.
-x FILE, --ref-tax FILE
                        tab-separated reference taxonomy file, required for
                        'cons' and 'otuid' classifiers.

VSEARCH-based consensus classifierspecific options:
--cons-id CONS_ID     sequence identity threshold (0.0 to 1.0, default 0.9).
--cons-maxhits CONS_MAXHITS
                        number of hits to consider (>=1, default 3).
--cons-minfrac CONS_MINFRAC
                        for each taxonomic rank, a specific taxa will be
```

(continues on next page)

(continued from previous page)

```
                        assigned if it is present in at least MINFRAC of the
                        hits (0.0 to 1.0, default 0.5).
--cons-mincov CONS_MINCOV
                        reject sequence if the fraction of alignment to the
                        reference sequence is lower than MINCOV. This
                        parameter prevents low-coverage alignments at the end
                        of the sequences (default 0.75).
--cons-strand {both,plus}
                        search both strands or the plus strand only (default
                        both).
--cons-threads THREADS
                        number of threads to use (1 to 256, default 1).

RDP Classifier/Database specific options:
--rdp-gene {16srrna,fungallsu,fungalits_warcup,fungalits_unite}
                        marker gene/region
--rdp-maxmem GB       maximum memory size for the java virtual machine in GB
                        (default 2)
--rdp-minconf RDP_MINCONF
                        minimum confidence value to assign taxonomy to a
                        sequence (default 0.8)

Examples

Classification of 16S sequences using the consensus classifier and
Greengenes:

    micca classify -m cons -i input.fasta -o tax.txt \
    --ref greengenes_2013_05/rep_set/97_otus.fasta \
    --ref-tax greengenes_2013_05/taxonomy/97_otu_taxonomy.txt

Classification of ITS sequences using the RDP classifier and the
UNITE database:

    micca classify -m rdp --rdp-gene fungalits_unite -i input.fasta \
    -o tax.txt

OTU ID matching after the closed reference OTU picking protocol:

    micca classify -m otuid -i otuids.txt -o tax.txt \
    --ref-tax greengenes_2013_05/taxonomy/97_otu_taxonomy.txt
```

CHAPTER 13

convert

```
usage: micca convert [-h] -i FILE -o FILE [-q FILE] [-d DEFAULTQ]
                     [-f INPUT_FORMAT] [-F OUTPUT_FORMAT]

micca convert converts between sequence file formats. See
http://biopython.org/wiki/SeqIO#File_Formats for a comprehnsive list
of the supported file formats.

Supported input formats:
abi, abi-trim, ace, embl, embl-cds, fasta, fasta-qual, fastq, fastq-illumina,
fastq-sanger, fastq-solexa, gb, genbank, genbank-cds, ig, imgt, pdb-atom,
pdb-seqres, phd, pir, qual, seqxml, sff, sff-trim, swiss, tab, uniprot-xml

Supported output formats:
embl, fasta, fastq, fastq-illumina, fastq-sanger, fastq-solexa, gb, genbank,
imgt, phd, qual, seqxml, sff, tab

optional arguments:
-h, --help            show this help message and exit

arguments:
-i FILE, --input FILE
                        input sequence file (required).
-o FILE, --output FILE
                        output sequence file (required).
-q FILE, --qual FILE  input quality file (required for 'fasta-qual' input
                        format.
-d DEFAULTQ, --defaultq DEFAULTQ
                        default phred quality score for format-without-quality
                        to format-with-quality conversion (default 40).
-f INPUT_FORMAT, --input-format INPUT_FORMAT
                        input file format (default fastq).
-F OUTPUT_FORMAT, --output-format OUTPUT_FORMAT
                        input file format (default fasta).
```

(continues on next page)

(continued from previous page)

```
Examples

Convert FASTA+QUAL files into a FASTQ (Sanger/Illumina 1.8+) file:

    micca convert -i input.fasta -q input.qual -o output.fastq \
    -f fasta-qual -F fastq

Convert a SFF file into a FASTQ (Sanger/Illumina 1.8+) file:

    micca convert -i input.sff -o output.fastq -f sff -F fastq
```

CHAPTER 14

filter

```
usage: micca filter [-h] -i FILE -o FILE [-e MAXEERATE] [-m MINLEN] [-t]
                    [-n MAXNS] [-f {fastq,fasta}]

micca filter filters sequences according to the maximum allowed
expected error (EE) rate %%. Optionally, you can:

* discard sequences that are shorter than the specified length
(suggested for Illumina overlapping paired-end (already merged)
reads) (option --minlen MINLEN);

* discard sequences that are shorter than the specified length AND
truncate sequences that are longer (suggested for Illumina and 454
unpaired reads) (options --minlen MINLEN --trunc);

* discard sequences that contain more than a specified number of Ns
(--maxns).

Sequences are first shortened and then filtered. Overlapping paired
reads with should be merged first (using micca-mergepairs) and then
filtered.

The expected error (EE) rate %% in a sequence of length L is defined
as (doi: 10.1093/bioinformatics/btv401):

                sum(error probabilities)
    EE rate %% = ------------------------ * 100
                           L

Before filtering, run 'micca filterstats' to see how many reads will
pass the filter at different minimum lengths with or without
truncation, given a maximum allowed expected error rate %% and maximum
allowed number of Ns.

micca-filter is based on VSEARCH (https://github.com/torognes/vsearch).
```

(continues on next page)

(continued from previous page)

```
optional arguments:
-h, --help            show this help message and exit

arguments:
-i FILE, --input FILE
                        input FASTQ file, Sanger/Illumina 1.8+ format
                        (phred+33) (required).
-o FILE, --output FILE
                        output FASTA/FASTQ file (required).
-e MAXEERATE, --maxeerate MAXEERATE
                        discard sequences with more than the specified expeced
                        error rate % (values <=1%, i.e. less or equal than one
                        error per 100 bases, are highly recommended).
                        Sequences are discarded after truncation (if enabled)
                        (default 1).
-m MINLEN, --minlen MINLEN
                        discard sequences that are shorter than MINLEN
                        (default 1).
-t, --trunc           truncate sequences that are longer than MINLEN
                        (disabled by default).
-n MAXNS, --maxns MAXNS
                        discard sequences with more than the specified number
                        of Ns. Sequences are discarded after truncation
                        (disabled by default).
-f {fastq,fasta}, --output-format {fastq,fasta}
                        file format (default fasta).

Examples

Truncate reads at 300 bp, discard low quality sequences
(with EE rate > 0.5%%) and write a FASTA file:

    micca filter -i reads.fastq -o filtered.fasta -m 300 -t -e 0.5
```

CHAPTER 15

filterstats

```
usage: micca filterstats [-h] -i FILE [-o DIR] [-t TOPN]
                        [-e MAXEERATES [MAXEERATES ...]] [-n MAXNS]

micca filterstats reports the fraction of reads that would pass for each
specified maximum expected error (EE) rate %% and the maximum number of
allowed Ns after:

* discarding sequences that are shorter than the specified length
(suggested for Illumina overlapping paired-end (already merged)
reads);

* discarding sequences that are shorter than the specified length AND
truncating sequences that are longer (suggested for Illumina and 454
unpaired reads);

Parameters for the 'micca filter' command should be chosen for each
sequencing run using this tool.

micca filterstats returns in the output directory 3 files:

* filterstats_minlen.txt: fraction of reads that would pass the filter after
the minimum length filtering;
* filterstats_trunclen.txt: fraction of reads that would pass the filter after
the minimum length filtering + truncation;
* filterstats_plot.png: plot in PNG format.

optional arguments:
-h, --help            show this help message and exit

arguments:
-i FILE, --input FILE
                        input FASTQ file, Sanger/Illumina 1.8+ format
                        (phred+33) (required).
-o DIR, --output DIR  output directory (default .).
```

(continues on next page)

(continued from previous page)

```
-t TOPN, --topn TOPN  perform statistics on the first TOPN sequences
                        (disabled by default)
-e MAXEERATES [MAXEERATES ...], --maxeerates MAXEERATES [MAXEERATES ...]
                        max expected error rates (%). (default [0.25, 0.5,
                        0.75, 1, 1.25, 1.5])
-n MAXNS, --maxns MAXNS
                        max number of Ns. (disabled by default).

Examples

Compute filter statistics on the top 10000 sequences, predicting
the fraction of reads that would pass for each maximum EE error
rate (default values):

    micca filterstats -i input.fastq -o stats -t 10000
```

CHAPTER 16

merge

```
usage: micca merge [-h] -i FILE [FILE ...] -o FILE [-s SEP] [-f {fastq,fasta}]

micca merge merges several FASTQ or FASTA files in a single file.
Different samples will be merged in a single file and sample names
will be appended to the sequence identifier
(e.g. >SEQID;sample=SAMPLENAME). Sample names are defined as the
leftmost part of the file name splitted by the first occurence of '.'
(-s/--sep option). Whitespace characters in names will be replaced
with a single character underscore ('_').

optional arguments:
-h, --help            show this help message and exit

arguments:
-i FILE [FILE ...], --input FILE [FILE ...]
                        input FASTQ/FASTA file(s) (required).
-o FILE, --output FILE
                        output FASTQ/FASTA file (required).
-s SEP, --sep SEP     Sample names are defined as the leftmost part of the
                        file name splitted by the first occurence of 'SEP'
                        (default .)
-f {fastq,fasta}, --format {fastq,fasta}
                        file format (default fastq).

Examples

Merge files in FASTA format:

    micca merge -i in1.fasta in2.fasta in3.fasta -o merged.fasta \
    -f fasta
```

CHAPTER 17

mergepairs

```
usage: micca mergepairs [-h] -i FILE [FILE ...] -o FILE [-r FILE]
                        [-l MINOVLEN] [-d MAXDIFFS] [-p PATTERN] [-e REPL]
                        [-s SEP] [-n] [--notmerged-fwd FILE]
                        [--notmerged-rev FILE] [-t THREADS]

micca mergepairs merges paired-end sequence reads into one sequence.

A single merging of a pair of FASTQ files can be simply performed
using both -i/--input and -r/--reverse options.

When the option -r/--reverse is not specified:

1. you can indicate several forward files (with the option -i/--input);

2. the reverse file name will be constructed by replacing the string
'_R1' in the forward file name with '_R2' (typical in Illumina
file names, see options -p/--pattern and -e/--repl);

3. after the merging of the paired reads, different samples will be
merged in a single file and sample names will be appended to the
sequence identifier (e.g. >SEQID;sample=SAMPLENAME), as in 'micca
merge' and 'micca split'. Sample names are defined as the leftmost
part of the file name splitted by the first occurence of '_'
(-s/--sep option). Whitespace characters in names will be replaced
with a single character underscore ('_').

micca mergepairs wraps VSEARCH (https://github.com/torognes/vsearch).
Statistical testing of significance is performed in a way similar to
PEAR (doi: 10.1093/bioinformatics/btt593). The quality of merged bases
is computed as in USEARCH (doi: 10.1093/bioinformatics/btv401).

By default staggered read pairs (staggered pairs are pairs where the 3'
end of the reverse read has an overhang to the left of the 5’ end
of the forward read) will be merged. To override this feature (and
```

(continues on next page)

(continued from previous page)

```
therefore to discard staggered alignments) set the -n/--nostagger
option.

optional arguments:
-h, --help            show this help message and exit

arguments:
-i FILE [FILE ...], --input FILE [FILE ...]
                        forward FASTQ file(s), Sanger/Illumina 1.8+ format
                        (phred+33) (required).
-o FILE, --output FILE
                        output FASTQ file (required).
-r FILE, --reverse FILE
                        reverse FASTQ file, Sanger/Illumina 1.8+ format
                        (phred+33).
-l MINOVLEN, --minovlen MINOVLEN
                        minimum overlap length (default 32).
-d MAXDIFFS, --maxdiffs MAXDIFFS
                        maximum number of allowed mismatches in the overlap
                        region (default 8).
-p PATTERN, --pattern PATTERN
                        when the reverse filename is not specified, it will be
                        constructed by replacing 'PATTERN' in the forward file
                        name with 'REPL' (default _R1).
-e REPL, --repl REPL  when the reverse filename is not specified, it will be
                        constructed by replacing 'PATTERN' in the forward file
                        name with 'REPL' (default _R2).
-s SEP, --sep SEP     when the reverse file name is not specified, sample
                        names are appended to the sequence identifier (e.g.
                        >SEQID;sample=SAMPLENAME). Sample names are defined as
                        the leftmost part of the file name splitted by the
                        first occurence of 'SEP' (default _)
-n, --nostagger       forbid the merging of staggered read pairs. Without
                        this option the command will merge staggered read
                        pairs and the 3' overhang of the reverse read will be
                        not included in the merged sequence.
--notmerged-fwd FILE  write not merged forward reads.
--notmerged-rev FILE  write not merged reverse reads.
-t THREADS, --threads THREADS
                        number of threads to use (1 to 256, default 1).

Examples

Merge reads with a minimum overlap length of 50 and maximum number
of allowed mismatches of 3:

    micca mergepairs -i reads1.fastq -r reads2.fastq -o merged.fastq \
    -l 50 -d 3

Merge several illumina paired reads (typically named *_R1*.fastq and
*_R2*.fastq):

    micca mergepairs -i *_R1*.fastq -o merged.fastq --notmerged-fwd \
    notmerged_fwd.fastq --notmerged-rev notmerged_rev.fastq
```

CHAPTER 18

msa

```
usage: micca msa [-h] -i FILE -o FILE [-m {muscle,nast}]
                [--muscle-maxiters MUSCLE_MAXITERS] [--nast-template FILE]
                [--nast-id NAST_ID] [--nast-threads NAST_THREADS]
                [--nast-mincov NAST_MINCOV] [--nast-strand {both,plus}]
                [--nast-notaligned FILE] [--nast-hits FILE] [--nast-nofilter]
                [--nast-notrim]

micca msa performs a multiple sequence alignment (MSA) on the input
file in FASTA format. micca msa provides two approaches for MSA:

* MUSCLE (doi: 10.1093/nar/gkh340). It is one of the most widely-used
multiple sequence alignment software;

* Nearest Alignment Space Termination (NAST) (doi:
10.1093/nar/gkl244). MICCA provides a very fast and memory
efficient implementation of the NAST algorithm. The algorithm is
based on VSEARCH (https://github.com/torognes/vsearch). It requires
a pre-aligned database of sequences (--nast-template). For 16S
data, a good template file is the Greengenes Core Set
(http://greengenes.lbl.gov/Download/Sequence_Data/Fasta_data_files/
core_set_aligned.fasta.imputed).

optional arguments:
-h, --help            show this help message and exit

arguments:
-i FILE, --input FILE
                        input FASTA file (required).
-o FILE, --output FILE
                        output MSA file in FASTA format (required).
-m {muscle,nast}, --method {muscle,nast}
                        multiple sequence alignment method (default muscle).

MUSCLE specific options:
```

(continues on next page)

(continued from previous page)

```
--muscle-maxiters MUSCLE_MAXITERS
                        maximum number of MUSCLE iterations. Set to 2 for a
                        good compromise between speed and accuracy (>=1
                        default 16).

NAST specific options:
--nast-template FILE  multiple sequence alignment template file in FASTA
                        format.
--nast-id NAST_ID     sequence identity threshold to consider a sequence a
                        match (0.0 to 1.0, default 0.75).
--nast-threads NAST_THREADS
                        number of threads to use (1 to 256, default 1).
--nast-mincov NAST_MINCOV
                        reject sequence if the fraction of alignment to the
                        template sequence is lower than MINCOV. This parameter
                        prevents low-coverage alignments at the end of the
                        sequences (default 0.75).
--nast-strand {both,plus}
                        search both strands or the plus strand only (default
                        both).
--nast-notaligned FILE
                        write not aligned sequences in FASTA format.
--nast-hits FILE      write hits on a TAB delimited file with the query
                        sequence id, the template sequence id and the
                        identity.
--nast-nofilter       do not remove positions which are gaps in every
                        sequenceces (useful if you want to apply a Lane mask
                        filter before the tree inference).
--nast-notrim         force to align the entire candidate sequence (i.e. do
                        not trim the candidate sequence to that which is bound
                        by the beginning and end points of of the alignment
                        span

Examples

De novo MSA using MUSCLE:

    micca msa -i input.fasta -o msa.fasta

Template-based MSA using NAST, the Greengenes alignment as
template (clustered at 97% similarity) 4 threads and a sequence
identity threshold of 75%:

    micca msa -i input.fasta -o msa.fasta -m nast --nast-threads 4 \
    --nast-template greengenes_2013_05/rep_set_aligned/97_otus.fasta
```

CHAPTER 19

otu

See *OTU picking and Denoising* for details.

```
usage: micca otu [-h] -i FILE [-o DIR] [-r FILE]
                 [-m {denovo_greedy,denovo_unoise,denovo_swarm,closed_ref,open_ref}]
                 [-d ID] [-n MINCOV] [-t THREADS] [-g {dgc,agc}] [-s MINSIZE]
                 [-a {both,plus}] [-c] [-S CHIM_ABSKEW]
                 [--swarm-differences SWARM_DIFFERENCES] [--swarm-fastidious]
                 [--unoise-alpha UNOISE_ALPHA]

micca otu assigns similar sequences (marker genes such as 16S rRNA and
the fungal ITS region) to operational taxonomic units (OTUs) or sequence
variants (SVs).

Trimming the sequences to a fixed position before clustering is
*strongly recommended* when they cover partial amplicons or if quality
deteriorates towards the end (common when you have long amplicons and
single-end sequencing).

Removing ambiguous nucleotides 'N' (with the option --maxns 0 in micca
filter) is mandatory if you use the de novo swarm clustering method.

micca otu provides the following protocols:

* de novo greedy clustering (denovo_greedy): useful for for the
identification of 97% OTUs;

* de novo unoise (denovo_unoise): denoise Illumina sequences using
the UNOISE3 protocol;

* de novo swarm (denovo_swarm): a robust and fast clustering method
(deprecated, it will be removed in version 1.8.0);

* closed-reference clustering (closed_ref): sequences are clustered
against an external reference database and reads that could not be
```

(continues on next page)

(continued from previous page)

```
matched are discarded.

* open-reference clustering (open_ref): sequences are clustered
against an external reference database and reads that could not be
matched are clustered with the 'de novo greedy' protocol.

Outputs:

* otutable.txt: OTU x sample, TAB-separated OTU table file,
containing the number of times an OTU is found in each sample.

* otus.fasta: FASTA file containing the representative sequences (OTUs);

* otuids.txt: OTU ids to original sequence ids (tab-delimited text
file);

* hits.txt: three-columns, TAB-separated file with matching sequence,
representative (seed) and identity (if available, else '*');

* otuschim.fasta (only for 'denovo_greedy', 'denovo_swarm' and
'open_ref' when --rmchim is specified): FASTA file containing the
chimeric otus.

optional arguments:
-h, --help            show this help message and exit

arguments:
-i FILE, --input FILE
                        input fasta file (required).
-o DIR, --output DIR  output directory (default .).
-r FILE, --ref FILE   reference sequences in fasta format, required for
                        'closed_ref' and 'open_ref' clustering methods.
-m {denovo_greedy,denovo_unoise,denovo_swarm,closed_ref,open_ref}, --method {denovo_
↪greedy,denovo_unoise,denovo_swarm,closed_ref,open_ref}
                        clustering method (default denovo_greedy)
-d ID, --id ID        sequence identity threshold (for 'denovo_greedy',
                        'closed_ref' and 'open_ref', 0.0 to 1.0, default
                        0.97).
-n MINCOV, --mincov MINCOV
                        reject sequence if the fraction of alignment to the
                        reference sequence is lower than MINCOV (for
                        'closed_ref' and 'open_ref' clustering methods,
                        default 0.75).
-t THREADS, --threads THREADS
                        number of threads to use (1 to 256, default 1).
-g {dgc,agc}, --greedy {dgc,agc}
                        greedy clustering strategy, distance (DGC) or
                        abundance-based (AGC) (for 'denovo_greedy' and
                        'open_ref' clustering methods) (default dgc).
-s MINSIZE, --minsize MINSIZE
                        discard sequences with an abundance value smaller than
                        MINSIZE after dereplication (>=1, default values are 2
                        for 'denovo_greedy' and 'open_ref', 1 for
                        'denovo_swarm' and 8 for 'denovo_unoise').
-a {both,plus}, --strand {both,plus}
                        search both strands or the plus strand only (for
                        'closed_ref' and 'open_ref' clustering methods,
```

(continues on next page)

(continued from previous page)

```
                        default both).

Chimera removal specific options:
-c, --rmchim          remove chimeric sequences (ignored in method
                        'closed_ref'
-S CHIM_ABSKEW, --chim-abskew CHIM_ABSKEW
                        abundance skew. It is used to distinguish in a three-
                        way alignment which sequence is the chimera and which
                        are the parents. If CHIM_ABSKEW=2.0, the parents
                        should be at least 2 times more abundant than their
                        chimera (defaults values are 16.0 for 'denovo_unoise',
                        2.0 otherwise).

Swarm specific options:
--swarm-differences SWARM_DIFFERENCES
                        maximum number of differences allowed between two
                        amplicons. Commonly used d values are 1 (linear
                        complexity algorithm), 2 or 3, rarely higher. (>=0,
                        default 1).
--swarm-fastidious    when working with SWARM_DIFFERENCES=1, perform a
                        second clustering pass to reduce the number of small
                        OTUs (recommended option).

UNOISE specific options:
--unoise-alpha UNOISE_ALPHA
                        specify the alpha parameter (default 2.0).

Examples

De novo clustering with a 97% similarity threshold and remove
chimeric OTUs:

    micca otu -i input.fasta --method denovo_greedy --id 0.97 -c

Open-reference OTU picking protocol with a 97% similarity
threshold, without removing chimeras in the de novo protocol step
and using 8 threads:

    micca otu -i input.fasta --method open_ref --threads 8 --id 0.97 \
    --ref greengenes_2013_05/rep_set/97_otus.fasta

De novo swarm clustering with the protocol using 4 threads:

    micca otu -i input.fasta --method denovo_swarm --threads 4 \
    --swarm-fastidious --rmchim --minsize 1
```

CHAPTER 20

root

```
usage: micca root [-h] -i FILE -o FILE [-m {midpoint,outgroup}]
                [-t TARGETS [TARGETS ...]]

micca root reroot the input tree:

* at the calculated midpoint between the two most distant tips of the
tree (--method midpoint);

* with the outgroup clade containing the given taxa (leaf nodes),
i.e. the common ancestor of the outgroup (--method outgroup).

optional arguments:
-h, --help            show this help message and exit

arguments:
-i FILE, --input FILE
                        input FASTA file (required).
-o FILE, --output FILE
                        output MSA file in FASTA format (required).
-m {midpoint,outgroup}, --method {midpoint,outgroup}
                        rooting method (default midpoint).
-t TARGETS [TARGETS ...], --targets TARGETS [TARGETS ...]
                        list of targets defining the outgroup (required for
                        the outgroup method).

Examples

Midpoint rooting:

    micca root -i input.tree -o input_rooted.tree

Rooting with outgroup:

    micca root -i input.tree -o input_rooted.tree -m outgroup DENOVO1 DENOVO2
```

CHAPTER 21

split

```
usage: micca split [-h] -i FILE -o FILE -b FILE [-n FILE] [-c FILE] [-s N]
                   [-e MAXE] [-t] [-f {fastq,fasta}]

micca split assign the multiplexed reads to samples based on their 5'
nucleotide barcode (demultiplexing) provided by the FASTA file
(--barcode). micca split creates a single FASTQ or FASTA file with
sample information (e.g. >SEQID;sample=SAMPLENAME) appended to the
sequence identifier. Barcode and the sequence preceding it is removed
by default, e.g.:

Barcode file:         Input file:

>SAMPLE1              >SEQ1
TCAGTCAG              TCAGTCAGGCCACGGCTAACTAC...
...                   ...

the output will be:

>SEQ1;sample=SAMPLE1
GCCACGGCTAACTAC...
...

optional arguments:
-h, --help            show this help message and exit

arguments:
-i FILE, --input FILE
                        input FASTQ/FASTA file (required).
-o FILE, --output FILE
                        output FASTQ/FASTA file (required).
-b FILE, --barcode FILE
                        barcode file in FASTA format (required).
-n FILE, --notmatched FILE
                        write reads in which no barcode was found.
```

(continues on next page)

(continued from previous page)

```
-c FILE, --counts FILE
                        write barcode counts in a tab-delimited file.
-s N, --skip N        skip N bases before barcode matching (e.g. if your
                        sequences start with the control sequence 'TCAG'
                        followed by the barcode, set to 4) (>=0, default 0).
-e MAXE, --maxe MAXE  maximum number of allowed errors (>=0, default 1).
-t, --notrim          do not trim barcodes and the sequence preceding it
                        from sequences.
-f {fastq,fasta}, --format {fastq,fasta}
                        file format (default fastq).

Examples

Split 'reads.fastq' and write the notmatched sequences in the
file 'notmatched.fastq':

    micca split -i input.fastq -o splitted.fastq -b barcode.fasta \
    -n notmatched.fastq
```

CHAPTER 22

stats

```
usage: micca stats [-h] -i FILE [-o DIR] [-n TOPN]

micca stats reports statistics on reads in a FASTQ file. micca stats
returns in the output directory 3 tab-delimited text files:

* stats_lendist.txt: length distribution;
* stats_qualdist.txt: Q score distribution;
* stats_qualsumm.txt: quality summary. For each read position, the
following statistics are reported:
- L: read position;
- NPctCum: percent of reads with at least L bases;
- QAv: average Q score at position L;
- EERatePctAv: average expected error (EE) rate %.

Moreover, micca stats returns the respective plots in PNG format,
stats_lendist_plot.png, stats_qualdist_plot.png, and
stats_qualsumm_plot.png.

optional arguments:
-h, --help            show this help message and exit

arguments:
-i FILE, --input FILE
                        input FASTQ file, Sanger/Illumina 1.8+ format
                        (phred+33) (required).
-o DIR, --output DIR  output directory (default .).
-n TOPN, --topn TOPN  perform statistics only on the first TOPN sequences
                        (disabled by default).

Examples

Compute statistics on the top 10000 sequences of input.fastq:

    micca stats -i input.fastq -o stats -n 10000
```

CHAPTER 23

tablebar

```
usage: micca tablebar [-h] -i FILE -o FILE [-r] [-t N] [--xticklabelsize SIZE]
                      [-f {pgf,ps,svg,rgba,raw,svgz,pdf,eps,png}]

micca tablebar generates a relative abundance bar plot from
OTU or taxa tables. The table must be an OTU/taxon x sample,
TAB-separated file (see 'micca otu').

optional arguments:
-h, --help            show this help message and exit

arguments:
-i FILE, --input FILE
                        input OTU table file (required).
-o FILE, --output FILE
                        output image file (required).
-r, --raw             plot raw values (i.e. counts) instead of relative
                        abundances.
-t N, --topn N        plot the top N abundant taxa (default 12).
--xticklabelsize SIZE
                        x tick label size (default 8).
-f {pgf,ps,svg,rgba,raw,svgz,pdf,eps,png}, --format {pgf,ps,svg,rgba,raw,svgz,pdf,eps,
↪png}
                        output file format (default png).

Example

    micca tablebar -i otutable.txt -o otutable_plot.png
```

CHAPTER 24

tablerare

```
usage: micca tablerare [-h] -i FILE -o FILE -d DEPTH [-r] [-s SEED]

Rarefy an OTU table by subsampling, with or without
replacement. Samples that have fewer counts then the depth are
omitted from the output table. OTUs that are not present in at
least one sample are omitted from the output table.

optional arguments:
-h, --help            show this help message and exit

arguments:
-i FILE, --input FILE
                        input OTU table file (required).
-o FILE, --output FILE
                        output rarefied OTU table file (required).
-d DEPTH, --depth DEPTH
                        sample depth (>0, required).
-r, --replace         subsample with replacement.
-s SEED, --seed SEED  random seed (default 0).

Examples

Rarefy an OTU table at a depth of 1000 sequences/sample:

    micca tablerare -i otutable.txt -o otutable_rare.txt -d 1000
```

CHAPTER 25

tablestats

```
usage: micca tablestats [-h] -i FILE [-o DIR] [-t STEP] [-r] [-s SEED]

micca tablestats reports a sample summary, an OTU summary and
the rarefaction curves for the input OTU table. The
rarefaction curves are evaluated using the interval of 'step'
(-t/--step) sample depths, always including 1 and the total
sample size.

micca filterstats returns in the output directory 4 files:

* tablestats_samplesumm.txt: samples summary;
* tablestats_otusumm.txt: OTUs summary;
* tablestats_rarecurve.txt: rarefaction curves in text format.
* tablestats_rarecurve_plot.txt: rarefaction curves in png format.

optional arguments:
-h, --help            show this help message and exit

arguments:
-i FILE, --input FILE
                        input FASTQ file, Sanger/Illumina 1.8+ format
                        (phred+33) (required).
-o DIR, --output DIR  output directory (default .).
-t STEP, --step STEP  sample depth interval (for rarefaction curves, default
                        500).
-r, --replace         subsample with replacement (for rarefaction curves).
-s SEED, --seed SEED  random seed (for rarefaction curves, default 0).

Examples

Compute OTU table statistics on otutable.txt:

    micca tablestats -i otutable.txt -o tablestats
```

CHAPTER 26

tabletotax

```
usage: micca tabletotax [-h] -i FILE -t FILE [-o DIR]

Given an OTU table and a taxonomy file, micca tabletotax
creates in the output directory a table for each taxonomic
level (taxtable1.txt, ..., taxtableN.txt). OTU counts are
summed together if they have the same taxonomy at the
considered level.

The OTU table must be an OTU x sample, TAB-separated OTU table
file (see 'micca otu'). The taxonomy file must be a
tab-delimited file where where rows are either in the form
(see 'micca classify'):

1. SEQID[TAB]k__Bacteria;p__Firmicutes;c__Clostridia;o__Clostridiales;f__;g__;
2. SEQID[TAB]Bacteria;Firmicutes;Clostridia;Clostridiales;;;
3. SEQID[TAB]Bacteria;Firmicutes;Clostridia;Clostridiales
4. SEQID[TAB]D_0__Bacteria;D_1__Firmicutes;D_2__Clostridia;D_3__Clostridiales;D_4__;D_
↪5__;

optional arguments:
-h, --help            show this help message and exit

arguments:
-i FILE, --input FILE
                        input OTU table file (required).
-t FILE, --tax FILE   input taxonomy file (required).
-o DIR, --output DIR  output directory (default .).

Examples

    micca tabletotax -i otutable.txt -t tax.txt -o taxtables
```

CHAPTER 27

tobiom

```
usage: micca tobiom [-h] -i FILE -o FILE [-t FILE] [-s FILE] [-u FILE]

micca tobiom converts the micca OTU table into BIOM Version
1.0 (JSON) format. Optionally, taxonomy and/or sample
information can be added.  When you convert on
(closed-reference) OTU table for PICRUSt, replace OTU IDs with
the original sequence IDs use the option -u/--otuids.

optional arguments:
-h, --help            show this help message and exit

arguments:
-i FILE, --input FILE
                        input OTU table file (required).
-o FILE, --output FILE
                        output BIOM file Version 1.0 (JSON) (required).
-t FILE, --tax FILE   add taxonomy information from a taxonomy file.
-s FILE, --sampledata FILE
                        add sample information from a sample data file.
-u FILE, --otuids FILE
                        replace OTU IDs with the original sequence IDs. Useful
                        when the closed-reference OTU picking protocol was
                        performed for PICRUSt

Example

    micca tobiom -i otutable.txt -o output.biom -t tax.txt -s sampledata.txt
```

CHAPTER 28

tree

```
usage: micca tree [-h] -i FILE -o FILE [-m {fasttree,muscle}] [--fasttree-gtr]
                [--fasttree-fastest]
                [--muscle-cluster {upgmb,upgma,neighborjoining}]

micca tree infers phylogenetic trees from alignments. It provides two
methods:

* FastTree (doi: 10.1371/journal.pone.0009490);
* MUSCLE (doi: 10.1093/nar/gkl244).

optional arguments:
-h, --help            show this help message and exit

arguments:
-i FILE, --input FILE
                        input FASTA file (required).
-o FILE, --output FILE
                        output tree in Newick format (required).
-m {fasttree,muscle}, --method {fasttree,muscle}
                        tree inference method (default fasttree).

FastTree specific options:
--fasttree-gtr        use the generalized time-reversible (GTR)+CAT model
                        instead of Jukes-Cantor+CAT (default False).
--fasttree-fastest    speed up the neighbor joining phase and reduce memory
                        usage recommended for >50,000 sequences) (default
                        False).

MUSCLE specific options:
--muscle-cluster {upgmb,upgma,neighborjoining}
                        clustering algorithm (default upgmb).

Examples

```

(continues on next page)

(continued from previous page)

```
Tree inference using FastTree and the generalized time-reversible
(GTR)+CAT model:

    micca tree -i input.fasta -o tree.tree --fasttree-gtr
```

CHAPTER 29

trim

```
usage: micca trim [-h] -i FILE -o FILE [-w FORWARD [FORWARD ...]]
                [-r REVERSE [REVERSE ...]] [-e MAXERATE] [-c] [-W] [-R]
                [-f {fastq,fasta}]

micca trim trims forward and reverse primers from a FASTQ/FASTA file
using Cutadapt (doi: 10.14806/ej.17.1.200) internally. Primer and the
sequence preceding (for forward) or succeding (for reverse) it are
removed. Optionally, reads that do not contain the primers (untrimmed
reads) can be discarded with the options -W/--duforward and
-R/--dureverse. Recommended options are:

* always discard reads that do not contain the forward primer
(-W/--duforward option);

* for overlapping paired-end (already merged) reads, also discard
reads that do not contain the reverse primer (using both
-W/--duforward and -R/--dureverse options).

IUPAC codes and multiple primers are supported.

optional arguments:
-h, --help            show this help message and exit

arguments:
-i FILE, --input FILE
                        input FASTQ or FASTA file (required).
-o FILE, --output FILE
                        output FASTQ or FASTA file (required).
-w FORWARD [FORWARD ...], --forward FORWARD [FORWARD ...]
                        trim forward primer(s). Only the best matching primer
                        is removed.
-r REVERSE [REVERSE ...], --reverse REVERSE [REVERSE ...]
                        trim reverse primer(s). Only the best matching primer
                        is removed.
```

(continues on next page)

(continued from previous page)

```
-e MAXERATE, --maxerate MAXERATE
                        maximum allowed error rate (default 0.1).
-c, --searchrc        search reverse complement primers too (default False).
-W, --duforward       discard untrimmed reads (reads that do not contain the
                        forward primer) (always recommended) (default False).
-R, --dureverse       discard untrimmed reads (reads that do not contain the
                        reverse primer) (suggested option for overlapping
                        paired-end already merged reads) (default False).
-f {fastq,fasta}, --format {fastq,fasta}
                        file format (default fastq).

Examples

454 or Illumina single-end reads: trim forward primer and discard reads
that do not contain it. Moreover, trim reverse primer:

    micca trim -i input.fastq -o trimmed.fastq -w AGGATTAGATACCCTGGTA \
    -r CRRCACGAGCTGACGAC -W

Illumina overlapping paired-end (already merged) reads: trim
forward and reverse primers. Reads that do not contain the forward
or the reverse primer will be discarded:

    micca trim -i reads.fastq -o trimmed.fastq -w AGGATTAGATACCCTGGTA \
    -r CRRCACGAGCTGACGAC -W -R
```