User-Manual

AcetoScan

(version 1.0)

Singh, A., Nylander, J. A. A., Schnürer, A., Bongcam-Rudloff, E., and Müller, B. (2020). High-Throughput Sequencing and Unsupervised Analysis of Formyltetrahydrofolate Synthetase (FTHFS) Gene Amplicons to Estimate Acetogenic Community Structure. Front. Microbiol. 11, 1–13. doi:10.3389/fmicb.2020.02066

User-manual edited on: 2020-09-04
Sign: Abhijeet Singh

Contents

AcetoScan

- Version: 1.0 (20200904)
- Last modified: Sep 04, 2020 11:50
- Sign: Abhijeet Singh (abhijeetsingh.aau@gmail.com)

Overview

AcetoScan is a software pipeline for the unsupervised analysis of high-throughput sequencing data obtained from formyltetrahydrofolate synthetase (FTHFS) gene amplicon sequencing. The pipeline is primarily designed for the analysis of data generated on Illumina MiSeq platform, however, it could potentially be used for the data generated on other sequencing platforms. AcetoScan can also process nucleotide multifasta sequences to filter out non-target sequences, assign taxonomy and generate a phylogenetic tree.

Dependencies

AcetoScan is built on following dependencies and requires the software versions equals to or higher than the mentioned versions:

- Cutadapt v2.9 (Martin 2017)
- VSEARCH v2.13.1 (Rognes *et al*. 2016)
- NCBI-blast+ v2.5.0+ (Camacho *et al*. 2009)
- Bioperl v1.7.2-3 (Stajich et al. 2002)
- MAFFT v7.307 (Katoh and Standley 2013)
- Fasttree 2.1.9 (Price, Dehal, and Arkin 2009)
- AcetoBase (Singh *et al*. 2019)
- R v3.5.2 (R Core Team 2011)
    - Phyloseq v1.24.2 (McMurdie and Holmes 2013)
    - ggplot2 v3.1.1 (Ginestet 2011)
    - plotly v4.9.0 (Sievert 2018)
    - RcolorBrewer v1.1.2 (Neuwirth 2014)
    - plyr v1.8.4 (Wickham 2014)
    - dplyr v0.8.0.1 (Wickham et al. 2017)
    - vegan (Dixon 2003)

Installation

AcetoScan version v1.0 is installation compatible with Debian/Ubuntu based systems. Other system specific methods can be used to install the dependency software followed by AcetoScan installation. To install AcetoScan and its dependencies on Debian/Ubuntu based systems following methods can be used:

Installing AcetoScan as a super user:

```
$ sudo ./INSTALL

$ acetoscan -h
```

Installing as a super user place the AcetoScan scripts in path `/usr/local/bin/` and will create directory `acetoscan` in path `/home/<user>/`, and three sub-directories. These sub-directories are named as: `acetoscan_bin` containing dependency binaries, `acetobase` containing AcetoBase reference protein database and `output_data` as a default output directory for `acetoscan` analysis.

Installing AcetoScan as a local user:

```
$ bash INSTALL

$ bash /home/<user>/acetoscan/acetoscan -h
```

If the installation needs to be done as normal user (without super user privilege), installation will create directory `acetoscan` in path `/home/<user>/`, and three sub-directories `acetoscan_bin`, `acetobase` and `output_data`. It is to be noted that some software dependencies needs to be installed manually if they are not already installed or if installation of AcetoScan is not done as super user.

Prerequisites

To run AcetoScan analysis, the raw data files must be present in an input directory or sub-directories in the compressed fastq format (Illumina Inc. 2015). The file name can contain several fields *i.e.* `Sample_<name>_<batch>_<date>_L001_R1_001.fastq.gz|bz2`, however, it is recommended to use short names *i.e.* `Sx_<date>_L001_R1_001.fastq.gz`, wherever possible for better visualization in the output plots.

The quality of the raw data should be visualized by the FastQC and MultiQC analysis. Samples which do not appear to be well sequenced in terms of number of reads, extremely bad sequence quality *et cetera* should be removed before starting the analysis.

Some analysis only requires FTHFS nucleotide sequences in multifasta format with no special requirements or modification. These analysis are further discussed in following text.

Description of analysis

Based on the information available for the published primer pairs targeting the FTHFS gene sequence, most of the primers generate amplicons greater than 600 base pairs (Leaphart and Lovell 2001; Ohashi et al. 2007; Müller, Sun, and Schnürer 2013; Müller et al. 2016). This means that the sequences generated for FTHFS amplicons (at present) on Illumina MiSeq cannot be merged. Therefore, AcetoScan processes the sequence data only for one type of reads as specified by the user (default = R1/forward reads). AcetoScan carries out unsupervised data analysis in four major steps and results into high quality and interactive plots. The analysis steps are further described below:

Table 1. Overview of AcetoScan analysis process

Step	Process	Task	Dependency
1	Quality control	Primer sequence trimming and quality filtering	Cutadapt
2	Sequence data analysis	Dereplication, denoising, chimera removal, clustering, OTU picking	VSEARCH
		Filtering non-target sequences and best frame analysis	Bioperl, NCBI-blast+, AcetoBase
		OTU table generation and taxonomic assignment	NCBI-blast+, VSEARCH, AcetoBase
3	Phylogenetic inference	Multiple sequence alignment	MAFFT
		Phylogenetic tree computation	Fasttree
4	Visualization	Plotting various plots for taxonomic abundance, alpha and beta diversity graphs, phylogenetic tree visualization	R & dependencies

Step 1. Quality control

In this step, the user specified raw sequence data (either forward or reverse read data) is subjected to adapter/primer sequence trimming and quality filtering. This step is dependent on software cutadapt version ≥1.18. The trimming is done by cutting specified number of bases (default = 24) from the 5’ end of the fastq sequences. The number of bases trimmed can be changed based on the length of the primer sequence. Filtering of the sequences is done according to the specified Phred quality

score threshold (default =20). In general, the sequence data having Phred quality score of ≥ 20 (99 % base call accuracy) is considered good enough for the sequence analysis, therefore set as default. The sequences are also filtered based on the minimum (default = 120) and maximum lengths (default = 300) of the fastq sequences.

Step 2. Sequence analysis

The second step is sub-divided into three sub-analysis process which require the dependency software VSEARCH version ≥ 2.13.1. The analysis sub-processes are described below:

I. Data curation

a. Dereplication

Dereplication of the sequence data refers to the removal of redundancies in the sequences and only keeping absolute unique sequences based on complete length and discarding duplicates. For dereplication step, multithreading is not supported in VSEARCH. Dereplication is done by the VSEARCH command `--derep_fulllength`. The minimum clustering threshold for dereplication is set to 2, to remove sequences appearing only once in whole data set being analysed.

b. Denoising

The sequence clustering and denoising is done based on the UNOISE algorithm version 3 implemented in VSEARCH command `--cluster_unoise`. In this step reads with sequencing and PCR error are removed and only biologically correct sequences are retained and clustered. The minimum cluster size is set to 2 as default. User can define the minimum cluster size using command `acetoscan` option -c.

c. Chimera filtering

Chimera sequences are removed from the denoised data using default values from VSEARCH command `--uchime3_denovo` based on the UCHIME2 algorithm. The non-chimera sequences are further used for the OTU picking.

d. OTU picking

Operational taxonomic units (OTU) are generated based on the minimum cluster size and minimum cluster threshold parameters specified

by the user. The default parameters for the clustering threshold is set to 80 % for the genus level resolution (Singh *et al*. 2019).

II. Data filtering

At this point, the non-target sequences *i.e.* sequences which are not FTHFS sequences are filtered out. The removal of sequences is done with blastx algorithm using AcetoBase reference protein database (https://acetobase.molbio.slu.se/download/ref/1). The sequence filter is done based on the evalue criteria which is set to 1e-3 as default, however, for increased accuracy user can change the evalue. Further, after discarding the non-targeted sequences, the retained FTHFS sequences are analysed for the longest reading frame without internal stop codons. This is done by AcetoScan using Bioperl as base package.

III. Generation of Abundance and taxonomic tables

The longest-best FTHFS sequences are used for the OTU table generation with the clustering threshold chosen in the OTU picking step. VSEARCH command `--usearch_global` generates the abundance table. Further, Taxonomy table is generated by applying the blastx algorithm with evalue criteria mentioned above on the frame checked FTHFS sequences against the AcetoBase protein database.

Step 3. Sequence alignment and phylogenetic inference

FTHFS OTU sequences in correct frame of codons are saved in fasta formatted file. These sequences are used for the global sequence alignment with MAFFT aligner with 5 rounds of UPGMA tree refinement and iteration specified by the user. A good alignment facilitates faster phylogenetic tree generations, therefore the iteration cycles for the multiple sequence alignment and phylogenetic bootstrap iterations are set as common parameter for both the steps and can be specified by the -B option (table 1). The default number of iterations in multiple sequence alignment and bootstraps in tree building is set to 1000 as default. Phylogenetic tree with the aligned sequences is generated with software Fasttree with Jukes-Cantor (JC) distance and maximum likelihood (ML) topology refinement with 10 rounds of nearest neighbor interchange (NNI) and 5 rounds of subtree pruning and re-grafting (SPR). The generalized time reversible (GTR) model with 10 rounds of CAT approximation and GAMMA rate heterogeneity and BIONJ distance optimization together with specified bootstrap iterations is applied for the phylogenetic tree generation.

Step 4. Data visualization

The final data generated in step 2 and 3 are used for the data visualization in form of various plots. The visualization of data is done with R environment where abundance table and taxonomy table are merged together with the phylogenetic tree and sample table to generate a phyloseq object with the package *phyloseq*. The Package *plyr*, *dplyr* and *vegan* are used for the data table modification and diversity analysis package. Barplots and heatmaps for all taxonomic levels (phylum to species) are generated with different abundance threshold which is specified in the respective plot. Alpha diversity analysis is done with package *phyloseq* for Observed, Shannon and Simpson diversity measure indices. Beta diversity is visualized in form of non-metric multidimensional scaling (NMDS) and principal coordinates analysis (PCoA) analysis. NMDS analysis is carried out with the Bray-Curtis dissimilarity distances and plotted in two different forms *i.e.* based on the phyla and sample. PCoA analysis (also known as multidimensional scaling - MDS) is carried out with the weighted UniFrac distances. Publication ready format plots are generated with package *ggplot2* and *RColorBrewer*. Interactive plots in html format to for visualization in web-browser are produced with the package *plotly*. Phylogenetic tree visualization and annotation is done at the phylum level.

AcetoScan pipeline

AcetoScan pipeline is primarily developed for the unsupervised analysis and visualization of raw sequencing data in compressed fastq format. However, if the user wants to process FTHFS nucleotide sequence data which are in fasta format and are generated by clone library construction and Sanger sequencing, AcetoScan harbors functionalities for this. AcetoScan pipeline has four different analysis commands *i.e.* `acetoscan`, `acetocheck`, `acetotax` and `acetotree`. These program executable scripts will be installed in directory based on the method of installation as discussed above. The programs available in AcetoScan pipeline can be seen by following command

```
$ acetoscan -X

# AcetoScan commands:

  acetoscan - for complete processing of raw sequence data
  acetocheck      - for processing fasta sequences and filtering out non-
target sequences
  acetotax   - acetocheck + taxonomic assignments
  acetotree - acetotax + phylogenetic tree generation
```

1. acetoscan

`acetoscan` is the main program of the pipeline and it requires the raw sequence data in compressed fastq format and results into ready to use graphs and plots. This program requires the user to give the path to the directory containing the raw sequence data. The only prerequisite of `acetoscan` is the compressed fastq format as discussed above. The options for `acetoscan` can be seen using help.

```
$ acetoscan -h

acetoscan -i /<input path>/ [-o /<output path>/] [-m 300] [-n 120] [-q 20]
[-l 24] [-r 1] [-t 0.80] [-c 2] [-e 1e-3] [-B 1000] [-P 8]


        -i      Input directory containing raw illumina data
        -o      Output directory
                        :default = /home/<user>/acetoscan/output_data
        -m      Maximum length of sequence after quality filtering
                        :default max_length = 300
        -n      Minimum length of sequence after quality filtering
                        :default min_length = 120
        -q      Quality threshold for the sequences
                        :default quality threshold = 20
        -l      Primer length
                        :default primer length = 24
        -r      Read type either forward or reverse reads
                        1 = forward reads (default), 2 = reverse reads
        -t      Clustering threshold
                        :default cluster threshold = 0.80 (80 %)
        -c      Minimum cluster size
                        :default minimum cluster size = 2
        -e      E-value
                        :default evalue = 1e-3
        -B      Bootstrap value
                        :default bootstrap = 1000
        -P      Parallel processes / threads
                        :default no. of parallels = all available threads
        -h      Print help
        -X      Print AcetoScan commands
        -v      Print AcetoScan version
        -C      Print AcetoScan citation
```

Running acetoscan on test data:

```
$ acetoscan -i /home/<user>/Desktop/test_data -o
/home/<user>/Desktop/test_result
```

Program `acetoscan` will result into two directories: **a)** `output_data` – in this directory the data processing files will be located. In case of an execution halt or process failure, the data or sub-directories can be accessed in this directory, **b)** `acetoscan_result` – which contains the 67 final result files after successful execution of `acetoscan`. The file generated as results by `acetoscan` is presented in the following tree:

```
/<path_to_output_directory>/acetoscan_result/
│
├── 0_acetoscan_<date>_<time>.log
├── 0_visualization_info.txt
├── 1_Phylum_abs_abundance.html
├── 1_Phylum_abs_abundance.pdf
├── 1_Phylum_abs_abundance.tif
├── 1_Phylum_barplot.html
├── 1_Phylum_barplot.pdf
├── 1_Phylum_barplot.tif
├── 2_Class_barplot.html
├── 2_Class_barplot.pdf
├── 2_Class_barplot.tif
├── 3_Order_barplot.html
├── 3_Order_barplot.pdf
├── 3_Order_barplot.tif
├── 4_Family_barplot.html
├── 4_Family_barplot.pdf
├── 4_Family_barplot.tif
├── 4_Family_heatmap.html
├── 4_Family_heatmap.pdf
├── 4_Family_heatmap.tif
├── 5_Genus_barplot.html
├── 5_Genus_barplot.pdf
├── 5_Genus_barplot.tif
├── 5_Genus_heatmap.html
├── 5_Genus_heatmap.pdf
├── 5_Genus_heatmap.tif
├── 6_Species_barplot.html
├── 6_Species_barplot.pdf
├── 6_Species_barplot.tif
├── 6_Species_heatmap.html
├── 6_Species_heatmap.pdf
├── 6_Species_heatmap.tif
├── 7_Absolute_abundance.pdf
├── 8_Relative_abundance.pdf
├── Alpha_diversity.html
├── Alpha_diversity.pdf
├── Alpha_diversity.tif
├── FTHFS_otu.aln
├── FTHFS_otu.fasta
├── FTHFS_otutab.csv
├── FTHFS_otu.tree
├── FTHFS_samtab.csv
├── FTHFS_taxtab.csv
```

```
├── FTHFS_tree1.html
├── FTHFS_tree1.pdf
├── FTHFS_tree1.tif
├── FTHFS_tree2.html
├── FTHFS_tree2.pdf
├── FTHFS_tree2.tif
├── NMDS_Phylum_1.html
├── NMDS_Phylum_1.pdf
├── NMDS_Phylum_1.tif
├── NMDS_Phylum_2.html
├── NMDS_Phylum_2.pdf
├── NMDS_Phylum_2.tif
├── NMDS_Sample_1.html
├── NMDS_Sample_1.pdf
├── NMDS_Sample_1.tif
├── NMDS_Sample_2.html
├── NMDS_Sample_2.pdf
├── NMDS_Sample_2.tif
├── weighted_unifrac_PCoA_2.html
├── weighted_unifrac_PCoA_2.pdf
├── weighted_unifrac_PCoA_2.tif
├── weighted_unifrac_PCoA.html
├── weighted_unifrac_PCoA.pdf
└── weighted_unifrac_PCoA.tif
```

2. acetocheck

`acetocheck` can be used to filter out FTHFS sequences, discard non-FTHFS sequences and check FTHFS sequences for internal stop codons. Only FTHFS sequences in frame without internal stop codon are selected and appear in the output file. `acetocheck` can use user-specific evalues for filtering the sequences, the default evalue is set to 1e-3.

```
$ acetocheck -h

acetocheck -i /path/<input_file> [-o /path/<output_file>] [-e 1e-3] [-P 8]


        -i      Input file - multifasta file
        -o      Output file
                        :default = acetocheck_<date>_<time>.fasta
        -e      E-value
                        :default evalue = 1e-3
        -P      Parallel processes/threads
                        :default no. of parallels = all available threads
        -h      Print help
        -X      Print AcetoScan commands
        -v      Print AcetoScan version
        -C      Print AcetoScan citation
```

3. acetotax

`acetotax` has two sub-processing steps which include `acetocheck` as first step followed by taxonomic assignments of the filtered sequences. Filtering and taxonomic assignment thresholds can be changed according to the user specific parameters.

```
$ acetotax -h

acetotax -i /path/<input_file>/ [-o /path/<output_file>/] [-e 1e-3] [-P 8]

      -i      Input_file
      -o      Output file
                     :default = acetotax_<date>_<time>.csv
                     :default = acetotax_<date>_<time>.fasta
      -e      E-value
                     :default evalue = 1e-3
      -P      Parallel processes/threads
                     :default no. of parallels = all available threads
      -h      Print help
      -X      Print AcetoScan commands
      -v      Print AcetoScan version
      -C      Print AcetoScan citation
```

4. acetotree

`acetotree` is the program to generate the phylogenetic tree from the FTHFS sequences. `acetotree` is based on `acetotax` followed by multiple sequence alignment of FTHFS sequences and phylogenetic tree generation. The bootstrap is the common iteration value for the multiple sequence alignment and the phylogenetic tree construction.

```
$ acetotree -h


acetotree  -i  /path/<input_file>  [-o  /path/<output_file>]  [-e  1e-3]  [-B
1000] [-P 8]


        -i      Input file - multifasta file
        -o      Output file
                        :default = acetotree_<date>_<time>.fasta
                        :default = acetotree_<date>_<time>.csv
                        :default = acetotree_<date>_<time>.aln
                        :default = acetotree_<date>_<time>.tree
        -e      E-value
                        :default evalue = 1e-3
        -B      Bootstrap value
                        :default bootstrap = 1000
        -P      Parallel processes/threads
                        :default no. of parallels = all available threads
        -h      Print help
        -X      Print AcetoScan commands
        -v      Print AcetoScan version
        -C      Print AcetoScan citation
```

AcetoScan Pipelline as a Docker image/container

AcetoScan pipeline is also available in Docker version, for the hassle-free execution of the process without caring for the dependency software. For this the only requirement is to install the Docker program on the local machine. This can be done by following the Docker installation tutorial at https://docs.docker.com/get-docker/. After successful installation of the Docker program, AcetoScan can be execute for the raw input data files.

1. Connecting the local data to Docker container

In order to run the AcetoScan in Docker container, the Docker container must have access to the raw data. To do this, the local volume (data location) can be mounted to the container. This can be done by following command:

```
$ sudo docker volume create --opt type=none --opt o=bind --opt device=/PATH/to/my/DATA --name MY_CUSTOM_NAME
```

Where:

`--opt device=` the path to your raw data

`--name` Any name user want to assign to the mounted volume

Example:

```
$ sudo docker volume create --opt type=none --opt o=bind --opt device=/home/abhi/Desktop/reads --name myDockerAcetoscan
```

Explanation:

`/home/abhi/Desktop/reads` is the path to the raw data on local computer

`myDockerAcetoscan` is the name of the mounted volume

2. Execution of AcetoScan in Docker container

Once the local data is connected to the Docker container, AcetoScan pipeline can be executed with the following command:

```
$ sudo docker run --rm -v MY_CUSTOM_NAME:/acetoscan/input_dir --entrypoint
acetoscan -it abhijeetsingh1704/acetoscan -i /acetoscan/input_dir
```

Where:

`MY_CUSTOM_NAME` is the name specified while mounting the local volume to the Docker container

Example:

```
$ sudo docker run --rm -v myDockerAcetoscan:/acetoscan/input_dir --
entrypoint acetoscan -it abhijeetsingh1704/acetoscan -i
/acetoscan/input_dir
```

Here:

`myDockerAcetoscan` custom name specified while mounting the local volume to the Docker container

- The additional options for the AcetoScan can be provided by the respective flags.

3. Execution of acetocheck, acetotax or acetotree in Docker container

1. acetocheck

```
$ sudo docker run --rm -v myDockerAcetoscan:/acetoscan/input_dir --
entrypoint acetocheck -it abhijeetsingh1704/acetoscan -i
acetoscan/input_dir/input_file.fasta -o
acetoscan/input_dir/output_file.fasta
```

2. acetotax

```
$ sudo docker run --rm -v myDockerAcetoscan:/acetoscan/input_dir --
entrypoint acetotax -it abhijeetsingh1704/acetoscan -i acetoscan/input_dir/
input_file.fasta -o acetoscan/input_dir/acetotax_out
```

3. acetotree

```
$ sudo docker run --rm -v myDockerAcetoscan:/acetoscan/input_dir --
entrypoint acetotree -it abhijeetsingh1704/acetoscan -i
acetoscan/input_dir/input_file.fasta -o acetoscan/input_dir/acetotree_out -
e 1e-3
-B 1000 -P 8
```

Bibliography

Camacho, Christiam, George Coulouris, Vahram Avagyan, Ning Ma, Jason Papadopoulos, Kevin Bealer, and Thomas L. Madden. 2009. “BLAST+: Architecture and Applications.” *BMC Bioinformatics* 10 (421). https://doi.org/10.1186/1471-2105-10-421.

Dixon, Philip. 2003. “VEGAN, a Package of R Functions for Community Ecology.” *Journal of Vegetation Science* 14 (6). https://doi.org/10.1111/j.1654-1103.2003.tb02228.x.

Ginestet, Cedric. 2011. “Ggplot2: Elegant Graphics for Data Analysis.” *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 174 (1): 245–46. https://doi.org/10.1111/j.1467-985x.2010.00676_9.x.

Illumina Inc. 2015. “MiSeq Reporter Generate FASTQ: Workflow Guide.” San Diego, California 92122 U.S.A.: Illumina, Inc. https://support.illumina.com/content/dam/illumina-support/documents/documentation/software_documentation/miseqreporter/miseq-reporter-generate-fastq-workflow-guide-15042322-01.pdf.

Katoh, Kazutaka, and Daron M. Standley. 2013. “MAFFT Multiple Sequence Alignment Software Version 7: Improvements in Performance and Usability.” *Molecular Biology and Evolution* 30 (4): 772–80. https://doi.org/10.1093/molbev/mst010.

Leaphart, Adam B, and Charles R Lovell. 2001. “Recovery and Analysis of Formyltetrahydrofolate Synthetase Gene Sequences from Natural Populations of Acetogenic Bacteria.” *Applied and Environmental Microbiology* 67 (3): 1392–95. https://doi.org/10.1128/AEM.67.3.1392.

Martin, Marcel. 2017. “Cutadapt Removes Adapter Sequences From High-Throughput Sequencing Reads.” https://cutadapt.readthedocs.io/en/stable/. https://doi.org/https://doi.org/10.14806/ej.17.1.200.

McMurdie, Paul J., and Susan Holmes. 2013. “Phyloseq: An R Package for Reproducible Interactive Analysis and Graphics of Microbiome Census Data.” *PLoS ONE* 8 (4): e61217. https://doi.org/10.1371/journal.pone.0061217.

Müller, Bettina, Li Sun, and Anna Schnürer. 2013. “First Insights into the Syntrophic Acetate-Oxidizing Bacteria - a Genetic Study.” *MicrobiologyOpen* 2 (1): 35–53. https://doi.org/10.1002/mbo3.50.

Müller, Bettina, Li Sun, Maria Westerholm, and Anna Schnürer. 2016. “Bacterial Community Composition and Fhs Profiles of Low- and High-Ammonia Biogas Digesters Reveal Novel Syntrophic Acetate-Oxidising Bacteria.” *Biotechnology for Biofuels* 9 (1): 1–18. https://doi.org/10.1186/s13068-016-0454-9.

Neuwirth, Erich. 2014. “RColorBrewer: ColorBrewer Palettes.” *User Manual*. CRAN. https://colorbrewer2.org.

Ohashi, Yuji, Tomoko Igarashi, Fumi Kumazawa, and Tomohiko Fujisawa. 2007. “Analysis of Acetogenic Bacteria in Human Feces with Formyltetrahydrofolate Synthetase Sequences.” *Bioscience and Microflora* 26 (2): 37–40. https://doi.org/10.12938/bifidus.26.37.

Price, Morgan N., Paramvir S. Dehal, and Adam P. Arkin. 2009. “Fasttree: Computing Large Minimum Evolution Trees with Profiles Instead of a

Distance Matrix." *Molecular Biology and Evolution* 26 (7): 1641–50. https://doi.org/10.1093/molbev/msp077.
R Core Team. 2011. "R: A Language and Environment for Statistical Computing." *R Foundation for Statistical Computing*. The R Foundation. http://www.r-project.org/.
Rognes, Torbjørn, Tomáš Flouri, Ben Nichols, Christopher Quince, and Frédéric Mahé. 2016. "VSEARCH: A Versatile Open Source Tool for Metagenomics." *PeerJ* 4 (October): e2584. https://doi.org/10.7717/peerj.2584.
Sievert, Carson. 2018. "Plotly for R." https://plotly-r.com.
Singh, Abhijeet, Bettina Müller, Hans Henrik Fuxelius, and Anna Schnürer. 2019. "AcetoBase: A Functional Gene Repository and Database for Formyltetrahydrofolate Synthetase Sequences." *Database : The Journal of Biological Databases and Curation*. https://doi.org/10.1093/database/baz142.
Stajich, Jason E., David Block, Kris Boulez, Steven E. Brenner, Stephen A. Chervitz, Chris Dagdigian, Georg Fuellen, et al. 2002. "The Bioperl Toolkit: Perl Modules for the Life Sciences." *Genome Research* 12 (10): 1611–18. https://doi.org/10.1101/gr.361602.
Wickham, Hadley. 2014. "R: Plyr." *R Core Team*. CRAN. https://www.rdocumentation.org/packages/plyr/versions/1.8.6.
Wickham, Hadley, Romain François, Lionel Henry, and Kirill Müller. 2017. "Dplyr: A Grammar of Data Manipulation." https://dplyr.tidyverse.org/.