Tourmaline: A containerized workflow for rapid and iterable amplicon sequence analysis using QIIME 2 and Snakemake

Luke R. Thompson [1,2,*], Sean R. Anderson [1,2], Paul A. Den Uyl [3], Nastassia V. Patin [2,4,†], Shen Jean Lim [2,4], Grant Sanderson [5] and Kelly D. Goodwin [2,†]

[1]Northern Gulf Institute, Mississippi State University, Mississippi State, MS 39762, USA
[2]Ocean Chemistry and Ecosystems Division, Atlantic Oceanographic and Meteorological Laboratory, National Oceanic and Atmospheric Administration, Miami, FL 33149, USA
[3]Cooperative Institute for Great Lakes Research, University of Michigan, Ann Arbor, MI 48108, USA
[4]Cooperative Institute for Marine and Atmospheric Studies, Rosenstiel School of Marine and Atmospheric Science, University of Miami, Miami, FL 33149, USA
[5]Marine Science Department, University of Hawaii, Hilo, HI 96720, USA
*Correspondence address. Luke R. Thompson, 4301 Rickenbacker Cswy, Miami, FL 33149, USA. E-mail: luke.thompson@noaa.gov
†Stationed at Southwest Fisheries Science Center, National Marine Fisheries Service, National Oceanic and Atmospheric Administration, La Jolla, CA 92037, USA.

Abstract

Background: Amplicon sequencing (metabarcoding) is a common method to survey diversity of environmental communities whereby a single genetic locus is amplified and sequenced from the DNA of whole or partial organisms, organismal traces (e.g., skin, mucus, feces), or microbes in an environmental sample. Several software packages exist for analyzing amplicon data, among which QIIME 2 has emerged as a popular option because of its broad functionality, plugin architecture, provenance tracking, and interactive visualizations. However, each new analysis requires the user to keep track of input and output file names, parameters, and commands; this lack of automation and standardization is inefficient and creates barriers to meta-analysis and sharing of results. **Findings:** We developed Tourmaline, a Python-based workflow that implements QIIME 2 and is built using the Snakemake workflow management system. Starting from a configuration file that defines parameters and input files—a reference database, a sample metadata file, and a manifest or archive of FASTQ sequences—it uses QIIME 2 to run either the DADA2 or Deblur denoising algorithm; assigns taxonomy to the resulting representative sequences; performs analyses of taxonomic, alpha, and beta diversity; and generates an HTML report summarizing and linking to the output files. Features include support for multiple cores, automatic determination of trimming parameters using quality scores, representative sequence filtering (taxonomy, length, abundance, prevalence, or ID), support for multiple taxonomic classification and sequence alignment methods, outlier detection, and automated initialization of a new analysis using previous settings. The workflow runs natively on Linux and macOS or via a Docker container. We ran Tourmaline on a 16S ribosomal RNA amplicon data set from Lake Erie surface water, showing its utility for parameter optimization and the ability to easily view interactive visualizations through the HTML report, QIIME 2 viewer, and R- and Python-based Jupyter notebooks. **Conclusion:** Automated workflows like Tourmaline enable rapid analysis of environmental amplicon data, decreasing the time from data generation to actionable results. Tourmaline is available for download at github.com/aomlomics/tourmaline.

Keywords: amplicon sequencing, metabarcoding, environmental DNA, eDNA, microbiome, meta-analysis

Background

Earth's environments are teeming with environmental DNA (eDNA): free and cellular genetic material from whole microorganisms [1, 2] or remnants of larger macroorganisms [3, 4]. This eDNA can be collected, extracted, and sequenced to reveal the identities and functions of the organisms that produced it. Amplicon sequencing (metabarcoding), whereby a short genomic region is amplified and sequenced using polymerase chain reaction (PCR) from an environmental or experimental community's eDNA, is a popular method for measuring taxonomic diversity of microbiomes and environmental samples [3, 5, 6]. PCR primers have been used to generate amplicons of the bacterial 16S ribosomal RNA (rRNA) gene in studies of human and animal-associated microbiota [7–9], as well as environmental microbiota [2, 10]. Other regions that are commonly targeted include the fungal internal transcribed spacer (ITS) regions between rRNA genes [11], the 18S rRNA gene of eukaryotes [12], the mitochondrial cytochrome oxidase I (COI) gene of invertebrate and vertebrate eDNA [13], and the mitochondrial 12S rRNA gene of fish [14]. Information gained from amplicon metabarcoding has far-reaching implications for human health (e.g., microbiome research), ecosystem function and conservation, and resource management [15, 16].

Computational workflows (pipelines) that run on local or networked computing resources or in the cloud have emerged as useful approaches to execute extended bioinformatics analyses [17]. Workflows wrap multiple tools and commands into a much smaller number of commands, with parameters often specified in a configuration file. Ideally, workflows allow for less time and effort spent on each separate analysis (i.e., scalability) and more reproducibility between analyses. Because workflows allow multiple data sets to be analyzed in parallel with standardized parameters, they provide opportunities for improved meta-analysis of micro-

biome or eDNA data sets [18–20]. Some of the amplicon workflows that have been developed are Anacapa [21, 22], Banzai [23], PEMA [24, 25], nf-core/ampliseq [26, 27], Cascabel [28, 29], dadasnake [30, 31], CoMA [32], ASAP 2 [33, 34], and tagseq [35]. Note here that we do not consider amplicon analysis packages like QIIME 2 [36, 37], MOTHUR [38], or OBITools [39] to be workflows, although they are very useful. Indeed, we believe that the most efficient workflows would take advantage of these existing packages and their built-in features. The abovementioned workflows have many excellent features, as compared previously [25], but none of them possesses all of the features that might be desired in a single workflow.

The ideal amplicon sequence analysis workflow, in our view, would build upon a modern amplicon analysis package, with advanced data formats, interactive visualization capabilities, and extensibility. QIIME 2, with its built-in provenance tracking, archive format, interactive visualizations, multiple interfaces including a Python API, and extensible plugin architecture, has become a popular package and is our package of choice. QIIME 2 supports DADA2 [40] and Deblur [41] plugins for denoising amplicon sequence data. The ideal amplicon workflow would also be built on a modern workflow management system to promote scalability and reproducibility. Snakemake [42] is a popular workflow management system in the bioinformatics community that manages input and output files in a defined directory structure, with commands defined in a Snakefile as "rules" and parameters and initial input files set by the user in a configuration file. Snakemake ensures that only the commands required for requested output files not yet generated are run, saving time and computation when rerunning part of a workflow. The ideal workflow would take advantage of the defined directory structure through downstream analysis capabilities like Jupyter notebooks for analysis and meta-analysis and support for parameter optimization. Outputs would be summarized with summary plots and tables, and all outputs would be presented in a single report (e.g., HTML) that could be shared with collaborators. Use of the workflow would be simplified by providing a containerized installation to enable deployment on multiple platforms while avoiding dependency issues. Finally, the workflow would provide clear step-by-step instructions with a tutorial using a small test data set.

Here, we present Tourmaline [43], an amplicon analysis pipeline that uses Snakemake to run QIIME 2 commands for core analysis and interactive visualization—plus workflow-specific commands that generate an HTML report of output and summary tables and figures of data and metadata—with rapid analysis aided by workflow iterability and scalability, support for multiple cores, a Docker container, and a detailed tutorial. After cloning the initial Tourmaline directory from GitHub and setting up the input files and parameters, only a few simple shell commands are required to execute the Tourmaline workflow. Outputs are stored in a defined directory structure that is the same for every Tourmaline run, facilitating data exploration, parameter optimization, downstream analysis, and meta-analysis across studies. Because of this defined directory structure, different runs that utilize different parameters (e.g., DADA2 truncation lengths) can be easily compared, facilitated by a helper script that makes a new copy of the Tourmaline directory from an existing one. Every Tourmaline run produces an HTML report containing a summary of metadata and outputs, with links to web-viewable QIIME 2 visualization files; the report facilitates evaluation of metadata (e.g., compliance with standards) and output (e.g., statistics about representative sequences and feature tables). A zipped run directory can be shared with collaborators, and relative links in the report are preserved, facilitating data exploration by experts and nonex-

perts alike. QIIME 2 artifact files can be fed directly into provided Python- and R-based Jupyter notebooks. In addition to running natively on Mac and Linux platforms, Tourmaline can be run in any computing environment using Docker containers. In this article, we describe the Tourmaline workflow and apply it to a downsampled 16S rRNA gene data set from surface waters of Western Lake Erie. The tutorial includes guidance on evaluating output to refine parameters for the workflow and showcases the HTML report, interactive visualizations, and Jupyter notebooks for biological insight into amplicon data sets.

Findings
Workflow
Overview

Tourmaline is a Snakemake-based bioinformatics workflow that operates in a defined directory structure (Fig. 1). Installation involves installing QIIME 2 and other dependencies or installing the Docker container. The starting directory structure is then cloned directly from GitHub and is built out through Snakemake commands, defined as "rules" in *Snakefile*. Tourmaline provides 7 high-level "pseudo-rules" for each of DADA2 paired-end, DADA2 single-end, and Deblur (single-end), running denoising and taxonomic and diversity analyses via QIIME 2 and other programs, encompassing commonly used analyses in eDNA/microbiome research. For each type of processing, there are 4 steps: (i) the *denoise* rule imports FASTQ data and runs denoising, generating a feature table and representative sequences; (ii) the *taxonomy* rule assigns taxonomy to representative sequences; (iii) the *diversity* rule does representative sequence curation, core diversity analyses, and alpha and beta group significance; and (iv) the *report* rule generates an HTML report of the metadata, inputs, outputs, and parameters. Steps 2–4 have 2 modes each, *unfiltered* and *filtered*, thus making 7 pseudo-rules total. The difference between the *unfiltered* and *filtered* commands is that in the *taxonomy_filtered* command, undesired taxonomic groups or individual sequences from the representative sequences and feature table are filtered (removed). The *diversity* and *report* rules are identical for *unfiltered* and *filtered* commands, except the outputs go into separate subdirectories. In addition to the 21 pseudo-rules (3 denoising methods with 7 pseudo-rules each), there are 47 regular rules defined in *Snakefile* that perform the actual QIIME 2, Python, and shell commands of the workflow (Supplementary Fig. S1).

Test data set

Tourmaline comes with a test data set of 16S rRNA gene (bacteria/archaea) amplicon data from surface waters of Western Lake Erie in summer 2018 (see Methods). The sequence data were subsampled to 1,000 sequences per sample to allow the entire workflow to run in ~10 minutes. This test data set is used throughout the article to demonstrate the capabilities of Tourmaline.

Documentation

Full instructions for using the Tourmaline workflow, including installation, cloning, and editing the config file, are described in the Tourmaline Wiki at [44]. Some experience with the command line, QIIME 2, and Snakemake is helpful to use Tourmaline; basic tutorials for each of these are provided at [45].

Installation

The workflow requires QIIME 2 (version 2021.2) plus several dependencies, which can be installed natively in a Conda environ-
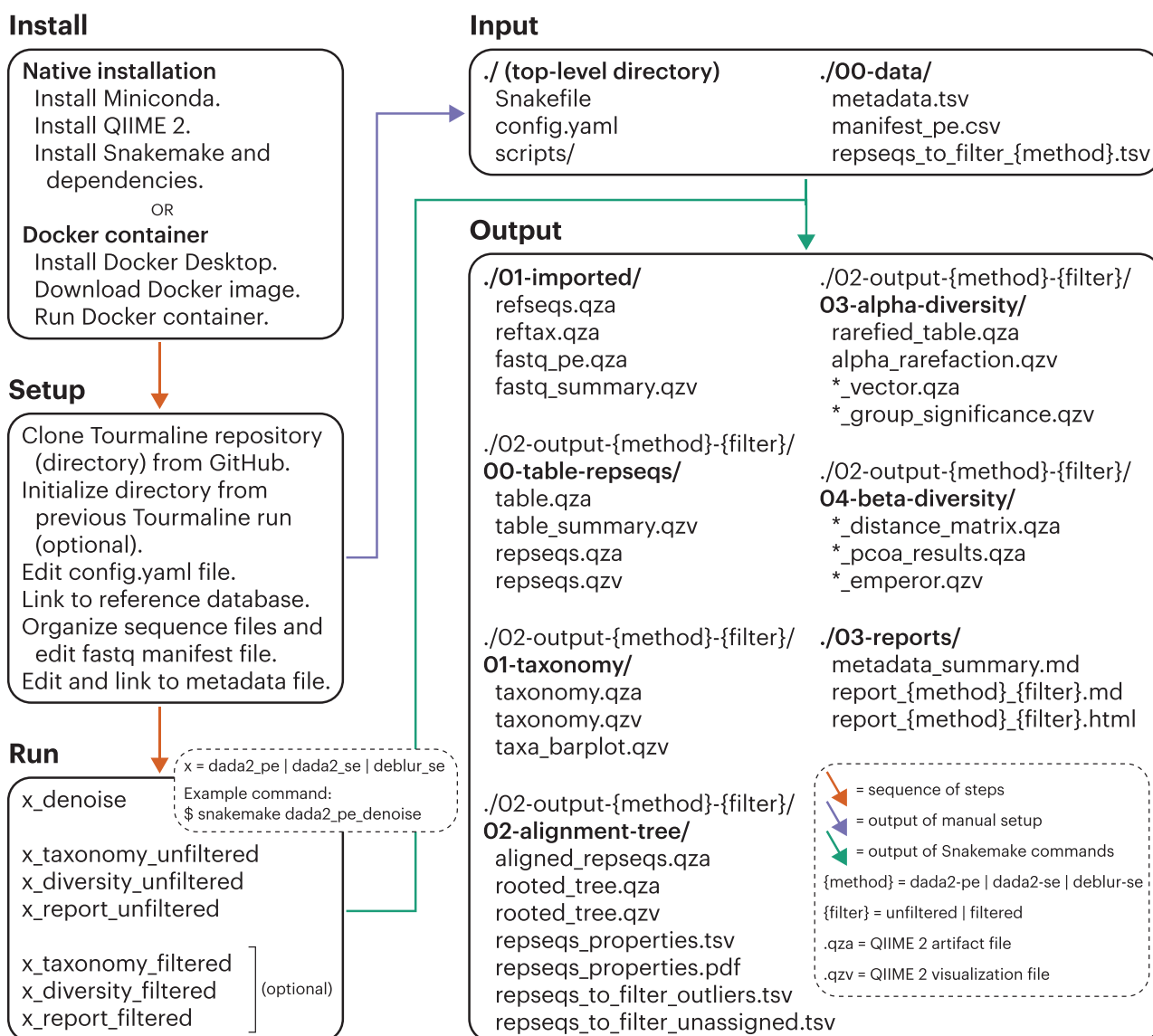
Figure 1: The Tourmaline workflow. Install natively (macOS, Linux) or using a Docker container. Set up by cloning the Tourmaline repository (directory) from GitHub, initializing the directory from a previous run (optional), editing the configuration file (*config.yaml*, Supplementary Table S1), creating symbolic links to the reference database files, organizing the sequence files and/or editing the FASTQ manifest file, and editing and creating a symbolic link to the metadata file. Run by calling the Snakemake commands for *denoise*, *taxonomy*, *diversity*, and *report*—or running just the *report* command to generate all output if the parameters do not need to be changed between individual commands. It is recommended but not required to run the *unfiltered* commands before the *filtered* commands. The primary input and output files are listed. Detailed instructions for each step are provided in the Tourmaline Wiki [44].

ment (instructions at [43]) or via a Docker container using the Docker image from DockerHub [46]. Tourmaline is installed by cloning the GitHub repository to the current directory with *git clone https://github.com/aomlomics/tourmaline*. This step is repeated any time a new iteration of Tourmaline is needed, and new copies can be initialized using a helper script (described below).

Snakefile

As a Snakemake workflow, Tourmaline has as its core files (i) a *Snakefile* that provides all the commands (rules) that comprise the workflow and (ii) a *config.yaml* file that provides the input files and parameters for the workflow. *Snakefile* contains all of the commands used by Tourmaline, which invoke QIIME 2 commands, have helper scripts (see below), or generate output directly. The main analysis features and options supported by Tourmaline, as specified in *Snakefile*, are as follows:

- FASTQ sequence import using a manifest file, or use a preimported FASTQ.qza file
- Denoising with DADA2 [40] (paired-end and single-end) and Deblur [41] (single-end)
- Feature classification (taxonomic assignment) with options of naive Bayes [47], consensus BLAST [48], and consensus VSEARCH [49]
- Feature filtering by taxonomy, sequence length, feature ID, and abundance/prevalence
- *De novo* multiple sequence alignment with MUSCLE [50], Clustal Omega [51], or MAFFT [52] (with masking) and tree building with FastTree [53]
- Outlier detection with odseq [54]

- Interactive taxonomy barplot
- Tree visualization using Empress [55]
- Alpha diversity, alpha rarefaction, and alpha group significance with four metrics: number of observed features, Faith's phylogenetic diversity, Shannon diversity, and Pielou's evenness
- Beta diversity distances, principal coordinates, Emperor [56] plots, and beta group significance (1 metadata column) with 4 metrics: unweighted and weighted UniFrac [57], Jaccard distance, and Bray–Curtis distance
- Robust Aitchison PCA (principal component analysis) and biplot ordination using DEICODE [58]

Config file

The configuration file *config.yaml* includes paths to input files and parameters for QIIME 2 commands and other steps. Default settings have been chosen to balance run performance and accuracy and to work with the test data. For user data, all parameters should be checked and possibly adjusted for appropriateness with the data set; see Supplementary Table S1, Fig. 1, and the Wiki section *Setup* for guidance.

Input files

Tourmaline requires 3 categories of input files: (i) Reference database: a FASTA file of reference sequences (*refseqs.fna*) and a tab-delimited file of taxonomy (*reftax.tsv*) for those sequences, or their imported QIIME 2 artifact equivalents (*refseqs.qza*, *reftax.qza*); (ii) Amplicon data: demultiplexed FASTQ sequence files and FASTQ manifest file(s) (*manifest_pe.csv*, *manifest_se.csv*) mapping sample names to the location of the sequence files, or their imported QIIME 2 equivalents (*fastq_pe.qza*, *fastq_se.qza*); and (iii) Metadata: a tab-delimited sample metadata file (*metadata.tsv*) with sample names in the first column matching those in the FASTQ manifest file. We recommend formatting metadata following the MIMARKS standard [59], and we have done so in the metadata file included with the test data set using the MIMARKS "water" environmental package. See the Wiki section *Setup* for guidance on input file paths and use of symbolic links to avoid storing multiple copies of large input files.

Run the workflow

The workflow is run using Snakemake commands. For example, if using DADA2 paired-end method without any filtering (see below), the commands would be (i) *snakemake dada2_pe_denoise*, (ii) *snakemake dada2_pe_taxonomy_unfiltered*, (iii) *snakemake dada2_pe_diversity_unfiltered*, and (iv) *snakemake dada2_pe_report_unfiltered*. Alternatively, the entire workflow can be run at once with the last command, *snakemake dada2_pe_report_unfiltered*.

Outputs

The outputs of each step of Tourmaline are described following a test run with the Lake Erie test data that come with the GitHub repository. For each command, the main parameters used and list of output files generated in those commands are provided (Fig. 2). Accompanying the list of output files is guidance for evaluating them to choose parameters for subsequent steps (Fig. 2), with screenshots of the Tourmaline-specific output files (Fig. 3) and both QIIME 2 and Tourmaline-specific output files (Supplementary Fig. S3). A video version of the tutorial is also available on YouTube [60].

Denoise

The first command is *snakemake dada2_pe_denoise* (Fig. 2), which imports the FASTQ files and reference database (if not already present in directory *01-imported*), summarizes the FASTQ data, runs denoising using DADA2, and summarizes the output. In addition to QIIME 2 visualizations of the feature table, representative sequences, and phylogenetic tree, Tourmaline generates a table and scatterplot (*repseqs_properties.tsv*, *repseqs_properties_describe.md*, and *repseqs_properties.pdf*; Fig. 3A–D) of representative sequence properties, including sequence length, number of gaps in the multiple sequence alignment, outlier status, taxonomy, and total number of observations in the observation table. Quality control can be performed using *fastq_summary.qzv* (Supplementary Fig. S3A) for quality scores and *reqseqs.qzv* (Supplementary Fig. S3C) or *repseqs_lengths.tsv* for representative sequence lengths. The helper script *fastqc_per_base_sequence_quality_dropoff.py* can be run on the output of FastQC and MultiQC to estimate and set DADA2 or Deblur truncation lengths (see below) and then rerun the *denoise* step. Based on the representative sequence lengths, filtering by sequence length can also be set, to be used later in the *filtered* commands. Choice of appropriate sampling (rarefaction) depths for the parameters "alpha_max_depth" and "core_sampling_depth," to be used in the *diversity* step, can be done by examining *table_summary_features.txt* (Fig. 3E), *table_summary_samples.txt* (Fig. 3F), and *table_summary.qzv* (Supplementary Fig. S3B).

Taxonomy

The second command is *snakemake dada2_pe_taxonomy_unfiltered* (Fig. 2), which assigns taxonomy to the representative sequences using a naive Bayes classifier or consensus BLAST or VSEARCH method and generates an interactive taxonomy table and an interactive barplot of sample taxonomic composition. Choice of taxonomic groups to be filtered by keyword, to be used later with *filtered* commands, can be done by examining *taxonomy.qzv* (Supplementary Fig. S3D) and *taxa_barplot.qzv* (Supplementary Fig. S3E).

Diversity

The third command is *snakemake dada2_pe_diversity_unfiltered* (Fig. 2), which aligns representative sequences using 1 of 3 methods, computes outliers using odseq [54], and builds a phylogenetic tree. This step generates lists of representative sequences that have unassigned taxonomy and were computed to be outliers, summarizes and plots the representative sequence properties, performs alpha rarefaction, and runs alpha diversity and beta diversity analyses and group significance tests using a suite of metrics. Filtering parameters can be checked by examining *rooted_tree.qzv* (Supplementary Fig. S3F) and *repseqs_properties.pdf* (Supplementary Fig. S3G), if desired. Whether sampling depth was sufficient can be evaluated with *alpha_rarefaction.qzv* (Supplementary Fig. S3I). Alpha and beta diversity patterns and statistically significant differences between groups can be evaluated with *observed_features_group_significance.qzv* (Supplementary Fig. S3J; other alpha diversity metrics are also provided), *unweighted_unifrac_emperor.qzv* (Supplementary Fig. S3H; other beta diversity metrics are also provided), and *beta_group_significance.qzv* (Supplementary Fig. S3K).

Report

The fourth and final command is *snakemake dada2_pe_report_unfiltered* (Fig. 2), which creates a comprehensive HTML report of parameters, metadata, inputs, outputs, and visu-

Parameters in config.yaml

Output to evaluate

snakemake dada2_pe_denoise

\# use manifest file to import fastq.gz sequence files
manifest_pe: 00-data/manifest_pe.csv

\# use pre-imported reference database
refseqs_qza: 01-imported/refseqs.qza
reftax_qza: 01-imported/reftax.qza

\# choose dada2 parameters based on fastq error profiles
dada2pe_trunc_len_f: 240
dada2pe_trunc_len_r: 190

fastq_summary.qzv
• median Q-score <30 occurs at fwd. position 267 and rev. position 233
  -> trimming at 240 and 190 is acceptable for this amplicon (<300 bp)
• 16 samples (fwd. & rev.) all have 1000 reads per sample (test dataset)

repseqs.qzv & repseqs_lengths.tsv
• of 301 repseqs, most are 253 bp and max is 255 bp except two that
  are much longer (416 bp, 417 bp) -> filter by length max 260 bp

table_summary.qzv
• of 16 samples, lowest count per sample is 511 -> set core sampling
  depth (rarefaction) to 500 (check again after filtering)

snakemake dada2_pe_taxonomy_unfiltered

\# choose taxonomic classification method (*)
classify_method: consensus-vsearch

taxonomy.qzv
• 10 repseqs are "Unassigned" and 2 repseqs are "d__Eukaryota"
  -> filter by keywords "unassigned,eukaryota"

taxa_barplot.qzv
• the contribution of Unassigned and Eukaryota groups is <10%;
  still want to filter them

snakemake dada2_pe_diversity_unfiltered

\# choose MSA parameters (*)
alignment_method: muscle
alignment_muscle_maxiters: 2
alignment_muscle_diags: -diags

\# choose outlier detection parameters (*)
odseq_distance_metric: linear
odseq_bootstrap_replicates: 100
odseq_threshold: 0.025

\# choose sampling (rarefaction) depth
core_sampling_depth: 500
alpha_max_depth: 500

\# choose beta group significance parameters (*)
beta_group_column: region
beta_group_method: permanova
beta_group_pairwise: --p-pairwise

rooted_tree.qzv
• feature metadata coloring confirms we should filter Unassigned and
  Eukaryota

repseqs_properties.pdf
• confirms we should filter Unassigned and Eukaryota and sequences
  longer than 260 bp; don't need to filter all outliers

alpha_rarefaction.qzv
• observed features plateaus at ~450–500 sequences per sample

observed_features_group_significance.qzv
• difference between regions (Open Water vs. Western Boundary) is not
  significant by Kruskal–Wallis, but filter size is significant

unweighted_unifrac_emperor.qzv
• separation by region (axis 2) and filter size (axes 1 & 2)

beta_group_significance.qzv
• distance based on region is significant

snakemake dada2_pe_report_unfiltered

\# choose theme for html report
report_theme: github

report_dada2-pe_unfiltered.html
• a summary of the results and metadata and links to output files are
  presented in this HTML report

snakemake dada2_pe_report_filtered

\# choose terms to filter from taxonomy   (**)
exclude_terms: unassigned,eukaryota

\# choose repseq length limits
repseq_min_length: 0
repseq_max_length: 260

table_summary.qzv
• of 16 samples, lowest count per sample is 507 -> it was ok to leave
  sampling (rarefaction) depth at 500

rooted_tree.qzv
• feature metadata coloring confirms Unassigned and Eukaryota were
  removed, and tree topology is more homogeneous

repseqs_properties.pdf
• confirms long sequences and Unassigned and Eukaryota were
  removed, resulting in fewer gaps in the multiple sequence alignment

report_dada2-pe_filtered.html
• a summary of the results and metadata and links to output files are
  presented in this HTML report

(*) these steps can be defined before
    starting the workflow, as they do not
    depend on the output of previous steps

(**) all steps are being run at once by
     using the report command

Figure 2: Step-by-step tutorial on Tourmaline using the provided test data, which are subsampled from the 16S rRNA amplicon data of a 2018 survey of Western Lake Erie. Key parameters in *config.yaml* and primary output for each command (pseudo-rule) are listed. Indicated output should be evaluated to determine the appropriate parameters for the next command. Evaluation of the primary outputs and rationale for parameter choice is shown for the test Lake Erie 16S rRNA data that come with the Tourmaline repository. See Supplementary Fig. S3 for screenshots of the primary output files.
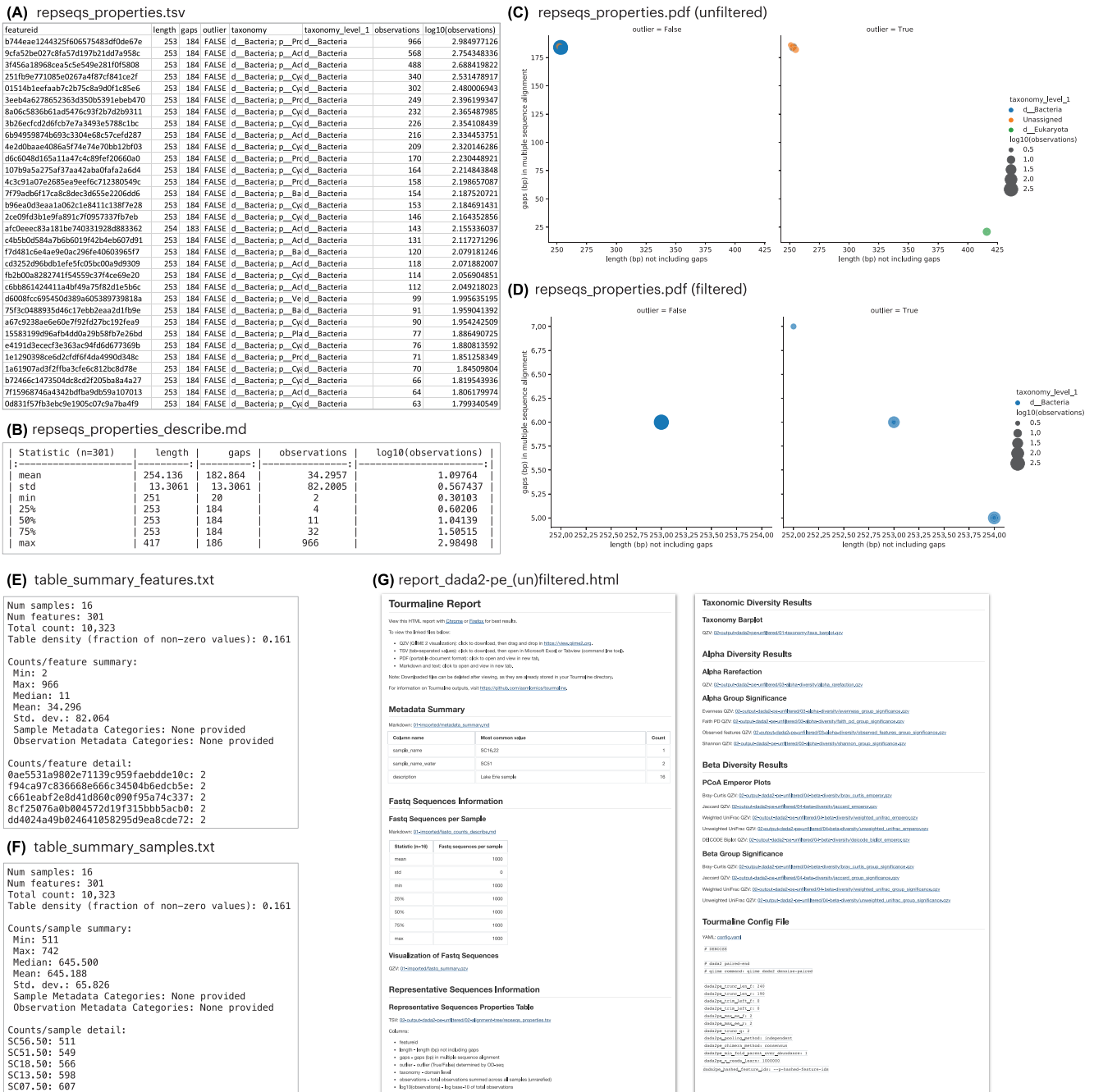
(A) repseqs_properties.tsv

featureid	length	gaps	outlier	taxonomy	taxonomy_level_1	observations	log10(observations)
b744eae1244325f606575483df0de67e	253	184	FALSE	d__Bacteria; p__Pro	d__Bacteria	966	2.984977126
9cfa52be027c8fa57d197b21dd7a958c	253	184	FALSE	d__Bacteria; p__Act	d__Bacteria	568	2.754348336
3f456a18968cea5c5e549e281f0f5808	253	184	FALSE	d__Bacteria; p__Act	d__Bacteria	488	2.688419822
251fb9e771085e0267a4f87cf841ce2f	253	184	FALSE	d__Bacteria; p__Cya	d__Bacteria	340	2.531478917
01514b1eefaab7c2b75c8a9d0f1c85e6	253	184	FALSE	d__Bacteria; p__Pro	d__Bacteria	302	2.480006943
3eeb4a6278652363d350b5391ebeb470	253	184	FALSE	d__Bacteria; p__Pro	d__Bacteria	249	2.396199347
8a06c5836b61ad5476c93f2b7d2b9311	253	184	FALSE	d__Bacteria; p__Cya	d__Bacteria	232	2.365487985
3b26ecfcd2d6fcb7e7a3493e5788c1bc	253	184	FALSE	d__Bacteria; p__Cya	d__Bacteria	226	2.354108439
6b94959874b693c3304e68c57cefd287	253	184	FALSE	d__Bacteria; p__Act	d__Bacteria	216	2.334453751
4e2d0baae4086a5f74e74e70bb12bf03	253	184	FALSE	d__Bacteria; p__Cya	d__Bacteria	209	2.320146286
d6c6048d165a11a47c4c89fef20660a0	253	184	FALSE	d__Bacteria; p__Pro	d__Bacteria	170	2.230448921
107b9a5a275af37aa42aba0fafa2a6d4	253	184	FALSE	d__Bacteria; p__Cya	d__Bacteria	164	2.214843848
4c3c91a07e2685ea9eef6c712380549c	253	184	FALSE	d__Bacteria; p__Cya	d__Bacteria	158	2.198657087
7f79adb6f17ca8c8dec3d655e2206dd6	253	184	FALSE	d__Bacteria; p__Ba	d__Bacteria	154	2.187520721
b96ea0d3eaa1a062c1e8411c138f7e28	253	184	FALSE	d__Bacteria; p__Cya	d__Bacteria	153	2.184691431
2ce09fd3b1e9fa891c7f0957337fb7eb	253	184	FALSE	d__Bacteria; p__Cya	d__Bacteria	146	2.164352856
afc0eeec83a181be740331928d883362	254	183	FALSE	d__Bacteria; p__Act	d__Bacteria	143	2.155336037
c4b5b0d584a7b6b6019f42b4eb607d91	253	184	FALSE	d__Bacteria; p__Act	d__Bacteria	131	2.117271296
f7d481c6e4ae9e0ac296fe40603965f7	253	184	FALSE	d__Bacteria; p__Ba	d__Bacteria	120	2.079181246
cd3252d96bdb1efe5fc05bc00a9d9309	253	184	FALSE	d__Bacteria; p__Cya	d__Bacteria	118	2.071882007
fb2b00a8282741f54559c37f4ce69e20	253	184	FALSE	d__Bacteria; p__Cya	d__Bacteria	114	2.056904851
c6b8861424411a4bf49a75f82d1e5b6c	253	184	FALSE	d__Bacteria; p__Act	d__Bacteria	112	2.049218023
d6008fcc695450d389a605389739818a	253	184	FALSE	d__Bacteria; p__Ve	d__Bacteria	99	1.995635195
75f3c0488935d46c17ebb2eaa2d1fb9e	253	184	FALSE	d__Bacteria; p__Ba	d__Bacteria	91	1.959041392
a67c9238ae6e60e7f92fd27bc192fea9	253	184	FALSE	d__Bacteria; p__Cya	d__Bacteria	90	1.954242509
15583199d96afb4dd0a29b58fb7e26bd	253	184	FALSE	d__Bacteria; p__Pla	d__Bacteria	77	1.886490725
e4191d3ececf3e363ac94fd6d673592b	253	184	FALSE	d__Bacteria; p__Cya	d__Bacteria	76	1.880813592
1e1290398ce6d2cfdf6f4da4990d348c	253	184	FALSE	d__Bacteria; p__Pro	d__Bacteria	71	1.851258349
1a61907ad3f2ffba3cfe6c812bc8d78e	253	184	FALSE	d__Bacteria; p__Cya	d__Bacteria	70	1.84509804
b72466c1473504dc8cd2f205ba8a4a27	253	184	FALSE	d__Bacteria; p__Cya	d__Bacteria	66	1.819543936
7f15968746a4342bdfba9db59a107013	253	184	FALSE	d__Bacteria; p__Cya	d__Bacteria	64	1.806179974
0d831f57fb3ebc9e1905c07c9a7ba4f9	253	184	FALSE	d__Bacteria; p__Cya	d__Bacteria	63	1.799340549

(B) repseqs_properties_describe.md

```
| Statistic (n=301) | length   | gaps     | observations | log10(observations) |
|:------------------|:---------|:---------|:-------------|:--------------------|
| mean              | 254.136  | 182.864  | 34.2957      | 1.09764             |
| std               | 13.3061  | 13.3061  | 82.2005      | 0.567437            |
| min               | 251      | 20       | 2            | 0.30103             |
| 25%               | 253      | 184      | 4            | 0.60206             |
| 50%               | 253      | 184      | 11           | 1.04139             |
| 75%               | 253      | 184      | 32           | 1.50515             |
| max               | 417      | 186      | 966          | 2.98498             |
```

(C) repseqs_properties.pdf (unfiltered)

(D) repseqs_properties.pdf (filtered)

(E) table_summary_features.txt

```
Num samples: 16
Num features: 301
Total count: 10,323
Table density (fraction of non-zero values): 0.161

Counts/feature summary:
 Min: 2
 Max: 966
 Median: 11
 Mean: 34.296
 Std. dev.: 82.064
 Sample Metadata Categories: None provided
 Observation Metadata Categories: None provided

Counts/feature detail:
0ae5531a9802e7139c959faebdde10c: 2
f94ca97c836668e666c34504b6edcb5e: 2
c661eabf2e8d41d860c090f95a74c337: 2
8cf25076a0b004572d19f315bbb5acb0: 2
dd4024a49b024641058295d9ea8cde72: 2
```

(F) table_summary_samples.txt

```
Num samples: 16
Num features: 301
Total count: 10,323
Table density (fraction of non-zero values): 0.161

Counts/sample summary:
 Min: 511
 Max: 742
 Median: 645.500
 Mean: 645.188
 Std. dev.: 65.826
 Sample Metadata Categories: None provided
 Observation Metadata Categories: None provided

Counts/sample detail:
SC56.50: 511
SC51.50: 549
SC18.50: 566
SC13.50: 598
SC07.50: 607
```

(G) report_dada2-pe_(un)filtered.html

Figure 3: Example of the main outputs of the Tourmaline workflow beyond the QIIME 2 outputs. Contents in panels A, E, F, and G are truncated. Screenshots of additional output files are provided in Supplementary Fig. S3. See Fig. 2 for commands, parameters, and guidance.

alizations in a single file. The file *report_dada2-pe_unfiltered.html* (Fig. 3G) can be viewed in a web browser, and the linked output files can be viewed in a browser or downloaded and opened with [61] (.qzv files) or Microsoft Excel (.tsv files). Whether metadata are compliant with metadata standards such as MIMARKS can be easily detected by viewing the metadata summary in the report, which lists each metadata column and its most common value.

Filtering

After reviewing the *unfiltered* results—the taxonomy summary and taxa barplot, the representative sequence summary plot and table, and the list of unassigned and potential outlier representative sequences—the user may wish to filter (remove) certain repre-

sentative sequences by taxonomic group or other properties. This is done by setting the filtering parameters in *config.yaml* and providing a list of any individual representative sequences to filter, then running the *filtered* commands of the workflow: *snakemake dada2_pe_taxonomy_filtered*, *snakemake dada2_pe_diversity_filtered*, and *snakemake dada2_pe_report_filtered* (Fig. 2). Among the *filtered* output, the user can check *table_summary.qzv* (Supplementary Fig. S3L) to ensure that the sampling depth after filtering did not exclude samples and examine *rooted_tree.qzv* (Supplementary Fig. S3N) and *repseqs_properties.pdf* (Supplementary Fig. S3O) to check that the desired representative sequences were filtered. All of the outputs can be viewed by opening *report_dada2-pe_filtered.html* (Supplementary Fig. S3M) in a web browser.

Downstream analysis and meta-analysis

For users who wish to analyze their output further using Jupyter notebooks, we provide Python and R notebooks preloaded with popular data analysis and visualization tools for those platforms. These notebooks come ready to run with Tourmaline output, using relative paths to take advantage of Tourmaline's defined output file structure. The notebooks are shown with the tutorial data set that comes with Tourmaline. We also provide a Python notebook for meta-analysis, containing commands to merge outputs from multiple Tourmaline runs and then perform diversity analyses on the merged files.

Python Jupyter notebook

The Python Jupyter notebook (Supplementary Fig. S2A) uses the QIIME 2 Visualization and Artifact object classes, loading Visualization and Artifact objects from the .qzv and .qza Tourmaline output files. Before running the notebook, the denoising method, filtering mode, and alpha and beta diversity metrics to be used can be specified by changing variable assignments at the beginning of the notebook. The notebook renders Visualization objects for the feature table summary, representative sequences summary, phylogenetic tree, taxonomy, taxa barplot, alpha diversity group significance, and beta diversity principal coordinates analysis (PCoA) Emperor plot. Artifact objects can be viewed as a Pandas [62] DataFrame or Series. The notebook generates Pandas DataFrames for the feature table, taxonomy, reference sequence properties, and metadata, as well as a Pandas Series for alpha diversity. Static plots are generated from some of these tables using Seaborn [63].

R Jupyter notebook

The R Jupyter notebook (Supplementary Fig. S2B) imports Tourmaline artifact (.qza) files using qiime2R [64] and uses common R packages for analyzing and visualizing amplicon sequencing data, including phyloseq [65], tidyverse [66], and vegan [67]. The notebook covers how to import QIIME 2 count and taxonomy artifact files from Tourmaline into an R environment, merge and manipulate the resulting data frames into a single phyloseq object, and estimate and plot diversity metrics and taxonomy barplots of the 16S community using phyloseq and other packages. As with the Python notebook, a set of variables can be specified at the beginning of the R notebook to define specific denoising, filtering, and diversity metrics. After reading in the metadata file and merging to a phyloseq object, we define plotting parameters that can be easily modified by the user to customize the R visualizations.

Meta-analysis notebook

The meta-analysis notebook (Supplementary Fig. S3C) guides the user through running Tourmaline on 2 separate data sets, merging the outputs (feature tables, representative sequences, and taxonomies) and metadata, and performing some basic diversity analyses on the merged output. For simplicity, the 2 data sets are derived from the test data that come with Tourmaline. The commands provided could be applied to any set of Tourmaline outputs that the user wishes to combine in a meta-analysis. The only requirement is that the sequenced region must be the same across the data sets for the results to make sense. This notebook is a simple example that demonstrates Tourmaline's capacity to facilitate merging of outputs and meta-analysis. Many additional analyses are possible on the merged output, such as demonstrated in published microbiome meta-analyses [2, 68].

Helper scripts and parameter optimization

Tourmaline comes with several helper scripts that are run automatically with the workflow or run directly by the user. See the Wiki section *Setup* for more information.

Initialize a new Tourmaline directory

From the main directory of a newly cloned Tourmaline directory, the script *initialize_dir_from_existing_tourmaline_dir.sh* will copy *config.yaml* and *Snakefile* from an existing tourmaline directory, remove the test files, and then copy the data files and symlinks from the existing Tourmaline directory. This is useful when performing a new analysis on the same data set. The user can clone a new copy of Tourmaline, run this script to copy everything from the old copy to the new one, and then make desired changes to the parameters.

Create a FASTQ manifest file

Two scripts help create the manifest file that points Tourmaline to the FASTQ sequence files. (i) *create_manifest_from_fastq_directory.py* creates a FASTQ manifest file from a directory of FASTQ files. (ii) *match_manifest_to_metadata.py* takes an existing FASTQ manifest file and generates 2 new manifest files (paired-end and single-end) corresponding to the samples in the provided metadata file.

Determine optimal truncation length

If FastQC and MultiQC have been run for Read 1 and Read 2, *fastqc_per_base_sequence_quality_dropoff.py* will determine the position where median per-base sequence quality drops below some fraction (default: 0.90) of its maximum value. This is useful for defining 3′ truncation positions in DADA2 and Deblur ("dada2pe_trunc_len_f", "dada2se_trunc_len", and "deblur_trim_length").

Parameter optimization

The helper scripts and Tourmaline's defined directory structure enable testing and comparison of different parameter sets to optimize a workflow. By making multiple copies of the directory and populating settings with *initialize_dir_from_existing_tourmaline_dir.sh* script, varying one or a small number of parameters, and running the workflow multiple times in parallel, outputs can be compared visually or programmatically to see the effects of parameter choices and choose a final set. To illustrate this, we analyzed the full data set of the 2018 Lake Erie 16S rRNA study (BioProject PRJNA679730 [69]). Running *fastqc_per_base_sequence_quality_dropoff.py* had suggested that a forward truncation length of 240 bp and reverse truncation length of 190 bp would strike a balance between sequence length and quality, but we wanted to test a full range of truncation lengths. We tested the effects of varying the forward and reverse truncation lengths from 100 bp to 250 bp in 50-bp increments on the distribution of representative sequence length (Supplementary Fig. S4A) and the number of reads assigned to Eukaryota (Supplementary Fig. S4B), a group potentially amplified by these primers but with longer representative sequences. This analysis helped choose a set of truncation lengths that would capture a large diversity of target organisms.

Parallelization and benchmarks

Thanks to efforts of developers of QIIME 2 and other software, Tourmaline supports multiple cores in steps that support them, including denoising, feature classification, multiple sequence alignment, tree building, and core diversity calculations. To eval-

uate runtimes with a real-world data set, we ran Tourmaline on the full data set of the 2018 Lake Erie 16S rRNA study [69], which is the data set from which the test data set was subsampled. This data set was sequenced with 2 × 300-bp Illumina MiSeq sequencing and consists of 96 samples having an average of 120,338 paired reads per sample, for a total of 11,552,448 paired reads. Processing was performed using the Tourmaline Docker container running on a 2017 iMac Pro with an 18-core 2.3-GHz Intel Xeon W processor and 64 GB RAM (32 GB RAM allocated for the Docker container). Speed improvements with parallelization were tested by running Snakemake with either 1 or 8 cores (parameter: --cores). Each main step in the workflow (*denoise*, *taxonomy*, *diversity*, and *report*; *unfiltered* commands) was run and timed separately. Times would be expected to be similar for *filtered* commands except that the *denoise* rule does not need to be rerun. The results (Table 1) show that a relatively large data set of ~100 samples with ~100,000 sequences per sample can be processed with a single core in ~5 hours. Dramatic speed improvements are possible with multiple cores, with this same data set being processed in ~2 hours when 8 cores were used.

Biological insights

The purpose of performing amplicon sequencing or metabarcoding is to reveal patterns of diversity, community structure, and biological (or environmental) drivers within diverse ecosystems. Whether the system of study is microbial communities in an environmental or biomedical setting or trace environmental DNA in an aquatic or terrestrial system, the kinds of biological questions being asked are similar. Tourmaline supports biological insight in 2 important ways: (i) by supporting the most popular analysis tools and packages in use today, with capacity to expand as new tools are developed, and (ii) by providing multiple ways to view the output, giving everyone from experts to novices a platform to visualize and query the output.

Through its core QIIME 2 functionality and downstream support for R and Python data science packages, Tourmaline enables analysis of the core metrics of microbial and eDNA diversity: taxonomic composition, within-sample diversity (alpha diversity), and between-sample diversity (beta diversity). Examining our analysis of the tutorial data set (Supplementary Fig. S3), we can see how Tourmaline facilitates insight into Western Lake Erie microbial communities. The interactive barplot (Supplementary Fig. S3E) provides rapid insights: the most abundant bacterial families in the 5.0-μm fraction are Sporichthyaceae and SAR11 Clade III; the most abundant bacterial family in the 0.22-μm fraction is Cyanobiaceae (the toxic cyanobacterial family Microcystaceae is less abundant), with the largest component assigned as chloroplasts, which can be filtered in a subsequent run; at the domain level, a small fraction of unassigned and Eukaryota-assigned sequences are observed, which can also be filtered. The alpha diversity results show that the 5.0-μm fraction has greater within-sample diversity (number of observed features) than the 0.22-μm fraction (Supplementary Fig. S3J) and that this diversity appears to be saturated, with a relatively small sampling depth of ~350 sequences per sample sufficient to observe these values (Supplementary Fig. S3I). However, because a large fraction of the 0.22-μm sequences was identified as chloroplast, filtering out those sequences in a future run would be warranted and provide more accurate diversity results. The beta diversity results show that 16S communities are distinguished both by location (Open Water vs. Western Boundary) and size fraction (0.22-μm vs. 5.0-μm) (Supplementary Fig. S3H). From this simple tutorial data set, we demonstrate the use of Tourmaline to analyze environmental amplicon data, in this case revealing the importance of pore size when filtering water samples for microbial sequencing and the presence of spatial variability (regardless of pore size) among microbial communities in Lake Erie.

The ability to view Tourmaline output files with multiple interfaces provides access to researchers with different backgrounds. For users experienced with the Unix command line, the diverse output file types, organized in a defined directory structure, can be queried and analyzed using a wide array of data science tools; anything that can be done with QIIME 2 output and other common sequence diversity output files types can be done with Tourmaline output. For data scientists most comfortable with Jupyter notebooks, the prebuilt Python and R notebooks come ready to work with Tourmaline output and rapidly enable biological discovery from amplicon data. For casual users, the web-based report and QIIME 2 visualizations provide a user-friendly on-ramp to view and interact with the data. This last mode of interacting with the output opens up amplicon analysis to a wider range of users than is typically possible, from collaborators to students to anyone with limited data science expertise. This increased accessibility can accelerate the pace of discovery by increasing the diversity of researchers able to work with the data.

Conclusions

Tourmaline provides a comprehensive platform for amplicon sequence analysis that enables rapid and iterable processing and inference of microbiome and eDNA metabarcoding data. It has multiple features that enhance usability and interoperability:

- **Portability.** Native support for Linux and macOS in addition to Docker containers, enabling it to run on desktop, cluster, and cloud computing platforms.
- **QIIME 2.** The core commands of Tourmaline, including the DADA2 and Deblur packages, are all commands of QIIME 2, one of the most popular amplicon sequence analysis software tools available. Users can print all of the QIIME 2 and other shell commands of a workflow before or while running the workflow.
- **Snakemake.** Managing the workflow with Snakemake provides several benefits:
  - **Configuration file** contains all parameters in one file, so the user can see what the workflow is doing and make changes for a subsequent run.
  - **Directory structure** is the same for every Tourmaline run, so the user always knows where outputs are.
  - **On-demand commands** mean that only the commands required for output files not yet generated are run, saving time and computation when rerunning part of a workflow.
- **Parameter optimization.** The configuration file and defined directory structure make it simple to test and compare different parameter sets to optimize a workflow.
- **Visualizations and reports—ready to share.** Every Tourmaline run produces an HTML report containing a summary of metadata and outputs, with links to web-viewable QIIME 2 visualization files. Zipped run directories can be shared with collaborators, with relative links in the report allowing easy access to the visualizations and other output files.
- **Downstream analysis.** Analyze the output of single or multiple Tourmaline runs programmatically, with qiime2R in R or

Table 1. Benchmarking and parallel processing results from running the full 2018 Lake Erie 16S rRNA data set through Tourmaline with either 1 or 8 cores using a Tourmaline Docker container allocated with 32 GB RAM running on an 18-core iMac Pro (2017). The Snakemake command used the parameter *--cores 1* or *--cores 8*, and parameters in *config.yaml* specifying the number of threads for individual rules were set to 1 or 8, respectively. Times reported are the elapsed real time between invocation and termination and are reported as HH:MM:SS. Times do not include the initial step of importing FASTQ files into a QIIME 2 archive (*fastq_pe.qza*), which took ~2 minutes. Parameters shown in the last column are those most relevant to the runtimes. Unless otherwise noted, the parameters used were the defaults in *config.yaml*.

Rule	Time (--cores 1)	Time (--cores 8)	Parameters and details
dada2_pe_denoise	02:05:43	00:38:10	method: dada2-pe 96 samples * 120,338 sequences per sample = 11,552,448 total sequences
dada2_pe_taxonomy_unfiltered	01:31:55	00:12:39	classify_method: consensus-vsearch 12,379 representative sequences
dada2_pe_diversity_unfiltered	01:18:09	01:13:49	alignment_method: muscle alignment_muscle_maxiters: 2 alignment_muscle_diags: -diags odseq_distance_metric: linear odseq_bootstrap_replicates: 100 odseq_threshold: 0.025 12,379 representative sequences (lengths: min 240, max 418, avg 258)
dada2_pe_report_unfiltered	00:00:05	00:00:05	–
Total	04:55:52	02:04:43	–

the QIIME 2 Artifact API in Python, using the provided R and Python Jupyter notebooks or other code.

- **Meta-analysis.** The standardized input and output file names and directory structure facilitate meta-analysis of multiple studies that have been analyzed through Tourmaline. The provided meta-analysis Jupyter notebook, written in Python, uses Pandas and the QIIME 2 Artifact API and provides a starting point for combining and coanalyzing the output of multiple Tourmaline runs.

Through its streamlined workflow and broad functionality, Tourmaline enables rapid response and biological discovery in any system where amplicon sequencing is applied, from biomedical and environmental microbiology to eDNA for fisheries and protected or invasive species. The QIIME 2–based interactive visualizations it generates allow users to quickly compare differences between samples and groups of samples in their taxonomic composition, within-sample diversity (alpha diversity), and between-sample diversity (beta diversity), which are core metrics of microbial and eDNA diversity. Tourmaline's unique HTML report and preloaded Jupyter notebooks provide ready access to the output, supporting less-experienced researchers and data scientists alike, and the output files are ready to be loaded into a variety of downstream tools in the QIIME 2 and phyloseq ecosystems. Future improvements to the workflow will include support for new QIIME 2 releases and plugins, better integration with Snakemake, possibly including Conda integration and connecting Snakemake's reporting ability with QIIME 2's provenance tracking, and enhanced support for cloud computing environments. With its existing features that balance usability, functionality, iterability, and scalability, and with continued development with support from the research community, Tourmaline will be a valuable and longstanding tool for amplicon sequence analysis.

Methods

Sample collection and DNA extraction

Water samples were collected using a long-range autonomous underwater vehicle (LRAUV, Monterey Bay Aquarium Research Institute) equipped with a third-generation environmental sample processor (3G-ESP, Monterey Bay Aquarium Research Institute) [70]. For each sample, water was filtered through stacked 5.0-μm (top) and 0.22-μM (bottom) Durapore filters (EMD Millipore, Burlington, MA, USA) held in custom 3G-ESP "archive" cartridges and preserved in-cartridge with RNAlater (Thermo Fisher Scientific, Waltham, MA, USA). DNA extraction was performed using the Qiagen (Qiagen, Hilden, Germany) DNeasy Blood and Tissue kit.

Amplicon sequencing

Extracted DNA was amplified using a BiooScientific NEXTFlex 16S V4 Amplicon-Seq Kit 2.0 (NOVA-520999/Custom NOVA-4203-04) (BiooScientific, Austin, TX, USA). Target-specific regions of the forward and reverse primers in the 16S V4 Amplicon-Seq kit were custom ordered to follow the Earth Microbiome Project 16S Illumina Amplicon Protocol: forward primer 515F 5′-GTGYCAGCMGCCGCGGTAA-3′ [71] and reverse primer 806R 5′-GGACTACNVGGGTWTCTAAT-3′ [72]. 16S rRNA amplicons were pooled and sequenced on an Illumina MiSeq with 2 × 300-bp chemistry at the University of Michigan Advanced Genomics Core [73]. Demultiplexed sequences were deposited in NCBI under BioProject PRJNA679730 [69].

Availability of Supporting Source Code

- Project name: Tourmaline
- Project home page: https://github.com/aomlomics/tourmaline

- Operating systems: macOS (native or Docker), Linux (native or Docker), Windows (Docker)
- Programming language: Python
- Other requirements: Conda or Docker
- License: 3-clause BSD license
- RRID: SCR_022465
- bio.tools ID: tourmaline

Availability of Supporting Data

- The test 16S data set (1,000 sequences per sample) is available directly from the GitHub repository at [43].
- Reference databases are available for 16S rRNA at [74] and for 18S–ITS rRNA at [75].
- Output for the tutorial using the included test data is available from Zenodo at [76].
- A snapshot of the GitHub repository is available from Zenodo at [77].

Additional Files

Supplementary Fig. S1. Directed acyclic graphs (DAGs) of the Tourmaline workflow for the DADA2 paired-end method from start to report with (a) *unfiltered* commands and (b) *filtered* commands. This figure was generated from the test data that comes with the repository by running the commands (a) *snakemake dada2_pe_report_unfiltered --dag | dot -Tpdf -Grankdir=LR -Gnodesep=0.1 -Granksep=0.1 > dag_pe_report_unfiltered.pdf* and (b) *snakemake dada2_pe_report_filtered --dag | dot -Tpdf -Grankdir=LR -Gnodesep=0.1 -Granksep=0.1 > dag_pe_report_filtered.pdf*. For a simpler graph, substitute *--rulegraph* for *--dag* in the above commands.

Supplementary Fig. S2. Screenshots of Tourmaline's included Python and R Jupyter notebooks running the provided test data. Both notebooks are designed to run out-of-the-box with the Tourmaline output from any data set. (A) The Tourmaline Python notebook loads and displays sample metadata, feature metadata (representative sequences properties and taxonomy), static plots generated by Seaborn, and interactive QIIME 2 visualizations. (B) The Tourmaline R notebook demonstrates how to load .qza files (counts and taxonomy) into R, merge files with metadata into a single phyloseq object, and generate high-quality visualizations of community diversity and taxonomy using phyloseq and a suite of tidyverse packages (e.g., ggplot2). (C) The Tourmaline meta-analysis notebook walks through the merging of 2 sets of Tourmaline outputs and performing some basic diversity analyses on the merged files. The number of processed data sets being merged in the meta-analysis can be increased by adding additional inputs to the commands.

Supplementary Fig. S3. Screenshots of the primary output files after running Tourmaline on the test data (see Fig. 2 for commands, parameters, and guidance). The visualization files (.qzv, .pdf, .html) are useful both for data evaluation and discovery and for biological insight.

Supplementary Fig. S4. Effect of truncation length parameters on (A) the distribution of representative sequence length and (B) the number of reads assigned to Eukaryota in the full 2018 Lake Erie 16S rRNA amplicon data.

Supplementary Table S1. Parameters in the configuration file, *config.yaml*, that the user may edit as necessary. Additional parameters not shown may also be edited. The default configuration file is provided in the top level of the GitHub repository. The file format of *config.yaml*, YAML (yet another markup language), is a simple markup language that is used by Snakemake to specify parameters for a workflow.

Abbreviations

ASV: amplicon sequence variant; bp: base pair; COI: cytochrome oxidase I; DAG: directed acyclic graph; eDNA: environmental DNA; ITS: internal transcribed spacer; NMDS: nonmetric dimensional scaling; PCoA: principal coordinates analysis; PCR: polymerase chain reaction; QIIME: Quantitative Insights Into Microbial Ecology.

Competing Interests

The authors declare that they have no competing interests.

Authors' Contributions

The Tourmaline workflow was designed and developed by L.R.T. Code was tested by L.R.T., N.V.P., S.R.A., and S.J.L. The Docker image was built by N.V.P. and L.R.T. Data analysis and visualization of the case study were done by S.R.A. Analysis notebooks were developed by L.R.T., S.R.A., and G.S. Samples were collected by P.A.D.U. and K.D.G. DNA was extracted and prepared for sequencing by P.A.D.U. The manuscript was written by L.R.T., S.R.A., P.A.D.U., S.J.L., and K.D.G.

References

1. The Human Microbiome Project Consortium. Structure, function and diversity of the healthy human microbiome. *Nature* 2012;**486**:207–14.
2. Thompson, LR, Sanders, JG, McDonald, D, *et al.* A communal catalogue reveals Earth's multiscale microbial diversity. *Nature* 2017;**551**:457–63.
3. Deiner, K, Bik, HM, Mächler, E, *et al.* Environmental DNA metabarcoding: Transforming how we survey animal and plant communities. *Mol Ecol* 2017;**26**:5872–95.
4. Compson, ZG, McClenaghan, B, Singer, GAC, *et al.* Metabarcoding from microbes to mammals: comprehensive bioassessment on a global scale. *Front Ecol Evol* 2020;**8**:581835.

5. Ruppert, KM, Kline, RJ, Rahman, MS. Past, present, and future perspectives of environmental DNA (eDNA) metabarcoding: A systematic review in methods, monitoring, and applications of global eDNA. *Global Ecol Conserv* 2019;**17**:e00547.

6. Zaiko, A, Martinez, JL, Schmidt-Petersen, J, *et al.* Metabarcoding approach for the ballast water surveillance—an advantageous solution or an awkward challenge? *Marine Pollution Bull* 2015;**92**:25–34.

7. Ahn, J, Sinha, R, Pei, Z, *et al.* Human gut microbiome and risk for colorectal cancer. *J Natl Cancer Inst* 2013;**105**:1907–11.

8. Turnbaugh, PJ, Ley, RE, Mahowald, MA, *et al.* An obesity-associated gut microbiome with increased capacity for energy harvest. *Nature* 2006;**444**:1027–31.

9. Kartzinel, TR, Hsing, JC, Musili, PM, *et al.* Covariation of diet and gut microbiome in African megafauna. *Proc Natl Acad Sci* 2019;**116**:23588–93.

10. Sunagawa, S, Coelho, LP, Chaffron, S, *et al.* Structure and function of the global ocean microbiome. *Science* 2015;**348**:1261359–9.

11. Abarenkov, K, Nilsson, RH, Larsson, K, *et al.* The UNITE database for molecular identification of fungi—recent updates and future perspectives. *New Phytologist* 2010;**186**:281–5.

12. de Vargas, C, Audic, S, Henry, N, *et al.* Eukaryotic plankton diversity in the sunlit ocean. *Science* 2015;**348**:1261605.

13. Leray, M, Yang, JY, Meyer, CP, *et al.* A new versatile primer set targeting a short fragment of the mitochondrial COI region for metabarcoding metazoan diversity: application for characterizing coral reef fish gut contents. *Front Zool* 2013;**10**:34.

14. Miya, M, Sato, Y, Fukunaga, T, *et al.* MiFish, a set of universal PCR primers for metabarcoding environmental DNA from fishes: detection of more than 230 subtropical marine species. *R Soc Open Sci* 2015;**2**:150088.

15. Halfvarson, J, Brislawn, CJ, Lamendella, R, *et al.* Dynamics of the human gut microbiome in inflammatory bowel disease. *Nat Microbiol* 2017;**2**:17004–4.

16. Thomsen, PF, Willerslev, E. Environmental DNA—an emerging tool in conservation for monitoring past and present biodiversity. *Biol Conserv* 2015;**183**:4–18.

17. Reiter, T, Brooks, PT, Irber, L, *et al.* Streamlining data-intensive biology with workflow systems. *GigaScience* 2021;**10**:giaa140. doi:10.1093/gigascience/giaa140.

18. Harper, LR, Buxton, AS, Rees, HC, *et al.* Prospects and challenges of environmental DNA (eDNA) monitoring in freshwater ponds. *Hydrobiologia* 2019;**826**:25–41.

19. Dickie, IA, Boyer, S, Buckley, HL, *et al.* Towards robust and repeatable sampling methods in eDNA–based studies. *Mol Ecol Resources* 2018;**18**:940–52.

20. Vangay, P, Burgin, J, Johnston, A, *et al.* Microbiome metadata standards: report of the National Microbiome Data Collaborative's workshop and follow-on activities. *mSystems* 2021;**6**. e01194–20. doi:10.1128/msystems.01194-20.

21. Anacapa GitHub repository. https://github.com/limey-bean/Anacapa. Accessed 21 April 2022.

22. Curd, EE, Gold, Z, Kandlikar, GS, *et al.* Anacapa Toolkit: an environmental DNAtoolkit for processing multilocus metabarcode datasets. *Methods Ecol Evol* 2019;**00**:1–7. doi:10.1111/2041-210x.13214.

23. Banzai GitHub repository. https://github.com/jimmyodonnell/banzai. Accessed 21 April 2022.

24. PEMA GitHub repository. https://github.com/hariszaf/pema. Accessed 21 April 2022.

25. Zafeiropoulos, H, Viet, HQ, Vasileiadou, K, *et al.* PEMA: a flexible Pipeline for Environmental DNA Metabarcoding Analysis of the 16S/18S ribosomal RNA, ITS, and COI marker genes. *GigaScience* 2020;**9**. giaa022. doi:10.1093/gigascience/giaa022.

26. nf-core/ampliseq GitHub repository. https://github.com/nf-core/ampliseq. Accessed 21 April 2022.

27. Straub, D, Blackwell, N, Langarica-Fuentes, A, *et al.* Interpretations of environmental microbial community studies are biased by the selected 16S rRNA (gene) amplicon sequencing pipeline. *Front Microbiol* 2020;**11**:550420.

28. CASCABEL GitHub repository. https://github.com/AlejandroAb/CASCABEL. Accessed 21 April 2022.

29. Asbun, AA, Besseling, MA, Balzano, S, *et al.* Cascabel: a flexible, scalable and easy-to-use amplicon sequence data analysis pipeline. bioRxiv 2019;809384. doi:10.1101/809384.

30. Dadasnake GitHub repository. https://github.com/a-h-b/dadasnake. Accessed 21 April 2022.

31. Weißbecker, C, Schnabel, B, Heintz-Buschart, A. Dadasnake, a Snakemake implementation of DADA2 to process amplicon sequencing data for microbial ecology. *GigaScience* 2020;**9**:giaa135.

32. Hupfauf, S, Etemadi, M, Juárez, MF-D, *et al.* CoMA—an intuitive and user-friendly pipeline for amplicon-sequencing data analysis. *PLoS ONE* 2020;**15**:e0243241.

33. ASAP 2. https://hts.iit.edu/asap2. Accessed 21 April 2022.

34. Tian, R, Imanian, B. ASAP 2: a pipeline and web server to analyze marker gene amplicon sequencing data automatically and consistently. *BMC Bioinformatics* 2022;**23**:27.

35. Tagseq GitHub repository. https://github.com/shu251/tagseq-qiime2-snakemake. Accessed 21 April 2022.

36. QIIME 2 GitHub repository. https://github.com/qiime2/qiime2. Accessed 21 April 2022.

37. Bolyen, E, Rideout, JR, Dillon, MR, *et al.* Reproducible, interactive, scalable and extensible microbiome data science using QIIME 2. *Nat Biotechnol* 2019;**37**:852–7.

38. Schloss, PD, Westcott, SL, Ryabin, T, *et al.* Introducing mothur: open-source, platform-independent, community-supported software for describing and comparing microbial communities. *Appl Environ Microbiol* 2009;**75**:7537–41.

39. Boyer, F, Mercier, C, Bonin, A, *et al.* obitools: a unix–inspired software package for DNA metabarcoding. *Mol Ecol Resources* 2016;**16**:176–82.

40. Callahan, BJ, McMurdie, PJ, Rosen, MJ, *et al.* DADA2: High-resolution sample inference from Illumina amplicon data. *Nat Methods* 2016;**13**:581–3.

41. Amir, A, McDonald, D, Navas-Molina, JA, *et al.* Deblur rapidly resolves single-nucleotide community sequence patterns. *mSystems* 2017;**2**:e00191–16. doi:10.1128/msystems.00191-16.

42. Köster, J, Rahmann, S. Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics (Oxford, England)* 2012;**28**:2520–2.

43. Tourmaline GitHub repository. https://github.com/aomlomics/tourmaline. Accessed 21 April 2022.

44. Tourmaline Wiki. https://github.com/aomlomics/tourmaline/wiki. Accessed 21 April 2022.

45. Tutorials GitHub repository. https://github.com/aomlomics/tutorials. Accessed 21 April 2022.

46. Tourmaline Docker container. https://hub.docker.com/repository/docker/aomlomics/tourmaline. Accessed 21 April 2022.

47. Bokulich, NA, Kaehler, BD, Rideout, JR, *et al.* Optimizing taxonomic classification of marker-gene amplicon sequences with QIIME 2's q2-feature-classifier plugin. *Microbiome* 2018;**6**:90.

48. Camacho, C, Coulouris, G, Avagyan, V, *et al.* BLAST+: architecture and applications. *BMC Bioinformatics* 2008;**10**:421–1.

49. Rognes, T, Flouri, T, Nichols, B, *et al.* VSEARCH: a versatile open source tool for metagenomics. *PeerJ* 2016;**4**:e2584.

50. Edgar, RC. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res* 2004;**32**:1792–7.

51. Sievers, F, Higgins, DG. Multiple sequence alignment methods. *Methods Mol Biol* 2014;**1079**:105–16.

52. Katoh, K, Standley, DM. MAFFT Multiple Sequence Alignment Software Version 7: improvements in performance and usability. *Mol Biol Evol* 2013;**30**:772–80.

53. Price, MN, Dehal, PS, Arkin, AP. FastTree: computing large minimum evolution trees with profiles instead of a distance matrix. *Mol Biol Evol* 2009;**26**:1641–50.

54. Jehl, P, Sievers, F, Higgins, DG. OD-seq: outlier detection in multiple sequence alignments. *BMC Bioinformatics* 2015;**16**: 269.

55. Cantrell, K, Fedarko, MW, Rahman, G, *et al.* EMPress enables tree-guided, interactive, and exploratory analyses of multi-omic data sets. *mSystems* 2021;**6**: e01216–20. doi:10.1128/msystems.01216-20.

56. Vázquez-Baeza, Y, Pirrung, M, Gonzalez, A, *et al.* EMPeror: a tool for visualizing high-throughput microbial community data. *GigaScience* 2013;**2**:16.

57. Lozupone, C, Lladser, ME, Knights, D, *et al.* UniFrac: an effective distance metric for microbial community comparison. *ISME J* 2010;**5**:169–72.

58. Martino, C, Morton, JT, Marotz, CA, *et al.* A novel sparse compositional technique reveals microbial perturbations. *mSystems* 2019;**4**:e00016–19.

59. Yilmaz, P, Kottmann, R, Field, D, *et al.* Minimum information about a marker gene sequence (MIMARKS) and minimum information about any (x) sequence (MIxS) specifications. *Nat Biotechnol* 2011;**29**:415–20.

60. Tourmaline Tutorial on YouTube. https://youtu.be/xKfOxrXBXYQ. Accessed 21 April 2022.

61. QIIME 2 View. https://view.qiime2.org. Accessed 21 April 2022.

62. McKinney, W. Data structures for statistical computing in Python. *Proc 9th Python Sci Conf* 2010;**445**:51–6.

63. Qalieh, MW, Botvinnik, O, O'Kane, D, *et al.* https://github.com/mwaskom/seaborn v0.8.1 (September 2017). 2017. doi:10.5281/zenodo.883859.

64. Bisanz, JE. qiime2R: Importing QIIME2 artifacts and associated data into R sessions. 2018. https://github.com/jbisanz/qiime2R.

65. Halfvarson, J, Brislawn, CJ, Lamendella, R, *et al.* Dynamics of the human gut microbiome in inflammatory bowel disease. *Nat Microbiol* 2017;**2**:17004.

66. Wickham, H, Averick, M, Bryan, J, *et al.* Welcome to the tidyverse. *J Open Source Softw* 2019;**4**:1686.

67. Oksanen, J, Blanchet, FG, Friendly, M, *et al.* Package "vegan": Community Ecology Package. 2020. https://cran.r-project.org/web/packages/vegan/index.html.

68. Delgado-Baquerizo, M, Oliverio, AM, Brewer, TE, *et al.* A global atlas of the dominant bacteria found in soil. *Science* 2018;**359**:320–5.

69. BioProject PRJNA679730. https://www.ncbi.nlm.nih.gov/bioproject/?term=prjna679730. Accessed 21 April 2022.

70. Pargett, DM, Birch, JM, Preston, CM, *et al.* Development of a mobile ecogenomic sensor, OCEANS 2015 - MTS/IEEE Washington. 2015, pp. 1–6. doi: 10.23919/OCEANS.2015.7404361.

71. Parada, AE, Needham, DM, Fuhrman, JA. Every base matters: assessing small subunit rRNA primers for marine microbiomes with mock communities, time series and global field samples. *Environ Microbiol* 2016;**18**:1403–14.

72. Apprill, A, McNally, S, Parsons, R, *et al.* Minor revision to V4 region SSU rRNA 806R gene primer greatly increases detection of SAR11 bacterioplankton. *Aquatic Microbial Ecol* 2015;**75**:129–37.

73. BRCF Advanced Genomics Core. https://cores.research.umich.edu/core/brcf-advanced-genomics-core/. Accessed 21 April 2022.

74. QIIME 2 Docs - Data resources. https://docs.qiime2.org/2021.2/data-resources/#silva-16s-18s-rrna. Accessed 21 April 2022.

75. UNITE - Resources. https://unite.ut.ee/repository.php. Accessed 21 April 2022.

76. Thompson L. (2021). Tutorial output for Tourmaline amplicon sequence processing workflow [Data set]. Zenodo. https://doi.org/10.5281/zenodo.5044532.

77. Thompson L. (2022). lukenoaa/tourmaline: Tourmaline release for GigaScience publication (v1.0.0). Zenodo. https://doi.org/10.5281/zenodo.6608988..