Tourmaline MMB UMCU

unknown

May 31, 2023

QUICKSTART GUIDE

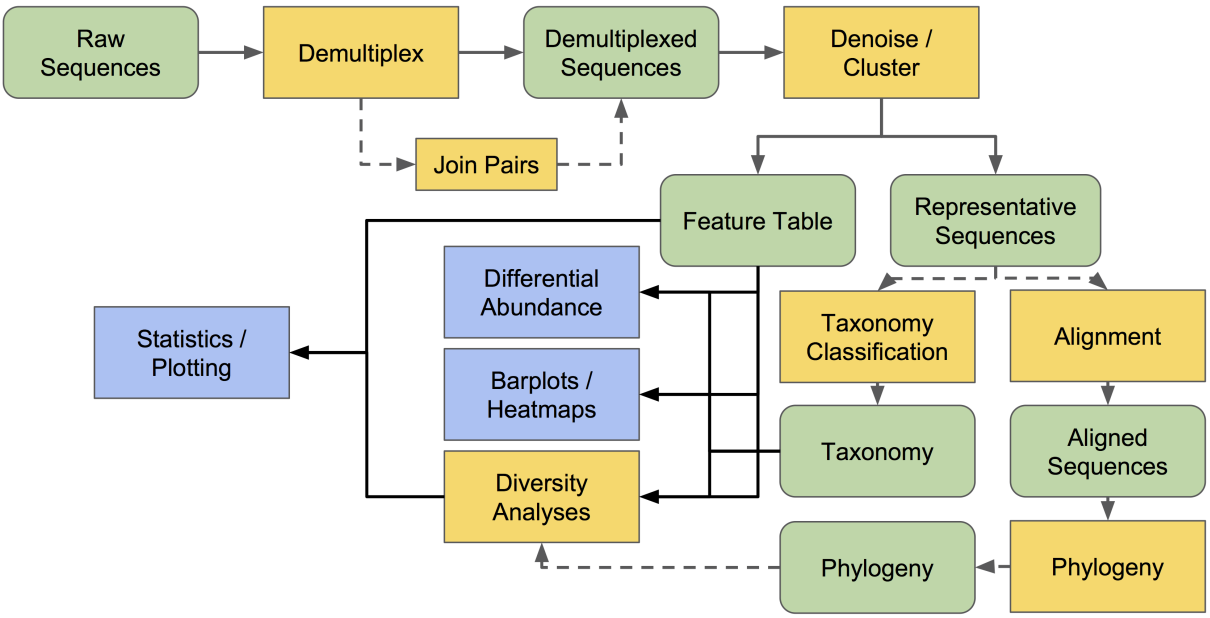
This is the Quick Start Guide on how to run the stable version of the MMB UMCU Tourmaline on the HPC, consisting of a quick overview of QIIME 2, a setup part and instructions of how to run the pipeline.

CHAPTER

ONE

OVERVIEW OF QIIME 2

The core commands of Tourmaline are all commands of QIIME 2, one of the most popular amplicon sequence analysis software tools available. Below is a workflow diagram of QIIME 2. For more in depth information, visit this page.



CHAPTER

TWO

SETUP

Clone the Tourmaline repository into your `data/` folder on the HPC and navigate to the cloned Tourmaline folder. For this purpose you need a gitlab account. Instructions to create an account, see http://143.121.18.159/doku.php?id=intro_to_git_gitlab

```
cd data/
git clone --depth 1 --branch v2.2 https://gitlab.com/malbert.rogers/tourmaline.git␣
↪tourmaline_project_name
cd tourmaline_project_name/
```

Replace 'tourmaline_project_name' with the name of your project.

HAVING TROUBLE WITH GIT CLONE? In case git clone does not work, you could try to download the reporsitory as a zip-file using the Download button (next to the Clone button), transfer this file to the HPC and unzip.

CHAPTER

THREE

RUNNING THE PIPELINE

There are some prerequisites that need to be followed before actually starting the pipeline.

3.1 *Transfer files*

Transfer the readfiles (R1.fastq.gz and R2.fastq.gz) and the metadata file (.txt) to the cloned folder `tourmaline_project_name/` using WinSCP (for instuctions see Transfer Files Using WinSCP) or FileZilla (for MacBook).

3.2 *FastQC*

After transfer of the readfiles to the HPC, it is recommended to first perform a quality control on the reads using FASTQC (for instructions see FastQC for read-quality control), to assess the quality of the readfiles. This will be necessary to determine the truncation values in the config file of the tourmline pipeline.

3.3 *Lay-out metadata and validation*

There are a few prerequisites for the metadata file to avoid errors running the pipeline. In general, if your aim is to run Tourmaline to obtain the taxonomy and feature tables and perform the statistical analysis in R, the advise is to make a metadata file that contains only three columns: 1) header should be 'sample_name' 2) 'BarcodeSequence' and 3) 'any name', this header will be used for the beta_group_column (see below in 'Edit config.yaml'). For column 3 you could use as header e.g. sample_source. The data in this column are not actually used. Note: for the Tourmaline pipeline the LinkerPrimerSequence is not necessary.

Important: don't use special characters, only dots, underscores and dashes are allowed. Validate the metadata file and check for possible warnings/errors:

```
./validate_metadata.py metadata_file_example.txt
```

After validation a new metadata.tsv file has been generated in the tourmaline_project_name/00-data folder. Please check before starting the pipeline. Alternative: in case you are 100% sure that your metadata file doesn't contain special characters, you can skip the validation step and directly transfer the file to the tourmaline_project_name/00-data folder. However, make sure that this file contains an .tsv extension. This folder always already contains a metadata file for the mock R1 and R2.fastq files in case you want to test the pipeline. However, you can overwrite this.

3.4 *Edit config.yaml*

Open and edit the configuration file, config.yaml:

```
nano config.yaml
```

Edit the the values for the following parameters:

```
R1_fastq_file:          ## 1-mock_data_pool_R1.fastq.gz is filled in as default
R2_fastq_file:          ## 1-mock_data_pool_R2.fastq.gz is filled in as default

dada2_trunc_len_f:      ## Truncation value for the original R2 (reverse) readfile,␣
↪default = 225
dada2_trunc_len_r:      ## Truncation value for the original R1 (forward) readfile,␣
↪default = 245

beta_group_column:      ## Any header of the metadata file, default = Type
```

- R1_fastq_file: forward (R1) readfile name including all extensions (example_L001_R1_001.fastq.gz)
- R2_fastq_file: reverse (R2) readfile name including all extensions (example_L001_R2_001.fastq.gz)
- dada2_trunc_len_f: truncation value for the R2 (reverse) readfile. This should be determined based on the results from a FastQC report. All bases after this value will be trimmed for all R2 reads. Please note that the reads in the FastQC report still contain barcode and primer sequences, while at this point in the Tourmaline pipeline both were already removed. Consequently, the reads would be approximately 36 bases (+/- 3) shorter from the front, resulting in a total length of around 264 bases.
- dada2_trunc_len_r: truncation value for the R1 (forward) readfile. This should be determined based on the results from a FastQC report. All bases after this value will be trimmed for all R1 reads. Please note that the reads in the FastQC report still contain barcode and primer sequences, while at this point in the Tourmaline pipeline both were already removed. Consequently, the reads would be approximately 36 bases (+/- 3) shorter from the front, resulting in a total length of around 264 bases.
- beta_group_column: a column header of the metadata file. This is required for the pipeline to run. You could pick a random column header or a column header of interest, if you would also like alpha- and beta-diversity analyses performed (on this specific column).

Optional: In the case that you would also like Tourmaline to generate alpha- and beta-diversity outputs, it's then recommended to also adapt the values for following parameters (more information on these parameters can be found in the configuration file):

```
core_sampling_depth:    ## Minimum number of sequences required for a sample to be␣
↪included in further diversity analyses. Default = 500
alpha_max_depth:        ## Maximum number of sequences for alpha diversity calculations.
↪ Also used as a maximum for the rarefaction curve. Default = 10000
```

Save and quit nano. You do this by pressing Ctrl + X (to quit), then press Y (to save) and finally press ENTER.

3.5 *Run Tourmaline*

Now the pipeline can be executed. For QIIME2 analyses to obtain features-tables and taxonomy* outputs, run the following command (see below for other possible options):

```
./sbatch_tourmaline.sh taxonomy
```

*Input options for running the pipeline are:

1. *demux*: only demultiplex the sequencing data, generating seperate fastq-files for each sample (recommended as a first step for projects' data that are split over multiple sequencing runs);
2. *denoise*: imports FASTQ data and runs denoising, generating a feature table and representative sequences;
3. *taxonomy*: assigns taxonomy to representative sequences (recommended option to run in most cases);
4. *diversity*: step does representative sequence curation, core diversity analyses, and alpha and beta group significance;
5. *report*: step generates an HTML report of the outputs plus metadata, inputs, and parameters. Also, the *report* step can be run immediately to run the entire workflow.

CHAPTER

FOUR

DESCRIPTION

For certain projects it might not be possible to run all the samples in a single sequence run. The samples are then divided over multiple sequence runs. The main problem here is however, the use of the same barcodes in these sequencing runs and the generation of the feature tables. This makes it necessary for a slightly different approach to analyzing the data using Tourmaline. In short, for each sequence run a seperate Tourmaline will be cloned to run only the demux step of Tourmaline. Next, the demultiplexed_renamed fastq files from run 1 are moved and added to the folder containing the demultiplexed_renamed fastq files from run 2. In addition, the metadata files from run 1 and 2 will be combined and copied to the 00-data folder. After these steps are completed the remaining of the pipeline can be run.

CHAPTER

FIVE

INSTRUCTIONS

For each sequence run, follow the set up in the Quick Start Guide and run (`./sbatch_tourmaline.sh demux`) in seperate cloned tourmaline folders:

1. Combine the fastq-files located in `00-data/fastq/demultiplexed_renamed/` in one of the run's demultiplexed_renamed folder. Example:

```
mv tourmaline_project_name_run1/00-data/fastq/demultiplexed_renamed/*.fastq.gz␣
↪tourmaline_project_name_run2/00-data/fastq/demultiplexed_renamed/
```

In this example, because the readfiles were moved from tourmaline_project_name_run1 to ``tourmaline_project_name_run2``, this should then also be the folder in which to execute further QIIME2 analyses using the Tourmaline pipeline (example shown below at step 3).

2. Combine the validated metadata.tsv files of the different runs and copy that to '00-data/metadata.tsv' *In the example, copy to tourmaline_project_name_run2/00-data

3. Run the Tourmaline pipeline (samples from the different runs will be analyzed together). Example:

```
./sbatch_tourmaline.sh taxonomy
```

CHAPTER

SIX

OUTPUT

After completion of a Tourmaline run, a number of different output folders and files will appear in the directory. In general, the most important files for further downstream analyses are located in the `02-output-dada2-pe-unfiltered` folder. This folder should contain, a.o.:

- ASV tables in tsv-format (`02-output-dada2-pe-unfiltered/00-table-repseqs/table*.tsv`)
- Taxonomy table in tsv-format (`02-output-dada2-pe-unfiltered/01-taxonomy/taxonomy.tsv`)
- QIIME2 visualization files (QZV-files).

TIP: A good way to quickly evaluate the results, would be to first have a look at the file `02-output-dada2-pe-unfiltered/00-table-repseqs/dada2_stats.qzv`. This file contains an overview on the different filtering steps that were performed. A low percentage of input that passed filtering could be an indication that the truncation settings were not optimal. In this case it is advisable to check the fastqc output files again. See below for instuctions on how to view these type of files.

CHAPTER

SEVEN

VIEWING QZV-FILES

To view .qzv output (QIIME 2 visualization) files, drag and drop these in https://view.qiime2.org. Empress trees (i.e. rooted_tree.qzv) may take more than 10 minutes to load. For a lot of visualization files in table format it is also possible to download these as tsv-files by clicking "Download metadata TSV file" in the top left corner.

CHAPTER

EIGHT

RUN DURATION

The whole workflow with test data should take ~30 minutes to complete. A normal dataset may take several hours to complete depending on the size and amount of samples.

CHAPTER

NINE

RE-RUNNING TOURMALINE

If you want to re-run and not save previous output, run the following command (while inside your project folder) to remove ALL previously generated output:

```
srun --pty bash
./scripts/remove_output.sh
```

Then re-run Tourmaline as before. Example:

```
./sbatch_run_tourmaline taxonomy
```

CHAPTER

TEN

TOURMALINE FEATURES

Tourmaline has several features that enhance usability and interoperability:

- **Portability.** Native support for Linux and macOS in addition to Docker containers.
- **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. You can print all of the QIIME 2 and other shell commands of your 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 you can see what your workflow is doing and make changes for a subsequent run.
  - **Directory structure** is the same for every Tourmaline run, so you always know where your outputs are.
  - **On-demand commands** mean that only the commands required for output files not yet generated are run, saving time and computation when re-running part of a workflow.
- **Parameter optimization.** The configuration file and standard directory structure make it simple to test and compare different parameter sets to optimize your workflow. Included code helps choose read truncation parameters and identify outliers in representative sequences (ASVs).
- **Visualizations and reports.** Every Tourmaline run produces an HTML report containing a summary of your metadata and outputs, with links to web-viewable QIIME 2 visualization files.
- **Downstream analysis.** Analyze the output of single or multiple Tourmaline runs programmatically, with qiime2R in R or the QIIME 2 Artifact API in Python, using the provided R and Python notebooks or your own code.

CHAPTER

ELEVEN

TOURMALINE

Tourmaline is an amplicon sequence processing workflow for Illumina sequence data that uses QIIME 2 and the software packages it wraps. Tourmaline manages commands, inputs, and outputs using the Snakemake workflow management system.

This version (Tourmaline-MMB-UMCU) is an adaptation of Tourmaline *developed specifically for the Medical Microbiology Department at the UMC Utrecht. The Tourmaline-MMB-UMCU has been adapted for the use of the 16S rRNA dual-index primers as developed by Fadrosh et al. (Microbiome, 2014) and uses not yet demultiplexed fastq files as input.*

11.1 Getting Started

The *Quick Start Guide* is a great place to start, containing short descriptions of the different steps needed to run the pipeline along with some example commands. In case you have multiple sequencing runs that you would like to merge, jump to *Merging Sequencing Runs*.

11.2 Citing Tourmaline

The Tourmaline paper is published in GigaScience:

Thompson, L. R., Anderson, S. R., Den Uyl, P. A., Patin, N. V., Lim, S. J., Sanderson, G. & Goodwin, K. D. Tourmaline: A containerized workflow for rapid and iterable amplicon sequence analysis using QIIME 2 and Snakemake. GigaScience, Volume 11, 2022, giac066, https://doi.org/10.1093/gigascience/giac066