MetaFlow|mics

Release 0.1

Dec 17, 2022

Contents:

CHAPTER 1

Cite

Cédric Arisdakessian, Sean B. Cleveland, and Mahdi Belcaid. 2020. MetaFlow|mics: Scalable and Reproducible Nextflow Pipelines forthe Analysis of Microbiome Marker Data. InPractice and Experience in Advanced Research Computing (PEARC '20), July 26–30, 2020,Portland, OR, USA.ACM, New York, NY, USA, 9 pages.

CHAPTER 2

Description

This repository is a collection of tools for 16S and ITS data analysis and was developed to support scientifics at the Pacific Biosciences Research Center to analyze microbial data. This work was funded by the Hawaii Data Science Institute.

Each analysis tool has been implemented using Nextflow to facilitate their use on any environment, and more specifically on a High Performance Computing Cluster or a Cloud service (GCP, AWS, . . .). Some configurations are already available in the *conf* folder (local, hpcc, google cloud). These configurations can be expanded for any other platform.

You will find three available app:

1. A demultiplexing app for single or paired barcodes.
2. A 16S pipeline for bacterial reads
3. An ITS pipeline for fungal reads

See the documentation for more details.

CHAPTER 3

Contribute

- Issue tracker: GitHub
- Source Code: GitHub

CHAPTER 4

License

Apache 2.0

4.1 Getting Started

4.1.1 Downloading the pipelines

To download the pipelines, simply clone the repository:

```
git clone https://github.com/hawaiidatascience/metaflowmics
cd metaflowmics/metaflowmics
```

4.1.2 Usage

Each pipeline work the same way. To get the available options, you can display the help using the following command (where <pipeline-name> refers to one of the 3 folders in *metaflowmics/metaflowmics*):

```
nextflow run <pipeline-name> --help
```

When running any of the pipelines, you will need to provide a few mandatory arguments:

1. The path to the input reads (see corresponding section for more details)
2. The profile defined in conf/*.config.

4.1.3 Choosing the pipeline parameters

Using the command line

You can change the parameters for a given pipeline by adding flags when running the pipeline. For instance, if you want to change the parameter *reads*, you can use the following syntax:

```
nextflow run Pipeline-16S --reads "/data/Hiseq01/demultiplexed/*_R{1,2}.fastq.gz"
```

The only exception is the *profile* parameter that only requires a single hyphen (i.e. -profile <profile_name>)

Using the configuration file

Alternatively, you can change the parameters value directly in the *.nextflow.config* file located in each application folder.

4.1.4 Dependencies

Docker / Singularity configuration (recommended)

Each of the available pipelines are were built using multiple computing languages and softwares. The dependencies are specified in the appropriate sections. Since setting up these environment can be time time consuming, and sometime complicated depending on the platform, we provide a unified docker instance for all pipelines. This instance can be then used by *Docker* or *Singularity* (if the user does not have sudo rights). This is th recommened way to use the pipeline.

The requirements to run the pipeline with the docker instance are minimal:

- *Bash(>=3.2)*
- java 8 up to 11
- Nextflow (>=19.10)
- Singularity can be compiled from source in the GitHub releases or installed via *apt-get* in the NeuroDebian repository.
- For the 16S and ITS pipelines, downloading the microbial / fungal database is also required. Refer to the corresponding sections for more information.

Custom configuration

Alternatively, you can install all required dependencies on your machine. The specific requirements for each pipeline is specified on the corresponding documentation.

4.1.5 Configuration profiles

To facilitate the configuration of the pipeline, run profiles are available in the *conf* folder for multiple use cases:

- The *singularity* and *docker* profile to run the pipeline locally
- The *local* profile does not use the docker instance and therefore requires the machine to have all the required packages and softwares.
- The *hpc* profile is set up for the slurm scheduler and uses the singularity image. This profile requires *singularity* to be installed on the HPCC as a module. See the *Configure for a HPCC* section for more information.
- The *gcp* profile to run the pipelines on the Google Cloud Platform. See the *Configure for Google Cloud Platform (GCP)* section for more information.

To select a given configuration, you just need to append *-profile <profile_name>* to the command. For custom configuration, see the *Computing resources* section.

4.2 Read demultiplexing

4.2.1 Pre-requisites

- Nextflow
- python(>=3.6) + libraries: Biopython, pandas, numpy, matplotlib, seaborn, argparse

4.2.2 Usage

Clone the repository:

```
git clone https://github.com/hawaiidatascience/nextflow_cmaiki.git
    cd nextflow_cmaiki/metagenomics-pipelines
```

Running the pipeline

To run the pipeline on your data, simply enter the following command:

```
nextflow run multithreaded-demux -profile <config> --inputdir "<path_to_reads>"
```

For more information about the available profiles, see the profiles section.

4.2.3 Demultiplexing steps

The demultiplexing pipeline processes the raw fastq sequences, splits them into separate samples and provides useful plots to facilitate the downstream analysis. The goal is to map each index read to a sample name. In the case of single barcoded reads, we only need to map one index to one barcode while in the paired case, both indexes need to map two associated barcodes. This algorithm uses a similar approach as done in Dada2 in that it builds an error model to calculate transition probability between 2 nucleotides using the illumina quality score. Using this model, the algorithm assign the index to the barcode with the highest likelihood. A hard threshold of 3 mismatches is set so that if an index has a distance of more than 3 to the most likely barcode, the index is dropped.

The demultiplexing algorithm is summarized below. Values in between "<" ">" correspond to default values of tunable parameters.

Inputs

This algorithm is expecting 3 to 5 input files for demultiplexing depending on whether: - Your reads are single-end or paired-end - Your reads have single or paired index

- 1 or 2 index fastq files (unzipped) matching the glob pattern **_I{1,2}*.fastq**
- 1 or 2 read fastq files (unzipped) matching the glob pattern **_R{1,2}*.fastq**
- 1 barcode file (extension: .csv), comma separated, with no header and 2 or 3 columns: (sample name, forward barcode and the reverse complement of reverse barcode for paired index)

Matching order prediction (paired indexes only)

Due to the variability of Illumina protocols, index pairs and barcodes pairs can match in-order or in reverse order. In other words, for a given sequencing protocol, two paired index reads *(i1, i2)* and a matching barcode pair *(b1, b2)*, we can either have an in-order match *(i1, i2) = (b1, b2)* or a reversed match *(i1, i2) = (b2, b1)*. The first step of the pipeline consists in guessing this association by extracting the most frequent index pairs, and comparing it to the barcodes pair. The idea is that errors happen randomly at a low rate, and thus the most frequent pairs should be real barcodes with no errors. If the most frequent in-order pairs are among the barcodes, then we process with the in-order mapping, and with the reverse otherwise.

Error model

In order to determine the transition probability from a barcode sequence to an index read, we need to compute the probability of each of the 4 possible nucleotides (A, C, G, T) to transition to one of the 5 possibilities (A, C, G, T, N). Illumina provides quality scores linked to this probability, but it is known to be highly variable. Therefore, we build error models for the forward and reverse indexes, in a similar fashion as done in [DADA2 Ref]. For each index read, we extract the barcodes with less than 2 mismatches, and store the transitions between each nucleotide pair at each position along with the illumina quality score. We repeat this procedure until 100k transitions are observed. We convert the origin x mutated x quality transition matrix (4 x 5 x 40) to probabilities so that the scores from an origin to a mutated nucleotide for a given score sums up to 1. To smooth the transitions and fill the unobserved transitions, we interpolate the transitions by fitting an isotonic model for each (origin, mutated) nucleotide pair. The isotonic model makes sure the curve is monotonic, and is therefore more robust to outliers than a polynomial regression as it is done is DADA2.

Index - Barcode matching

This step is done on concurrent processes by splitting the original file in multiple files of fixed size (default: 100k). We compare each index read with all barcodes and extract the most likely match. To compute this probability, we use the above-mentioned error model. The probability is simply the product each transition probability between each nucleotide pair between the index and the barcode, and keep the best barcode match if there are less than 3 nucleotide mismatches between the two. Once the mapping is done, we simply move the sequencing read to their corresponding sample file. The sequencing reads are split in the same manner as the index reads to run the demultiplexing on concurrent processes.

Figures

The pipeline provides two useful figures. First, we run the FASTQC tool to provide the overall quality distribution at each read position, therefore providing the necessary information to choose the right truncation parameters when quality filtering the reads:

The second plot shows the sample size distribution (log transformed) which can be used to determine a subsampling threshold later in the analysis (see the subsampling section):

4.3 Microbial 16S pipeline

This pipeline

4.3.1 Pre-requisites

- Nextflow

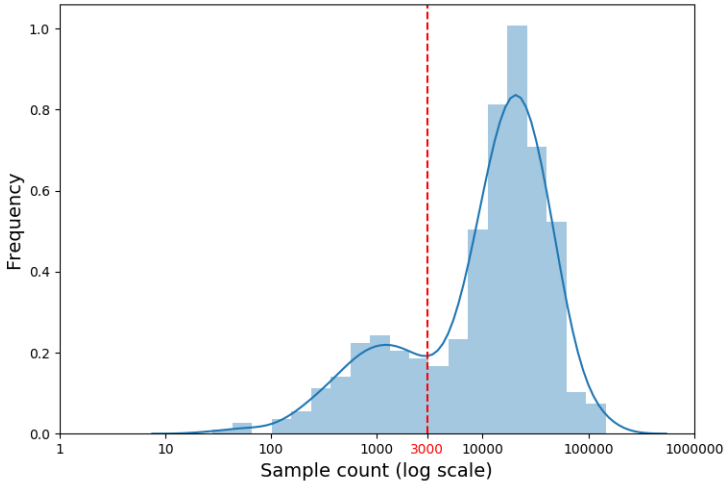

Fig. 1: Distribution of log sample sizes

- Docker or Singularity

If using the available containers, no additional tool is required. However, in case such containerization tool are not available, the following dependencies need to be installed:

- python(>=3.6) + libraries: biopython, pandas, matplotlib, seaborn, bokeh, h5py, scikit-learn, scipy
- R(>=3.5) + libraries: ggplot2, lulu, dada2, seqinr, stringr, ShortRead, doParallel, ape, phyloseq, vegan, igraph
- Mothur (tested with v1.44)

For more details, see the dockerfile.

Download the software

Clone the repository:

```
git clone https://github.com/hawaiidatascience/metaflowmics.git
```

Silva Database

In addition, you will need to download the Silva reference database available on the Mothur website:

```
mkdir -p databases/silva/v138 && cd databases/silva/v138
wget https://mothur.s3.us-east-2.amazonaws.com/wiki/silva.nr_v138.tgz
tar -xvzf silva.nr_v138.tgz && rm -f silva.nr_v138.tgz
```

For species assignments, the database is available on Zenodo (silva_species_assignment_v138.fa.gz)

4.3.2 Usage

To run the pipeline on your data, simply enter the following command:

```
nextflow run 16S-pipeline -profile <config> --reads "<path_to_reads/glob_pattern>" --
↪referenceAln databases/silva/v138/silva.nr_v138.align --referenceTax databases/
↪silva/v138/silva.nr_v138.tax
```

The input reads need to be in the *.fastq* format (preferably gzipped) in a single folder. Reads can be single or paired-end. In the former case, the flag *–singleEnd* must be set and in the latter case, the glob pattern needs to group the R1 and R2 reads using the syntax **R{1,2}**.

For more information about the available profiles, see the profiles section.

4.3.3 Outputs

After running the pipeline, the results are stored in the output directory you chose (default is 16S_pipeline_outputs). The results are organized the following way:

- The folder *Results/* groups all of pipeline final outputs:
  - The subfolder *figures/* provides useful insights into the data outputs.
  - The subfolder *main/* contains the pipeline data that is ready for further analysis, and includes:
    - The abundance table: A (sample x OTU) abundance table (tsv formatted) with the *.shared* extension.
    - The taxonomy table: A (OTU x 6 taxonomic tanks) table (tsv formatted) with the *.taxonomy* extension.
    - The OTU sequences: A fasta formatted file with the representative sequences for each OTU.
    - Optionally, a mothur "database" file that combines all of the previous 3 files and a *.biom* file.
  - The subfolder *raw/* contains the same information as *main*, before the subsampling, taxa filtering, multipletons filter and lulu steps.
  - The subfolder *postprocessing/* provides alpha and beta diversity metrics. For more details, see 16S post-processing.
- The folder *Misc/* is mainly for troubleshooting purposes. It groups all of the intermediate results from the pipeline, one sub-folder per step.

4.3.4 16S pipeline steps

The 16S analysis pipeline is summarized below. Values in between "<" ">" correspond to default values of tunable parameters.

Inputs

Reads need to be demultiplexed and gzipped

Read filtering and denoising (Dada2)

Reads are truncated at positions <*220*> / <*190*> or at the first occurrence of a base of quality <*2*> or lower. Reads matching the phiX genome are <*discarded*>, as well as reads with an expected number of errrors above <*3*>. Reads shorter than <*20bp*> are filtered out. Finally, samples with less than <*50*> reads are discarded. Error models and denoising are then performed on each sample independently

Read merging (Dada2)

Paired reads are merged if they overlap by at least <*20bp*> with <*1bp*> mismatch at most.

Contig filtering (Mothur)

Contigs are aligned against the silva reference database. Discard any sequence with an alignment shorter than <*50bp*>, as well as sequences starting after where <*95%*> of the sequences start, or end before <*95%*> of the sequences end.

Chimeric contigs are removed using Mothur's implementation of VSEARCH.

OTU clustering (Mothur)

OTU are clustered at similarity levels <*100%, 97%*> (100% means no clustering).

Taxa filter

Lineages are assigned to each individual sequence using the SILVA reference database. Any sequence matching <*mitochondria, chloroplasts, unknown*> annotations are removed.

Multipletons filter

OTU with a total abundance of <*2*> or below are discarded.

Subsampling

We perform sample normalization by subsampling each sample to the same level. Samples with a size below this level are discarded. By default, the subsampling level is defined as the <*10th*> percentile of the sample sizes, and a hard threshold is set if this value goes below <*5000*>. The recommended approach is to determine this value before the analysis and a custom subsampling level can be set. This step can be skipped.

Co-occurrence pattern correction

A daughter OTU is merged with its parent if:

- they share at least <*97%*> similarity
- *daughter_abundance* < *parent_abundance*: in all samples (<*"min"*>) or in average ("avg").
- the relative co-occurence (proportion of time the daughter is present when the parent is present) must be at least <*1*>

Consensus classification and representative sequences extraction

Using the remaining sequences, we choose a representative sequence for each OTU cluster as the most abundant sequence in the cluster. For each taxonomic rank, OTU's taxonomy is assigned as the majority vote in the OTU cluster. If the consensus vote is lower than 51%, no taxonomy is assigned at the given rank.

Summaries

- (samples x pipeline steps) table with the number of remaining sequences in each sample at each step
- Figures
  1. (top OTUs x samples) bi-clustered heatmap with phylum, class and order information.
  2. scatter plot of OTUs abundance vs prevalence, one facet per phylum.
  3. scatter plot of OTUs abundance vs prevalence for proteobacteria, one facet per class.
  4. barplot of relative taxonomy composition at Phylum level for each sample. In a metadata table is provided, this plots represents the composition for each level of the provided factor.

Postprocessing

For each clustering threshold, we compute alpha and beta diversity metrics (see mothur calculators for a full description of these acronyms)

- Alpha diversity: *nseqs*, *sobs*, *chao*, *shannon*, *shannoneven*
- Beta diversity: *braycurtis*, *thetayc*, *sharedsobs*, *sharedchao*

In addition, we compute the phylogenetic tree using FastTree and compute the UniFrac distances using the R's phyloseq package implementing the Fast UniFrac algorithm.

4.4 Fungal ITS pipeline

4.4.1 Pre-requisites

- Nextflow
- python(>=3.6) + packages: Biopython, pandas, matplotlib, seaborn, ITSxpress
- R(>=3.5) + libraries: ggplot2, lulu, dada2, seqinr, stringr, ShortRead, doParallel, ape, phyloseq
- install VSEARCH, HMMER and BBTools

Download the software

Clone the repository:

```
git clone https://github.com/hawaiidatascience/metaflowmics.git
cd metaflowmics/metaflowmics
```

4.4.2 Usage

To run the pipeline on your data, simply enter the following command:

```
nextflow run ITS-pipeline -profile <config> --reads "<path_to_reads/glob_pattern>"
```

The input reads need to be in the *.fastq* format (preferably gzipped) in a single folder. Reads can be single or paired-end. In the latter case, the glob pattern needs to group the R1 and R2 reads using the syntax "*R{1,2}*", and the flag *–paired_end* must be set.

For more information about the available profiles, see the profiles section.

4.4.3 ITS pipeline steps

The ITS analysis pipeline is summarized below. Values in between "<" ">" correspond to default values of tunable parameters.

Inputs

Reads need to be demultiplexed and gzipped

ITS extraction

The target region *<ITS1>* is extracted using ITSxpress. Reads missing either the left or the right flanking regions are discarded. If reads are paired, both reads are merged before ITS region extraction. Since reverse reads are often of very low quality, default mode is single-end.

Contig filtering (DADA2)

Contigs containing 'N' nucletodes are discarded, as well as contigs smaller than *<20bp>*. Then, we use *filterAndTrim()* and keep reads with at most *<3>* expected errors. We further filter the contigs and remove any chimeric sequence using VSEARCH.

Denoising (DADA2)

learnErrors(), dada(): Error models and denoising are performed on each sample independently.

OTU clustering (VSEARCH)

OTU are clustered at similarity levels *<100%, 97%>* (100% means no clustering).

Co-occurrence pattern correction (LULU)

A daughter OTU is merged with its parent if:

- they share at least *<97%>* similarity
- daughter_abundance < parent_abundance in all samples (*<min>*) or in average (*avg*).
- the relative co-occurence (proportion of time the daughter is present when the parent is present) must be at least *<1>*

Classification (DADA2)

Lineages are assigned to each individual sequence using the UNITE reference database. The assignment is performed with the naive Bayesian classifier method and sequences with a confidence at a given taxonomic rank lower than <50%> are not assigned.

Summaries

(samples x pipeline steps) table with the number of remaining sequences in each sample at each step

4.5 Computing resources

We saw that we can change the default parameters of any application by modifying the *nextflow.config* file. However, the computing resources are defined separately in the *conf* folder and they should be tuned to your work environment. You will find there multiple pre-filled configuration to help you. If you are working on a remote server or your local computer, the recommended configuration is the *conf/docker* configuration or, if you do not have sudo access, the *conf/singularity* configuration. Otherwise, configuration for a HPCC or Google Cloud Platform are provided.

4.5.1 Main parameters

- queueSize: maximum number of jobs to launch in parallel
- errorStrategy: Error handling. Available choices are *terminate*, *finish*, *ignore* and *retry*. Note that some errors (such as segfaults) can sometime be hard to recover from, and the pipeline might exit abruptly even with this parameter set to ignore.
- maxRetries: When *errorStrategy* is set to retry, it corresponds to the maximum number of times a process instance can be re-submitted in case of failure.

In addition, the pipelines define three categories of processes: low / medium / high computation. These resources can be set for either type of resource by changing their values in the corresponding fields. For example, if you wish to change the number of cpus for high computation processeses, you just need to change:

```
withLabel: high_computation {
   cpus = 3 // Change the value here
}
```

- cpus: Number of cpus
- memory: Maximum amount of memory
- time: Maximum running time

4.5.2 Configure for a HPCC

HPCC require a few more specific parameters to be set:

- *executor*: The HPCC scheduler. Default is SLURM, but many others are available. For a comprehensive list see nextflow documentation
- *queue*: List of queues where you wish to submit the processes. Multiple queues are separated with commas.
- *module*: Singularity module location.

Note: If you are working on a squashed root system, you will need to pull the repository before running the pipeline. This is likely the case if you get the error *Failed to pull singularity image* when trying to run the pipeline. You will also need to set up a docker account.

```
module load path/to/singularity/module
mkdir -p ~/.singularity_images.cache
mkdir -p /tmp/sg
export SINGULARITY_CACHEDIR=/tmp/sg
singularity pull --docker-login /tmp/sg/nakor-pipeline-env.img docker://nakor/
↪pipeline-env
singularity pull --docker-login /tmp/sg/alexcoppe-fastx.img docker://alexcoppe/fastx
singularity pull --docker-login /tmp/sg/pegi3s-fasttree.img docker://pegi3s/fasttree
mv /tmp/sg/*.img ~/.singularity_images.cache
```

4.5.3 Configure for Google Cloud Plateform (GCP)

In order to use Google Cloud Platform, you need to set up an account on the Google Cloud website. Then, you will need to:

1. Create a project
2. Create a storage bucket

The last thing you need is to provide nextflow a way to access your account via a token. The instruction for creating the token are extracted from nextflow documentation:

- Open the Google Cloud Console
- Go to APIs & Services → Credentials
- Click on the Create credentials (blue) drop-down and choose Service account key, in the following page
- Select an existing Service account or create a new one if needed
- Select JSON as Key type
- Click the Create button and download the JSON file giving a name of your choice e.g. *creds.json*

Finally define the following variable replacing the path in the example with the one of your credentials file just downloaded:

```
export NXF_MODE=google
export GOOGLE_APPLICATION_CREDENTIALS=/path/your/file/creds.json
```

You are almost set to use the GCP platform. The last remaining thing you need to do is set the configuration file in *conf/gcp.conf* and set your project information, the location and type of machines you want to use:

- *project*: name of your project on GCP
- *zone*: Location of the cloud instances. See GCP documentation for a list of available locations.
- *machineType*: specs of the instance you wish to use. A list of available instances is available on the GCP documentation.

For more details about the GCP configuration, see the nextflow documentation

4.5.4 Configure for Amazon Web Service

Not implemented yet. If you are interested to set it, have a look at the nextflow documentation

4.6 Monitoring your runs with Nextflow Tower

Nextflow Tower is a monitoring tool for your nextflow runs. Through a web interface, it tracks all processes launched by your nextflow run and returns some useful statistics on your jobs, such as the CPU and RAM usage or the running time.

To include this monitoring tool in your runs, you will need to:

- Create sign in nextflow tower website
- Generate a token:
  1. Once you are logged in, click on *Your tokens* on the top left of the screen
  2. Click on *New token*
- Provide this token to nextflow (as described in tower documentation) by either:
  - Setting the environment variable *TOWER_ACCESS_TOKEN* to your token value
  - Setting the nextflow configuration variable, *tower.accessToken* to your token value

Then, you can simply run your nextflow pipelines with the *with-tower* flag. The summaries will appear on your personal account on Nextflow Tower website

4.7 Linking with the AgaveToGo user interface

CHAPTER 5

Indices and tables

- genindex
- modindex
- search