Table of Contents

- 1 Home
  - 1.1 Introduction
  - 1.2 Pipeline summary
    - 1.2.1 Pre-processing
    - 1.2.2 Assembly
    - 1.2.3 Taxonomic classification
    - 1.2.4 Functional annotation
  - 1.3 Quick Start
  - 1.4 Databases and references
    - 1.4.1 Alignment
    - 1.4.2 Taxonomic classification
    - 1.4.3 Functional annotation
    - 1.4.4 Host reference
  - 1.5 Documentation
  - 1.6 Credits
  - 1.7 Citations
- 2 dalmolingroup/euryale: Usage
  - 2.1 Introduction
  - 2.2 Samplesheet input
    - 2.2.1 Multiple runs of the same sample
    - 2.2.2 Full samplesheet
  - 2.3 Running the pipeline
    - 2.3.1 Updating the pipeline
    - 2.3.2 Reproducibility
  - 2.4 Core Nextflow arguments
  - 2.5 Custom configuration
    - 2.5.1 Resource requests
    - 2.5.2 Updating containers (advanced users)
  - 2.6 Running in the background
  - 2.7 Nextflow memory requirements
- 3 dalmolingroup/euryale: Output

- 3.1 Introduction
- 3.2 Pipeline overview
  - 3.2.1 Kaiju and Kraken2
  - 3.2.2 Krona
  - 3.2.3 MicroView
  - 3.2.4 Diamond
  - 3.2.5 Annotate
  - 3.2.6 MEGAHIT
  - 3.2.7 MultiQC
  - 3.2.8 Pipeline information
- **4 dalmolingroup/euryale: Citations**
  - 4.1 Pipeline tools
  - 4.2 Software packaging/containerisation tools
- I Reference
  - **5 dalmolingroup/euryale pipeline parameters**
    - 5.1 Input/output options
    - 5.2 Skip Steps
    - 5.3 Decontamination
    - 5.4 Alignment
    - 5.5 Taxonomy
    - 5.6 Functional
    - 5.7 Assembly
    - 5.8 Reference genome options
    - 5.9 Download Entry
    - 5.10 Max job request options
    - 5.11 Generic options

1 Home

nf-core CI passing docs passing

nextflow DSL2 ≥22.10.1 run with docker run with singularity

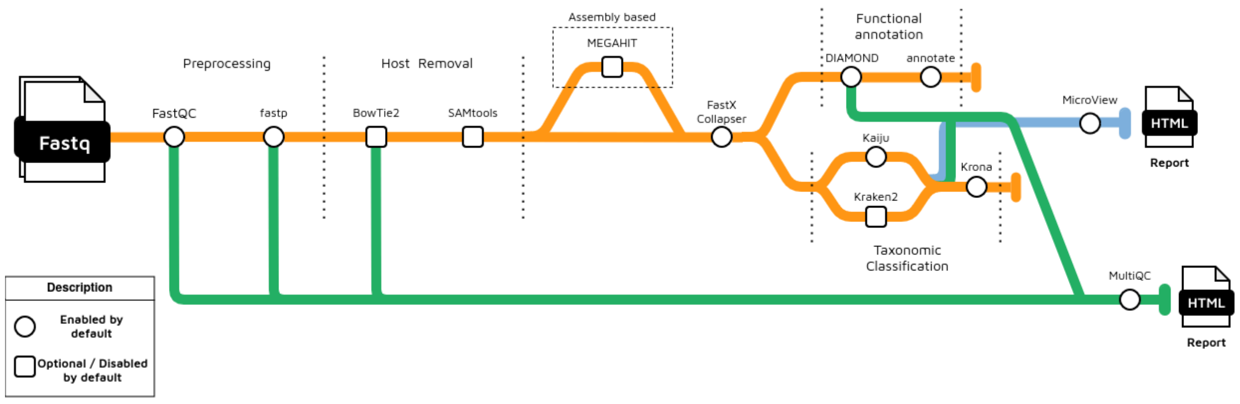
EURYALE

1.1 Introduction

dalmolingroup/euryale is a pipeline for taxonomic classification and functional annotation of metagenomic reads. Based on MEDUSA.

The pipeline is built using Nextflow, a workflow tool to run tasks across multiple compute infrastructures in a very portable manner. It uses Docker/Singularity containers making installation trivial and results highly reproducible. The Nextflow DSL2 implementation of this pipeline uses one container per process which makes it much easier to maintain and update software dependencies. Where possible, these processes have been submitted to and installed from nf-core/modules in order to make them available to all nf-core pipelines, and to everyone within the Nextflow community!

1.2 Pipeline summary



1.2.1 Pre-processing

- Read QC (`FastQC`)
- Read trimming and merging (`fastp`)
- (*optionally*) Host read removal (`BowTie2`)
- Duplicated sequence removal (`fastx collapser`)
- Present QC and other data (`MultiQC`)

1.2.2 Assembly

- (*optionally*) Read assembly (`MEGAHIT`)

1.2.3 Taxonomic classification

- Sequence classification (`Kaiju`)
- Sequence classification (`Kraken2`)
- Visualization (`Krona`)

1.2.4 Functional annotation

- Sequence alignment (`DIAMOND`)
- Map alignment matches to functional database (`annotate`)

1.3 Quick Start

1. Install `Nextflow` (`>=22.10.1`)
2. Install any of `Docker` , `Singularity` (you can follow this tutorial), `Podman` , `Shifter` or `Charliecloud` for full pipeline reproducibility *(you can use* `Conda` *both to install Nextflow itself and also to manage software within pipelines. Please only use it within pipelines as a last resort; see docs)*.
3. Download the pipeline and test it on a minimal dataset with a single command:

```
nextflow run dalmolingroup/euryale -profile test,YOURPROFILE --outdir <OUTDIR>
```

Note that some form of configuration will be needed so that Nextflow knows how to fetch the required software. This is usually done in the form of a config profile (`YOURPROFILE` in the example command above). You can chain multiple config profiles in a comma-separated string.

- The pipeline comes with config profiles called `docker`, `singularity`, `podman`, `shifter`, `charliecloud` and `conda` which instruct the pipeline to use the named tool for software management. For example, `-profile test,docker`.
- Please check nf-core/configs to see if a custom config file to run nf-core pipelines already exists for your Institute. If so, you can simply use `-profile <institute>` in your command. This will enable either `docker` or `singularity` and set the appropriate execution settings for your local compute environment.
- If you are using `singularity`, please use the `nf-core download` command to download images first, before running the pipeline. Setting the `NXF_SINGULARITY_CACHEDIR` or `singularity.cacheDir` Nextflow options enables you to store and re-use the images from a central location for future pipeline runs.
- If you are using `conda`, it is highly recommended to use the `NXF_CONDA_CACHEDIR` or `conda.cacheDir` settings to store the environments in a central location for future pipeline runs.

- Start running your own analysis!

```
nextflow run dalmolingroup/euryale \
  --input samplesheet.csv \
  --outdir <OUTDIR> \
  --kaiju_db kaiju_reference \
  --reference_fasta diamond_fasta \
  --host_fasta host_reference_fasta \
  --id_mapping id_mapping_file \
  -profile <docker/singularity/podman/shifter/charliecloud/conda/institute>
```

1.4 Databases and references

A question that pops up a lot is: Since Euryale requires a lot of reference parameters, where can I find these references?

One option is to execute EURYALE's download entry, which will download the necessary databases for you. This is the recommended way to get started with the pipeline. This uses the same sources as EURYALE's predecessor MEDUSA.

```
nextflow run dalmolingroup/euryale \
    --download_functional \
  --download_kaiju \
  --download_host \
  --outdir <output directory> \
  -entry download \
  -profile <docker/singularity/podman/shifter/charliecloud/conda/institute>
```

Check out the full documentation for a full list of EURYALE's download parameters. In case you download the Kraken2 database (`--download_kraken`), make sure to extract it using the following command before using it in the pipeline:

```
tar -xvf kraken2_db.tar.gz
```

Below we provide a short list of places where you can find these databases. But, of course, we're not limited to these references: Euryale should be able to process your own databases, should you want to build them yourself.

1.4.1 Alignment

For the alignment you can either provide `--diamond_db` for a pre-built DIAMOND database, or you can provide `--reference_fasta`. For reference fasta, by default Euryale expects something like NCBI-nr, but similarly formatted reference databases should also suffice.

1.4.2 Taxonomic classification

At its current version, Euryale doesn't build a reference taxonomic database, but pre-built ones are supported.

- If you're using Kaiju (the default), you can provide a reference database with `--kaiju_db` and provide a .tar.gz file like the ones provided in the official Kaiju website. We have extensively tested Euryale with the 2021 version of the nr database and it should work as expected.
- If you're using Kraken2 (By supplying `--run_kraken2`), we expect something like the pre-built .tar.gz databases provided by the Kraken2 developers to be provided to `--kraken2_db`.

1.4.3 Functional annotation

We expect an ID mapping reference to be used within annotate. Since we're already expecting by default the NCBI-nr to be used as the alignment reference, the ID mapping data file provided by Uniprot should work well when provided to `--id_mapping`.

1.4.4 Host reference

If you're using metagenomic reads that come from a known host's microbiome, you can also provide the host's genome FASTA to `--host_fasta` parameter in order to enable our decontamination subworkflow. Ensembl provides easy to download genomes that can be used for this purpose. Alternatively, you can provide a pre-built BowTie2 database directory to the `--bowtie2_db` parameter.

1.5 Documentation

The dalmolingroup/euryale documentation is split into the following pages:

- Usage
  - An overview of how the pipeline works, how to run it and a description of all of the different command-line flags.
- Output
  - An overview of the different results produced by the pipeline and how to interpret them.

1.6 Credits

dalmolingroup/euryale was originally written by João Cavalcante.

We thank the following people for their extensive assistance in the development of this pipeline:

- Diego Morais (for developing the original MEDUSA pipeline)

1.7 Citations

> J. V. F. Cavalcante, I. Dantas de Souza, D. A. A. Morais and R. J. S. Dalmolin, "EURYALE: A versatile Nextflow pipeline for taxonomic classification and functional annotation of metagenomics data," 2024 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), Natal, Brazil, 2024, pp. 1-7, doi: 10.1109/CIBCB58642.2024.10702116.

This pipeline uses code and infrastructure developed and maintained by the nf-core community, reused here under the MIT license.

> **The nf-core framework for community-curated bioinformatics pipelines.**
>
> Philip Ewels, Alexander Peltzer, Sven Fillinger, Harshil Patel, Johannes Alneberg, Andreas Wilm, Maxime Ulysse Garcia, Paolo Di Tommaso & Sven Nahnsen.
>
> *Nat Biotechnol.* 2020 Feb 13. doi: 10.1038/s41587-020-0439-x.

2 dalmolingroup/euryale: Usage

Documentation of pipeline parameters is generated automatically from the pipeline schema and can be found in the reference section

2.1 Introduction

2.2 Samplesheet input

You will need to create a samplesheet with information about the samples you would like to analyse before running the pipeline. Use this parameter to specify its location. It has to be a comma-separated file with 3 columns, and a header row as shown in the examples below.

```
--input '[path to samplesheet file]'
```

2.2.1 Multiple runs of the same sample

The `sample` identifiers have to be the same when you have re-sequenced the same sample more than once e.g. to increase sequencing depth. The pipeline will concatenate the raw reads before performing any downstream analysis. Below is an example for the same sample sequenced across 3 lanes:

```
sample,fastq_1,fastq_2
CONTROL_REP1,AEG588A1_S1_L002_R1_001.fastq.gz,AEG588A1_S1_L002_R2_001.fastq.gz
CONTROL_REP1,AEG588A1_S1_L003_R1_001.fastq.gz,AEG588A1_S1_L003_R2_001.fastq.gz
CONTROL_REP1,AEG588A1_S1_L004_R1_001.fastq.gz,AEG588A1_S1_L004_R2_001.fastq.gz
```

2.2.2 Full samplesheet

The pipeline will auto-detect whether a sample is single- or paired-end using the information provided in the samplesheet. The samplesheet can have as many columns as you desire, however, there is a strict requirement for the first 3 columns to match those defined in the table below.

A final samplesheet file consisting of both single- and paired-end data may look something like the one below. This is for 6 samples, where `TREATMENT_REP3` has been sequenced twice.

```
sample,fastq_1,fastq_2
CONTROL_REP1,AEG588A1_S1_L002_R1_001.fastq.gz,AEG588A1_S1_L002_R2_001.fastq.gz
CONTROL_REP2,AEG588A2_S2_L002_R1_001.fastq.gz,AEG588A2_S2_L002_R2_001.fastq.gz
CONTROL_REP3,AEG588A3_S3_L002_R1_001.fastq.gz,AEG588A3_S3_L002_R2_001.fastq.gz
```

Column	Description
`sample`	Custom sample name. This entry will be identical for multiple sequencing libraries
`fastq_1`	Full path to FastQ file for Illumina short reads 1. File has to be gzipped and have t
`fastq_2`	Full path to FastQ file for Illumina short reads 2. File has to be gzipped and have t

An example samplesheet has been provided with the pipeline.

2.3 Running the pipeline

The typical command for running the pipeline is as follows:

```
nextflow run dalmolingroup/euryale --input samplesheet.csv --outdir <OUTDIR> -profil
```

This will launch the pipeline with the `docker` configuration profile. See below for more information about profiles.

Note that the pipeline will create the following files in your working directory:

```
work                # Directory containing the nextflow working files
<OUTDIR>            # Finished results in specified location (defined with --outdir)
.nextflow_log       # Log file from Nextflow
# Other nextflow hidden files, eg. history of pipeline runs and old logs.
```

2.3.1 Updating the pipeline

When you run the above command, Nextflow automatically pulls the pipeline code from GitHub and stores it as a cached version. When running the pipeline after this, it will always use the cached version if available - even if the pipeline has been updated since. To make sure that you're running the latest version of the pipeline, make sure that you regularly update the cached version of the pipeline:

```
nextflow pull dalmolingroup/euryale
```

2.3.2 Reproducibility

It is a good idea to specify a pipeline version when running the pipeline on your data. This ensures that a specific version of the pipeline code and software are used when you run your pipeline. If you keep using the same tag, you'll be running the same version of the pipeline, even if there have been changes to the code since.

First, go to the dalmolingroup/euryale releases page and find the latest pipeline version - numeric only (eg. `1.3.1`). Then specify this when running the pipeline with `-r` (one hyphen) - eg. `-r 1.3.1`. Of course, you can switch to another version by changing the number after the `-r` flag.

This version number will be logged in reports when you run the pipeline, so that you'll know what you used when you look back in the future. For example, at the bottom of the MultiQC reports.

2.4 Core Nextflow arguments

> **NB:** These options are part of Nextflow and use a *single* hyphen (pipeline parameters use a double-hyphen).

2.4.1 `-profile`

Use this parameter to choose a configuration profile. Profiles can give configuration presets for different compute environments.

Several generic profiles are bundled with the pipeline which instruct the pipeline to use software packaged using different methods (Docker, Singularity, Podman, Shifter, Charliecloud, Conda) - see below.

> We highly recommend the use of Docker or Singularity containers for full pipeline reproducibility, however when this is not possible, Conda is also supported.

Note that multiple profiles can be loaded, for example: `-profile test,docker` - the order of arguments is important! They are loaded in sequence, so later profiles can overwrite earlier profiles.

If `-profile` is not specified, the pipeline will run locally and expect all software to be installed and available on the `PATH`. This is *not* recommended, since it can lead to different results on different machines dependent on the computer enviroment.

- `test`
- A profile with a complete configuration for automated testing
- Includes links to test data so needs no other parameters other than `--outdir`
- `docker`
- A generic configuration profile to be used with Docker
- `singularity`
- A generic configuration profile to be used with Singularity
- `podman`
- A generic configuration profile to be used with Podman
- `shifter`
- A generic configuration profile to be used with Shifter
- `charliecloud`
- A generic configuration profile to be used with Charliecloud
- `conda`
- A generic configuration profile to be used with Conda. Please only use Conda as a last resort i.e. when it's not possible to run the pipeline with Docker, Singularity, Podman, Shifter or Charliecloud.

2.4.2 `-resume`

Specify this when restarting a pipeline. Nextflow will use cached results from any pipeline steps where the inputs are the same, continuing from where it got to previously. For input to be considered the same, not only the names must be identical but the files' contents as well. For more info about this parameter, see this blog post.

You can also supply a run name to resume a specific run: `-resume [run-name]`. Use the `nextflow log` command to show previous run names.

2.4.3 `-c`

Specify the path to a specific config file (this is a core Nextflow command). See the nf-core website documentation for more information.

2.5 Custom configuration

2.5.1 Resource requests

Whilst the default requirements set within the pipeline will hopefully work for most people and with most input data, you may find that you want to customise the compute resources that the pipeline requests. Each step in the pipeline has a default set of requirements for number of CPUs, memory and time. For most of the steps in the pipeline, if the job exits with any of the error codes specified here it will automatically be resubmitted with higher requests (2 x original, then 3 x original). If it still fails after the third attempt then the pipeline execution is stopped.

For example, if the pipeline is failing after multiple re-submissions of the `DIAMOND_BLASTX` process due to an exit code of `137` this would indicate that there is an out of memory issue:

```
[62/149eb0] NOTE: Process `EURYALE:ALIGNMENT:DIAMOND_BLASTX (WT_REP1)` terminated wi
Error executing process > 'EURYALE:ALIGNMENT:DIAMOND_BLASTX (WT_REP1)'

Caused by:
    Process `EURYALE:ALIGNMENT:DIAMOND_BLASTX  (WT_REP1)` terminated with an error e

Command executed:
    diamond \
    blastx \
    --threads 2 \
    --db $DB \
    --query test_minigut_sample2.fasta \
    --outfmt 6 qseqid sseqid pident length mismatch gapopen qstart qend sstart send
    --more-sensitive --top 3 --compress 1 \
    --out test_minigut_sample2.txt \
    --log

Command exit status:
    137

Command output:
    (empty)

Command error:
    .command.sh: line 9:  30 Killed
Work dir:
    /home/pipelinetest/work/9d/172ca5881234073e8d76f2a19c88fb

Tip: you can replicate the issue by changing to the process work dir and entering th
```

2.5.1.1 For beginners

A first step to bypass this error, you could try to increase the amount of CPUs, memory, and time for the whole pipeline. Therefor you can try to increase the resource for the parameters `--max_cpus`, `--max_memory`, and `--max_time`. Based on the error above, you have to increase the amount of memory. Therefore you can go to the parameter documentation of rnaseq and scroll down to the `show hidden parameter` button to get the default value for `--max_memory`. In this case 128GB, you than can try to run your pipeline again with `--max_memory 200GB -resume` to skip all process, that were already calculated. If you can not increase the resource of the complete pipeline, you can try to adapt the resource for a single process as mentioned below.

2.5.1.2 Advanced option on process level

To bypass this error you would need to find exactly which resources are set by the `DIAMOND_BLASTX` process. The quickest way is to search for `process DIAMOND_BLASTX` in the dalmolingroup/euryale Github repo. We have standardised the structure of Nextflow DSL2 pipelines such that all module files will be present in the `modules/` directory and so, based on the search results, the file we want is `modules/nf-core/diamond/blastx/main.nf`. If you click on the link to that file you will notice that there is a `label` directive at the top of the module that is set to `label process_high`. The Nextflow `label` directive allows us to organise workflow processes in separate groups which can be referenced in a configuration file to select and configure subset of processes having similar computing requirements. The default values for the `process_high` label are set in the pipeline's `base.config` which in this case is defined as 72GB. Providing you haven't set any other standard nf-core parameters to **cap** the maximum resources used by the pipeline then we can try and bypass the `DIAMOND_BLASTX` process failure by creating a custom config file that sets at least 72GB of memory, in this case increased to 300GB. The custom config below can then be provided to the pipeline via the `-c` parameter as highlighted in previous sections.

```
process {
    withName: 'EURYALE:ALIGNMENT:DIAMOND_BLASTX' {
        memory = 300.GB
    }
}
```

> **NB:** We specify the full process name i.e. `EURYALE:ALIGNMENT:DIAMOND_BLASTX` in the config file because this takes priority over the short name (`DIAMOND_BLASTX`) and allows existing configuration using the full process name to be correctly overridden.
>
> If you get a warning suggesting that the process selector isn't recognised check that the process name has been specified correctly.

2.5.2 Updating containers (advanced users)

The Nextflow DSL2 implementation of this pipeline uses one container per process which makes it much easier to maintain and update software dependencies. If for some reason you need to use a different version of a particular tool with the pipeline then you just need to identify the `process` name and override the Nextflow `container` definition for that process using the `withName` declaration. For example, in the dalmolingroup/euryale pipeline a tool called Kraken2 is being used. You can override the default container used by the pipeline by creating a custom config file and passing it as a command-line argument via `-c custom.config`.

1. Check the default version used by the pipeline in the module file for Kraken2
2. Find the latest version of the Biocontainer available on Quay.io
3. Create the custom config accordingly:
4. For Docker:

   ```
   nextflow
   process {
       withName: KRAKEN2 {
           container = 'quay.io/biocontainers/kraken2:2.1.3--pl5321hdcf5f25_2'
       }
   }
   ```

5. For Singularity:

   ```
   nextflow
   process {
       withName: KRAKEN2 {
           container = 'https://depot.galaxyproject.org/singularity/kraken2%3A2.1.3-
   -pl5321hdcf5f25_2'
       }
   }
   ```

6. For Conda:

   ```
   nextflow
   process {
       withName: PANGOLIN {
           conda = 'bioconda::kraken2=2.1.3'
       }
   }
   ```

> **NB:** If you wish to periodically update individual tool-specific results (e.g. Kraken2) generated by the pipeline then you must ensure to keep the `work/` directory otherwise the `-resume` ability of the pipeline will be compromised and it will restart from scratch.

2.6 Running in the background

Nextflow handles job submissions and supervises the running jobs. The Nextflow process must run until the pipeline is finished.

The Nextflow `-bg` flag launches Nextflow in the background, detached from your terminal so that the workflow does not stop if you log out of your session. The logs are saved to a file.

Alternatively, you can use `screen` / `tmux` or similar tool to create a detached session which you can log back into at a later time. Some HPC setups also allow you to run nextflow within a cluster job submitted your job scheduler (from where it submits more jobs).

2.7 Nextflow memory requirements

In some cases, the Nextflow Java virtual machines can start to request a large amount of memory. We recommend adding the following line to your environment to limit this (typically in `~/.bashrc` or `~./bash_profile`):

```
NXF_OPTS='-Xms1g -Xmx4g'
```

3 dalmolingroup/euryale: Output

3.1 Introduction

This document describes the output produced by the pipeline. Most of the plots are taken from the MultiQC report, which summarises results at the end of the pipeline.

The directories listed below will be created in the results directory after the pipeline has finished. All paths are relative to the top-level results directory.

3.2 Pipeline overview

The pipeline is built using Nextflow and processes data using the following steps (steps in **italics** don't run by default):

- Kaiju and/or *Kraken2* - Taxonomically classify reads or contigs
- Krona - Visualize the taxonomic classification for each sample.
- MicroView - Visualize the taxonomic diversity for each sample.
- Diamond - Alignment reads and contigs against a reference database (such as NCBI-nr).
- Annotate - Functional annotation of alignment matches.
- *MEGAHIT* - Assembled contigs.
- MultiQC - Aggregate report describing results and QC from the whole pipeline
- Pipeline information - Report metrics generated during the workflow execution

3.2.1 Kaiju and Kraken2

▶ Output files

Kaiju is a software to perform fast taxonomic classification of metagenomic sequencing reads using a protein reference database.

Kraken2 is the second version of the Kraken taxonomic sequence classification system.

3.2.2 Krona

▶ Output files

- Krona is a tool to interactively explore metagenomes and more from a web browser.

3.2.3 MicroView

▶ Output files

- MicroView is a reporting tool for aggregating results from taxonomic classification analyses.

3.2.4 Diamond

▶ Output files
- DIAMOND is an accelerated BLAST compatible local sequence aligner.

3.2.5 Annotate

▶ Output files
- Annotate is a tool to annotate each query using the best alignment for which a mapping is known.

3.2.6 MEGAHIT

▶ Output files
- MEGAHIT is an ultra-fast and memory-efficient (meta-)genome assembler

3.3 *DIAMOND database*

▶ Output files
- This output is present if you add the `--save_db` parameter.
- DIAMOND is an accelerated BLAST compatible local sequence aligner.

3.3.1 MultiQC

▶ Output files
MultiQC is a visualization tool that generates a single HTML report summarising all samples in your project. Most of the pipeline QC results are visualised in the report and further statistics are available in the report data directory.

Results generated by MultiQC collate pipeline QC from supported tools e.g. FastQC. The pipeline has special steps which also allow the software versions to be reported in the MultiQC output for future traceability. For more information about how to use MultiQC reports, see http://multiqc.info.

3.3.2 Pipeline information

▶ Output files

Nextflow provides excellent functionality for generating various reports relevant to the running and execution of the pipeline. This will allow you to troubleshoot errors with the running of the pipeline, and also provide you with other information such as launch commands, run times and resource usage.

4 dalmolingroup/euryale: Citations

4.1 EURYALE

J. V. F. Cavalcante, I. Dantas de Souza, D. A. A. Morais and R. J. S. Dalmolin, "EURYALE: A versatile Nextflow pipeline for taxonomic classification and functional annotation of metagenomics data," 2024 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), Natal, Brazil, 2024, pp. 1-7, doi: 10.1109/CIBCB58642.2024.10702116.

4.2 nf-core

Ewels PA, Peltzer A, Fillinger S, Patel H, Alneberg J, Wilm A, Garcia MU, Di Tommaso P, Nahnsen S. The nf-core framework for community-curated bioinformatics pipelines. Nat Biotechnol. 2020 Mar;38(3):276-278. doi: 10.1038/s41587-020-0439-x. PubMed PMID: 32055031.

4.3 Nextflow

Di Tommaso P, Chatzou M, Floden EW, Barja PP, Palumbo E, Notredame C. Nextflow enables reproducible computational workflows. Nat Biotechnol. 2017 Apr 11;35(4):316-319. doi: 10.1038/nbt.3820. PubMed PMID: 28398311.

4.4 Pipeline tools

- FastQC
- MultiQC

  Ewels P, Magnusson M, Lundin S, Käller M. MultiQC: summarize analysis results for multiple tools and samples in a single report. Bioinformatics. 2016 Oct 1;32(19):3047-8. doi: 10.1093/bioinformatics/btw354. Epub 2016 Jun 16. PubMed PMID: 27312411; PubMed Central PMCID: PMC5039924.

4.5 Software packaging/containerisation tools

- Anaconda

Anaconda Software Distribution. Computer software. Vers. 2-2.4.0. Anaconda, Nov. 2016. Web.

- Bioconda

  Grüning B, Dale R, Sjödin A, Chapman BA, Rowe J, Tomkins-Tinch CH, Valieris R, Köster J; Bioconda Team. Bioconda: sustainable and comprehensive software distribution for the life sciences. Nat Methods. 2018 Jul;15(7):475-476. doi: 10.1038/s41592-018-0046-7. PubMed PMID: 29967506.

- BioContainers

  da Veiga Leprevost F, Grüning B, Aflitos SA, Röst HL, Uszkoreit J, Barsnes H, Vaudel M, Moreno P, Gatto L, Weber J, Bai M, Jimenez RC, Sachsenberg T, Pfeuffer J, Alvarez RV, Griss J, Nesvizhskii AI, Perez-Riverol Y. BioContainers: an open-source and community-driven framework for software standardization. Bioinformatics. 2017 Aug 15;33(16):2580-2582. doi: 10.1093/bioinformatics/btx192. PubMed PMID: 28379341; PubMed Central PMCID: PMC5870671.

- Docker
- Singularity

  > Kurtzer GM, Sochat V, Bauer MW. Singularity: Scientific containers for mobility of compute. PLoS One. 2017 May 11;12(5):e0177459. doi: 10.1371/journal.pone.0177459. eCollection 2017. PubMed PMID: 28494014; PubMed Central PMCID: PMC5426675.

I. Reference

5 dalmolingroup/euryale pipeline parameters

A pipeline for metagenomic taxonomic classification and functional annotation. Based on MEDUSA.

5.1 Input/output options

Define where the pipeline should find input data and save output data.

Parameter	Description
`input`	Path to comma-separated file containing information about the samples in ▶ Help
`outdir`	The output directory where the results will be saved. You have to use absol
`email`	Email address for completion summary. ▶ Help
`multiqc_title`	MultiQC report title. Printed as page header, used for filename if not other
`save_dbs`	Save DIAMOND db to results directory after construction

5.2 Skip Steps

Choose to skip pipeline steps

Parameter	Description	Type	Default	Require
`skip_classification`	Skip taxonomic classification	`boolean`		
`skip_alignment`	Skip alignment	`boolean`		
`skip_functional`	Skip functional annotation	`boolean`		
`skip_host_removal`	Skip host removal	`boolean`		
`skip_microview`	Skip MicroView report	`boolean`		
`skip_preprocess`	Skip Preprocessing steps	`boolean`		

5.3 Decontamination

Parameter	Description	Type	D
`host_fasta`	Host FASTA to use for decontamination	`string`	
`bowtie2_db`	Pre-built bowtie2 index. Directory where index is located.	`string`	

5.4 Alignment

Parameter	Description	Type	Default	Required
`reference_fasta`	Path to FASTA genome file.	`string`		
`diamond_db`	Path to pre-built DIAMOND db.	`string`		

5.5 Taxonomy

Parameter	Description	Type	Default	Required	Hidden
`kaiju_db`	Kaiju database	`string`		True	
`kraken2_db`	Kraken2 database	`string`			
`run_kaiju`	Run Kaiju classifier	`boolean`	True		
`run_kraken2`	Run Kraken2 classifier	`boolean`			

5.6 Functional

Parameter	Description	Ty
`id_mapping`	Path to ID mapping file to be used for the Functional annotation	`s`
`minimum_bitscore`	Minimum bitscore of a match to be used for annotation	`in`
`minimum_pident`	Minimum identity of a match to be used for annotation	`in`
`minimum_alen`	Minimum alignment length of a match to be used for annotation	`in`
`maximum_evalue`	Maximum evalue of a match to be used for annotation	`nu`

5.7 Assembly

Parameter	Description	Type	Default	Required	Hidden
`assembly_based`		`boolean`			

5.8 Reference genome options

Reference genome related files and options required for the workflow.

Parameter	Description	Type	Defau
`genome`	Name of iGenomes reference. ▶ Help	`string`	
`igenomes_base`	Directory / URL base for iGenomes references.	`string`	s3://ng
`igenomes_ignore`	Do not load the iGenomes reference config. ▶ Help	`boolean`	
`fasta`		`string`	

5.9 Download Entry

Parameter	Description	Type
`download_functional`	Whether to dowload functional references	`boolean`
`download_kaiju`	Whether to dowload the Kaiju reference db	`boolean`
`download_kraken`	Whether to dowload the Kraken2 reference db	`boolean`
`download_host`	Whether to download the host reference genome	`boolean`
`functional_db`	Functional reference URL (download entry)	`string`
`functional_dictionary`	Functional dictionary URL (download entry)	`string`
`kaiju_db_url`	Kaiju reference URL (download entry)	`string`
`kraken2_db_url`	Kraken2 reference URL (download entry)	`string`
`host_url`	Host FASTA reference URL (download entry)	`string`

5.10 Max job request options

Set the top limit for requested resources for any single job.

Parameter	Description	Ty
`max_cpus`	Maximum number of CPUs that can be requested for any single job. ▶ Help	`in`
`max_memory`	Maximum amount of memory that can be requested for any single job. ▶ Help	`st`
`max_time`	Maximum amount of time that can be requested for any single job. ▶ Help	`st`

5.11 Generic options

Less common options for the pipeline, typically set in a config file.

Parameter	Description
`help`	Display help text.
`version`	Display version and exit.
`publish_dir_mode`	Method used to save pipeline results to output directory. ▶ Help
`email_on_fail`	Email address for completion summary, only when pipeline fa ▶ Help
`plaintext_email`	Send plain-text email instead of HTML.
`max_multiqc_email_size`	File size limit when attaching MultiQC reports to summary er
`monochrome_logs`	Do not use coloured log outputs.
`hook_url`	Incoming hook URL for messaging service ▶ Help
`multiqc_config`	Custom config file to supply to MultiQC.
`multiqc_logo`	Custom logo file to supply to MultiQC. File name must also b
`multiqc_methods_description`	Custom MultiQC yaml file containing HTML including a meth
`tracedir`	Directory to keep pipeline Nextflow logs and reports.
`validate_params`	Boolean whether to validate parameters against the schema

Parameter	Description
`show_hidden_params`	Show all params when using `--help` ▶ Help
`schema_ignore_params`	