diag_pipelines Documentation

Release

Sacha Laurent

Mar 12, 2018

CONTENTS:

CHAPTER

ONE

COOKBOOK

Routine procedures for diagnostic purposes using microbial genomics and metagenomics.

Workflows for epidemiology, anti-microbial resistance genotyping and virulence factors identification have been implemented using the Snakemake workflow management system with bioconda integration for software dependency. Docker images of main releases are available.

1.1 Dependencies

Docker

```
docker pull metagenlab/diag_pipelines:ring_trial_v0.1.3
```

1.2 General use

Once you have pulled the docker image on your computer:

```
docker run -t --rm --mount source="$(pwd)",target=/home/pipeline_user/
data,type=bind metagenlab/diag_pipelines:ring_trial_v0.1.3 sh -c 'snakemake
--snakefile $pipeline_folder/workflows/ring_trial/pipeline.rules --use-conda
--conda-prefix $conda_folder --configfile config.yaml'
```

Update the config file for your needs.

1.3 Generating files of interest

The pipeline works by asking the generation of the files of interest for a particular analysis. Consult the full documentation to know what files can be generated. Main examples are provided below:

```
docker run -t --rm --mount source="$(pwd)",target=/home/pipeline_user/
data,type=bind metagenlab/diag_pipelines:ring_trial_v0.1.3 sh -c 'snakemake
--snakefile $pipeline_folder/workflows/ring_trial/pipeline.rules --use-conda
--conda-prefix $conda_folder --configfile config.yaml config.yaml quality/
multiqc/self_genome/multiqc_report.html'
```

This will assemble and annotate every samples, and generate a multiqc report for all samples.

```
docker run -t --rm --mount source="$(pwd)",target=/home/pipeline_user/
data,type=bind metagenlab/diag_pipelines:ring_trial_v0.1.3 sh -c 'snakemake
--snakefile $pipeline_folder/workflows/ring_trial/pipeline.rules --use-conda
--conda-prefix $conda_folder --configfile config.yaml virulence_summary.xlsx'
```

This will generate a summary excel file for the virulence factors of the samples, based on the virulence factors annotated in the file defined on the config file.

```
docker run -t --rm --mount source="$(pwd)",target=/home/pipeline_user/data,
type=bind metagenlab/diag_pipelines:ring_trial_v0.1.3 sh -c 'snakemake
--snakefile $pipeline_folder/workflows/ring_trial/pipeline.rules
--use-conda --conda-prefix $conda_folder --configfile config.yaml typing/
freebayes_joint_genotyping/core_ridom/33148/bwa/distances_snp.xlsx'
```

This will generate a snp-distance matrix of all samples, only on the core genome calculated with parsnp and with all complete genomes of the species defined in the *taxid* variable of the config file, mapped on the assembly (from the NCBI Assembly database) whose *id* is 33148 (*Staphylococcus aureus* COL substrain, reference genome for the ridom cgMLST scheme) with bwa

```
docker run -t --rm --mount source="$(pwd)",target=/home/pipeline_user/
data,type=bind metagenlab/diag_pipelines:ring_trial_v0.1.3 sh -c 'snakemake
--snakefile $pipeline_folder/workflows/ring_trial/pipeline.rules --use-conda
--conda-prefix $conda_folder --configfile config.yaml typing/mlst/summary.
xlsx'
```

This will generate an Excel summary file of the MLST of all samples, based on the software mlst)

CHAPTER

TWO

GENERALITIES

2.1 Defining variables

As a general rules, any `variable` referenced in this documentation must be either:

- Defined in the yaml config file that is passed to snakemake by `--configfile`
- Defined directly in the snakemake command by `--config variable=$value`

2.2 Logging functions

Archiving processes are defined in the file `workflows/logging.rules`. The variable `logging_folder` must be defined in the `config.yaml` or passed to snakemake with `--config`. Each time an effective snakemake run is started, a folder named with the current UTC datetime. A different number of files will be copied there, so that replication of the run is possible:

- The snakefile passed to snakemake
- The config file
- The full command used, copied into the file `cmd.txt`
- The parameter files defining the SRA and local samples, if they exist

The logs of every command run during the execution of the workflow will then be stored in this folder.

2.3 Determining sample names

Samples for the run will be determined in the file `workflows/making_sample_dataset.rules`.

2.3.1 Local samples

Local samples will be determined based on the tabulated file whose full path must be passed to the variable `local_samples` in the `config.yaml` or through `--config` on the snakemake command. It must contain at least two columns: *SampleName* and *ScientificName*.

Table 2.1: Local data example

SampleName	ScientificName
S10	Staphylococcus aureus
S1	Staphylococcus aureus

For each entry, there must be in the folder defined by the `link_directory` variable, two files (for paired reads) or only one (for single reads) whose filename starts by one and only one entry of the *SampleName* columns. For instance, the files `S10_001_R1_L001.fastq.gz` and `S10_001_R2_L001.fastq.gz` in the folder defined by the `link_directory` variable will be matched to the sample name `S10`. The matching is performed by using regular expressions to end the search at non alphanumeric characters or by the end of the word, thus the sample name `S1` will actually not match `S10_001_R1_L001.fastq.gz` nor `S10_001_R2_L001.fastq.gz`.

If needed, an *OldSampleName* column can be added to the file, when the read filenames and the desired new sample names can not be matched simply by testing the identity at the start of both names.

Table 2.2: Local data example with old sample names

SampleName	ScientificName	OldSampleName
S10	Staphylococcus aureus	Staaur-10
S1	Staphylococcus aureus	Staaur-1

In this case, the files `Staaur-10_S10_L001_R1_001.fastq.gz` and `Staaur-10_S10_L001_R2_001.fastq.gz` in the folder defined in `link_directory` will be matched to the sample name `S10`. Similarly, `Staaur-1` will actually not match `Staaur-10_S10_L001_R1_001.fastq.gz`.

2.3.2 SRA samples

SRA samples will be determined based on the tabulated file whose full path must be passed to the variable `sra_samples`. The RunInfo files that can be downloaded through the SRA NCBI database can be directly passed without any modification. Otherwise, four columns must be defined.

Table 2.3: SRA data example

Run	SampleName	LibraryLayout	ScientificName
ERR1140788	Mycobacterium_tuberculosis_N0145-Lineage_2	paired	Mycobacterium tuberculosis
SRR006916	Mycobacterium_tuberculosis_K21-Lineage_1	single	Mycobacterium tuberculosis

CHAPTER

THREE

WORKFLOWS

Current available workflows are implemented in the folder `workflows`. Each workflow will depend on `rules`, stored in the folder of the same name, and can also depend on other workflows. `rules` are sorted with respect to their general function in different folders.

3.1 Core genome determination

Core genomes can be calculated by three different means.

3.1.1 Ridom

cgMLST scheme from ridom can be extracted directly for theses species

Table 3.1: Available cgMLST schemes from ridom

Species	Taxonomy ID	Ridom ID	Reference genome assembly ID
Staphylococcus aureus	1280	141106	33148
Mycobacterium tuberculosis	1773	741110	538048
Listeria monocytogenes	1639	690488	264498
Escherichia coli	562	5064703	79781
Klebsiella pneumoniae	573	2187931	31388
Enterococcus faecium	1352	991893	526908
Acinetobacter baumannii	470	3956907	39528
Legionella pneumophila	446	1025099	30068

A bed file is constructed from the locus target file, constructing coordinates from the start and length columns of the csv file file available on the ridom website.

3.1.2 Enterobase

cgMLST scheme from enterobase is extracted for *Salmonella enterica*:

Table 3.2: Available cgMLST schemes from enterobase

Species	Taxonomy ID	Enterobase ID	Reference genome assembly ID	Scheme
Salmonella enterica	28901	SALwgMLST	359488	cgMLSTv1

A bed file for the reference genome 359488, based on the locus tag present in this genome is constructed. For instance, over the 3002 locus of the *Salmonella* cgMLSTv1, 69 come from a different genome than the reference 359488.

3.1.3 ParSNP

For species unavailable on either resource, core genome can be calculated using parsnp and the complete genomes of the species available on RefSeq

3.2 Assembly and quality

Aggregates rules for assembling genomes and performing various quality control checks. Required parameters:

- `cov_cutoff`: contigs whose coverage is below this cutoff will be excluded from the final assembly
- `adapter_file_name`: look for the adaptor for this library preparation kit (possible values)
- `adapter_removal_param1`, `adapter_removal_param2`, `adapter_removal_param3`: parameters for adapter trimming (reference)
- `minimum_quality_base`: leading and trailing bases below this quality will be removed
- `minimum_read_length`: reads shorter than this threshold after trimming will be discarded (be careful when using reads from SRA!)

Deliverables:

- `quality/multiqc/self_genome/multiqc_report.html`: quality control report based on the results of **fastqc**, **trimmomatic**, **qualimap**, **quast** and **prokka** for every sample
- `samples/{sample_name}/annotation/`: folder containing all annotation files from the `prokka` software

3.3 Resistance

Depends on the *Assembly and quality* workflow.

Required parameters:

- `resistance_prediction_softwares`: list of software for genetic resistance assessment. Possible values: `mykrobe` and `rgi`.

Deliverables:

- `samples/{sample_name}/annotation/resistance/rgi.tsv`: results files for RGI
- `samples/{sample_name}/annotation/resistance/mykrobe.tsv`: results file for mykrobe

3.4 Virulence

Depends on the *Assembly and quality* workflow.

Required parameters:

- `virulence_factors`: file with list of uniprot accession of virulence factors. An example is available in the folder `data/staph/db/`

Deliverables:

- `virulence_summary.xlsx`: summary of virulence proteins found in every samples.

3.5 Epidemiology

Depends on the *Assembly and quality* workflow (for determining the Sequence Types).

Required parameters:

- `minimum_coverage_for_calling`: minimum of coverage for considering a genomic position when counting differences between samples. Any position (SNP or non-SNP when compared to the reference) having a lower coverage will be masked
- `minimum_alternate_fraction_for_calling`: minimum ratio of observations favouring a SNP over observations not favouring a SNP. Any SNPs not meeting this criteria will also be masked

Deliverables:

- `typing/{snp_caller}/core_{ridom or enterobase}/{reference_genome}/bwa/distance_snp_mst_no_st.svg`: Minimum spanning tree of the distance in snps between every sample over the core genome as defined by ridom or enterobase. Available species and values for reference genomes are listed in the files in `data/core_genome_dbs/`. If the species under consideration has a multiple locus sequence type available, `typing/{snp_caller}/core_{ridom or enterobase}/{reference_genome}/bwa/distance_snp_mst_with_st.svg` can be generated with the ST of each sample.
- `phylogeny/{snp_caller}/core_{ridom or enterobase}/{reference_genome}/bwa/phylogeny_no_st.svg`: A phylogeny based on the alignments of the core SNPs, using RAxML. Available species and values for reference genomes are listed in the files in `data/core_genome_dbs/`. If the species under consideration has a multiple locus sequence type available, `phylogeny/{snp_caller}/core_{ridom or enterobase}/{reference_genome}/bwa/phylogeny_with_st.svg` can be generated with the ST of each sample.
- `quality/multiqc/mapping_to_{reference_genome}/multiqc_report.html`: multiqc report of **qualimap**, **fastqc** and **trimmomatic** of every samples when mapping against the reference. Check for quality control.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search