The GOSgene WGS analysis pipeline

Georg W. Otto [*]

The UCL Institute of Child Health

March 6, 2023

Contents

1 Introduction

This is a pipeline for aligning whole-genome sequencing (WGS) reads to a genome reference and variant calling (Figures 1 and 2). The alignment parts follows alignment pipeline standards described in [1] and in the github pages of CCDG, Baylor HGSC and the Center for Statistical Genetics (University of Michigan).

The variant calling and variant refinement part of the pipeline follows GATK Best Practices, in particular the single sample and cohort workflows.

2 Pipeline workflow

The workflow implemented in this pipline is outlined in figs. 1 and 2. QC metrics are generated using `FastQC` [2]. Alignment of sequencing reads to the reference

[*]g.otto@ucl.ac.uk

genome is carried out using `bwa mem` [3]. Bam files are processed, sorted and indexed using `SAMBLASTER` [4], `Samtools` [5] and `Sambamba` [6]. Duplicate read pairs are marked by `Picard tools` [7]. Base quality calibration is performed using `BaseRecalibrator`, part of the `GATK` software package [8]. Variant calling, genotyping, calibration and indexing is carried out using `GATK` software.

3 Pipeline structure

This pipeline is a python package consisting of scripts to run the pipeline and a collection of python modules which define classes and methods (see class diagram xx). It consists of two parts: (1) An analysis pipeline that works on a per-sample level, aligning and processing sequencing reads and performing variant calling, resulting in a single-sample `GVCF` file (fig. 1). (2) A cohort-level analysis pipeline that combines `GVCF` files from a family or cohort and performs variant recalibration and genotyping (fig. 2).

The pipeline is designed to run on a variety of computing environments, e.g. on an `SGE` queueing system or an interactive shell. In order to achieve this, the pipeline workflows consist of two levels: Wrapper scripts (`initialize_sample_analysis.py`, `initialize_cohort_analysis.py`), which take the sample information from a text file and configurations from a yaml file. They create the commands, environments and data structures for the pipeline scripts (`sample_analysis.py`, `cohort_analysis.py`) and send the jobs to the cluster queue or run them directly as a new process on the server. This is outlined in figs. 3 and 4.

1. WGS sample analysis
    (a) initialization script `initialize_sample_analysis.py`
    (b) pipeline script `sample_analysis.py`
2. Cohort analysis
    (a) initialization script `initialize_cohort_analysis.py`
    (b) pipeline script `cohort_analysis.py`

Use of scratch directory When running the pipeline on the Computer Science cluster, it is recommended to move input data to the temporary scratch directory of each node, and also to write output data there. The pipeline takes care of the copying of data to and from the scratch directory. In order to do this, a path to the scratch directory has to be given in the yaml config file e.g.

```
scratchdir: /scratch0/
```

If the scratchdir variable is empty

```
scratchdir: ''
```

or has the boolean value `False`

```
scratchdir: False
```

no temporary directory will be used and data are read from and written to their original location.

Use of environment modules The pipeline supports the use of environment modules to control to load the programs used in the path. This is configured in the configuration file. There is an option `modules`, which can be set to `False`. In that case environment modules are not used, and it is assumed that every program needed is in the path.

4 The WGS sample analysis pipeline

4.1 WGS sample analysis initialization

The sample analysis pipeline (fig. 1) is initialized and started by the command

```
initialize_sample_analysis.py --config-file <file> --run-mode <\
    mode>
```

The command line options are:

`--config-file, -c` [mandatory] the path to the yaml configuration file containing parameters how to run the pipeline. A sample configuration file can be found in `config/config-sample-analysis.yml`. Please copy this file and modify it according to your needs.

`--run-mode, -r` [optional, default: test] The pipeline can be run in different run modes, these are

- `test` dry run without initializing the `sample_analysis.py` program. Prints out the command that would be send to the cluster queue.
- `server` starts `sample_analysis.py` for each sample as a subprocess on the machine the wrapper script is running.
- `cluster` sends a command for each sample to the cluster queue in order to run `sample_analysis.py`.

`--help, -h` [optional] prints a short message containing the vailable command line options.

Auxiliary files needed

`samples-file` A tab-delimited text file, (containing one header line). It containes `fastq` file annotations, with each line representing a `fastq` file or (in case of paired-end sequencing) a `fastq` file pair. The table can contain any number of columns, but the components processed by the pipeline are the first two or three columns: The first column is the sample name. A sample name can occur in multiple lines, because a sample can have multiple `fastq` files or file pairs. All `fastq` files with the same sample name, i.e. read pairs and technical replicates, will be combined in the same bam file. The second column is the file name of the `fastq` file in case of single end sequencing or of the forward `fastq` file in case of paired-end sequencing. The third column is the file name of the `fastq` reverse file (unique) in case of paired-end sequencing.

example `data/samples.txt`:

```
sample     fastq_1                  fastq_2
sample-1   sample-1_1_R1.fastq.gz   sample-1_1_R2.fastq.gz
sample-1   sample-1_2_R1.fastq.gz   sample-1_2_R2.fastq.gz
sample-1   sample-1_3_R1.fastq.gz   sample-1_3_R2.fastq.gz
sample-2   sample-2_1_R1.fastq.gz   sample-2_1_R2.fastq.gz
sample-2   sample-2_2_R1.fastq.gz   sample-2_2_R2.fastq.gz
sample-2   sample-2_3_R1.fastq.gz   sample-2_3_R2.fastq.gz
```

The path to the samples file is set in the configuration file in the variable `samples-file`.

4.2 The sample analysis pipeline

Usually, the sample analysis pipeline is started by the `initialize_sample_analysis.py` script (section 4.1). However the pipeline can be started directly (for example for tests) for a single sample using the command:

```
sample_analysis.py --config-file <file> --fastq-object-file <\
    file> --run-mode <mode>
```

The command line options are:

`--config-file, -c` [mandatory] the path to the yaml configuration file containing parameters how to run the pipeline. A sample configuration file can be found in `config/config-sample-analysis.cfg`. Please copy this file and modify it according to your needs.

`--fastq-object-file, -f` the path to a file containing the serialized `Fastq` object of the sample, in `pickle` format.

`--run-mode, -r` [optional, default: test] The pipeline can be run in different run modes, these are

- `test` for debugging, does not delete the input `.pkl` file and does not delete the scratch directory upon pipeline completeion.
- `server` configurations for running the pipeline directly on the server where it is invoked (e.g. not in a cluster queue).
- `cluster` configurations for queuing and running the pipeline on a sge cluster.

`--help, -h` [optional] prints a short message containing the vailable command line options.

`--version, -v` [optional] prints the version of the pipeline.

5 The WGS cohort analysis pipeline

5.1 WGS cohort analysis initialization

The cohort analysis pipeline (Figure 2) is initialized and started by the command

```
initialize_cohort_analysis.py --config-file <file> --run-mode <\
    mode>
```

The command line options are:

`--config-file, -c` [mandatory] the path to the yaml configuration file containing parameters how to run the pipeline. A sample configuration file can be found in `config/config-cohort-analysis.yml`. Please copy this file and modify it according to your needs.

`--run-mode, -r` [optional, default: test] The pipeline can be run in different run modes, these are

- `test` dry run without initializing the `cohort_analysis.py` program. Prints out the command that would be send to the cluster queue.
- `server` starts `cohort_analysis.py` for each sample as a subprocess on the machine the wrapper script is running.
- `cluster` sends a command for each sample to the cluster queue in order to run `cohort_analysis.py`.

`--help, -h` [optional] prints a short message containing the vailable command line options.

Auxiliary files needed

`cohorts-file` A tab-delimited text file, with sample annotations (containing one header line). The table can contain any number of columns, but the components processed by the pipeline are the first two columns: The first column is the cohort name, the second column is the name of the corresponding sample `GVCF` files. A cohort can consist of one or (usually) more samples.

example `data/cohorts.txt`:

```
cohort     vcf
cohort-1   sample-1.vcf.gz
cohort-1   sample-2.vcf.gz
cohort-2   sample-3.vcf.gz
cohort-2   sample-4.vcf.gz
```

The path to the samples file is set in the configuration file in the variable `cohorts-file`.

5.2 The cohorts analysis pipeline

Usually, the cohorts analysis pipeline is started by the `initialize_cohorts_analysis.py` script (section 5.1). However the pipeline can also be started directly (for example for tests) for a single sample using the command:

```
cohort_analysis.py --config-file <file> --fastq-object-file <\
    file> --run-mode <mode>
```

The command line options are:

`--config-file, -c` [mandatory] the path to the yaml configuration file containing parameters how to run the pipeline. A sample configuration file can be found in `config/config-cohort-analysis.yml`. Please copy this file and modify it according to your needs.

`--vcf-object-file, -f` the path to a file containing the serialized `Vcf` object of the cohort, in `pickle` format.

`--run-mode, -r` [optional, default: test] The pipeline can be run in different run modes, these are

  `test` for debugging, does not delete the input `.pkl` file and does not delete the scratch directory upon pipeline completion.

  `server` configurations for running the pipeline directly on the server where it is invoked (e.g. not in a cluster queue).

  `cluster` configurations for queuing and running the pipeline on a sge cluster.

`--help, -h` [optional] prints a short message containing the vailable command line options.

`--version, -v` [optional] prints the version of the pipeline.

5.3 Tests

The pipeline comes with a set of test data , i.e. fastq files with a small number of reads and reference files, which are located in the `data` directory. The `test` directory contains a number of test scripts to test pipeline functionality with the test data. Most of these test scripts are "Self-contained", i.e. do not need a configuration file.

`test-sample-analysis.py` test that runs the sample analysis with test data, sample-1.

```
test-sample-analysis.py
```

To test a pipeline run on the cluster, the pipeline can be run with the test data and the configuration file configured appropriatedly.

6 Getting help

Help on how to run the higher level scripts `initialize_sample_analysis.py` and `sample_analysis.py` can be found using the options `-h` or `--help`:

```
python initialize_sample_analysis.py --help
```

Classes and methods are documented using docstring, so for example to get documentation of the fastq module, the Fastq class and the fastqList method of the Fastq class, use:

```
import fastq
help(fastq)
help(fastq.Fastq)
help(fastq.Fastq.fastqList)
```

References

1. Regier, A. A. *et al.* Functional equivalence of genome sequencing analysis pipelines enables harmonized variant calling across human genetics projects. *bioRxiv* (Feb. 2018).

2. *FastQC* `http://www.bioinformatics.babraham.ac.uk/projects/fastqc/` (2015).
3. Li, H. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *arXiv.* arXiv: `1303.3997 [q-bio.GN]` (Mar. 16, 2013).
4. Faust, G. G. & Hall, I. M. SAMBLASTER: fast duplicate marking and structural variant read extraction. *Bioinformatics* **30,** 2503–2505 (May 2014).
5. Li, H. *et al.* The Sequence Alignment/Map format and SAMtools. eng. *Bioinformatics* **25,** 2078–2079 (Aug. 2009).
6. Tarasov, A. *et al.* Sambamba: fast processing of NGS alignment formats. *Bioinformatics* **31,** 2032–2034 (Feb. 2015).
7. *Picard* `https://broadinstitute.github.io/picard/`.
8. *GATK: Genome Analysis Toolkit* `https://gatk.broadinstitute.org/hc/en-us`.

List of Figures

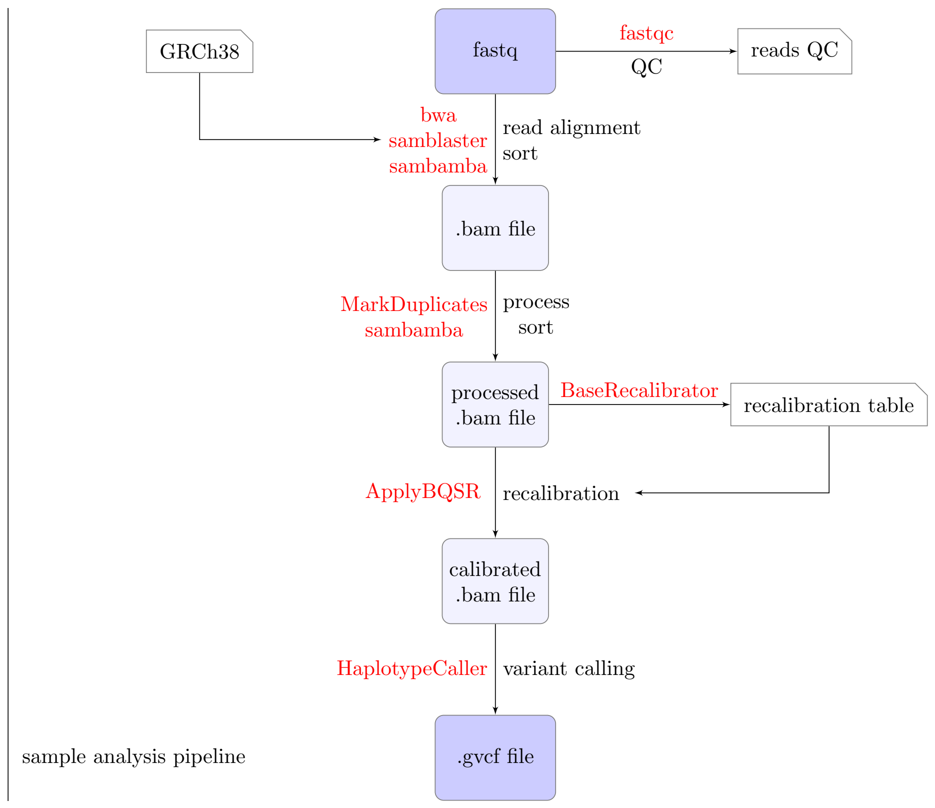


Figure 1: Flowchart of sample analysis

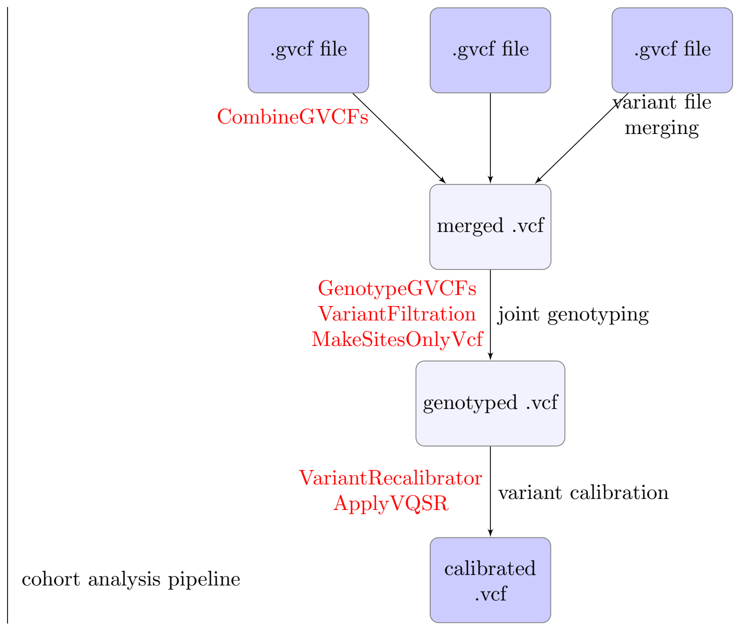


Figure 2: Flowchart of cohort analysis

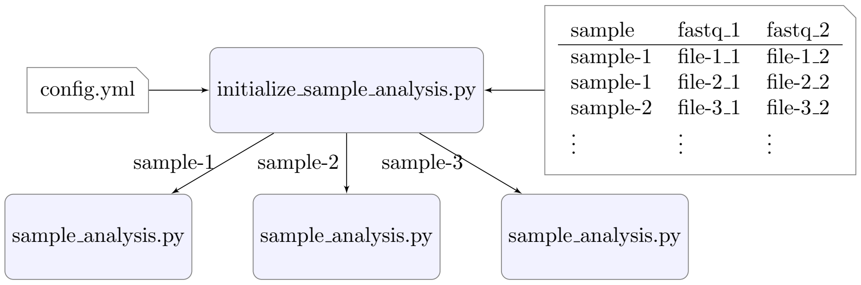
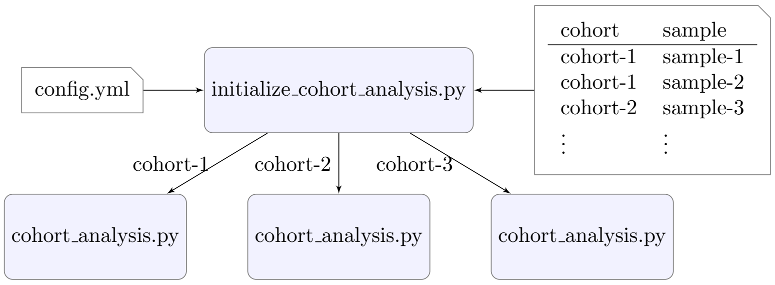


Figure 3: Pipeline process deployment, samples



cohort	sample
cohort-1	sample-1
cohort-1	sample-2
cohort-2	sample-3
⋮	⋮

Figure 4: Pipeline process deployment, cohort