A Snakemake workflow for WGS data analyses of Chlamydia trachomatis genome

Whole genome sequencing data analyses of Chlamydia trachomatis genome from clinical samples was performed using Snakemake - a Python-based pipeline management tool for creation of reproducible workflows.

General information

The pipeline requires the reference genome of human host and Chlamydia trachomatis. The reference genome for Chlamydia used in this pipeline is the pan genome, which was generated by seq-seq-pan (Jandrasits et al., 2018) by combining Chlamydial reference genomes of serovars A, B, D, E, F, G, J and K. The index files for *C. trachomatis* reference genome can be created by the following command and should be saved in the same folder as the reference genome.

bowtie2-build ../mac/samples/reference/ct_genome_consensus.fasta
../mac/samples/reference/ct_pan

samtools faidx ../mac/samples/reference/ ct_genome_consensus.fasta

picard CreateSequenceDictionary R=../mac/samples/reference/ ct_genome_consensus.fasta
O=../mac/samples/reference/ ct_genome_consensus.dict

BOWTIE2 is used to map the Chlamydia sequences from the isolates and then the mapped sequences are post-processed, preparing the input files for variant calling with GATK. The output files from GATK for all the samples are merged and then SNPs and INDELs are called in separate files. Both, SNPs and INDELs are filtered and a table with all the required information from the SNPs and INDELs is generated separately. Quality checks are performed at various stages of the analysis and towards the end a final report is generated that lists the statistics for each performed analysis step.

Platform specific installation notes

The implementation was tested on a Linux and a MACOS system. The RAM-intensive jobs like BBNORM and GATK, will most likely be executed on a cluster, or has long runtime on a 2.9 GHz Intel Core i5 processor (personal laptop). Mostly, the environment built using Miniconda is analogous between the two platforms, however some packages require platform-specific implementation of dependencies (e.g. libgcc and libcxx). Therefore, I have included different environment YAML-files for the two implementations. Since the normalization step using BBNORM takes too long to perform on 2.9 GHz Intel Core i5, so I have included the normalization step output files for the test run with the samples (CTD_1 and CTD_2). The fastq files and the output files from the normalization step were uploaded on onedrive due to file size. The link for the same is provided below.

Configuring the workflow

The configuration file config.yaml offers some flexibility to the user – some (or all) of the parameters can be passed onto the executable using the specified fields. The user can keep or remove the duplicates from the sequencing data by changing the parameter in the config file.

The read groups can be named to suit the user requirement. MULTIQC parameter field refers to a separate configuration file stored in the rules folder, which provides an order of the reports that are generated during the execution of the workflow. Furthermore, an on/off switch is provided for the execution of BBNORM. If the coverage is homogeneous then normalisatoion step is not advisable. But since in these samples, it was not homogeneous (FASTQC report) so the default is set to 1, but can be changed. To introduce this option in the workflow, an if-statement has been added in the mapping step with BOWTIE2 and when generating the MULTIQC report.

Step by step instructions:

1). If you don't have conda on your computer you can install the Miniconda Python3 distribution from this link (https://conda.io/en/latest/miniconda.html). Make sure you answer yes to the question whether conda shall be put into your PATH.

2). Obtain the copy this workflow
git clone https://github.com/casrssk/WGS-analysis-in-C_trachomatis.git

Make it your current working directory
cd WGS-analysis-in-C_trachomatis.git

3). To avoid duplication of folders common to Linux and MACOS platform, these have been placed in the parent directory. The user can move them into the samples folder (MACOS or Linux) depending on the implementation platform. Make samples folder as your current directory for running the rest of the pipeline. The annotation file is too large to be put on github and so I am proving a link from onedrive from where the folder can be downloaded
https://1drv.ms/u/s!AoZAmrQJEC56gQKjbO-kElU8uZcZ?e=JvYqHv

The sample data and the normalized data can also be downloaded from this link
https://1drv.ms/u/s!AoZAmrQJEC56gzvzGdza5UFPnA-t?e=IUOGol

4). You have to set up conda channels and then create a virtual environment and install all the dependencies into it using the following command.

cd <path to the folder containing environment.yml>
conda install –channel "AgBiome" package
conda config –append channels AgBiome
conda env create -c conda-forge -c bioconda –c defaults –c R -f environment.yml

5). Execute the workflow by first testing it using dry-run option
snakemake –n

If it runs without errors
snakemake –cores <specify the number of cores>

6). After successful completion, you can investigate the results by
snakemake –report report.html