Learning experience

Our group first discussed about the functions we need to run in the snakemake script. We want to reproduce the whole process by which starts with downloading the sample sequence and reference sequence. Then TrimmomaticPE function is used to remove low quality sequences. Next, use bowtie2 to align the reads to the reference genome, generating sam files. After that, convert the sam file to bam file in a bid to decrease the file size for future analysis. The whole flow chart of our workflow has been uploaded to the group page as dag.svg file.

1. Downloading

The first step is the downloading step, where we found to use the sra-tools function to download the sample trials by setting the accession numbers as the parameters. It will download both forward and reverse fastq files based on the provided accession numbers. However, we encountered unwanted problems when using this docker image of sra-tools. One sample (SRR2584863) cannot be completely downloaded through this function, which will generate 'file imcomplete' error when being trimmed. It may due to this sra-tool downloads the .fastq file instead of .fastq.gz file. In this case, we simply reduce the accession numbers used as two accession numbers can also prove the reproductivity of our script.

2. Trim

In the trim step, docker images from staphb and biocontainers were first choices while no manual instruction can be found in their homepage. As a result, we turned to another docker image named chrishah/trimmomatic-docker:0.38, where relatively instruction can be found including how to refer the adapter file in the shell code. We also encountered some strange permission related issues which can be solved by signing in as root.

3. Bowtie2

In order to run bowtie2 in one snakemake script, we have to make sure the index will be built before the bowtie2 rule. We managed to do that by specifying one of the inputs in the bowtie2 rule as Ec606.1.bt2 which is also defined as one output of index building rule and will be generated automatically after index building. Through building this input & output connection, the bowtie2 index will be built before running bowtie2. In this case, the input and output are both not presented in the shell directive, while it can still restrict the running order of the two rules.

4. sam to bam & sort

The sam files generated after bowtie2 are not sorted, which have to be compressed into bam files before sorting. It seems that these two steps have been

giving us problems that are difficult to solve. The bam files generated by the docker images we tried seems to be broken and returns white errors during sorting, which may point to the previous converting step. We finally determined that the docker image for samtools may contain problems after successfully running the problematic steps with locally installed samtools.

5. Upload vcf files to s3 bucket

The variants calling steps were normal when using docker images. Therefore, we can obtain the results we want in our local storage of the EC2 instance. Based on the function 'aws s3 cp' function, we can upload our results to s3 bucket. However, in order to use an output as the final input of the rule all, we need to generate something can be detected by snakemake when uploading our results. We managed to let our script create an empty txt file when performing the uploading process by using 'touch' function. So that, one txt file will be created every time after successfully uploading the vcf file, which can be used as a trick to run the uploading rule.