Linux-based SNP calling analysis pipeline for calling variants from paired end FASTQ files

Reid Wiggins

Video Tutorial Can Be Found Here:
https://drive.google.com/drive/folders/1w9ERjJPhsNr1u96LrCO_11C7OpGp6Z2s?usp=sharing

And here:

https://studentuml-my.sharepoint.com/:f:/g/personal/frederic_chain_uml_edu/El1dQCPq_J5DjV2kCux2tj0Bzh0g-JPQczPKi0vaQYCoTg?e=GBfv1e

GOAL

Create a pipeline to call SNPs from pair-end sequencing reads versus a reference genome.

DATA SELECTION

Two datasets are necessary for the SNP calling pipeline:

(1) a **reference genome**
(2) **Next Generation Sequencing (NGS) paired-end sequences**

The reference genome is used to align the NGS reads for identifying variants. The reference genome in this tutorial is the GRCh38.p13 human reference genome available from the RefSeq NCBI database.

The sample NGS pair-end sequences used in this tutorial were obtained from the NCBI SRA database under accession number SRR9916705. This RNA-seq data was originally collected using the Illumina NextSeq 500 platform as part of an NCBI screening experiment. Multiple cell samples were taken from 57 random individuals and chromosome 11 was selectively sequenced at a relatively high coverage of 10X. Everyone was sampled 3 times and processed individually. This coverage level has adequate power for calling SNPs.

Of note, the sample NGS reads were sequenced using RNA-seq and indirectly represent the transcriptome rather than the primary DNA sequence. Additionally, the relative expression of different regions of the genome creates artifacts within SNP detection; SNPs within highly expressed genes will be called with relative certainty given the high read depth compared to SNPs within relatively unexpressed genes (e.g. with low read depth). The variable coverage of the RNA-seq reads is a result of variance in expression levels and alternative splicing. The optimal dataset for variant calling would be a high coverage DNA-seq experiment, however the variant calling pipeline will still work with RNA-seq data and for most combinations of reference/NGS sample data. This dataset was chosen to specifically target chromosome 11.

LINUX PIPELINE

SETUP

Tools Needed – Miniconda, SRAtoolkit

SRAtoolkit can be found here: https://github.com/ncbi/sra-tools

Miniconda can be found here: https://github.com/conda/conda-docs/blob/master/docs/source/miniconda.rst

The purpose of this Linux pipeline is to detect SNPs within a targeted genome sequence. This process is referred to as "SNP calling". To accomplish this, a Linux command-line was used along with numerous individually developed tools. The command line offers a highly customizable and efficient infrastructure for genomic data analysis and file manipulation. I performed all analyses on a Linux virtual box on my personal computer. The virtual machine software used (VirtualBox) can be found here: https://www.virtualbox.org/wiki/Downloads.

It is useful to download and install the Miniconda-3 software into the Linux command-line before beginning data-analysis. Miniconda ("conda") is a third-party software designed to allow for the easy download and installation of tools directly from and into the command line.

The first step of the process is to obtain the reference genome from the NCBI database. The reference genome can be downloaded from the command line:

wget ftp://ftp.ncbi.nlm.nih.gov/refseq/H_sapiens/annotation/GRCh38_latest/refseq_identifiers/GRCh38_latest_genomic.fna.gz

```
(SNPcall) reid@reid-VirtualBox:~/project$ wget ftp://ftp.ncbi.nlm.nih.gov/refseq/H_sapiens/annotation/GRC
h38_latest/refseq_identifiers/GRCh38_latest_genomic.fna.gz
```

This command will save the reference genome as a *.fna.gz file to the current working directory. This is a compressed FASTA format.

Next, obtain the NGS sequence data, which can be done from the Linux command line using the SRAtoolkit package. The SRAtoolkit can be installed through conda:

```
(SNPcall) reid@reid-VirtualBox:~/project$ conda install sra-tools
```

To download the experimental sequencing data from the NCBI SRA database, the *-prefetch* command will download the file based on accession number SRR9916705:

```
(SNPcall) reid@reid-VirtualBox:~/project$ prefetch SRR9916705
```

The download will be saved to the current working directory as SRR9916705.sra, the SRA file format containing both pair-end reads.

The SRR9916705.sra file must be converted into *.fastq.gz format to be compatible for quality check and alignment. The SRAtoolkit has a command that separates the *.sra file into two fastq.gz (compressed fastq) files based on read pair:

```
(SNPcall) reid@reid-VirtualBox:~/project$ fastq-dump --gzip --defline-qual '+' --split-
e SRR9916705.sra
```

The fastq-dump command reads the SRR9916705.sra file and saves SRR9916705_1.fastq.gz and SRR9916705_2.fastq.gz to the current working directory. These files are the compressed fastq format of both pair-end reads.

At this point all the data needed for the pipeline are accessible to the Linux command line. The data are also in the current format to proceed with the pipeline.

QUALITY CONTROL (Figure 1)

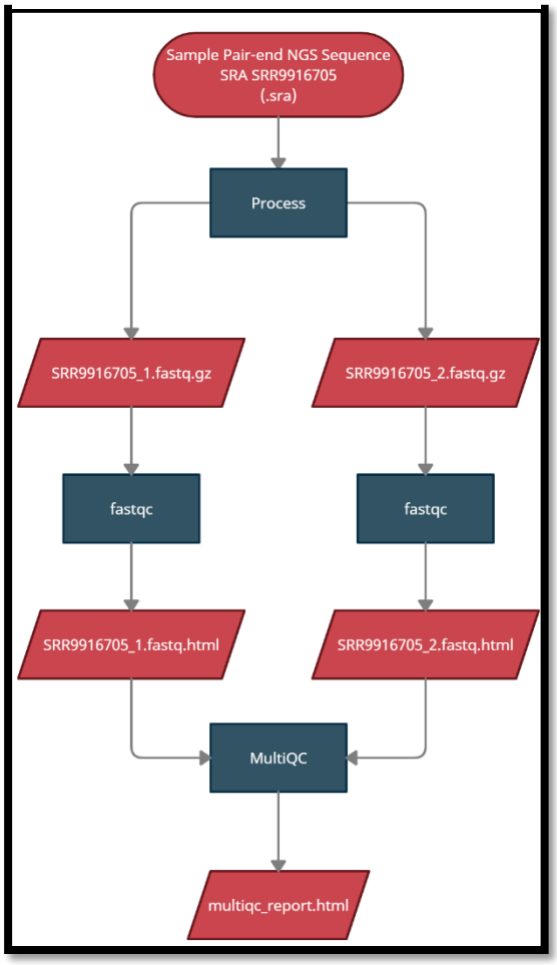
Tools Needed – FastQC, MultiQC

FastQC can be found here: https://github.com/s-andrews/FastQC.

MultiQC can be found here: https://github.com/ewels/MultiQC.

This flow chart represents the process for obtaining sample FASTQ files from the SRA database and the quality control portion of the pipeline that includes FastQC and MuliQC. A read trimming/filtering step was not performed in this pipeline because the QC results suggested it was unecessary.

Figure 1 - Quality Control



Two tools will be used to run quality checks on the SRR9916705.fastq.gz files directly from the command line: FastQC and MultiQC. Both tools can easily be installed via the command line using the conda installer. The fastqc report can be generated with the following:

```
(SNPcall) reid@reid-VirtualBox:~/project$ fastqc *.fastq.gz
```

This command runs a FastQC analysis on all files in the current directory that end with .fastq.gz. This will run the analysis on both pair-end reads. The generated report can be viewed in a web browser and my results are found here in html/csl format:

SRR99167105_1 ------- https://drive.google.com/file/d/1rk4M5EWEAhxwVJM7-EAZGScPzKN2Q7zf/view?usp=sharing.

SRR99167105_2 -------https://drive.google.com/file/d/1IPY0g8j8BAyGn35VNtHDsXD--vkuxUlc/view?usp=sharing

My FastQC reports indicate that no further filtering was required.

An additional quality control report which further considers both generated FastQC reports can be generated using the MultiQC tool that produces a quality report based on the two FastQC reports from both pair-end reads. To run the MultiQC report:

```
(SNPcall) reid@reid-VirtualBox:~/project$ multiqc .
```

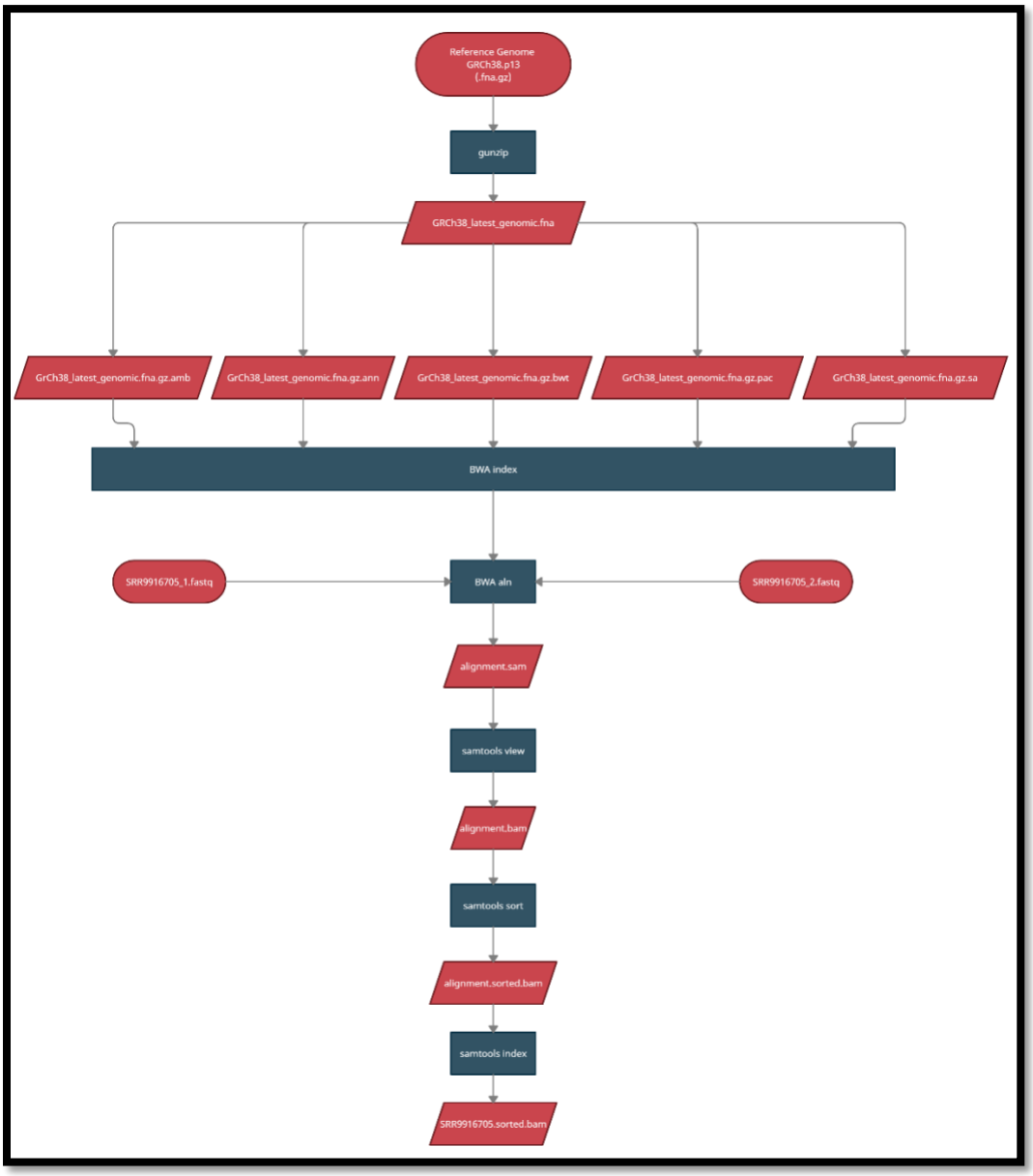
The '.' argument tells the program to automatically look in the current working directory for any files compatible with the tool. The command should return a script saying that "two files have been found" and produce a multiqc report in the current working directory. The multiqc report will be generated as multiqc_report.html in the home directory. My report can be found here in html/csl format: https://drive.google.com/file/d/1OBmkXwZQzt9XdAjCkw89bw-1VYfd4C7I/view?usp=sharing

ALIGNMENT (Figure 2)

Tools Needed – bwa

Bwatools can be found here: https://github.com/lh3/bwa

Figure 2 - Sequence Alignment



This flowchart illustrates the alignment portion of the pipeline. The reference genome is first indexed and then used to align the previously obtained FASTQ sample files to the reference genome. Additionally, the resulting SAM file is compressed and filtered into a sorted.bam file.

After obtaining the data and ensuring quality of the sample pair-end reads; the reads now need to be mapped against the reference genome. In most cases, adapter and read trimming and filtering are performed, but in our case we will directly proceed to read alignment. The alignment process is essentially two steps: creating an index from the reference genome and aligning the reads against the reference genome. Both steps can be completed using the bwa library.

Indexing the reference genome is a step that allows the alignment tool to align the sequence more efficiently. This is accomplished using the Burrows-Wheeler Transform (BWT), A commonly used computer algorithm that recompiles the reference genome to indices. The bwa binary for performing the BWT on the reference genome is the command bwa index. This library is loaded into linux environment via conda install. All input files can be unzipped using the -*gunzip* command before indexing, this step is not required for proper function. The following command will create an index from the reference genome in the form of 5 index files in the current working directory:

```
(SNPcall) reid@reid-VirtualBox:~/project$ bwa index GRCh13_latest_genomic.fna index
```

The reference genome is the first argument for the command in the uncompressed FASTA format. All index files created in the home directory will be named with the prefix -index.

Once the index has been created it is time to align the two paired-end NGS sequences to the reference genome. This is accomplished using the bwa command -*mem* which aligns the two paired-end reads to the reference genome:

```
(SNPcall) reid@reid-VirtualBox:~/SNPdemo$ bwa mem GRCh38_latest_genomic.fna SRR9916705_1.fastq.gz SRR9916705_2.
fastq.gz > aln_pe.sam
```

This command will align the two paired-end sequences to the indexed reference genome and store the resulting alignment.sam file into the current working directory.

POST-ALIGNMENT PROCESSING

Tools Needed – Samtools

samtools can be found here: https://github.com/samtools

The alignment file must be compressed, sorted, and indexed before moving onto the variant calling portion of the pipeline.

The first step is to compress the alignment.sam file into a .bam file format:

```
(base) reid@reid-VirtualBox:~/project$ samtools view -b alignment.sam > alignment.bam
```

The -b argument sets the output file to BAM format and saves the file to the current working directory.

The next step in preparing the alignment .bam for SNP calling is to sort the file. This is done to allow for the program to call the SNPs in order of locus on the reference sequence:

```
(base) reid@reid-VirtualBox:~/project$ samtools sort -o SRR9916705.sorted.bam alignment.bam
```

This command sorts the alignment file and outputs the sorted file into the working directory.

The final step before calling SNPs is to index the sorted BAM file so that the data are easier to process/access. This can again be accomplished with samtools:

```
(base) reid@reid-VirtualBox:~/project$ samtools index SRR9916705.sorted.bam
```

This will index the sorted BAM file and create a SRR9916705.sorted.bai file. This file will be used in the SNP calling portion of the pipeline.

To analyze the SRR9916705.sorted.bam file you can use the samtools flagstat command to get statistics about the alignment:

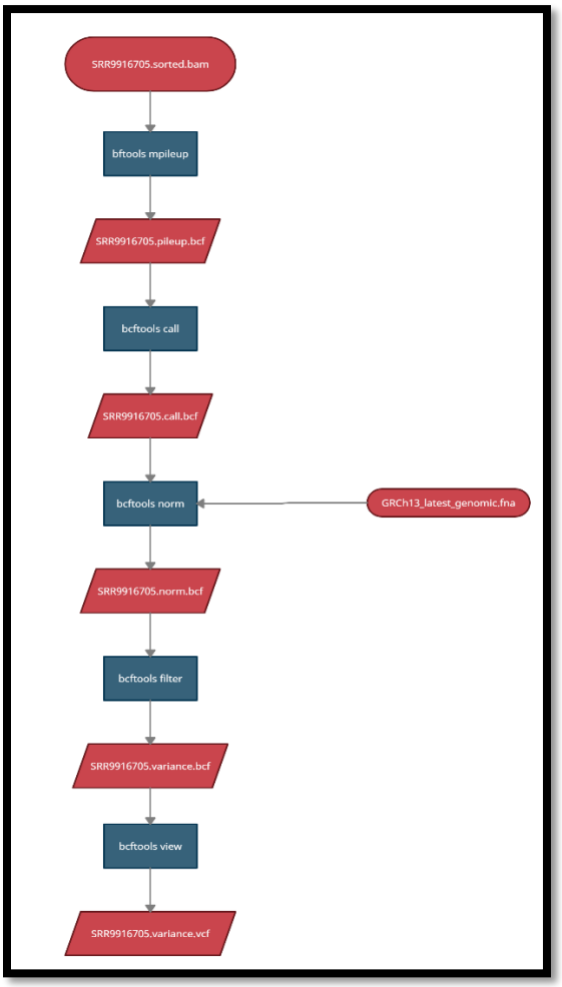
```
(base) reid@reid-VirtualBox:~/project$ samtools flagstat SRR9916705.sorted.bam
46856949 + 0 in total (QC-passed reads + QC-failed reads)
0 + 0 secondary
2053 + 0 supplementary
0 + 0 duplicates
21739184 + 0 mapped (46.39% : N/A)
46854896 + 0 paired in sequencing
23427448 + 0 read1
23427448 + 0 read2
47298 + 0 properly paired (0.10% : N/A)
48314 + 0 with itself and mate mapped
21688817 + 0 singletons (46.29% : N/A)
884 + 0 with mate mapped to a different chr
252 + 0 with mate mapped to a different chr (mapQ>=5)
```

SNP VARIANT CALLING AND FILTERING (Figure 3)

Tools Needed - BCFtools

Bcftools can be found here: https://github.com/samtools/bcftools

Figure 3 - Variant Calling



This flowchart illustrates the variant calling portion of the pipeline. The sorted.bam file is compiled into a .bcf file and then filtered for SNPs. The result is a .vcf file.

The first step in calling variants from the sorted.bam file is to run the mpileup function. This is a command that is part of the BCFtools package. The mpileup command compiles variants between the sequenced reads and the reference genome. The bcftools mpileup command will output a BCF file, which is the binary variant call format. The command used is as follows:

```
(base) reid@reid-VirtualBox:~/project$ bcftools mpileup -Ou -f reference.fna -o SRR.9916705.pileup.bcf SR
R9916705.sorted.bam
```

The -Ou argument dictates that the output file is the uncompressed vcf file as opposed to the compressed bcf file. This format is easier to work with and does not require the use of a compression algorithm. The -f argument is the input for the reference genome. The -o argument dictates the name of the output file.

The next step is calling the variants from the generated SRRR9916705.pileup.bcf file. This can be accomplished using the bcftools -call command:

```
(base) reid@reid-VirtualBox:~/project$ bcftools call -m -v -Ou -o SRR9916705.call.bcf SRR9916705.pileup.b
cf
```

The -m option tells bcftools to use the default BCFtools calling method as opposed to using the original consensus calling method found within SAMtools. The -v option tells BCFtools to only output the detected variants and remove all loci that do not have variants. The -Ou option tells BCFtools to output the file in bcf format as opposed to vcf. The -o option dictates the name of the output file that will be saved to the current working directory following completion of the command.

The bcf file should be realigned to the reference genome to ensure accurate variant positioning. This is accomplished using the bcftools -norm command:

```
(base) reid@reid-VirtualBox:~/project$ bcftools norm -Ou -f GRCh13_latest_genomic.fna -d all -o SRR991670
5.norm.bcf SRR9916705.call.bcf
```

The -Ou option again dictates that the output file be in the binary BCF format. The -f option sets the reference genome that the input SRR9916705.call.bcf file is being aligned against. The -d option tells the function to consider all positions during the process of the function. The output file is saved as SRR9916705.norm.bcf in the current working directory.

The final step in the variant calling process is to filter the SRR9916705.variants.bcf file to detect SNPs. BCFtools again has a -filter command that makes the process very simple:

```
(SNPcall2) reid@reid-VirtualBox:~/SNPdemo$ bcftools filter -Ob -e 'QUAL<40 || DP<10 || GT!="0/1"' -o SRR9916705
.variants.bcf SRR9916705.norm.bcf
```

In this case the -Ob argument creates the output of the command in the compressed version and saves it as SRR9916705.variants.bcf in the current working directory. The -e option sets the filtering command to exclude from the output any variants that do not meet the proceeding filtering criteria. Alternatively, the command will accept the -i option which would include the

filtering criteria. The QUAL portion of the filtering filters out any nucleotide position with a quality score less than 40. The DP portion of the filtering criteria sets the minimum read depth at 10X for the potential variant to be included. The GT!=”0/1” portion of the filtering criteria excludes any heterozygous variants.

An additional conversion of the output SRR9916705.variants.bcf to a non-binary VCF format makes the file human-readable. This can be accomplished using BCFtools view:

```
(base) reid@reid-VirtualBox:~/project$ bcftools view -Ov -o SRR9916705.variance.vcf SRR
9916705.variance.bcf
```

The -Ov option specifies the output file as a non-binary VCF format. The -o option names the output VCF file that will be saved into the current working directory. The final SRR9916705.variance.vcf file can be found here: https://drive.google.com/file/d/1v35hKVyFdm_o4qtamVoqTc_2CfaTerd_/view?usp=sharing

REFERENCES

H.I. Avi-Itzhak, X. Su, F.M. De La Vega. Selection of Minimum Subsets of Single Nucleotide Polymorphisms to Capture Haplotype Block Diversity. Pacific Symposium on Biocomputing 8:466-477(2003)

DeStefano V, Martinelli I, Mannucci P, et al. The risk of recurrent deep venous thromboembolism among heterozygous carriers of the G20210A prothrombin gene mutation**.** Br J Hematol**.** 2001; *113*: 630–635