# Programming for Bioinformatics | BIOL7200

## Week 6 Exercise

September 24, 2018

This week's assignment is the first "real" programming task and is a wrap up for shell. Everything that you have learned so far, all the bits and pieces, will be used this week to create a pipeline.

### Instructions for submission

- This assignment is due Monday, September 30 2019 at 11:59 pm, late submissions will receive a 0.
- Download test input files here: D2-DS3_paired1.fq.gz and D2-DS3_paired2.fq.gz
- The reference genome is Human genome assembly hg38, Chromosome 17. You can find this on the Golden Path. Once you download it, uncompress it.
- Fork this repository; make it private; clone and edit the skeleton provided; add, commit, and push your work back to GitLab. *Do not send merge requests*
- You will be required to run your pipeline, *with realignment and the -z flag*, and upload the resulting VCF (named `gtusername.vcf.gz`) and GATK logs (named `gtusername.log`) to Canvas.
- You will also be required to convert the VCF file you generate into a BED file (as specified here: Converting VCF to BED). Again upload these files (`snps.txt` and `indels.txt`) to Canvas
- Your code and test output will be automatically graded using the continuous integration system (a.k.a The Autograder)
- Your Canvas submissions will be graded based on their correctness when compared to a reference VCF and BED files produced by the TAs

### SNP-calling pipeline

You will be required to create a pipeline for calling Single Nucleotide Polymorphisms (SNPs).

*Underlying biology and problem description*: We are living in the post-genomic era where sequencing genomes now takes under a day; thousands of genomes have been sequenced and assembled. Many (not all) organisms, whether prokaryote or eukaryote, now have a complete reference (or a canonical) genome sequenced and assembled already. Given that assembling genomes *de novo* requires substantial expertise along with computational and human resources, a more practical approach of analyzing genomes is to map the genomic reads to an existing reference. This approach is also referred to as genome re-sequencing.

Once you have your reads mapped to an existing reference strain, the process can diverge into multiple directions depending on the nature and objective of the project. One common direction is to call variants. Variants are simply the bases in your sample's genome that are different from the corresponding bases in the reference genome. These differences can (or cannot) lead to a range of phenotypic differences either directly (as is the case in amino acid changes or nonsynonymous changes) or indirectly (a change of base in the splice site leading to change in splicing pattern or change in the regulatory region leading to enhanced or depleted gene expression). In this exercise, we will write a pipeline for mapping genomic reads to a reference genome and calling SNPs from the mapping. You have seen most of these commands before, be sure to use the WGS/WES Mapping to Variant Calls – Version 1.0 as a guide

## Pipeline requirements:

The pipeline will have the following steps:

1. Align FASTQ reads to a reference genome to create an alignment file
2. Processing the alignment file (file format conversion, sorting, alignment improvement)
3. Calling the variants

You will be using the following tools for the development of the pipeline:

1. bwa for the alignment
2. samtools/HTS package for processing and calling variants
3. GATK for improving the alignment. You must use GATK v3.7.0, available on the Archived version page

The workflow that you will be adopting is outlined here: WGS/WES Mapping to Variant Calls – Version 1.0

We've chosen this workflow, along with a slightly older version of GATK, because it is straight forward and used widely in the field. The newer version GATK (GATK4) uses a more complicated workflow but has not yet been standardized for many applications. You will find that older versions of software are used for many years because a great deal of work has gone into verifying their outputs in, for example, a clinical setting. Updating this pipeline to the newest version is a single afternoon's work if you really need GATK4.

## Code requirements:

1. Comment your script! We cannot stress this enough. Commenting in programming is as important as the code itself. Even the best in the field do it and so should you. By

commenting I simply mean this – have a brief explanation in your code to explain exactly what you are doing to your future self or someone else who is looking into your code. Here is an example if you still do not understand what we mean: Wikipedia Article on Comments

2. Input command line options:

| Option | Meaning |
|---|---|
| `-a` | Input reads file – pair 1 |
| `-b` | Input reads file – pair 2 |
| `-r` | Reference genome file |
| `-e` | Perform read re-alignment |
| `-o` | Output VCF file name |
| `-f` | Mills file location |
| `-z` | Output VCF file should be gunzipped (*.vcf.gz) |
| `-v` | Verbose mode; print each instruction/command to tell the user what your script is doing right now |
| `-i` | Index your output BAM file (using `samtools index`) |
| `-h` | Print usage information (how to run your script and the arguments it takes in) and exit |

3. Required functionality:
    1. Mapping
    2. Improvement
        a. Realign
        b. Index bam
    3. Variant Calling
    4. File Checks (see below)

4. File checks:
    a. Do the input reads files exist? If either of them doesn't exist, prompt which one is missing and exit.
    b. Does the reference genome exist? If not, prompt and exit.
    c. Does the output VCF file already exist? Prompt that the output VCF file already exists and ask the user what he/she wants to do – do they want to overwrite the existing file or exit the program? The input of their response can be received using the `read` command.

5. Although you will be working with human data here, your script should be flexible enough to accommodate any species of data, e.g., cichlid genomes. The only way of doing this is to make sure you do not hardcode things.

6. Variable names: Please name your variables exactly as specified in this table. The autograder will use these variable names to validate your input. You can, and will have to, define other variables while writing your scripts but the following **must** be defined in your `getopts` block.

| Variable Name | Meaning |
|---|---|
| `reads1` | Location of first reads file |
| `reads2` | Location of second reads file |
| `ref` | Location of reference genome (uncompressed) |
| `output` | Name of output file |
| `millsFile` | Location of Mills Indel File |
| `realign` | Variable for saving status of `-e` (1 or 0) |
| `gunzip` | Variable for saving status of `-z` (1 or 0) |
| `v` | Variable for saving status of `-v` (1 or 0) |
| `index` | Variable for saving status of `-i` (1 or 0) |
| `answer` | Variable for checking whether the input file needs to be overwritten |

**Please do not change the names of the functions in the skeleton**. The autograder will rely on finding these exact functions in your script. Again, you can define additional functions as you see fit.

## Create BED from VCF

For this part of the assignment, you can use the different bash utilities you have learned so far (`awk`, `sed`, etc.) to convert your VCF to the following format:

```
Chromosome  Start Stop  Length
```

Stay with me! To do this, you need to remove the headers from your VCF file (any line that starts with a #). Next, you need to get the difference in length between the REF and ALT in your VCF file.

Eg:

This is what part of a VCF file will look like:

```
...
#CHROM   POS       ID        REF       ALT       QUAL      FILTER   INFO      FORMAT
sample_1
chr17      151035    .              T              C              225          .
DP=27;VDB=0.17647;SGB=-0.692914;MQSB=1;MQ0F=0;AC=2;AN=2;DP4=0,0,11,14;MQ=60
GT:PL                                                                 1/1:255,75
chr17      151041    .              G              A              225          .
DP=27;VDB=0.161205;SGB=-0.692976;MQSB=1;MQ0F=0;AC=2;AN=2;DP4=0,0,12,14;MQ=60
GT:PL                                                                 1/1:255,78
chr17      152076    .              A              T              225          .
DP=23;VDB=0.828962;SGB=-0.692562;MQSB=1;MQ0F=0;AC=2;AN=2;DP4=0,0,9,13;MQ=60
GT:PL                                                                 1/1:255,66
chr17       155729    .              AGTGT      AGT           225          .
INDEL;IDV=10;IMF=0.909091;DP=11;VDB=0.907553;SGB=-
0.676189;MQSB=0.964642;MQ0F=0;AC=2;AN=2;DP4=0,0,7,4;MQ=60               GT:PL
1/1:255,33,0
...
```

Here, we can see that there are 3 SNPs (Single Nucleotide Polymorphisms) and 1 indel.

Upon conversion, we want you to create a file that looks like this:

```
...
chr17                    151035                    151035                    0
chr17                    151041                    151041                    0
chr17                    152076                    152076                    0
chr17                   155729                    155727                    -2
...
```

Let's look at what happened above.

| Column | Explanation |
|---|---|
| 1. *Chromosome* | Lists the Chromosome on which the mutation is seen; First column of the VCF file |
| 2. *Start* | Lists the starting position of the mutation - this is the POS field from the VCF file |
| 3. *Stop* | Lists the end position of the mutation; Formula: LengthOf(Alternate) - LengthOf(Reference) + *Start* |
| 4. *Length* | LengthOf(Alternate) - LengthOf(Reference) |

Almost there!

A commonly encountered problem when dealing with genomic data is the way chromosomes are represented. Some programs/sources write them as *chr17* while some will just write them as *17*. Assume that you have to change the way chromosomes are represented in your output file to just numbers (removing chr).

This is what the file needs to look like:

```
...
17                 151035                151035                0
17                 151041                151041                0
17                 152076                152076                0
17                155729                155727                -2
...
```

Finally, we want to write out the SNPs and indels into different files. *i.e.* we want two files like:

indels.txt

```
...
17                155729                155727                -2
...
```

snps.txt

```
...
17                 151035                151035                0
17                 151041                151041                0
17                 152076                152076                0
...
```

All the steps above can be accomplished using a single (albeit long) one-liner in bash.

## Submission Instructions:

- Once you generate the summary files as described above, upload them as *snps.txt* (for the file with information about SNPs) and *indels.txt* (for the file with information about indels) on Canvas.

Again: this assignment is due Monday, September 30, 2019, at 11:59 pm, late submissions will receive a 0.