Programming for Bioinformatics
BIOL 8803 B
October 5th, 2015

This is a wrap up week for shell. Everything that you have learned so far, all the bits and pieces, will be used this week to create a pipeline.

SNP-calling pipeline

This week, you will be required to create a pipeline for calling Single Nucleotide Polymorphisms.

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. Most (not all) of the 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 a substantial expertise, computational and human resources, a more practical approach of analyzing genomes that is being widely adopted worldwide is to map the genomic reads to an existing reference. This approach is also referred as genome resequencing.

Having the genomic 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 in your genome of interest. You have done variant calling in the past (week 4). Variants are simply the bases in your genome of interest that are different from the corresponding bases in the reference genome. These differences can (or cannot) yield to a range of phenotypic differences either directly (as is the case in amino acid changes or non-synonymous 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. Since you have written most parts of the script in the past, I do not anticipate a lot of trouble here.

Pipeline requirements:
The pipeline will have the following steps:

1) Aligning genomic reads to the reference genome
2) Processing the alignment file (conversion, sorting, alignment improvement)
3) Calling the variants

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

bwa for the alignment
http://bio-bwa.sourceforge.net/

samtools/HTS package for processing and calling variants
http://www.htslib.org/

GATK for improving the alignment
https://www.broadinstitute.org/gatk/

The workflow that you will be adopting is outlined here:
http://www.htslib.org/workflow/

Code requirements:

1) **Comment your script! I 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 what you are doing exactly to your future self or someone else who is looking into your code. Here is an example if you still do not understand what I mean:
http://en.wikipedia.org/wiki/Comment_(computer_programming)

2) Input command line options:

-a	Input reads file – mate 1
-b	Input reads file – mate 2
-g	Reference genome file
-o	The output VCF file
-e	Do reads re-alignment
-z	If the 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 and exit i.e., how to run your script and what are the arguments it takes in

3) File checks:
Before the script starts running, it should perform the following file checks:
   a) Does the input reads file exist? If any of them doesn't exist, prompt which one is missing and exit
   b) Does the reference genome exist? If not, prompt and exist
   c) Does the output VCF file exist? Prompt that the output VCF file exist and ask user what he wants to do – does he wants to overwrite the existing file or exit the program. The input can be received using the `read` command.

4) Although you will be working with human data here, your script should be flexible enough to accommodate any species data say a cichlid genome. The only way of doing this is to make sure you **do not hardcode things**.

a_variable=$(some command) puts the result of some command into a_variable

Input files for testing your script:
You can find input file on http://jordan.biology.gatech.edu/biol8803b/ to test your script: D2-DS3_paired1.fq, D2-DS3_paired2.fq
The reference genome is human genome assembly hg19 chromosome 17

Deliverables

- Code: week7.sh
- Text files:
  1) a VCF file containing your final output
  2) a README.txt file containing a brief description on your final results: What are those results? How did you get them? What do they mean?

Additional Instructions on code submission
Before you start writing and testing your code, you should first create a directory structure looks like:

```
week7/
        |------ week7.sh
        |------ wrapper7.sh
        |------ data/
                    |------ D2-DS3_paired1.fq
                    |------ D2-DS3_paired2.fq
                    |------ chr17.fa
                    |------ Mills_and_1000G_gold_standard.indels.hg19.sites.vcf
        |------ dependencies/
                    |------ GenomeAnalysisTK.jar
                    |------ {Optional: Other dependencies (file size should not exceed t-square limit*)}
```

- The files/directories in black should be submitted on t-square (any additional file other than week7.sh will be considered as "Other dependencies").
- The files/directories in grey do **NOT** need to submitted on t-square.
- `BWA, Samtools, java` and `bcftools` are assumed to be in your path.
- **How it works:** When your code is tested/graded, we will first create a directory structure with all grey files/directories listed above. Then the black files/directories from your t-square submission will be placed in corresponding paths as shown above.
- **Make sure the non-optional file/directory names in your directory are EXACTLY the same as shown above.**

*http://info.t-square.gatech.edu/node/253