Programming for Bioinformatics

BIOL 7200

October 17th, 2016

This is a wrap up week for shell. Everything that you have learned so far, all of 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 (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. 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://software.broadinstitute.org/gatk/>

The workflow that you will be adopting is outlined here: <http://www.htslib.org/workflow/> (WGS/WES Mapping to Variant Calls - Version 1.0)

**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 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 I mean: <https://en.wikipedia.org/wiki/Comment_(computer_programming)>
2. Input command line options:

|  |  |
| --- | --- |
| -a | Input reads file – pair 1 |
| -b | Input reads file – pair 2 |
| -r | 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 | 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 |

1. Required and optional functionalities

You should implement ALL required functionalities for users to run. You will get 5 bonus points for this exercises if you also implement the optional functionalities. Remember to assign additional options (which are not the ones already used in the **2)** section) to allow user to choose whether if they want to use those additional functionalities. Comment the optional functionalities in your code and explain it clearly in the README file.

* Mapping(required)
* Improvement
  + Realign (required)
  + Base recalibrate (optional)
  + Merge reads (optional)
  + Index bam (required)
* Variant Calling (required)

1. File checks:

Before the script starts running, it should perform the following file checks:

* 1. Do the input reads files exist? If either of them don’t exist, prompt which one is missing and exit.
  2. Does the reference genome exist? If not, prompt and exit.
  3. 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.

1. 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**.

a\_variable=$(<some command>) puts the result of <some command> into a\_variable

**Input files for testing your script:**

You can find input reads files to test your script at <http://jordan.biology.gatech.edu/biol7200/>

The input reads files are D2-DS3\_paired1.fq and D2-DS3\_paired2.fq

The reference genome is the human genome assembly hg38, chromosome 17

**Deliverables**

* Code: week7.sh
* Text files:

1. a VCF file containing your final output
2. a README.txt file containing
   1. Instructions on how to run your pipeline
   2. The structure of your pipeline directories (could be similar to the structure shown below)
   3. Explain what your results are
   4. Briefly describe how you got those results and what they mean

* Other files and dependencies (see the last section “**Instructions for submission**”)

**Additional Instructions on code submission**

Before you start writing and testing your code, you should *first create a directory structure as shown below:*

week7/ # Your root directory for this exercise

|------ week7.sh

|------ README.txt

|------ data/

|------ D2-DS3\_paired1.fq

|------ D2-DS3\_paired2.fq

|------ chr17.fa

|------ Mills1000G.b38.vcf

|------ lib/ #where your dependency file(s) should go to

|------ GenomeAnalysisTK.jar

|------ [**Other dependencies**]#file size should not exceed t-square limit\*

|------ tmp/ #where your temp file(s) should go to

|------ output/ #where ALL your final output file(s) should go to

|------ output.vcf.gz

|------ [**Other final output file(s) to be submitted**]

* You should have **both black and grey** files/directories on your computer.
* The files/directories **in black** should **be submitted to 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, bcftools and tabix are assumed to be **in your path**.
* **How it works:** When your code is tested/graded, we will have a directory structure as shown above with all *grey files/directories*. Then the *black files/directories* from your t-square submission will be placed in corresponding paths as shown above.

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

**Instructions for submission:**

**Compress your week7 directory (*i.e.* your root directory for this exercise) including only the necessary files/subdirectories(as shown in black on the previous page) in ONE tar or zip file and submit on T-Square.**