Practical exercises: Variant calling

An online version of this document can be found here https://tinyurl.com/yb5ak2xb. Please feel free to add comments if anything is unclear or incorrect. The answers to the exercises can be found at the end of this document. However, try to figure out answers yourself as that is the most effective way to learn.

Exercise 1: Making sense of the input data

We will need the aligned sequences in SAM, BAM or CRAM format and the reference genome in fasta format. To list the files in the current directory, type

```
ls -lh
```

The listing shows two mouse data fragments, strains A/J and NZO, and the chromosome 19 of the mouse reference genome.

Before variant calling, it is important to check the data we'll be working with. Using the command below, get some statistics from the bams

```
samtools stats -r GRCm38_68.19.fa A_J.bam > A_J.stats
samtools stats -r GRCm38_68.19.fa NZO.bam > NZO.stats
plot-bamstats -r GRCm38_68.19.fa.gc -p A_J.graphs/ A_J.stats
plot-bamstats -r GRCm38_68.19.fa.gc -p NZO.graphs/ NZO.stats
```

1.1 Open the plots `A_J.graphs/index.html` and `NZO.graphs/index.html` in your firefox browser (e.g., `firefox A_J.graphs/index.html`). What is the percentage of mapped reads in both strains? Check the insert size, GC content, per-base sequence content and quality per cycle graphs. Do all look reasonable?

Exercise 2: Generating pileup

The command `samtools mpileup` prints the read bases that align to each position in the reference genome. On the command line, try this

```
samtools mpileup -f GRCm38_68.19.fa A_J.bam | less -S
```

Each line corresponds to a position on the genome. The columns are: chromosome, position, reference base, read depth, read bases (dot `.` and comma `,` indicate match on the forward and on the reverse strand; `ACGTN` and `acgtn` a mismatch on the forward and the reverse strand) and the final column is the base qualities encoded into characters. The caret symbol `^` marks the start of a read, the dollar sign `$` the end of a read, deleted bases are represented by asterisk `*`.

2.1 What is the read depth at position `10001994`? (Rather than scrolling to the position, use the substring searching capabilities of `less`: press `/`, then type `10001994` followed by `enter` to find the position.)

2.2 What is the reference and the alternate base at the position? How many reference and how many non-reference bases there are?

This output can be used for a simple consensus calling. One rarely needs this type of output. Instead, for a more sophisticated variant calling method, go to the next section.

Exercise 3: Generating genotype likelihoods and calling variants

The `bcftools mpileup` command can be used to generate genotype likelihoods. (Beware: the command `mpileup` is present in both `samtools` and `bcftools`, but in both they do different things. While `samtools mpileup` produces the text pileup output from the previous exercise, `bcftools mpileup` generates a VCF with genotype likelihoods.)

Run the following command (when done, press `q` to quit the viewing mode):

```
bcftools mpileup -f GRCm38_68.19.fa A_J.bam | less -S
```

This generates an intermediate output which contains genotype likelihoods and other raw information necessary for variant calling. This output is usually streamed directly to the caller like this

```
bcftools mpileup -f GRCm38_68.19.fa A_J.bam | bcftools call -m | less -S
```

> **3.1** The output above contains both variant and non-variant positions. Check the `Input/output options` section of the `bcftools call` usage page and see if there is an option to print out only variant sites.

The `INFO` and `FORMAT` fields of each entry tells us something about the data at the position in the genome. It consists of a set of key-value pairs with the tags being explained in the header of the VCF file (see the `##INFO` and `##FORMAT` lines in the header).

Let `mpileup` output more information. For example, we can ask it to add the `FORMAT/AD` tag which informs about the number of high-quality reads that support alleles listed in REF and ALT columns. The list of all available tags can be printed with the command

```
bcftools mpileup -a ?
```

Now let's run the variant calling again, this time adding the `-a AD` option. We will also add the `-Ou` option so that it streams a binary uncompressed BCF into call. This is to avoid the unnecessary CPU overhead of formatting the internal binary format to plain text VCF only to be immediately formatted back to the internal binary format again

```
bcftools mpileup -a AD -f GRCm38_68.19.fa A_J.bam -Ou | bcftools call -mv -o
out.vcf
```

Examine the VCF file output using the unix command less

```
less -S out.vcf
```

> **3.2** What is the reference and the alternate base at the position 10001994?
>
> **3.3** What is the total raw read depth at the position 10001994? (Note that this number may be different from the values we obtained earlier, because some low quality reads or bases might have been filtered previously.)
>
> **3.4** What is the number of high-quality reads supporting the SNP call at position 10001994? How many reads support the reference allele and how many support the alternate allele? Look up the `AD` tag in the FORMAT column: the first value gives the number of reference reads and the second gives the number of non-reference reads.
>
> **3.5** What sort of event is at position 10003649?

Exercise 4: Variant filtering

In the series of commands we will learn how to extract information from VCFs and how to filter the raw calls. We will use the `bcftools` commands again. Most of the commands accept the `-i, --include` and

`-e, --exclude` options (https://samtools.github.io/bcftools/bcftools.html#expressions) which will come handy when filtering using fixed thresholds. We will estimate the quality of the callset by calculating the ratio of transitions and transversions (https://en.wikipedia.org/wiki/Transversion).

When drafting commands, it is best to build them gradually. This prevents errors and allows to verify that they work as expected. Let's start with printing a simple list of positions from the VCF using the bcftools query command (https://samtools.github.io/bcftools/bcftools.html#query) and pipe through the head command to limit the printed output to the first few lines:

```
bcftools query --format 'POS=%POS\n' out.vcf | head
```

As you could see, the command expanded the formatting expression `POS=%POS\n` in the following way: for each VCF record the string `POS=` was copied verbatim, the string `%POS` was replaced by the VCF coordinate stored in the POS column, and then the newline character `\n` ended each line. (Without the newline character, positions from the entire VCF would be printed on a single line.)

Now add the reference and the alternate allele to the output. They are stored in the REF and ALT column in the VCF, and let's separate them by a comma:

```
bcftools query -f'%POS %REF,%ALT\n' out.vcf | head
```

In the next step add also the quality (`%QUAL`), genotype (`%GT`) and sequencing depth (`%AD`) to the output. Note that FORMAT tags must be enclosed within square brackets `[...]` to iterate over all samples in the VCF. (Check the Extracting per-sample tags section in the manual https://samtools.github.io/bcftools/howtos/query.html for a more detailed explanation why the square brackets are needed.)

```
bcftools query -f'%POS %QUAL [%GT %AD] %REF %ALT\n' out.vcf | head
```

Now we are able to quickly extract important information from the VCFs. Now let's filter rows with QUAL smaller than 30 by adding the filtering expression `--exclude 'QUAL<30'` or `--include 'QUAL>=30'` like this:

```
bcftools query -f'%POS %QUAL [%GT %AD] %REF %ALT\n' -i'QUAL>=30' out.vcf | head
```

Now compare the result with the output from the previous command, were the low-quality lines removed? In the next step limit the output to SNPs and ignore indels by adding the `type="snp"` condition to the filtering expression. Because both conditions must be valid at the same time, we request the AND logic using the `&&` operator:

```
bcftools query -f'%POS %QUAL [%GT %AD] %REF %ALT\n' -i'QUAL>=30 && type="snp"'
out.vcf | head
```

4.1 Can you print SNPs with `QUAL` bigger than `30` and require at least `25` alternate reads in the `AD` tag?
Remember, the first value of the `AD` tag is the number of reference reads, the second is the number of alternate reads, therefore you will need to query the second value of the `AD` tag. The first value can be queried as `AD[0]` and the second as `AD[1]` (the allele indexes are zero-based). In case of FORMAT fields, also the queried sample must be selected as `AD[sample:subfield]`. Therefore add to the expression the condition `AD[0:1] >= 25` to select the first (and in our case the only one) sample or `AD[*:1] >= 25` to select any sample for which the condition is valid.

Now we are able to filter our callset. In order to evaluate the quality, we will use `bcftools stats` to calculate the ratio of transitions vs transversions. We start by checking first what is the ts/tv of the raw unfiltered callset. The `stats` command produces a text output, we extract the field of interest as follows:

```
bcftools stats out.vcf | less
bcftools stats out.vcf | grep TSTV
bcftools stats out.vcf | grep TSTV | cut -f5
```

4.2 Calculate ts/tv of the set filtered as above by adding `-i 'QUAL>=30 && AD[*:1]>=25'` to the `bcftools stats` command. (Here the asterisk followed by a colon tells the program to apply the filtering to all samples. At least one sample must pass in order for a site to pass.) After applying the filter, you should observe an increased ts/tv value.

4.3 Can you do the reverse and find out the ts/tv of the **removed** sites? Use the `-e` option instead of `-i`. The ts/tv of the removed low-quality sites should be lower.

4.4 The test data come from an inbred homozygous mouse, therefore any heterozygous genotypes are most likely mapping and alignment artefacts. Can you find out what is the ts/tv of the heterozyous SNPs? Do you expect higher or lower ts/tv? Use the filtering expression `-i 'GT="het"'` to select sites with heterozygous genotypes.

Another useful command is `bcftools filter` which allows to "soft filter" the VCF: instead of removing sites, it can annotate the `FILTER` column to indicate sites which fail. Apply the above filters to produce a final callset, adding also the `--SnpGap` and the `--IndelGap` option to filter variants in close proximity to indels:

```
bcftools filter -s LowQual -i'QUAL>=30 && AD[*:1]>=25' -g8 -G10 out.vcf -o
out.flt.vcf
```

Exercise 5: Variant normalization

The same indel variant can be represented in different ways. For example, consider the following 2bp deletion. Although the resulting sequence does not change, the deletion can be placed at two different positions within the short repeat:

```
          12345
          TTCTC
POS=1     T--TC
POS=3     TTC--
```

In order to be able to compare indels between two datasets, we left-align such variants.

5.1 Use the `bcftools norm` command to normalize the filtered callset. Note that you will need to provide the `--fasta-ref` option. Check in the output how many indels were realigned.

Exercise 6: Multi-sample variant calling

In many types of experiments we want to sequence multiple samples and compare their genetic variation. The single-sample variant calling we have done so far has the disadvantage of not providing information about reference genotypes. Because only variant sites are stored, we are not able to distinguish between records missing due to reference genotypes versus records missing due to lack of coverage.

6.1 Type `ls *bam` to check that there are two BAM files in the directory. Can you modify the calling command from **Exercise 3** to use both BAM files? This time we want to output the calls as a compressed BCF. Write the output to a BCF file called `multi.bcf` and index the file afterwards.

6.2 Apply the same filters as before and write the output to a BCF file called `multi.filt.bcf`, then index the file.

6.3 What is the ts/tv of the raw calls and of the filtered set?

6.4 What is the ts/tv of the removed sites?

Exercise 7: Data visualization

It is often useful to visually inspect a SNP or indel of interest in order to assess the quality of the SNP call and interpret the genomic context of the SNP. We can use the `IGV` tool to view some of the SNPs positions from the VCF file. Assuming the location of the `IGV` launching script relative to your current working directory is `igv.sh,` type on the command line

```
igv.sh
```

Set the reference genomes to be mouse (`mm10/GRCm38`) by clicking on the `Genomes` then `Load Genomes from Server` buttons and select `mm10`. On the file menu down the left, select `Load from File` and select the BAM file `A_J.bam`. In the top bar, enter the position `chr19:10,001,874-10,002,017` to view the SNP bases at position `10001946`.

7.1 How many forward aligned reads support the SNP call? Note: hover the mouse pointer over the coverage bar at the top (or click, depending on the IGV settings) to get this information.

7.2 Check in the VCF if this SNP was called by `bcftools`? (Use `bcftools view -H -r chr19:10001946 multi.filt.bcf` to verify.) Did it pass the filters?

7.3 In the top bar, enter the position 'chr19:10072443' to view the SNP at position 10072443. Was the SNP also called by bcftools? Did it pass the filters? Does this look like a real SNP? Why?

Exercise 8: Functional consequences

There are several popular programs available for predicting functional consequences. Here we will use the lightweight `bcftools csq` command. It is haplotype-aware and can correctly predict consequences for frame-restoring and other compound variants such as MNPs. On input it requires a VCF, the fasta reference file and a GFF file with the gene model. Because our data is not phased, we will provide the `--phase` option (which does not actually phase the data, but tells the program to make an assumption about the phase):

```
bcftools view -i 'FILTER="PASS"' multi.filt.bcf | bcftools csq -p m -f
GRCm38_68.19.fa -g Mus_musculus.part.gff3.gz -Ob -o multi.filt.annot.bcf

bcftools index multi.filt.annot.bcf
```

8.1 The `bcftools csq` command annotated the VCF with a new INFO tag `BCSQ`. Use the `bcftools query -f '%BCSQ\n'` command to extract the consequence at position `19:10088937`. What is the functional annotation at this site? What is the amino acid change?

Answers to exercises:

1.1 You will find there is a consistent GC bias. However, in our case this is OK because the stats was produced from a small BAM fragment only, with a locally different GC content.

2.1 66 reads

2.2 The reference allele is `A` and the alternate allele is `G`. The upper/lower case letters indicate the forward/reverse orientation of the read.

3.1 Add the `-v` option to the command:

```
bcftools mpileup -f GRCm38_68.19.fa A_J.bam | bcftools call -mv | less -S
```

3.2 Look up the REF and ALT columns: the reference base is A, the alternate allele is G.
3.3 Look up the tag `DP` in the INFO column: there were 69 raw reads at the position.
3.4 There are 0 reference and 66 alternate high-quality reads.
3.5 Five bases TGTGG were inserted after the T at position 10003649

4.1 The complete command is

```
bcftools query -f'%POS %QUAL [%GT %AD] %REF %ALT\n' -i'QUAL>=30 &&
type="snp" && AD[*:1]>=25' out.vcf | head
```

4.2 The complete command is

```
bcftools stats -i'QUAL>=30 && AD[*:1]>=25' out.vcf | grep TSTV | cut -f5
```

4.3 The complete command is

```
bcftools stats -e'QUAL>=30 && AD[*:1]>=25' out.vcf | grep TSTV | cut -f5
```

4.4 The complete command is

```
bcftools stats -i 'GT="het"' out.vcf | grep TSTV | cut -f5
```

5.1 The complete command is

```
bcftools norm -f GRCm38_68.19.fa out.flt.vcf -o out.flt.norm.vcf
```

6.1 Use the commands

```
bcftools mpileup -a AD -f GRCm38_68.19.fa *.bam -Ou | bcftools call -mv -Ob
-o multi.bcf
bcftools index multi.bcf
```

6.2 Use the commands

```
bcftools filter -s LowQual -i'QUAL>=30 && AD[*:1]>=25' -g8 -G10 multi.bcf
-Ob -o multi.filt.bcf
bcftools index multi.filt.bcf
```

6.3 Use the commands

```
bcftools stats multi.filt.bcf | grep TSTV | cut -f5
bcftools stats -i 'FILTER="PASS"' multi.filt.bcf | grep TSTV | cut -f5
```

6.4 Use the command

```
bcftools stats -e 'FILTER="PASS"' multi.filt.bcf | grep TSTV | cut -f5
```

7.1 75 reads total, 39 on the forward and 36 on the reverse strand.
7.2 Use the command

```
bcftools view -H -r 19:10001946 multi.filt.bcf
```

7.3 It is an alignment artefact, the aligner prefered two SNPs instead of a long deletion. The call was removed by filtering only because we excluded calls in close proximity of indels:

```
bcftools view -H -r 19:10072443 multi.filt.bcf
```

8.1 The C>T mutation changes the amino acid at position 163 in the protein sequence, from valine to isoleucine.