Variant calling tutorial

SAMtools is a suite of commands for dealing with databases of mapped reads. You'll be using it quite a bit throughout the course. It includes programs for performing variant calling (mpileup-bcftools).

- Calling variants in reads mapped by bowtie
  - Load SAMtools
  - Prepare your directories
  - Index the FASTA reference file
  - Convert mapped reads from SAM to BAM, sort, and index
  - Call genome variants
- Optional Exercises
  - Calling variants in reads mapped by BWA or Bowtie2
  - Filtering VCF files with grep
  - Comparing the results of different mappers using bedtools
  - Further Optional Exercises
- From here...

Calling variants in reads mapped by bowtie

Right now, we'll be using it to call variants (find mutations) in the re-sequenced *E. coli* genome from the Mapping tutorial. You will need the output SAM files from that tutorial to continue here.

If you do not have the output from the Mapping tutorial, run these commands to copy over the output that would have been produced. Then, you can immediately start this tutorial!

```
cds
mkdir intro_to_mapping
cd intro_to_mapping
cp -r $BI/ngs_course/intro_to_mapping/bowtie .
cp -r $BI/ngs_course/intro_to_mapping/bwa .
cp -r $BI/ngs_course/intro_to_mapping/bowtie2 .
```

We assume that you are still working in the main directory called `intro_to_mapping` data that you created on `$SCRATCH`.

Load SAMtools

Load the SAMtools module (if not already loaded).

```
module load samtools
```

Can you figure out what version of samtools is loaded on TACC and where it is installed?

This should work:

```
samtools
which samtools
```

Prepare your directories

Create a new output directory called `samtools_bowtie` or whatever makes sense to you.

Let's copy over just the read alignment file in the SAM format and the reference genome in FASTA format to this new directory, so that we don't have so many files cluttering our space.

```
mkdir samtools_bowtie
cp bowtie/SRR030257.sam samtools_bowtie
cp bowtie/NC_012967.1.fasta samtools_bowtie
```

Index the FASTA reference file

First, you need to **index** the reference file. (This isn't indexing it for read mapping. It's indexing it so that SAMtools can quickly jump to a certain base in the reference.)

Then run this command to index the reference file.

```
samtools faidx samtools_bowtie/NC_012967.1.fasta
```

Take a look at the new *.fai file that was created by this command. Any idea what some of the numbers mean?

```
less samtools_bowtie/NC_012967.1.fasta.fai
```

Hint: Type q to exit less.

Convert mapped reads from SAM to BAM, sort, and index

SAM is a text file, so it is slow to access information about how any given read was mapped. SAMtools and many of the commands that we will run later work on BAM files (essentially GZIP compressed binary forms of the text SAM files). These can be loaded much more quickly. Typically, they also need to be sorted, so that when the program wants to look at all reads overlapping position 4,129,888, it can easily find them all at once without having to search through the entire BAM file.

Convert from SAM to BAM format.

```
samtools view -b -S -o samtools_bowtie/SRR030257.bam samtools_bowtie/SRR030257.sam
```

Sort and index the BAM file.

```
samtools sort samtools_bowtie/SRR030257.bam samtools_bowtie/SRR030257.sorted
samtools index samtools_bowtie/SRR030257.sorted.bam
```

This is a really common sequence of commands, so you might want to add it to your personal cheat sheet.

- What new files were created by these commands?

  List the contents of the output directory

  ```
  ls samtools_bowtie
  ```

  Expected output

  ```
  NC_012967.1.fasta      SRR030257.sorted.bam.bai
  NC_012967.1.fasta.fai  SRR030257.sam
  SRR030257.bam          SRR030257.sorted.bam
  ```

- Why didn't we name the output `SRR030257.sorted.bam` in the `samtools sort` command?
  Samtools appends an extra .bam to whatever we put here, so it would have created SRR030257.sorted.bam.bam, and we would have had to make a joke about the Flintstones.
- Can you guess what a *.bai file is?
  Sure enough, it's the index file for the BAM file.

Hint: You might be tempted to `gzip` BAM files when copying them from one computer to another. Don't bother! They are already internally compressed, so you won't be able to shrink the file. On the other hand, compressing SAM files will save a fair bit of space.

Call genome variants

Now we use the `mpileup` command from `samtools` to compile information about the bases mapped to each reference position.

Output BCF file. This is a binary form of the text Variant Call Format (VCF).

This is one command. Put it all on one line.

```
samtools mpileup -u -f samtools_bowtie/NC_012967.1.fasta samtools_bowtie/SRR030257.sorted.bam > samtools_bowtie/SRR030257.bcf
```

What are all the options doing?

Convert BCF to human-readable VCF:

This is one command. Put it all on one line.

```
bcftools view -v -c -g samtools_bowtie/SRR030257.bcf > samtools_bowtie/SRR030257.vcf
```

What are these options doing?

Take a look at the `samtools_bowtie/SRR030257.vcf` file using `less`. It has a nice header explaining what the columns mean. Below this are the rows of data describing potential genetic variants.

Optional Exercises

Calling variants in reads mapped by BWA or Bowtie2

Follow the same directions to call variants in the BWA or Bowtie2 mapped reads.

Just be sure you don't write over your old files. Maybe create new directories like `samtools_bwa` and `samtools_bowtie2` for the output in each case.

You could also try running all of the commands from inside of the `samtools_bwa` directory, just for a change of pace.

Filtering VCF files with grep

VCF format has alternative Allele Frequency tags denoted by AF1. Try the following command to see what values we have in our files.

```
grep AF1 samtools_bowtie/SRR030257.vcf
```

Optional: For the data we are dealing with, predictions with an allele frequency not equal to 1 are not really applicable. (The reference genome is haploid. There aren't any heterozygotes.) How can we remove these lines from the file?

What does the -v flag do in grep?

```
grep -v *something*
```

```
cat input.vcf | grep AF1=1 > output.vcf
```

Is not practical, since we will lose vital VCF formatting and may not be able to use this file in the future.

```
cat input.vcf | grep -v AF1=0 > output.vcf
```

Will preserve all lines that don't have a AF1=0 value and is one way of doing this.

```
sed -i '/AF1=0/ d' input.vcf
```

Is a way of doing it in-line and not requiring you to make another file. (But it writes over your existing file!)

Comparing the results of different mappers using bedtools

You will need the output from #Calling variants in reads mapped by BWA or Bowtie2 to complete this exercise.

Often you want to compare the results of variant calling on different samples or using different pipelines. Bedtools is a suite of utility programs that work on a variety of file formats, one of which is conveniently VCF format. It provides many ways of slicing, dicing, and comparing the information in VCF files. For example, we can use it to find out what predictions are the same and which are different from the variant calling on reads mapped with different programs.

Set up a new output directory and copy the respective VCF files to it, renaming them so that we know where they came from:

```
mkdir comparison
cp samtools_bowtie/SRR030257.vcf comparison/bowtie.vcf
cp samtools_bwa/SRR030257.vcf comparison/bwa.vcf
cp samtools_bowtie2/SRR030257.vcf comparison/bowtie2.vcf
cd comparison
```

Use the subcommands `bedtools intersect` and `bedtools subtract` we can find equal and different predictions between mappers. Try to figure out how to to do this on your own first. **Hint:** Remember that adding `> output.vcf` to the end of a command will pipe the output that is to the terminal into a file, so that you can save it.

Load Bedtools.

```
module load bedtools
```

Finding common mutations.

```
bedtools intersect -a bowtie.vcf -b bwa.vcf > common_bowtie_bwa.vcf
```

Finding mutations that are unique for each mapper.

```
bedtools subtract -a bowtie.vcf -b common_bowtie_bwa.vcf > unique_bowtie.vcf
bedtools subtract -a bwa.vcf -b common_bowtie_bwa.vcf > unique_bwa.vcf
```

Further Optional Exercises

- Which mapper finds more variants?
- Can you figure out how to filter the VCF files on various criteria, like coverage, quality, ... ?
- How many high quality mutations are there in these *E. coli* samples relative to the reference genome?

From here...

*Look at how the reads supporting these variants were aligned to the reference genome in the Integrative Genomics Viewer (IGV) tutorial.
*Look into more sophisticated Variant+calling+with+GATK.