Lab 7
Single nucleotide and structural variation

Ewa Szczurek
MIMUW
Szczurek@mimuw.edu.pl

I. Single nucleotide variation

The data for this tutorial has been produced from human whole genomic DNA. Only reads that have mapped to a 2MB part of chromosome 20 have been used, to make the data suitable for an tutorial in feasable timescale. There are about one million 100bp reads in the dataset, produced on an Illumina HiSeq2000 (30x coverage, paired-end). This data was generated as part of the 1000 Genomes project:
http://www.1000genomes.org/

Exercise 1

Mapping reads to the genome using Bowtie 2.

Bowtie2 is a powerful tool with many options, which you can study in full at home by reading
http://bowtie-bio.sourceforge.net/bowtie2/manual.shtml
and trying out
http://bowtie-bio.sourceforge.net/bowtie2/manual.shtml#getting-started-with-bowtie-2-lambda-phage-example
The main difference to Bowtie (two labs ago) is that Bowtie2 can work with longer reads, pared-end reads and produces gapped alignments. In the commands below you will have to make sure that the paths to folders and files are correctly adjusted!

- **Download Bowtie 2**

http://sourceforge.net/projects/bowtie-bio/files/bowtie2/2.2.6/bowtie2-2.2.6-linux-x86_64.zip
and unzip to folder bowtie2-2.2.6.

- **Collect data.** We will need the following files with Paired end reads, which are downloadable from

http://bioputer.mimuw.edu.pl/~bartek/TSG2/lab04/
NA12878.hiseq.wgs_chr20_2mb.30xPE.fastq_1.fastqsanger
NA12878.hiseq.wgs_chr20_2mb.30xPE.fastq_2.fastqsanger

And the human chromosome 20
http://hgdownload.cse.ucsc.edu/goldenpath/hg19/chromosomes/chr20.fa.gz

- **Build an index** of the human chromosome 20 using Bowtie 2. This will take ca. 1 minute to run.

Unzip the fasta file with the chromosome sequence to chr20.fa.
In the command line in the Bowtie2 directory, type (adjusting proper path to file chr20.fa)

```
./bowtie2-build chr20.fa chr20
```

This will generate several files with the FM index of human chromosome 20 in the bowtie2-2.2.6 directory.

❖ **Map the reads.** This will take several minutes.
If you had access to many cpus you can use them with -p option in the general command
`/path-to-bowtie-programs/bowtie2 -p <# cpu> -x <genome index prefix> <fastq file>  > <output filename>`

We will use the command
```
./bowtie2 -x chr20\
-1 NA12878.hiseq.wgs_chr20_2mb.30xPE.fastq_1.fastqsanger \
-2 NA12878.hiseq.wgs_chr20_2mb.30xPE.fastq_2.fastqsanger \
-S chr20MappingBowtie.sam
```

The -1 and -2 options allow providing data from paired-end sequencing.
The -x option sets the prefix of names of the FM index files
The -S option drives the output to a sam-formatted file.

Exercise 2

Visualizing the mapping using IGV.

❖ **Use samtools view to convert the SAM file into a BAM file**. BAM is a binary format corresponding to the SAM text format. You can use the preinstalled samtools program in the computer lab. Alternatively you can do the same in the samtools directory by invoking ./samtools. Run:

```
./samtools view -bS chr20MappingBowtie.sam >
chr20MappingBowtie.bam
```

❖ **Use samtools sort to convert the BAM file to a sorted BAM file**.

```
./samtools sort chr20MappingBowtie.bam \
chr20MappingBowtie.sorted
```

We now have a sorted BAM file called chr20mapping.sorted
Sorted BAM is a useful format because the alignments are (a) compressed, which is convenient for long-term storage, and (b) sorted, which is conveneint for variant discovery.

❖ **Use samptools to create an index of the sorted mapping**
```
./samtools index chr20MappingBowtie.sorted.bam
```

This wil create a file `chr20MappingBowtie.sorted.bai`, which is required by IGV to view the mapping when large dataset is loaded.

❖ **You can download the mapping files**
`chr20MappingBowtie.sorted.bam`
`chr20MappingBowtie.sorted.bam`.bai
from
http://mimuw.edu.pl/~szczurek/TSG2/07_lab/

❖ **Run IGV:**
```
sh igv.sh
```

View mapping: Load `chr20MappingBowtie.sorted.bam`

You need to zoom in enough and make sure you scroll to the very far left of the chromosome.

In maximum zoom you will see color-coding for SNVs and sequencing errors.
In less zoom you will see color-coding for reads. By default IGV uses read color-coding to flag anomalous insert sizes, to detect insertions/deletions.
http://www.broadinstitute.org/igv/interpreting_insert_size
You can switch to visualize paired end orientations, to detect inversion, duplication or translocation events
http://www.broadinstitute.org/software/igv/interpreting_pair_orientations.

Scroll around and zoom in and out in the IGV genome viewer to get a feel for genomic data. Note that coverage is variable, with some regions getting almost no coverage.

- Try *chr20:1,870,686-1,880,895* - if you zoom right in to base resolution you'll see that this region is very GC rich, meaning it's hard to sequence. Unfortunately it also contains the first few exons of a gene...
- Try chr20:963,238-963,897 – do you think the SNP in the middle is a variant to trust?

Exercise 3

Generating variant calls using mpileup in samtools.

- **Run mpileup**

```
./samtools mpileup -g -f ../chr20.fa
chr20MappingBowtie.sorted.bam >chr20SAMvariants.bcf
```

  - -g: directs SAMtools to output genotype likelihoods in the binary call format (BCF). This is a compressed binary format.
  - -f: directs SAMtools to use the specified reference genome. A reference genome must be specified, and here we specify the reference sequence for chromosome 20

The mpileup command automatically scans every position supported by an aligned read, computes all the possible genotypes supported by these reads, and then computes the probability that each of these genotypes is truly present in our sample. It does not call variants.

Bcftools applies the prior and does the actual calling. It can also concatenate BCF files, index BCFs for fast random access and convert BCF to VCF.

- **Download the newest version of bcftools from**
  http://www.htslib.org/download/
  Unpack
  Make

- **Run bcftools to call variants**

```
./bcftools call -c -v -O v chr20SAMvariants.bcf -o
chr20BCFTOOLSvariants.vcf
```

  -v, --variants-only

output variant sites only
-O, --output-type *b|u|z|v*
Output compressed BCF (*b*), uncompressed BCF (*u*), compressed VCF (*z*), uncompressed VCF (*v*). Use the -Ou option when piping between bcftools subcommands to speed up performance by removing unecessary compression/decompression and VCF←→BCF conversion.
-o, --output FILE
-m, --multiallelic-caller
alternative, new model for multiallelic and rare-variant calling designed to overcome known limitations in -c calling model (conflicts with -c)

❖ **Visualize variants with IGV**
Load the generated vcf file to IGV.
You can download this file from
http://mimuw.edu.pl/~szczurek/TSG2/07_lab/

IGV displays VCF variants as coloured bars; homozygotes are solid bars, heterozygotes are two-colour .

❖ Scroll left and right to see the variants
❖ Look at chr20:963,238-963,897. Was the variant with the heterozygous SNP called?

Exercise 4

Processing files and variant calling with GATK
For more introduction (optional) follow workshop videos/pdfs
http://www.broadinstitute.org/partnerships/education/broade/best-practices-variant-calling-gatk-1
and a tutorial
http://hpc.ilri.cgiar.org/mkatari-bioinformatics-august-2013-gatknotes

You will need to adjust for proper paths to files when running these commands!!!

❖ **Download reference genome**
The human chromosome 20
http://hgdownload.cse.ucsc.edu/goldenpath/hg19/chromosomes/chr20.fa.gz

❖ **Download GATK and Picard**
Picard: download the latest version of the software package.
https://github.com/broadinstitute/picard/releases/

```
tar xjf picard-tools-1.140.zip
cd picard-tools-1.140/
java –jar picard.jar –h
```

This should print out some version and usage information.

To use Picard, we call on Java itself as the main program, then specify which jar file to use, knowing that one jar file = one tool. This applies to all Picard tools; to use them you will always build your command lines like this:

```
java -jar picard.jar <ToolName> [options]
```

This means you first make the call to Java itself as the main program, then specify the picard.jar file, then specify which tool you want, and finally you pass whatever other arguments (input files, parameters etc.) are needed for the analysis.

GATK. Download the latest version of the software package.
In order to access the downloads, you need to register for a free account on the GATK support forum. You will also need to read and accept the license agreement before downloading the GATK software package.

You can unzip the downloaded file
```
cd ../GenomeAnalysisTK-3.4-46
```
and test if properly installed
```
java -jar GenomeAnalysisTK.jar –h
```

You will use GATK with the general command
java -jar GenomeAnalysisTK.jar -T <ToolName> [arguments]

There are some strict rules that GATK follows thus resulting in a pipeline that needs several steps.

- **Create a dictionary of the reference (in Picard folder)**

```
java -jar picard.jar CreateSequenceDictionary \
  R= chr20.fa \
  O=chr20.dict
```

- **Create an index file of the reference**
  ./samtools index chr20.fa

You can download `chr20MappingBowtie.bam` from
http://mimuw.edu.pl/~szczurek/TSG2/07_lab/

- **Sort using Pcard**

```
java -jar picard.jar SortSam \
  INPUT= chr20MappingBowtie.bam \
  OUTPUT= chr20MappingBowtie.sortedPicard.bam \
  SORT_ORDER=coordinate
```

- **Create a read group** that describes where the reads are coming from. This is required for GATK

```
java -jar picard.jar AddOrReplaceReadGroups\
 INPUT= chr20MappingBowtie.sortedPicard.bam \
 OUTPUT= chr20MappingBowtie.sortedPicard.RG.bam \
 RGLB= NA12878\
 RGPL=Ilumina \
 RGPU=None \
 RGSM= NA12878
```

- **Dedup**. This will remove any reads that map to the same exact place. It is helpful to get rid of artifacts.

```
java -jar picard.jar MarkDuplicates\
 INPUT= chr20MappingBowtie.sortedPicard.RG.bam \
 OUTPUT= chr20MappingBowtie.sortedPicard.RG.dedup.bam \
 METRICS_FILE= dedup.metrics \
 REMOVE_DUPLICATES=TRUE \
```

```
ASSUME_SORTED=TRUE \
MAX_FILE_HANDLES_FOR_READ_ENDS_MAP=1000
```

- **Index and realign around indels**

```
./samtools index chr20MappingBowtie.sortedPicard.RG.dedup.bam
```

To run the following command, make sure that the chr20.dict file is where the chr20.fa file is, and that the samtools index file is where the bam file is (preferably both in the same folder where `GenomeAnalysisTK` is).

```
#identifying indels (adjust for proper paths to files!)
java -jar GenomeAnalysisTK.jar \
   -T RealignerTargetCreator \
   -R chr20.fa \
   -I chr20MappingBowtie.sortedPicard.RG.dedup.bam \
   -o chr20MappingBowtie.forRealign.intervals

 java -jar GenomeAnalysisTK.jar \
   -T IndelRealigner \
   -R chr20.fa \
   -I chr20MappingBowtie.sortedPicard.RG.dedup.bam \
  -targetIntervals chr20MappingBowtie.forRealign.intervals \
   -o chr20MappingBowtie.realign.bam
```

- **Compare the realigned BAM file to original BAM around an indel**
  - Open the realigned BAM file in IGV
  - It should appear as a new frame below the already opened previous BAM
  - Generally, the new BAM should appear identical to the old BAM except in the realigned regions.
  - Find some regions with indels that have been realigned (use the `chr20MappingBowtie.forRealign.intervals` file from the first step of realignment, it has a list of the realigned regions)
  - Check region chr20:1,163,895-1,163,976.
  - Describe the differences in the alignments visible in this region (between the original and the realigned).

- **`Sort and index`**

```
./samtools sort chr20MappingBowtie.realign.bam
chr20MappingBowtie.realign.sorted

./samtools index chr20MappingBowtie.realign.sorted.bam
```

- Finally! **run GATK.**

```
java -jar GenomeAnalysisTK.jar \
   -T HaplotypeCaller\
   -I chr20MappingBowtie.realign.sorted.bam\
   -R chr20.fa \
   --output_mode EMIT_VARIANTS_ONLY \
   -ploidy 2 \
   -o chr20MappingBowtie.realign.sorted.vcf
```

- You can Download the file containing GATK-realigned variants from http://mimuw.edu.pl/~szczurek/TSG2/07_lab/

- **Visualize with IGV.** You should see the original and the realigned alignments, and the variants called by bcftools and GATK.
- You can also use the option of direct upload of files from url
- What are the differences of GATK to the bcftools calls? Look at chr20:1,127,767-1,127,906.
- How do you think one could improve the overlap between different tools, and decide which variants are true?
- Observe https://docs.google.com/document/pub?id=1CuKkKylVDb03tnN7RSWl5EUzleetn0ctjmvaidPKLxM - h.sdayjdqzl1wg

Homework

Please report answers to questions written in red by email to Szczurek@mimuw.edu.pl.

- **Filter the variants**

```
java -jar GenomeAnalysisTK.jar \
 -R chr20.fa \
 -T VariantFiltration \
 -o chr20MappingBowtie.realign.sorted.f.vcf \
 --variant chr20MappingBowtie.realign.sorted.vcf \
 --filterExpression "QD < 2.0 || MQ < 40.0 || FS > 60.0" \
 --filterName "mannyfilter"
```

- For own pleasure, you can read more about filtering with GATK at https://www.broadinstitute.org/gatk/guide/tagged?tag=filtering

- Load the filtered variants to IGV. Now you should be able to view: the original Bowtie2 alignment, the GATK-reaigned alignment, the bcftools-called variants, and the GATK-called and filtered variants.

  - Describe the differences in variants which are called between Mpileup and GATK in the region chr20:1,163,895-1,163,976.

  - See which variants are filtered out – they are grayed out on the IGV display.

  - Go to region chr20:1,890,836-1,901,408. Given what you can infer about the coverage and the alignment quality in that region from IGV, describe why are many reads in this region filtered out.

II. Structural variation

Exercise 1

We will now look at structural variants in one of the chromosomes identified by the 1000 Genomes consortium using Circos.

- **Login to server adela, login tsg2**
- **Create your own directory, e.g.**
  - `mkdir ewaszczurek`
- **Extract deletions** from one of the ALL..vcf into a separate file (e.g. using grep and cut commands)

- **Extract insertions** from one of the ALL..vcf into a separate file

- Generate files containing counts of the variants in bins of length 500 across the chromosome in Circos format:

  position1 \t position2 \t count

  You will need to write a python /R script that will parse the separated deletion/insertion files and calculate the counts.

- Using Circos:
  - Visualize the counted deletions as a heatmap
  - Visualize the counted insertions as a histogram

To run circos, you need to go to the circus directory and use bin/circus executable
`bin/circos –conf conf.file`
where the conf file contains encoded instructions regarding your image.

Useful tutorials: http://circos.ca/tutorials/lessons/