From fastq to vcf

Note: Some commands continue on the next line. These commands have backslashes \ in the end of the line to indicate that the newline character (not shown) should be ignored. You should be able to copy (Ctrl+c) the full length command and then paste it in the console window. An alternative is to copy one line at time, leaving the backslashes out. In a script file, newline characters have to escaped with backslashes.

Working with CSC servers. Those having a VM can skip this and move to "0.1".

Some exercises may work on CSC computers.

```
ssh -X username@taito-shell.csc.fi
```

Find software packages with command `module spider name`:

```
module spider bwa
```

Load and unload software with options `load` and `unload`. However, some modules have other dependencies:

```
module load bwa/0.7.12
```

Suggests 'Try: "module spider bwa/0.7.12" to see how to load the module(s)'. When we do that:

```
module spider bwa/0.7.12
```

we see that package "bwa/0.7.12" requires package "gcc/4.7.2". We therefore load that package first and then bwa:

```
module load gcc/4.7.2
module load bwa/0.7.12

$ bwa

Program: bwa (alignment via Burrows-Wheeler transformation)
Version: 0.7.12-r1039
Contact: Heng Li <lh3@sanger.ac.uk>

Usage:   bwa <command> [options]

Command: index         index sequences in the FASTA format
        mem           BWA-MEM algorithm
....
```

At least one program is missing from CSC servers, GATK. It is legal to use it there if you install it by yourself.

```
mkdir ~/gatk
cd ~/gatk
wget http://wasabiapp.org/vbox/scripts/GenomeAnalysisTK-3.5.tar.bz2
tar xjf GenomeAnalysisTK-3.5.tar.bz2

pwd
#to me this outputs: /homeappl/home/username/gatk
```

This is the **path** below. Create file "gatk" and write into that file:

```
#!/bin/sh
/usr/bin/java -Xmx2g -Djava.io.tmpdir=/tmp -jar path/GenomeAnalysisTK.jar $@
```

Make it executable and test that it works:

```
chmod u+x gatk
./gatk -h
```

0.1 Get your data

Get the common data:

```
mkdir ~/session_3
cd ~/session_3
wget http://wasabiapp.org/vbox/data/session_3/file1.tgz
tar xzf file1.tgz
```

This file is 60MB in size and used for the exercises during the class (called "SUBSET OF DATA" below).

If you have not downloaded "file2.tgz" before the lecture, it's better to leave the analysis of full data as home work. Exercises can be done with the smaller data set.

You should also download the full data set (317MB) and perform the analysis of your own sample with that. For that do this:

```
wget http://wasabiapp.org/vbox/data/session_3/file2.tgz
tar xzf file2.tgz
```

Choose then a sample of your own (a number between 2 and 25) from here: http://doodle.com/poll/88kgv89wwigh2bhz

Remember your choice and work with that sample from now. In the following commands, change the sample number to your own one:

- if you number is "27", you need to change "sample-1" to "sample-27" but keep the rest of the command intact: `sample-1_1.fq.gz` becomes `sample-27_1.fq.gz`

```
mkdir ~/session_3/data
cd ~/session_3/data
```

Edit the following command to use your own sample (replace "27" with your sample number):

```
num=27
wget http://wasabiapp.org/vbox/data/session_3/sample/$num/sample-$num""_1.fq.gz
wget http://wasabiapp.org/vbox/data/session_3/sample/$num/sample-$num""_2.fq.gz
```

If you struggle with the commands above, open the URL (http://wasabiapp.org/vbox/data/session_3/sample/) in a browser and download the right data set that way.

Look at the input data

```
cd ~/session_3

ls -lh data/sample-1_?.fq.gz
ls -lh reference/ninespine.fa

less -S data/sample-1_1.fq.gz
less -S reference/ninespine.fa
```

0.2. Get familiar with the programs

```
bwa
bwa mem

samtools
samtools view

bcftools
bcftools index

bgzip

gatk -h
which gatk
less /usr/local/bin/gatk

fastqc -h
```

---

1. Preparations for the analyses

We do fasta dictionary, fasta index, mapping index

```
cd ~/session_3

samtools dict reference/ninespine.fa > reference/ninespine.dict

samtools faidx reference/ninespine.fa

bwa index reference/ninespine.fa
```

Bwa creates multiple files:

- .amb is text file, to record appearance of N (or other non-ATGC) in the ref fasta.
- .ann is text file, to record ref sequences, name, length, etc.
- .bwt is binary, the Burrows-Wheeler transformed sequence.
- .pac is binary, packaged sequence (four base pairs encode one byte).
- .sa is binary, suffix array index.

Compare the sizes of the original reference and the index files:

```
ls -sh reference/ninespine.fa*
```

Look into dictionary and fasta index files:

```
less -S reference/ninespine.dict
less -S reference/ninespine.fa.fai
```

Check the quality of fastq data

```
mkdir ~/session_3/qc

fastqc data/sample-1_1.fq.gz -o qc
firefox qc/sample-1_1_fastqc.html

fastqc data/sample-1_2.fq.gz -o qc
firefox qc/sample-1_2_fastqc.html
```

---

2. Mapping

"bwa mem" is very robust, we use it with default options. See `bwa mem` for the available options.

bwa:

- readgroup (sample data) has to be provided
- split reads option probably unnecessary

output:

- default output sam, we need bam for later analyses
- we can drop unmapped reads. (What does this mean? See `samtools view` and option "-F". Google "sam format".)

Option A: (bad, takes lots of disk space) mapping and format conversion separately

```
bwa mem -M -R "@RG\tID:sample-1\tSM:sample-1\tPL:illumina\tLB:sample-1\tPU:1" \
    reference/ninespine.fa \
    data/sample-1_1.fq.gz data/sample-1_2.fq.gz > data/sample-1_bwa.sam

samtools view -F4 -h -Obam -o data/sample-1_bwa.bam data/sample-1_bwa.sam
```

Option B: (better) format conversion integrated with mapping

```
bwa mem -M -R "@RG\tID:sample-1\tSM:sample-1\tPL:illumina\tLB:sample-1\tPU:1" \
    reference/ninespine.fa data/sample-1_1.fq.gz data/sample-1_2.fq.gz \
| samtools view -F4 -h -Obam -o data/sample-1_bwa.bam
```

Look into the bam file:

```
samtools view data/sample-1_bwa.bam | less -S
```

Compare the fastq and bam files. What do you notice?

```
less -S data/sample-1_1.fq.gz | head -1
samtools view data/sample-1_bwa.bam | head -1
```

Something else:

To understand the sam format, we use another data set:

- mapping of short data is often non-unique
- data used to be shorter, less of a problem nowadays

Get another data set. These are a couple of years old rad-tag sequences:

```
mkdir ~/session_3/radtag
cd ~/session_3/radtag
wget http://wasabiapp.org/vbox/data/session_3/radtag/contigs.fa
cd ~/session_3
```

Compare the output of following commands:

```
bwa mem reference/ninespine.fa radtag/contigs.fa | grep -v "@SQ" | less -S
bwa mem reference/ninespine.fa radtag/contigs.fa -a | grep -v "@SQ" | less -S
```

Do you spot differences?

---

3. Sort reads by coordinate, remove duplicates and index

For subsequent analyses the reads have to be sorted by position. We want to remove the reads that map to the exactly same positions. Why we want to do this?

```
samtools sort -T /tmp/sample-1 -O bam -o data/sample-1_bwa_sorted.bam \
data/sample-1_bwa.bam

samtools rmdup data/sample-1_bwa_sorted.bam data/sample-1_bwa_rmdup.bam

samtools index data/sample-1_bwa_rmdup.bam data/sample-1_bwa_rmdup.bai
```

Compare the fastq and bam files. What do you notice?

```
less -S data/sample-1_1.fq.gz | head -1
samtools view data/sample-1_bwa.bam | head -1
samtools view data/sample-1_bwa_rmdup.bam | head -1
```

Look into the bam file using the bam browser:

```
samtools tview data/sample-1_bwa_rmdup.bam reference/ninespine.fa \
-p ctg7180000005484:13100
```

Press ? to see commands. Show bases, colour them by nucleotide, scroll to 20,000.

---

4. Realign the reads around indels

One problem of bwa is that it aligns the reads separately and do not consider the alignment of other reads. This can cause errors around indels. We use GATK to realign those regions.

See https://www.broadinstitute.org/gatk/guide/tooldocs/index for information.

Create first a list of regions to realign, then do it

```
gatk \
    -T RealignerTargetCreator \
    -R reference/ninespine.fa \
    -I data/sample-1_bwa_rmdup.bam \
    -o data/sample-1_intervals.list
```

```
gatk \
    -T IndelRealigner \
    -R reference/ninespine.fa \
    -I data/sample-1_bwa_rmdup.bam \
    -targetIntervals data/sample-1_intervals.list \
    -o data/sample-1_realn.bam
```

Look into the new bam file:

```
samtools tview data/sample-1_realn.bam reference/ninespine.fa \
-p ctg7180000005484:20000
```

---

5. Call variants: genomic vcf

We use GATK. That is written in java and has some overhead in the start. Unfortunately it is not designed for unix-style piping but it can be forced to do some using /dev/stdout.

Depending on the computer, calling of full data can take 15-20 min or longer. We start with a smaller set; you should do full data after that or at home.

SUBSET OF DATA:

Get a smaller set, just the first three contigs. Make a bed file (google: "bed ucsc" to see what *bed* means):

```
head -3 reference/ninespine.fa.fai | awk '{OFS="\t";print $1,1,$2}' \
> data/first3.bed
```

Select a subset of bam. (Check "`samtools view`" and meaning of option "-L".)

```
samtools view -h -L data/first3.bed data/sample-1_realn.bam > data/sample-1_f3.bam

samtools sort -T /tmp/sample-1 -O bam -o data/sample-1_f3_sorted.bam \
data/sample-1_f3.bam

samtools index data/sample-1_f3_sorted.bam
```

Option A: (bad, takes lots of disk space) calling and compression separately

```
gatk \
    -T HaplotypeCaller \
    -R reference/ninespine.fa \
    -I data/sample-1_realn.bam \
    -gt_mode DISCOVERY \
    -ERC GVCF \
    -stand_emit_conf 3 \
```

```
    -stand_call_conf 10 \
    -GQB 10 -GQB 20 -GQB 30 -GQB 40 -GQB 50 \
    --variant_index_type LINEAR \
    --variant_index_parameter 128000 \
    -o data/sample-1_f3.calls.gvcf

bgzip data/sample-1_f3.calls.gvcf
```

Option B: (better) compression integrated with calling

```
gatk \
    -T HaplotypeCaller \
    -R reference/ninespine.fa \
    -I data/sample-1_f3_sorted.bam \
    -gt_mode DISCOVERY \
    -ERC GVCF \
    -stand_emit_conf 3 \
    -stand_call_conf 10 \
    -GQB 10 -GQB 20 -GQB 30 -GQB 40 -GQB 50 \
    --variant_index_type LINEAR \
    --variant_index_parameter 128000 \
    -o /dev/stdout \
| bgzip -s - > data/sample-1_f3.calls.gvcf.gz
```

Look into the bam file:

```
bcftools view -H data/sample-1_f3.calls.gvcf.gz | less -S
```

FULL DATA:

Option A: (bad, takes lots of disk space) calling and compression separately

```
gatk \
    -T HaplotypeCaller \
    -R reference/ninespine.fa \
    -I data/sample-1_realn.bam \
    -gt_mode DISCOVERY \
    -ERC GVCF \
    -stand_emit_conf 3 \
    -stand_call_conf 10 \
    -GQB 10 -GQB 20 -GQB 30 -GQB 40 -GQB 50 \
    --variant_index_type LINEAR \
    --variant_index_parameter 128000 \
    -o data/sample-1.calls.gvcf

bgzip data/sample-1.calls.gvcf
```

Option B: (better) compression integrated with calling

```
gatk \
```

```
    -T HaplotypeCaller \
    -R reference/ninespine.fa \
    -I data/sample-1_realn.bam \
    -gt_mode DISCOVERY \
    -ERC GVCF \
    -stand_emit_conf 3 \
    -stand_call_conf 10 \
    -GQB 10 -GQB 20 -GQB 30 -GQB 40 -GQB 50 \
    --variant_index_type LINEAR \
    --variant_index_parameter 128000 \
    -o /dev/stdout \
| bgzip -s - > data/sample-1.calls.gvcf.gz
```

Look into the bam file:

```
bcftools view -H data/sample-1.calls.gvcf.gz | less -S
```

---

6. Joint-call variants: vcf

We use 20 other gvcf files, two from each population.

SUBSET OF DATA:

Index the bam files (TBI-format)

```
bcftools index -t data/sample-1_f3.calls.gvcf.gz

for f in `ls gvcf_f3/*gvcf.gz`; do
   bcftools index -t $f
done
```

Make the following command in a text editor (e.g. "geany"). Either type the missing 18 file names manually (stupid!) or write a script that writes them for you.

```
gatk \
    -T GenotypeGVCFs \
    -R reference/ninespine.fa \
    -V data/sample-1_f3.calls.gvcf.gz \
    -V gvcf_f3/popA-17.calls.gvcf.gz \

(fill this bit with the file names)

    -V gvcf_f3/popJ-18.calls.gvcf.gz \
    -o /dev/stdout \
```

```
| bcftools view -Oz -o f3.calls.vcf.gz &
```

Hint. We can modify the loop above to write the file names:

```
for f in `ls gvcf_f3/*gvcf.gz`; do
   echo " -V $f \\"
done
```

Notice that backslash has a special meaning and has to be written twice.

Either copy the command to a console to run or execute it using `source <filename>`.

Look into the bam file:

```
bcftools view -H f3.calls.vcf.gz | less -S
```

FULL DATA:

Index the bam files (TBI-format)

```
bcftools index -t data/sample-1.calls.gvcf.gz

for f in `ls gvcf/*gvcf.gz`; do
   bcftools index -t $f
done
```

Make the following command in a text editor (e.g. "geany"). Either type the missing 18 file names manually (stupid!) or write a script that writes them for you.

```
gatk \
    -T GenotypeGVCFs \
    -R reference/ninespine.fa \
    -V data/sample-1.calls.gvcf.gz \
    -V gvcf/popA-17.calls.gvcf.gz \

(fill this bit with the file names)

    -V gvcf/popJ-18.calls.gvcf.gz \
    -o /dev/stdout \
| bcftools view -Oz -o all.calls.vcf.gz &
```

Look into the bam file:

```
bcftools view -H all.calls.vcf.gz | less -S
```

---

7. Vcf filtering

We continue from this next time. What do the following commands mean?

```
bcftools view -H -m3 f3.calls.vcf.gz | less -S

bcftools view -H -m2 -M2 f3.calls.vcf.gz | less -S

bcftools view -H -m2 -M2 -V indels,mnps f3.calls.vcf.gz| less -S

bcftools filter -g 10 f3.calls.vcf.gz | bcftools view -H -m2 -M2 -V indels,mnps \
| less -S


bcftools view -m2 -M2 -V indels,mnps f3.calls.vcf.gz \
| bcftools query -f '%CHROM\t%POS\t%REF\t%ALT[\t%SAMPLE=%GT]\n' | less -S
```