

CLINICAL NGS DATA ANALYSIS WORKSHOP

Genomic Medicine 2019 Nordic

Team:

Martin Larsen
Mark Burton
Lars Andersen
Klaus Brusgaard
Qin Hao
Ieva Miceikaite

November 12, 2019

Contents

1	Introduction	1
2	Installation and Setup of Programs	2
2.1	Access to SDU Cloud and Installation of NGS Analysis Tools	2
2.2	Install and Setup Varseq	3
3	Data Inspection and Initial Quality Control with FastQC	3
4	Alignment with BWA	5
5	Mark Duplicate Reads	6
6	Base Recalibration	8
7	Quality Control and Statistics	10
8	Variant Calling with HaplotypeCaller	11

List of Scripts

1	Setup conda and install NGS analysis tools	3
2	Data inspection and FastQC	4
3	Map reads to reference and sort the resulting bam file	6
4	Mark duplicate reads	7
5	Base recalibration	10
6	Quality control and statistics of bam files	11
7	Calling variants with HaplotypeCaller	12

1 Introduction

This manuscript provides an introduction to the GATK Best Practices for performing variant discovery analysis in next-generation sequencing (NGS) data. The GATK Best Practices and tools are the most recommended to use for variant discovery analysis in humans. This manuscript covers a short summary of the latest GATK Best Practices (last updated July 14, 2019) found here: <https://software.broadinstitute.org/gatk/best-practices>.

An overview of the GATK pipeline is provided in Figure 1. This manuscript covers the steps from fastq file (raw reads from the sequencing) to sorted bam files containing the alignment information, then to vcf files containing the discovered variants (i.e. the steps framed in the red box). The subsequent variant annotation, filtering and prediction will be covered in later parts of the workshop.

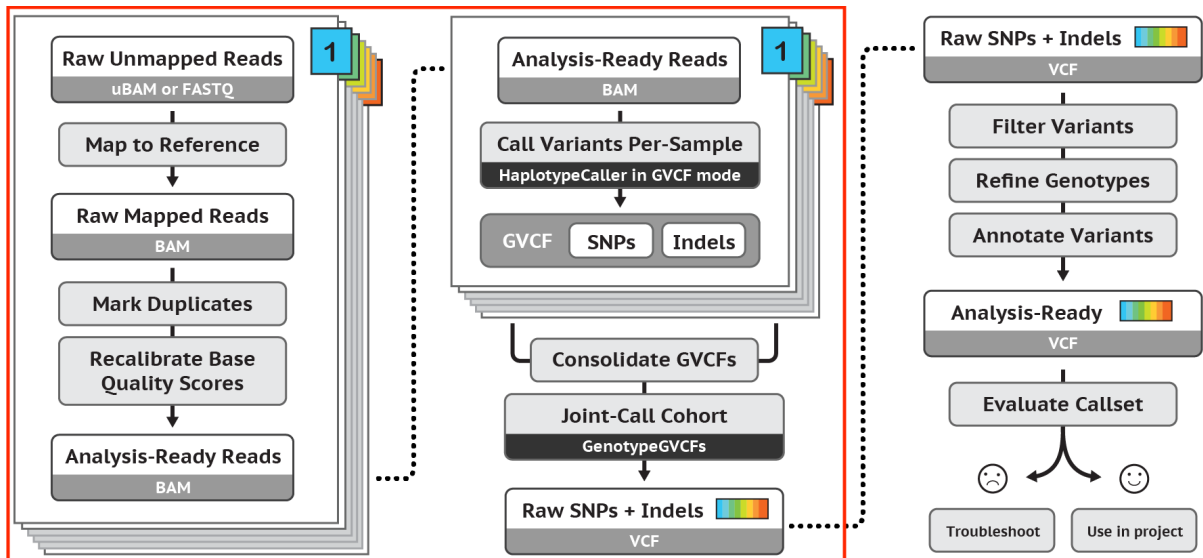


Figure 1: GATK pipeline for variant discovery analysis. The steps framed by the red box will be covered in this manuscript.

The tools used for the variant discovery analysis are **bwa** for alignment of the sequencing reads to a reference genome and **GATK**, which consists of a collection of tools, used for sorting and marking of duplicates of the bam files as well as the variant discovery resulting in the vcf files. Furthermore, **FastQC**, **Qualimap** and **MultiQC** are used for quality control and visual representation hereof. A presentation of the data formats (fastq, bam and vcf files) and the tools will be given during the workshop. The presentation can be found in the powerpoint slides called **ngs_data_analysis_workshop_2018.pptx**.

In this workshop we will perform the NGS data analysis on the SDU Cloud system located at the escience center, University of Southern Denmark. While the computing cluster is very powerful, the NGS data analysis could also be performed on a laptop or desktop computer if we only want to analyze a limited number of samples with low sequencing depth. However, it is recommended to use a server or HPC (high-performing cluster) when larger datasets should be analyzed.

In this workshop we will run the NGS data analysis pipeline on trio exome NGS data from a father, a mother and their son. The exomes has been sequenced on a Novaseq 6000. Now, we want to uncover the variant(s) causing the pathogenic phenotype for the son. Note, due to time limitations we only run the data analysis pipeline on a single chromosome (chromosome 21). However, we will provide the resulting files for the full exome analysis for the variant interpretation.

2 Installation and Setup of Programs

Before we can get started on our NGS data analysis, we need to get access to SDU Cloud where we will perform the first part of the analysis. Afterwards, we will install and setup Varseq that we will use for variant annotation and filtering.

2.1 Access to SDU Cloud and Installation of NGS Analysis Tools

Step 1: Login to SDU Cloud

To access SDU Cloud go to the website: <https://cloud.sdu.dk>. Click on ‘More login options’ and sign using the username and password provided to you.


Step 2: Check workshop data is available

Go to shares, and click ‘Accept’ to the workshop data under ‘Shared with Me’ (if this is not available to you let us know). Now click on Files, and check under the Shares folder that the workshop folder is there.

Step 3: Open terminal window on SDU Cloud

Now we will run a computing job on SDU Cloud. In this case, we want to get access to a terminal window in which we can run the commands for NGS data analysis. To start a job, click on Apps and locate JupyterLab. Then click on JupyterLab (v. 1.2.2) and then click the ‘Run Application’ button. Under ‘Job name’ type a name for the job e.g. workshop, then enter how long you want the job to run e.g. 10 hours. Under ‘Select additional folders to use’ click ‘Add folder’ and type on the box saying ‘No file selected’. Then locate the workshop folder under the shares folder and press on ‘Select’ next to it. Then click submit. When the green button with the text ‘Go to web interface’ appear, click on it. Now you should see the JupyterLab interface. Click the Terminal icon at the bottom of the page under the section ‘Other’. Now you should be presented with a black screen (a terminal window) in which we can type commands.

Step 4: Install NGS analysis tools

Now, that we have a terminal window we can install the tools needed for NGS data analysis. We do this by copy-and-pasting the commands from Script 1 (next page) into the terminal window. We suggest to copy the commands line-by-line and pressing  (the Return key) after pasting each line to run the command. Try to see what happens in the terminal when you run each command.

The lines (or text) starting with the # symbol are comments providing a brief description of what the commands do i.e. they do not execute anything. You can also put the # symbol in front of any command you do not wish to run.

[illegible]

```
### NGS Workshop - Data inspection and FastQC ###

# The location of the workshop data
project=/work/workshop

# Path to data, scripts, reference genome and resource directories
DATA=${project}/data
REF=${project}/reference
RES=${project}/resources
SCRIPTS=${project}/scripts

# Let's go to our working directory,
# we do this with the cd command (cd = change directory)
work=/work/analysis_results
cd ${work}

# Let's look inside the data, scripts, reference genome
# and resource directories
ls $DATA
ls $REF
ls $RES
ls $SCRIPTS

# Take a quick look in one of our data files
# (should be similar to the one shown in the presentation)
zcat $DATA/HG002_son_R1_001.fastq.gz | head

# We will create a directory to store all our quality control reports
mkdir -p quality_control

# Let's check the quality of our fastq files
# in the data directory using fastqc

# son
fastqc $DATA/HG002_son_R1_001.fastq.gz -o quality_control
fastqc $DATA/HG002_son_R2_001.fastq.gz -o quality_control

# father
fastqc $DATA/HG003_father_R1_001.fastq.gz -o quality_control
fastqc $DATA/HG003_father_R2_001.fastq.gz -o quality_control

# mother
fastqc $DATA/HG004_mother_R1_001.fastq.gz -o quality_control
fastqc $DATA/HG004_mother_R2_001.fastq.gz -o quality_control

# Now check that we have created HTML reports for the fastq files
# using the ls command
ls quality_control
```

4 Alignment with BWA

The first actual processing step of our NGS pipeline is to map each read-pair to the reference genome. We do this by executing the tool **bwa** (the **mem** version of the tool). Afterwards, we pipe (the **|** symbol) i.e. send the aligned data from **bwa mem** to GATK's SortSam tool. This tool converts the data to bam and sorts the aligned reads according to genomic coordinate. We do this for each sample in our analysis i.e. we run the same commands for the father, mother and son, respectively. Now, run the commands in Script 3

Note: When a line ends with **** the command continues on the next line. When you copy-and-paste commands line-by-line you should copy all lines together that ends with a **** until the first line appears without it.

```
### NGS Workshop - Alignment with BWA ###

# The location of the workshop data
project=/work/workshop

# Let's go to our working directory with the cd command
work=/work/analysis_results
cd ${work}

### Alignment and Sorting ###

# Path to reference genome and data
REF=${project}/reference_genome/human_g1k_v37_decoy.fasta
DATA=${project}/data

# Step 1: Align reads to reference genome with BWA

# Step 1.1: First we need to define Read groups information
RG_son="@RG\tID:son\tLB:Illumina\tPL:HiSeq\tSM:son\tPU:C2FAGACXX"
RG_father="@RG\tID:father\tLB:Illumina\tPL:HiSeq\tSM:father\tPU:C2FAGACXX"
RG_mother="@RG\tID:mother\tLB:Illumina\tPL:HiSeq\tSM:mother\tPU:C2FAGACXX"

# Step 1.2: Use bwa to align reads to reference genome

# son
bwa mem -t 4 -M -R ${RG_son} $REF $DATA/HG002_son_R1_001.fastq.gz \
$DATA/HG002_son_R2_001.fastq.gz > HG002_son.sam

# father
bwa mem -t 4 -M -R ${RG_father} $REF $DATA/HG003_father_R1_001.fastq.gz \
$DATA/HG003_father_R2_001.fastq.gz > HG003_father.sam

# mother
bwa mem -t 4 -M -R ${RG_mother} $REF $DATA/HG004_mother_R1_001.fastq.gz \
$DATA/HG004_mother_R2_001.fastq.gz > HG004_mother.sam

# Let's take a look in our working directory to see
# we have created new sam files with the aligned reads
ls

# Let's check the what's inside one of these sam files
samtools view HG002_son.sam | head -n 20
```

```

# Step 2: Use SortSam to sort the aligned reads by coordinate
# and convert to bam file to save space

# son
gatk --java-options -Xmx2G SortSam \
--INPUT=HG002_son.sam \
--OUTPUT=HG002_son_sorted.bam \
--VALIDATION_STRINGENCY=LENIENT \
--SORT_ORDER=coordinate \
--CREATE_INDEX=TRUE

# father
gatk --java-options -Xmx2G SortSam \
--INPUT=HG003_father.sam \
--OUTPUT=HG003_father_sorted.bam \
--VALIDATION_STRINGENCY=LENIENT \
--SORT_ORDER=coordinate \
--CREATE_INDEX=TRUE

# mother
gatk --java-options -Xmx2G SortSam \
--INPUT=HG004_mother.sam \
--OUTPUT=HG004_mother_sorted.bam \
--VALIDATION_STRINGENCY=LENIENT \
--SORT_ORDER=coordinate \
--CREATE_INDEX=TRUE

# We should now have created three bam files with extension _sorted.bam
# Again we check this with the ls command
ls

# Let's also check that the contents of the bam files are now sorted
samtools view HG002_son_sorted.bam | head -n 20

```

Script 3: Map reads to reference and sort the resulting bam file

5 Mark Duplicate Reads

This second processing step is again performed on a per-sample basis and consists of identifying read pairs that are likely to have originated from duplicates of the same original DNA fragments through some artifactual processes e.g. PCR. Since these duplicates provides no additional information but require a lot of processing time we exclude these from the analysis. Script 4 runs GATK's Markduplicates program which tags all but one of the read pairs with each set of duplicates such that they can be ignored during the base recalibration and variant discovery process.

```

### NGS Workshop - Mark duplicate reads ###

# The location of the workshop data
project=/work/workshop

# Let's go to our working directory with the cd command
work=/work/analysis_results
cd ${work}

```



```

#### MarkDuplicates ####

# We use the picard tool (now part of gatk4) called MarkDuplicate
# to mark duplicate reads introduced during e.g. PCR

# son
gatk --java-options -Xmx2G MarkDuplicates \
--INPUT=HG002_son_sorted.bam \
--OUTPUT=HG002_son_sorted_nodup.bam \
--METRICS_FILE=quality_control/HG002_son_duplicate_metrics.txt \
--CREATE_INDEX=TRUE \
--REMOVE_DUPLICATES=FALSE \
--OPTICAL_DUPLICATE_PIXEL_DISTANCE=2500 # Should be set to 2500 for
# patterned flowcells like Illumina Hiseq 3000/4000 and Novaseq 6000
# and set to 100 for unpatterned flowcells like HiSeq 2500

# father
gatk --java-options -Xmx2G MarkDuplicates \
--INPUT=HG003_father_sorted.bam \
--OUTPUT=HG003_father_sorted_nodup.bam \
--METRICS_FILE=quality_control/HG003_father_duplicate_metrics.txt \
--CREATE_INDEX=TRUE \
--REMOVE_DUPLICATES=FALSE \
--OPTICAL_DUPLICATE_PIXEL_DISTANCE=2500

# mother
gatk --java-options -Xmx2G MarkDuplicates \
--INPUT=HG004_mother_sorted.bam \
--OUTPUT=HG004_mother_sorted_nodup.bam \
--METRICS_FILE=quality_control/HG004_mother_duplicate_metrics.txt \
--CREATE_INDEX=TRUE \
--REMOVE_DUPLICATES=FALSE \
--OPTICAL_DUPLICATE_PIXEL_DISTANCE=2500

# Let's see that we have created three new bam files with extension
_sorted_nodup.bam
ls

# We can see duplicates with samtools by typing (head shows first 10 cases)
samtools view -F 0x400 HG002_son_sorted_nodup.bam | head

# To see everything but duplicates type (head shows first 10 cases)
samtools view -f 0x400 HG002_son_sorted_nodup.bam | head

# Remove old sam files as they are no longer needed and takes up space
# rm -f HG002_son.sam
# rm -f HG003_father.sam
# rm -f HG004_mother.sam

# Remove old bam files as they are no longer needed and takes up space
# rm -f HG002_son_sorted.ba{m,i}
# rm -f HG003_father_sorted.ba{m,i}
# rm -f HG004_mother_sorted.ba{m,i}

```

Script 4: Mark duplicate reads

6 Base Recalibration

The aim of the third processing is to help distinguish true variants from false positives. This step consists of applying machine learning to detect and correct for patterns of systematic errors in the base quality scores, which are confidence scores emitted by the sequencer for each base. Base quality scores play an important role in weighing the evidence for or against possible variant alleles during the variant discovery process, so it's important to correct any systematic bias observed in the data. Biases can originate from biochemical processes during library preparation and sequencing, from manufacturing defects in the chips, or instrumentation defects in the sequencer.

Note that the Base recalibration requires known-sites to run. We have already placed three databases of known-sites i.e. dbsnp, mills_1000G and 1000G_indels on SDU Cloud in the folder workshop→resources. These known-sites are provided by GATK and can be found in their resource bundle here: <https://software.broadinstitute.org/gatk/download/bundle>

To run the base recalibration algorithm use the commands found in Script 5.

```
#### NGS Workshop - Base Recalibration ####

# The location of the workshop data
project=/work/workshop

# Let's go to our working directory with the cd command
work=/work/analysis_results
cd ${work}

# Path to reference genome and resource files
REF=${project}/reference_genome/human_g1k_v37_decoy.fasta
BED=${project}/resources/workshop_chr21_regions_g1k.bed
dbsnp=${project}/resources/dbsnp_138.b37.vcf
mills_1000g=${project}/resources/Mills_and_1000G_gold_standard.indels.b37.vcf
phase1_1000g=${project}/resources/1000G_phase1.indels.b37.vcf

#### Base recalibration ####

# Base recalibration is a two step process: First find bases to
recalibrate, second apply the recalibration (it's actually three steps but
last step is optional and only produces a statistics of our recalibration)

# Step 1: Generate recalibration table

# son
gatk --java-options -Xmx2G BaseRecalibrator \
-R=$REF \
-I=HG002_son_sorted_nodup.bam \
--known-sites=${dbsnp} \
--known-sites=${mills_1000g} \
--known-sites=${phase1_1000g} \
-L=$BED \
-O=quality_control/HG002_son_pre_recalibration.grp

# father
gatk --java-options -Xmx2G BaseRecalibrator \
-R=$REF \
-I=HG003_father_sorted_nodup.bam \
--known-sites=${dbsnp} \
--known-sites=${mills_1000g} \
```

```

--known-sites=${phase1_1000g} \
-L=$BED \
-O=quality_control/HG003_father_pre_recalibration.grp

# mother
gatk --java-options -Xmx2G BaseRecalibrator \
-R=$REF \
-I=HG004_mother_sorted_nodup.bam \
--known-sites=${dbSNP} \
--known-sites=${mills_1000g} \
--known-sites=${phase1_1000g} \
-L=$BED \
-O=quality_control/HG004_mother_pre_recalibration.grp

# Step 2: Apply recalibration

# son
gatk --java-options -Xmx2G ApplyBQSR \
-R=$REF \
-I=HG002_son_sorted_nodup.bam \
--bqsr-recal-file=quality_control/HG002_son_pre_recalibration.grp \
-L=$BED \
-O=HG002_son_recalibrated.bam

# father
gatk --java-options -Xmx2G ApplyBQSR \
-R=$REF \
-I=HG003_father_sorted_nodup.bam \
--bqsr-recal-file=quality_control/HG003_father_pre_recalibration.grp \
-L=$BED \
-O=HG003_father_recalibrated.bam

# mother
gatk --java-options -Xmx2G ApplyBQSR \
-R=$REF \
-I=HG004_mother_sorted_nodup.bam \
--bqsr-recal-file=quality_control/HG004_mother_pre_recalibration.grp \
-L=$BED \
-O=HG004_mother_recalibrated.bam

# Step 3: Generate recalibration table of newly recalibrated bases
# (optional step)

# son
gatk --java-options -Xmx2G BaseRecalibrator \
-R=$REF \
-I=HG002_son_recalibrated.bam \
--known-sites=${dbSNP} \
--known-sites=${mills_1000g} \
--known-sites=${phase1_1000g} \
-L=$BED \
-O=quality_control/HG002_son_post_recalibration.grp

# father
gatk --java-options -Xmx2G BaseRecalibrator \
-R=$REF \
-I=HG003_father_recalibrated.bam \

```

```

--known-sites=${dbsnp} \
--known-sites=${mills_1000g} \
--known-sites=${phase1_1000g} \
-L=$BED \
-O=quality_control/HG003_father_post_recalibration.grp

# mother
gatk --java-options -Xmx2G BaseRecalibrator \
-R=$REF \
-I=HG004_mother_recalibrated.bam \
--known-sites=${dbsnp} \
--known-sites=${mills_1000g} \
--known-sites=${phase1_1000g} \
-L=$BED \
-O=quality_control/HG004_mother_post_recalibration.grp

# We will check the results of our recalibration together with
# the alignment statistics we create in the next step.

# Again, we remove old bam files so they don't take up space
# rm -f HG002_son_sorted_nodup.ba{m,i}
# rm -f HG003_father_sorted_nodup.ba{m,i}
# rm -f HG004_mother_sorted_nodup.ba{m,i}

```

Script 5: Base recalibration

7 Quality Control and Statistics

In this step we perform quality control on the aligned and recalibrated bam files. We use two tools for this. The first is GATK's CollectHsMetrics that collects various metrics. The second tool is qualimap that also collects various metrics for alignment, coverage, GC content and insertion size. The last command runs the tool MultiQC that combines all quality control metrics from all samples to a single HTML report, we can later investigate.

Now, collect the quality control metrics and generate the HTML report by running the commands in Script 6.

After successfully running the commands we can go to the quality_control directory and find the multiqc HTML report to investigate the quality of our alignment, deduplication and recalibration.

```

### NGS Workshop - Quality control and statistics ###

# The location of the workshop data
project=/work/workshop

# Let's go to our working directory with the cd command
work=/work/analysis_results
cd ${work}

# Path to reference genome and resources
REF=${project}/reference_genome/human_g1k_v37_decoy.fasta
BED=${project}/resources/workshop_chr21_regions_g1k.bed
INTERVALS=${project}/resources/workshop_chr21_regions_g1k.interval_list

### Quality Control and Bam File Metrics ###

```

```

# Collect Hs Metrics with gatk

# son
gatk --java-options -Xmx2G CollectHsMetrics \
--INPUT=HG002_son_recalibrated.bam \
--REFERENCE_SEQUENCE=$REF \
--OUTPUT=quality_control/HG002_son_HsMetrics.txt \
--BAIT_INTERVALS=$INTERVALS \
--TARGET_INTERVALS=$INTERVALS

# father
gatk --java-options -Xmx2G CollectHsMetrics \
--INPUT=HG003_father_recalibrated.bam \
--REFERENCE_SEQUENCE=$REF \
--OUTPUT=quality_control/HG003_father_HsMetrics.txt \
--BAIT_INTERVALS=$INTERVALS \
--TARGET_INTERVALS=$INTERVALS

# mother
gatk --java-options -Xmx2G CollectHsMetrics \
--INPUT=HG004_mother_recalibrated.bam \
--REFERENCE_SEQUENCE=$REF \
--OUTPUT=quality_control/HG004_mother_HsMetrics.txt \
--BAIT_INTERVALS=$INTERVALS \
--TARGET_INTERVALS=$INTERVALS

# Collect various metrics and generate HTML report using Qualimap

# son
qualimap bamqc -bam HG002_son_recalibrated.bam -gff $BED -nt 4 -c \
-sd -outdir quality_control/HG002_son_qualimap --java-mem-size=2G

# father
qualimap bamqc -bam HG003_father_recalibrated.bam -gff $BED -nt 4 -c \
-sd -outdir quality_control/HG003_father_qualimap --java-mem-size=2G

# mother
qualimap bamqc -bam HG004_mother_recalibrated.bam -gff $BED -nt 4 -c \
-sd -outdir quality_control/HG004_mother_qualimap --java-mem-size=2G

# Combine all our statistics to one report with multiqc
multiqc quality_control -o quality_control \
-c ${project}/resources/multiqc_config.yaml

```

Script 6: Quality control and statistics of bam files

8 Variant Calling with HaplotypeCaller

In the last step we run GATK's HaplotypeCaller to call SNPs and indels and output a single vcf file called **trio-variants.vcf.gz** that contains variants for all samples.

Small description of the HaplotypeCaller from GATK's webpage: The HaplotypeCaller is capable of calling SNPs and indels simultaneously via local de-novo assembly of haplotypes in an active region.

In other words, whenever the program encounters a region showing signs of variation, it discards the existing mapping information (obtained from bwa mem) and completely reassembles the reads in that region. This allows the HaplotypeCaller to be more accurate when calling regions that are traditionally difficult to call, for example when they contain different types of variants close to each other

We can run the HaplotypeCaller by executing the commands in Script 7.

```
### NGS Workshop - HaplotypeCaller ###

# The location of the workshop data
project=/work/workshop

# Let's go to our working directory (if you're not already there)
work=/work/analysis_results
cd ${work}

# Path to reference genome and regions file
REF=${project}/reference_genome/human_g1k_v37_decoy.fasta
BED=${project}/resources/workshop_chr21_regions_g1k.bed

### HaplotypeCaller ###

# Call variants on samples using GATK's HaplotypeCaller

gatk --java-options -Xmx2g HaplotypeCaller \
-R=$REF \
-I=HG002_son_recalibrated.bam \
-I=HG003_father_recalibrated.bam \
-I=HG004_mother_recalibrated.bam \
-L=$BED \
-O=trio-variants.vcf.gz

# Now, let's see that we created our trio-variants.vcf.gz file
ls

# Let's count the number of variants we found
# zcat: allows to look into compressed files
# grep -v: filters out header lines (starting with #)
# wc -l: counts number of lines in file
zcat trio-variants.vcf.gz | grep -v ^# | wc -l

# At last, let's also take a quick look of the contents in the vcf file
# with all of the discovered variants
zcat trio-variants.vcf.gz | more
```

Script 7: Calling variants with HaplotypeCaller