**WES**

**Running the Pipeline**

**To submit a script on cluster:**

qsub script.sh

**To check the jobs:**

qstat

**To delete a job:**

qdel job\_number

* To open/create a script, type “vi scriptname.sh”
  + For example, “vi step1.sh” and press enter
* This should open a new text file – press “a” or “i” to edit the file
* Press “esc” and then type “:wq” to exit the file and save
* This script is now ready to submit – do this with qsub scriptname.sh

## Quality Check

**Tool used- Fastqc/0.11.7(https://www.bioinformatics.babraham.ac.uk/projects/fastqc)**

Check the quality of the fastq files obtained from the sequencing facility and see if the basic statistics is passed i.e the quality score of reads is above 30 phred score.

#!/bin/bash  
#PBS -N Quality\_Check  
#PBS -S /bin/bash  
#PBS -l walltime=96:00:00  
#PBS -l nodes=1:ppn=4  
#PBS -l mem=40gb  
#PBS -o /scratch/username/your\_path/log.quality.out  
#PBS -e /scratch/username/your\_path/log.quality.err

module load java-jdk/1.10.0\_1

module load fastqc/0.11.7

echo START

mkdir QC

for input in `ls /scratch/\*\*\*/\*fastq.gz`; do

echo $input

fastqc -t 8 $input --outdir /path/to/QC

done

echo END

## STEP 1

## Generate an unmapped BAM from FASTQ

### **Tool used- Picard(2.8.1)( https://gatk.broadinstitute.org/hc/en-us/articles/360036351132-FastqToSam-Picard-)**

### Convert FASTQ to uBAM and add read group information using FastqToSam.For unmapped BAM files you can assign read groups which let you specify example what platform was used (Sanger, Illumina, Roche 454, IonTorrent, etc) and the sample name (e.g. Cancer vs Healthy). (<https://blastedbio.blogspot.co.uk/2011/10/fastq-must-die-long-live-sambam.html> )

We are analyzing pair-end whole exome sequencing data. The crucial part for the first step of the pipeline is the file names, which in this case is:

HLMM2BBXX\_15607628\_S19\_L003\_R1\_001.fastq.gz

HLMM2BBXX\_15607628\_S19\_L003\_R2\_001.fastq.gz

Where reads for a forward primer are in the R1 file and reads start with the reverse primer should be in R2 file.

From the file name, we can get information to assign read groups, such as sample name (S19), lane number (3) and platform unit (HLMM2BBXX.3).

In the script presented below, I used a “for” loop, which takes each R1 fastq file, combines it with the R2 pair and gives proper read groups to each pair of fastq files. In this way we don’t have to do it for each fastq pair separately.

Final script, which you can modify to find what you need:

#!/bin/bash  
#PBS -N Step1  
#PBS -S /bin/bash  
#PBS -l walltime=96:00:00  
#PBS -l nodes=1:ppn=4  
#PBS -l mem=40gb  
#PBS -o /scratch/username/your\_path/log.step1.out  
#PBS -e /scratch/username/your\_path/log.step1.err

module load java-jdk/1.8.0\_92

module load picard/2.8.1

echo START

for input in `ls /scratch/\*\*\*/\*\_L004\_R1\_001.fastq.gz`; do platform\_unit=("`zless $input |head -n1 | awk -F ':' '{print $3 "." $4}'`"); java -Djava.io.tmpdir=${input%\_R1\_001.fastq.gz}"\_001\_tmp" - jar /apps/software/java-jdk-1.8.0\_92/picard/2.8.1/picard.jar FastqToSam FASTQ=$input FASTQ2=${input%\_R1\_001.fastq.gz}"\_R2\_001.fastq.gz" OUTPUT=${input%\_R1\_001.fastq.gz}"\_ubam.bam" READ\_GROUP\_NAME=\*\*\* SAMPLE\_NAME=\*\*\* LIBRARY\_NAME=L1 PLATFORM\_UNIT=$platform\_unit PLATFORM=illumina ;done

echo END

**STEP 2**

**Mark adapter sequences using MarkIlluminaAdapters**

**Tool used - Picard 2.8.1 (** [**https://gatk.broadinstitute.org/hc/en-us/articles/360037434291-MarkIlluminaAdapters-Picard-**](https://gatk.broadinstitute.org/hc/en-us/articles/360037434291-MarkIlluminaAdapters-Picard-) **)**

[MarkIlluminaAdapters](https://broadinstitute.github.io/picard/command-line-overview.html#MarkIlluminaAdapters) adds the XT tag to a read record to mark the 5' start position of the specified adapter sequence and produces a metrics file. Some of the marked adapters come from concatenated adapters that randomly arise from the primordial soup that is a PCR reaction. Others represent read-through to 3' adapter ends of reads and arise from insert sizes that are shorter than the read length.

(<https://software.broadinstitute.org/gatk/documentation/article.php?id=6483> )

Again we don’t want to mark adapters for each sample separately, so this “for” loop will take each uBAM file created in previous step and run MarkIlluminaAdapters for it.

Final script:

#!/bin/bash

#PBS -N **step2**

#PBS -S /bin/bash

#PBS -l walltime=**72:00:00**

#PBS -l nodes=1:ppn=4

#PBS -l mem=**40gb**

#PBS -o /scratch/username/your\_path/log.step2.out

#PBS -e /scratch/username/your\_path/log.step2.err

module load java-jdk/1.8.0\_92

module load picard/2.8.1

for input in `ls /scratch/\*\*\*/\*\_ubam.bam`; do java - Djava.io.tmpdir=${input%\_ubam.bam}"\_tmp\_step2" -jar /apps/software/java-jdk-1.8.0\_92/picard/2.8.1/picard.jar MarkIlluminaAdapters I=$input O=${input%\_ubam.bam}"\_step2.bam" M=${input%\_ubam.bam}"\_step2\_metrics.txt" ;done

echo END

## STEP 3

## Align reads with BWA-MEM and merge with uBAM using MergeBamAlignment

This step actually pipes three processes, performed by three different tools. The three tools we use are SamToFastq, BWA-MEM and MergeBamAlignment.

### **3A**

### **Convert BAM to FASTQ and discount adapter sequences using SamToFastq**

**Tool used-Picard 2.8.1**

We use additional options to effectively remove previously marked adapter sequences, in this example marked with an XT tag. By specifying CLIPPING\_ATTRIBUTE=XT and CLIPPING\_ACTION=2.

For our paired reads example file we set SamToFastq's INTERLEAVE to true. During the conversion to FASTQ format, the query name of the reads in a pair are marked with /1 or /2 and paired reads are retained in the same FASTQ file.

Another “for” loop to avoid repeating everything separately for each file.

Final script:

#!/bin/bash  
#PBS -N Step3A  
#PBS -S /bin/bash  
#PBS -l walltime=80:00:00  
#PBS -l nodes=1:ppn=4  
#PBS -l mem=40gb  
#PBS -o /scratch/username/your\_path/log.step3a.out

#PBS -e /scratch/username/your\_path/log.step3a.err

module load java-jdk/1.8.0\_92

module load picard/2.8.1

echo START

for input in `ls /scratch/\*\*\*/\*\_step2.bam`; do java - Djava.io.tmpdir=${input%\_step2.bam}"\_tmp\_step3a\_BWA" -jar /apps/software/java-jdk-1.8.0\_92/picard/2.8.1/picard.jar SamToFastq I=$input FASTQ=${input%\_step2.bam}"\_3a\_forBWA.fastq" CLIPPING\_ATTRIBUTE=XT CLIPPING\_ACTION=2 INTERLEAVE=true NON\_PF=true; done

echo END

### **3B**

### **Align reads and flag secondary hits using BWA-MEM**

**Tool used-BWA 0.7.15**

**Reference Genome Used- hg38**

BWA alignment requires an indexed reference genome file. If you are downloading the reference from the web site, which I mentioned, you will find all the indexes there, so you can just download them and don’t worry.

But if you need it there is few tips:

To index the human genome for BWA, we apply BWA's index function on the reference genome file.

/apps/software/gcc-6.2.0/bwa/0.7.15/bwa index -a bwtsw /gpfs/data/godley-lab/path\_to\_reference/reference.fa.gz

We need five index files with the extensions amb, ann, bwt, pac and sa.

Command, which might be useful if you run out of time or memory:

/apps/software/gcc-6.2.0/bwa/0.7.15/bwa bwt2sa /gpfs/data/godley-lab/ path\_to\_reference/reference.fa.gz.bwt /gpfs/data/godley-lab/path\_to\_refrence/reference.fa.gz.sa

After you have reference and index files, you can run next “for” loop to align reads with BWA MEM.

Final script:

#!/bin/bash  
#PBS -N Step3B  
#PBS -S /bin/bash  
#PBS -l walltime=80:00:00  
#PBS -l nodes=1:ppn=4  
#PBS -l mem=40gb  
#PBS -o /scratch/username/your\_path/log.step3b.out

#PBS -e /scratch/username/your\_path/log.step3b.err

module load gcc/6.2.0

module load java-jdk/1.8.0\_92

module load picard/2.8.1

module load bwa/0.7.15

echo START

for input in `ls /scratch/\*\*\*/\*\_3a\_forBWA.fastq`; do /apps/software/gcc-6.2.0/bwa/0.7.15/bwa mem -M -t 7 -p /gpfs/data/godley-lab/WES\_analysis/reference\_human\_38/Homo\_sapiens.GRCh38.dna.toplevel.fa.gz $input > ${input%\_3a\_forBWA.fastq}"\_3b\_bwa\_mem.sam" ; done

echo END

### **3C**

### **Restore altered data and apply & adjust meta information using MergeBamAlignment**

**Tools used-picard 2.8.1 (** [**https://gatk.broadinstitute.org/hc/en-us/articles/360037225832-MergeBamAlignment-Picard-**](https://gatk.broadinstitute.org/hc/en-us/articles/360037225832-MergeBamAlignment-Picard-) **)**

The tool merges defined information from the unmapped BAM (uBAM, step 1) with that of the aligned BAM (step 3) to conserve read data, e.g. original read information and base quality scores. The tool also generates additional meta information based on the information generated by the aligner, which may alter aligner-generated designations, e.g. mate information and secondary alignment flags.

*Before running this part we need to have dictionary for the reference genome, if you didn’t download it, this command will help you to do that:*

*java -jar /apps/software/java-jdk-1.8.0\_92/picard/2.8.1/picard.jar CreateSequenceDictionary REFERENCE=/gpfs/data/godley-lab/refrence/refrence.fa.gz OUTPUT=/gpfs/data/godley-lab/reference/reference.dict*

*The dictionary file is formatted like a SAM header, describing the contents of your reference FASTA file. After you got it, you can run the next step.*

Final script:

#!/bin/bash  
#PBS -N Step3C  
#PBS -S /bin/bash  
#PBS -l walltime=80:00:00  
#PBS -l nodes=1:ppn=4  
#PBS -l mem=40gb  
#PBS -o /scratch/username/your\_path/log.step3c.out

#PBS -e /scratch/username/your\_path/log.step3c.err

module load java-jdk/1.8.0\_92

module load picard/2.8.1

for input in `ls /scratch/\*\*\*/\*\_3b\_bwa\_mem.sam`; do java - Djava.io.tmpdir=${input%\_3b\_bwa\_mem.sam}"\_tmp\_step3c\_BWA" -jar /apps/software/java-jdk-1.8.0\_92/picard/2.8.1/picard.jar MergeBamAlignment R=/gpfs/data/godley- lab/WES\_analysis/reference\_human\_38/Homo\_sapiens.GRCh38.dna.toplevel.f a.gz ALIGNED\_BAM=$input UNMAPPED\_BAM=${input%\_3b\_bwa\_mem.sam}"\_ubam.bam" OUTPUT=${input%\_3b\_bwa\_mem.sam}"\_mapped.bam" CREATE\_INDEX=true ADD\_MATE\_CIGAR=true CLIP\_ADAPTERS=false CLIP\_OVERLAPPING\_READS=true INCLUDE\_SECONDARY\_ALIGNMENTS=true MAX\_INSERTIONS\_OR\_DELETIONS=-1 PRIMARY\_ALIGNMENT\_STRATEGY=MostDistant ATTRIBUTES\_TO\_RETAIN=XS ; done

echo END

The maximum number of insertions or deletions permitted for an alignment to be included. Alignments with more than this many insertions or deletions will be ignored. Set to -1 to allow any number of insertions or deletions.

### **STEP 4**

### **Mark duplicates**

**Tools used-picard 2.8.1(https://gatk.broadinstitute.org/hc/en-us/articles/360037052812-MarkDuplicates-Picard-)**

Since we do not need sorting, because our BAM files are already sorted, we directly go to mark duplicated.

To check if your files are sorted:

samtools view –H bam\_file\_name.bam

If you will see:

@HD VN:1.5 SO:coordinate

SO:coordinate means the file is sorted by coordinate. If it would be not sorted then you will see: SO:unsorted.

 About marking duplicates, the idea here is that during the sequencing process, the same DNA fragments may be sequenced several times. The resulting duplicate reads are not informative and should not be counted as additional evidence for or against a putative variant.

The duplicate marking process (sometimes called \*\*dedupping\*\* in bioinformatics slang) does not remove the reads, but identifies them as duplicates by adding a flag in the read's SAM record. Most GATK tools will then ignore these duplicate reads by default, through the internal application of a read filter.

*In this part we can combine different lines together (****L002, L003*** *ect.)*

The script for marking duplicates.

#!/bin/bash  
#PBS -N Step4  
#PBS -S /bin/bash  
#PBS -l walltime=80:00:00  
#PBS -l nodes=1:ppn=4  
#PBS -l mem=40gb  
#PBS -o /scratch/username/your\_path/log.step4.out

#PBS -e /scratch/username/your\_path/log.step4.err

module load java-jdk/1.8.0\_92

module load picard/2.8.1

for input in `ls /scratch/\*\*\*/\*L003\_mapped.bam`; do java - Djava.io.tmpdir=${input%\_mapped.bam}"\_tmp\_step4" -jar /apps/software/java-jdk-1.8.0\_92/picard/2.8.1/picard.jar MarkDuplicates INPUT=$input INPUT=${input%\_L003\_mapped.bam}"\_L004\_mapped.bam" OUTPUT=${input%\_301\_276\_S261\_L003\_mapped.bam}"\_step4.bam" METRICS\_FILE=${input%\_301\_276\_S261\_L003\_mapped.bam}"\_step4\_metrics.txt " OPTICAL\_DUPLICATE\_PIXEL\_DISTANCE=2500 CREATE\_INDEX=true ; doneecho END

In this case I was combining two lines L002 and L003, please be careful, how many lines do you have.

### **STEP 5**

### **Recalibrate Bases**

**Tool used-GATK(3.7)**

Variant calling algorithms rely heavily on the quality scores assigned to the individual base calls in each sequence read. These scores are per-base estimates of error emitted by the sequencing machines. The scores produced by the machines are subject to various sources of systematic technical error, leading to over- or under-estimated base quality scores in the data.

Base quality score recalibration (BQSR) is a process in which we apply machine learning to model these errors empirically and adjust the quality scores accordingly. This allows us to get more accurate base qualities, which in turn improves the accuracy of our variant calls.

The base recalibration process involves two key steps: first the program builds a model of covariation based on the data and a set of known variants, then it adjusts the base quality scores in the data based on the model. The known variants are used to mask out bases at sites of real (expected) variation, to avoid counting real variants as errors. Outside of the masked sites, every mismatch is counted as an error. The rest is mostly accounting.

To get a set of known variants, you will need to download dbSNP (The Single Nucleotide Polymorphism Database)

It is **important** to use the same version of reference in known variants (here we are using all the time version 38).

### **5A**

### **Analyze patterns of covariation in the sequence dataset**

**Tool used – GATK BaseRecalibrator (** [**https://gatk.broadinstitute.org/hc/en-us/articles/360036898312-BaseRecalibrator**](https://gatk.broadinstitute.org/hc/en-us/articles/360036898312-BaseRecalibrator) **)**

This creates a GATK Report file called recal\_data.table containing several tables. These tables contain the covariation data that will be used in a later step to recalibrate the base qualities of your sequence data.

And here is the script:

#!/bin/bash  
#PBS -N Step5a  
#PBS -S /bin/bash  
#PBS -l walltime=80:00:00  
#PBS -l nodes=1:ppn=4  
#PBS -l mem=40gb  
#PBS -o /scratch/username/your\_path/log.step5a.out

#PBS -e /scratch/username/your\_path/log.step5a.err

module load java-jdk/1.8.0\_92

module load gatk/3.7

echo START

for input in `ls /scratch/\*\*\*/\*step4.bam`; do java -jar /apps/software/java-jdk-1.8.0\_92/gatk/3.7/GenomeAnalysisTK.jar -T BaseRecalibrator -R /gpfs/data/godley- lab/WES\_analysis/reference\_human\_38/Homo\_sapiens.GRCh38.dna.toplevel.f a -I $input -knownSites /gpfs/data/godley-lab/WES\_analysis/dbsnp/All\_20170710\_sort.vcf -o ${input%\_step4.bam}"\_step5a\_BaseRecalibrator.table" -U ALLOW\_SEQ\_DICT\_INCOMPATIBILITY; done

echo END

\*\*\*\*Important note: you can use “-U ALLOW\_SEQ\_DICT\_INCOMPATIBILITY”, which helps to avoid the error about incompatibility of contigs in reference and dbSNP. But I would try to avoid it, if possible.

### **5D**

### **Apply the recalibration to your sequence data**

Tool used-GATK

This creates a .bam file containing all the original reads, but now with exquisitely accurate base substitution, insertion and deletion quality scores. By default, the original quality scores are discarded in order to keep the file size down.

Notice how this step uses a very simple tool, PrintReads, to apply the recalibration. What’s happening here is that we are loading in the original sequence data, having the GATK engine recalibrate the base qualities on-the-fly thanks to the -BQSR flag (as explained earlier), and just using PrintReads to write out the resulting data to the new file.

The script to create the new **bam** files:

#!/bin/bash  
#PBS -N Step5d  
#PBS -S /bin/bash  
#PBS -l walltime=80:00:00  
#PBS -l nodes=1:ppn=4  
#PBS -l mem=40gb  
#PBS -o /scratch/username/your\_path/log.step5d.out

#PBS -e /scratch/username/your\_path/log.step5d.err

module load java-jdk/1.8.0\_92

module load gatk/3.7

echo START

for input in `ls /scratch/\*\*\*/\*step4.bam`; do java -jar /apps/software/java-jdk-1.8.0\_92/gatk/3.7/GenomeAnalysisTK.jar -T PrintReads -R /gpfs/data/godley- lab/WES\_analysis/reference\_human\_38/Homo\_sapiens.GRCh38.dna.toplevel.fa -I $input -BQSR ${input%\_step4.bam}"\_step5a\_BaseRecalibrator.table" -o ${input%\_step4.bam}"\_step5d.bam" ; done

echo END

That makes the header of the .bam file same, what in the next step makes a nice one .vcf file with all the variants.

Or play around with:

samtools view -H sample1.bam | sed "s/SM:[^\t]\*/SM:sample1/g" | samtools reheader – sample1.bam > sample1\_header.bam

## STEP 6

## Variant Discovery

## Tool used- GATK HaplotypeCaller(https://gatk.broadinstitute.org/hc/en-us/articles/360037225632-HaplotypeCaller)

## Calling variants is the last step (next you can do any kind of filtering).

## After that step we are getting the .vcf files with all the variants.

## Here I include example script, which you will have to modify for your needs. In this case I didn’t use for loop, just I manually choose, what do I want to include.

## Here is a script:

#!/bin/bash  
#PBS -N Step6  
#PBS -S /bin/bash  
#PBS -l walltime=80:00:00  
#PBS -l nodes=1:ppn=4  
#PBS -l mem=40gb  
#PBS -o /scratch/username/your\_path/log.step6.out

#PBS -e /scratch/username/your\_path/log.step6.err

module load java-jdk/1.8.0\_92

module load gatk/3.7

echo START

for input in `ls /scratch/\*\*\*/\*step5d.bam`; do java -jar /apps/software/java-jdk-1.8.0\_92/gatk/3.7/GenomeAnalysisTK.jar -T HaplotypeCaller -R /gpfs/data/godley- lab/WES\_analysis/reference\_human\_38/Homo\_sapiens.GRCh38.dna.toplevel.f a -I $input --genotyping\_mode DISCOVERY -o ${input%\_step5d.bam}".vcf"; done

echo END

## STEP 7

## Variant annotation

Tool used – **Annovar(https://annovar.openbioinformatics.org/en/latest/)**

#!/bin/bash

#PBS -N Step7

#PBS -S /bin/bash

#PBS -l walltime=24:00:00

#PBS -l nodes=1:ppn=4

#PBS -l mem=40gb

#PBS -o /scratch/username/your\_path/log.step7.out

#PBS -e /scratch/username/your\_path/log.step7.err

module load gcc/4.9.4

module load perl/5.18.4

echo START

for input in `ls /scratch/\*\*\*/\*vcf`; do /gpfs/data/godley-lab/WES\_analysis/annovar/table\_annovar.pl $input /gpfs/data/godley-lab/WES\_analysis/annovar/humandb/ -buildver hg38 -out ${input%.vcf} -arg '-splicing 15',,,,,,,, -remove -protocol refGene,cytoBand,exac03,gnomad\_exome,gnomad\_genome,kaviar\_20150923,clinvar\_20170905,avsnp147,dbnsfp30a -operation g,r,f,f,f,f,f,f,f -nastring . -vcfinput ; done

echo END