Variant Calling

Ty Medina

18 May 2018

Contents

Introduction

NGS sequencing data from 3 samples was provided for germline variant calling analysis. Each sample is a normal biopsy from a Head and Neck Squamous Cell Carcinoma (HNSCC) patient.

The data was generated as 100bp paired-end reads from an Illumina HiSeq platform, and was been presented as 8 FASTQ files, one sample having been split across 2 runs. These 4 runs are as follows:

- HN51
- HN60
- HN72_AC254KACXX
- HN72_AH0LENADXX

In this analysis, germline variant calling was performed using two methods. The first method makes use of the Samtools software suite, while the second was adapted from GATK best practices guidelines. Both methods utilized the same basic trimming procedure, then followed through similar steps of alignment, read-pair verification, duplicate removal, and variant calling. Commands used for each step, as well as a description of arguments, are shown for each process. Scripts for both processes can be found at the end of this report.

Raw Data Preprocessing

Quality Control

The raw data in FASTQ format was first checked for quality control using the FastQC tool. Overall quality reports for each FASTQ file were then combined using the MultiQC tool.

```
fastqc -o QC_files HN51_P1.fastq
multiqc -o QC_files QC_files
```

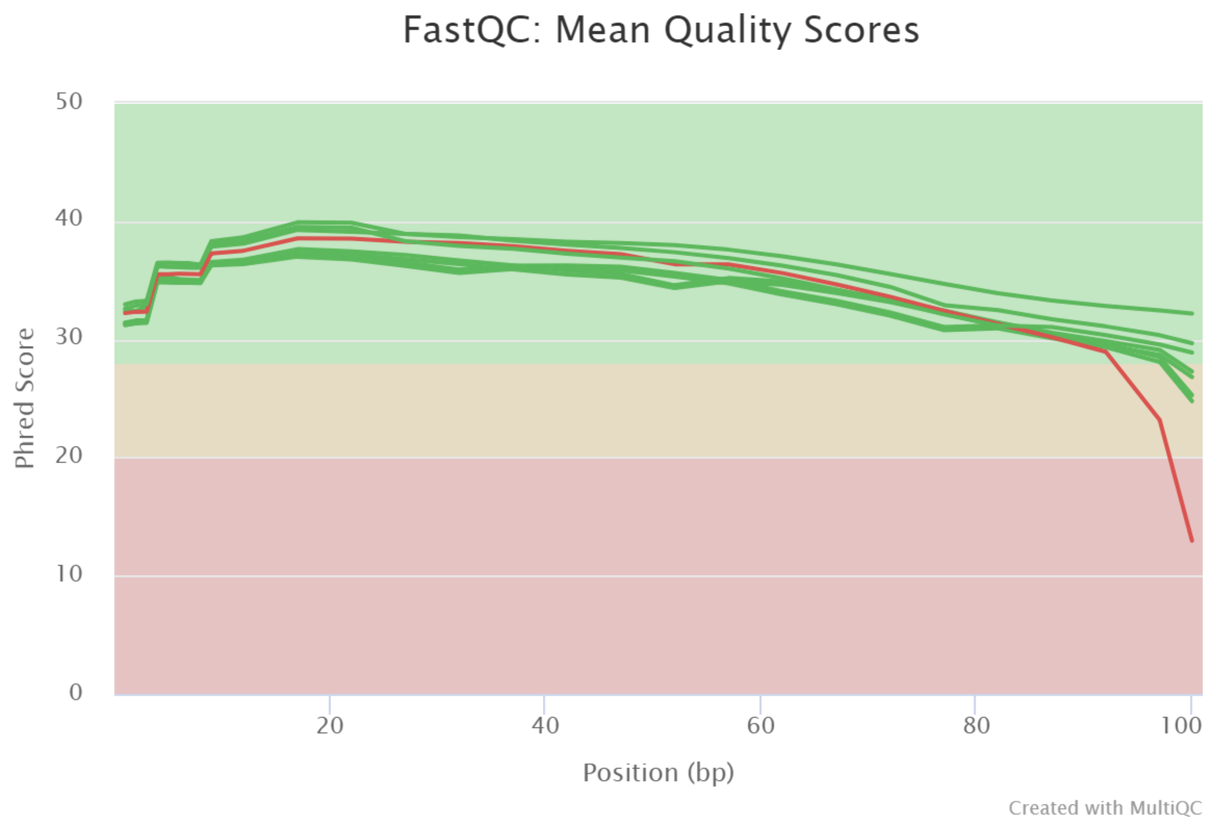
Trimming

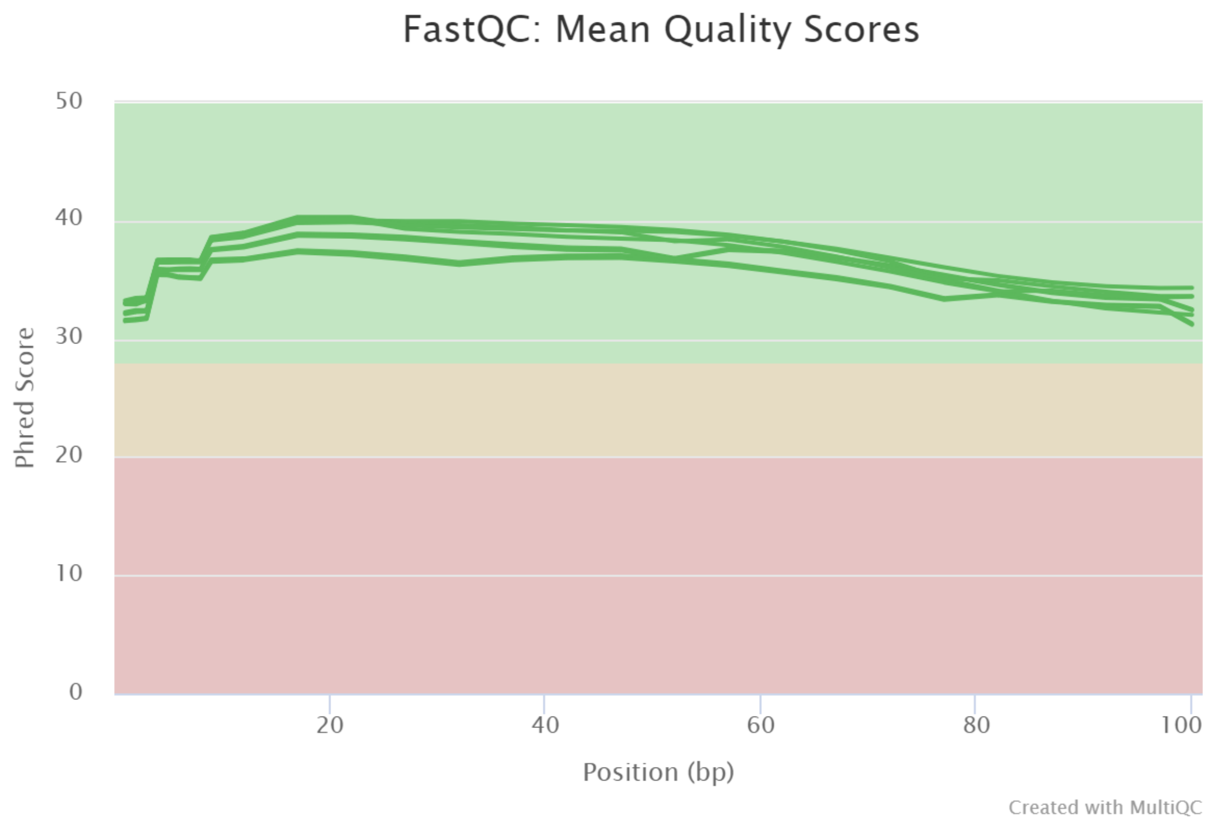
To ensure that only high-quality reads were used in the analysis, Trimmomatic was used to remove any Illumina adapter sequences, trim low-quality ends from each fragment, and remove any fragments that were too short from each FASTQ file. Because the data was paired-end, Trimmomatic was run in PE-mode, allowing it to also remove fragments whose pairs did not survive trimming.

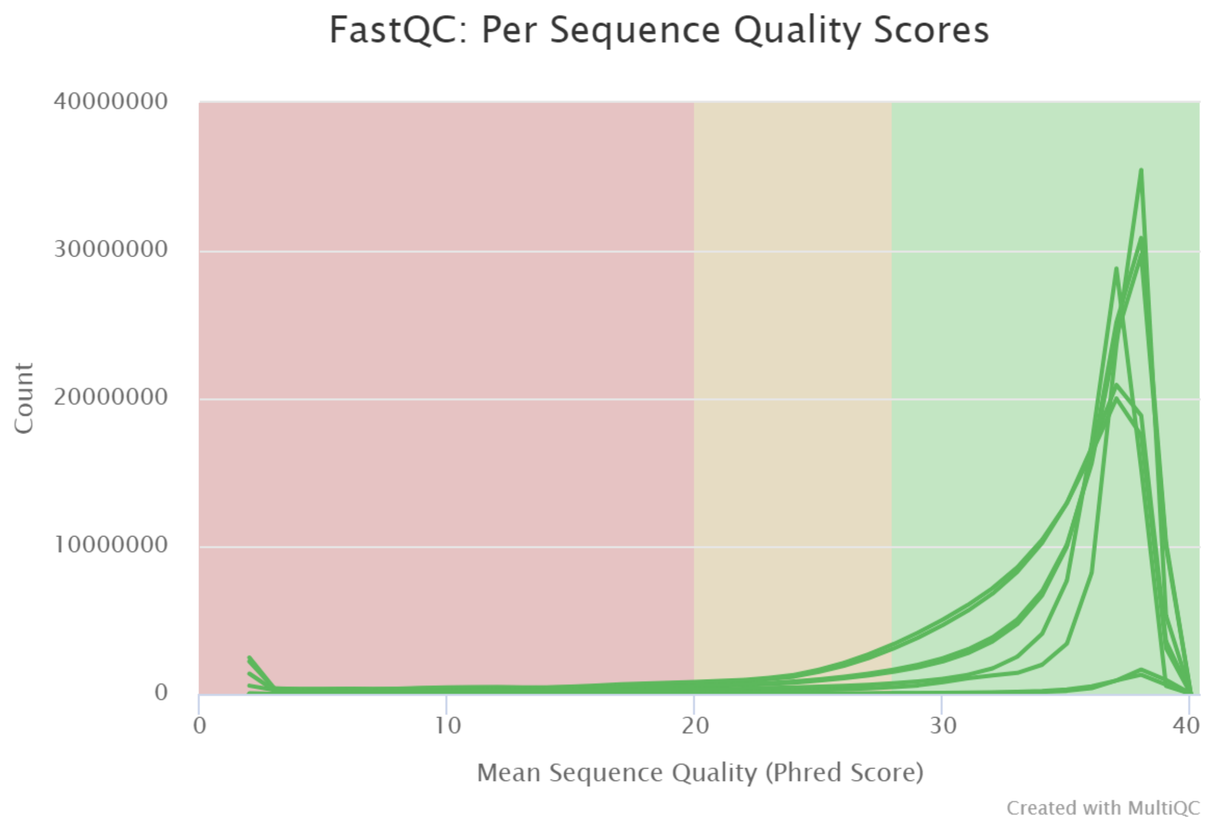
```
java -jar trimmomatic.jar PE -phred33 -trimlog trimlog.txt \
     HN51_P1.fastq HN51_P2.fastq \
     HN51_P1_trimmed.fastq HN51_P1_unpaired.fastq \
     HN51_P2_trimmed.fastq HN51_P2_unpaired.fastq \
     ILLUMINACLIP:Trimmomatic/adapters/TruSeq3-PE.fa:2:30:10 \
     LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15 MINLEN:36
```

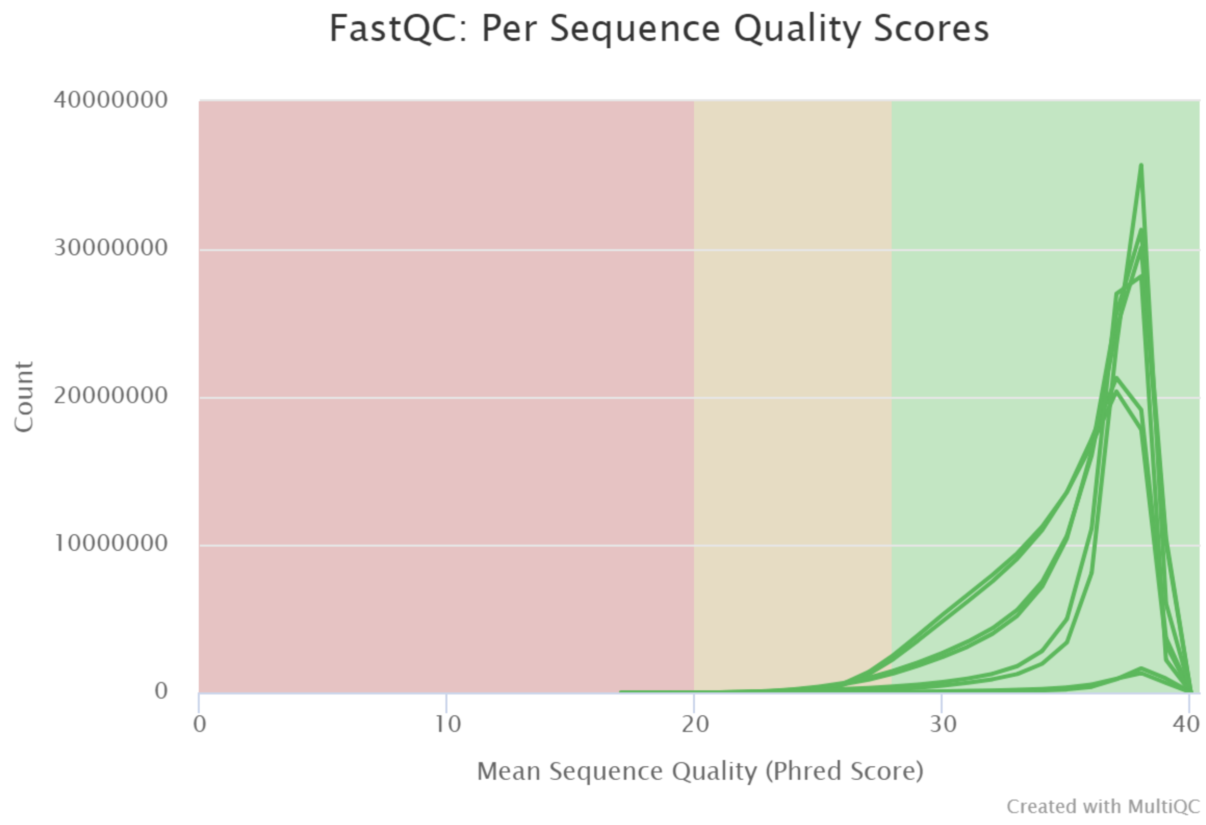
Quality Control, Post-Trimming

After the trimming procedure, FastQC and MultiQC were used again to check the resulting new quality of each file. Before and after comparisons are shown on the following pages, showing overall improvements in mean quality scores, the number of high quality reads, and the amount of N-called bases.

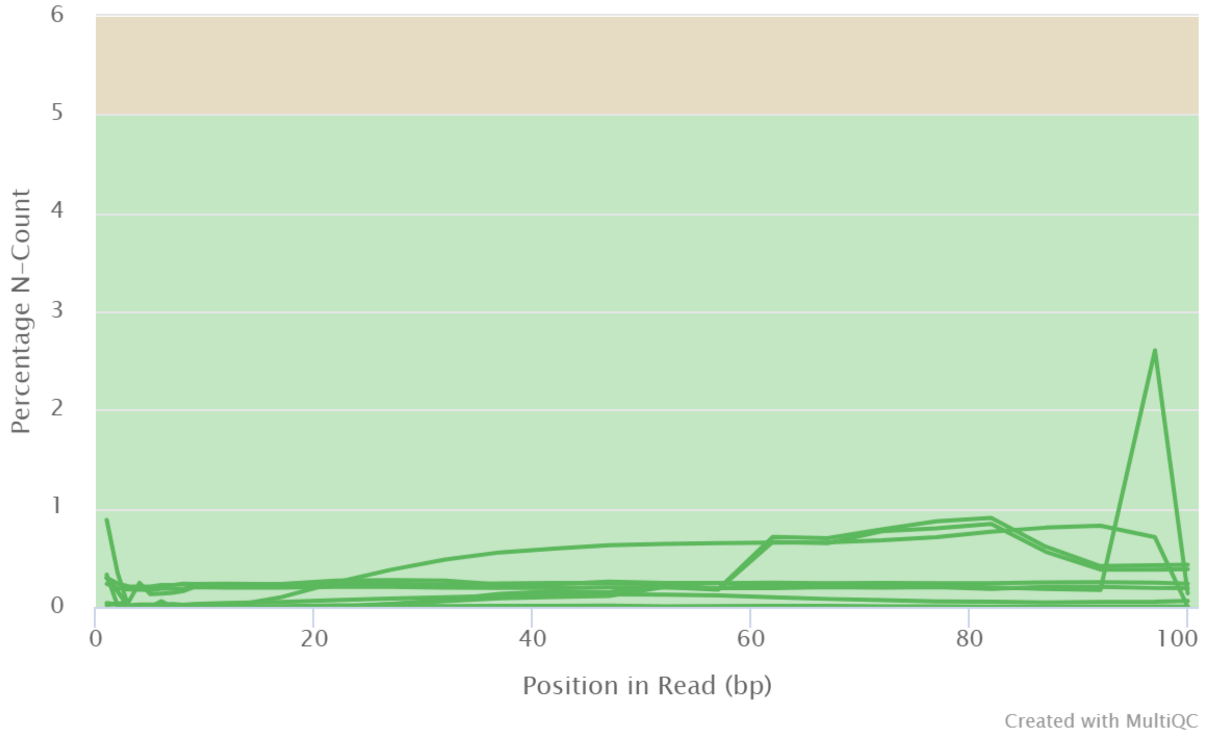

FastQC: Mean Quality Scores
Phred Score
50
40
30
20
10
0
20
40
60
80
100
Position (bp)
Created with MultiQC


FastQC: Mean Quality Scores
Phred Score
50
40
30
20
10
0
20
40
60
80
100
Position (bp)
Created with MultiQC

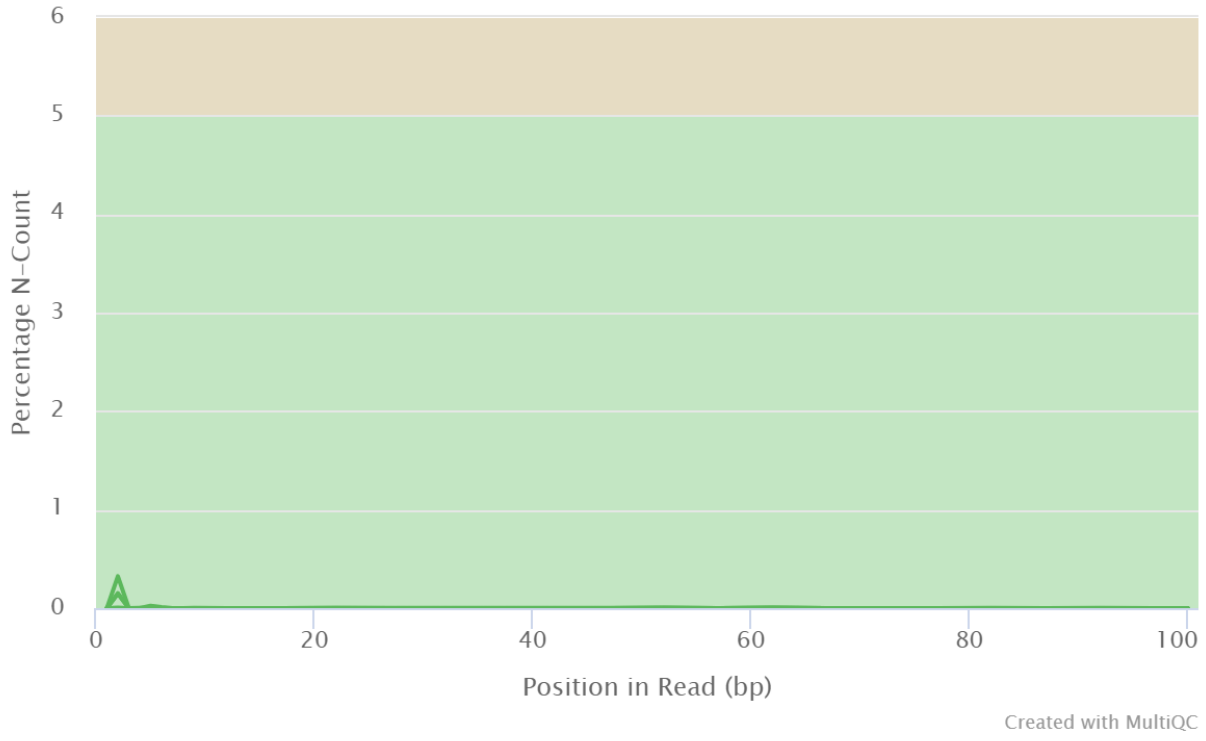

FastQC: Per Sequence Quality Scores
40000000
30000000
20000000
10000000
0
Count
0
10
20
30
40
Mean Sequence Quality (Phred Score)
Created with MultiQC


FastQC: Per Sequence Quality Scores
40000000
30000000
20000000
10000000
0
Count
0
10
20
30
40
Mean Sequence Quality (Phred Score)
Created with MultiQC

FastQC: Per Base N Content



FastQC: Per Base N Content



Samtools Procedure

The following outlines the variant calling procedure performed using the Samtools software suite, which includes both the eponymous Samtools as well as BCFtools.

Reference Indexing

The first step performed was read alignment. In this process, paired FASTQ files were passed to BWA-MEM. The Burrows-Wheeler Aligner, using its Maximal Exact Matches algorithm, aligns by seeding exact matches to the genome before extending them using the Smith-Waterman affine-gap penalty algorithm. Before alignment could proceed, the reference genome was first indexed using the "bwa index" command. The "-a bwtsw" option was used to specify the Smith-Waterman algorithm for whole genome usage, rather than the default algorithm for smaller datasets.

```
bwa index -a bwtsw hg19.fa
```

Alignment

After obtaining the index files for the reference genome, the actual alignment was performed. The options used are as follows:

- -t: Number of threads to use for parallel computation
- -R: The readgroup to use for the output file header, which breaks down into these components:
  - ID: The unique ID for this read group, given here as the unique Illumina flowcell number and lane
  - SM: The sample number
  - PL: The platform used to generate the NGS data
  - LB: The library preparation of the run, assumed to be the same as the sample in this data

The last 3 arguments shown are the reference genome prefix and the 2 paired FASTQ files, followed by output redirection to a SAM file.

```
bwa mem -t 4 -R "@RG\tID:BD1LYPACXX.3\tSM:HN51\tPL:ILLUMINA\tLB:HN51" hg19.fa \
  HN51_P1_trimmed.fastq HN51_P2_trimmed.fastq > HN51.sam
```

The first two reads of the output SAM file are shown here. Note that fields have been broken into multiple lines and shortened for readability in this document. Lines are as follows:

1. Read info, containing the following tab-separated values:
   - The unique read name (paired reads are given the same name)
   - Bitwise-flag containg read information, such as whether the read is paired or strandedness
   - Reference sequence that the read aligned to (in this case, which chromosome)
   - Start position of the alignment
   - Cigar string, providing information on how the alignment was processed
   - Reference sequence that the mate read aligned to, which is "=" if mates aligned to the same chromosome
   - Start position of the mate read, which can be roughly interpreted as the end of the alignment
   - Length of the alignment, which is determined from the start of each mate, and is negative for the complimentary, reverse mate
2. Sequence
3. Quality score of each base call
4. Optional tags containing additional information

HWI-ST0844:229:D1LYPACXX:3:1101:1199:1723 83 chr1 158746592 60 58M = 158746337 -313
GTAAACTATAGCAAGTGTTCGGTCAAGGGTCAGGGAATAGCTCTTCTTTAGCCGCACA
EAEFHE>GGGCFDGF;G<HFHHCGIIGIHDDEGHDHGHHCBBGHFEFFFHAFDDB=:1
NM:i:0 MD:Z:58 AS:i:58 XS:i:0 RG:Z:BD1LYPACXX.3

HWI-ST0844:229:D1LYPACXX:3:1101:1199:1723 163 chr1 158746337 60 101M = 158746592 313
GAGAAAATCATAAAGAAATGAAGTCTTTGAAGAGGTGAAAGGAAATGAAATTCAGAGCAT…
@@@FFFDEFDH?HGHHH:FHEHHHHJG9HECEAFG:EFDE>F>FIIHDFC?EHGH?D…
NM:i:0 MD:Z:101 AS:i:101 XS:i:20 RG:Z:BD1LYPACXX.3

SAM to BAM Conversion

While the SAM file format is human-readable, the binary BAM format is used for analysis due to its efficiency and smaller size. In this step, the output SAM files were converted to BAM format. The two options, "-S" and "b" indicate a SAM input and a BAM output.

```
samtools view -Sb HN51.sam > HN51.bam
```

Pair Verification

To ensure that reads were each correctly assigned to their mate, the Samtools "fixmate" tool was used to check for and correct incorrect mate-pair flags. Prior to using this tool, the reads are first sorted by read name. This ensures that reads with the same read name, which are mates, are sorted next to each other. The options used for Samtools "sort" are shown here:

- -n: Specifies to sort by name
- -m: Specifies the amount of RAM memory usable

Note that the "-m" flag was set to ~10 GB, instead of the default ~500 MB, to prevent an overly large number of temporary files from being created from memory dumps. After sorting by name, the "fixmate" tool was then used.

```
samtools sort -n -m 10000000000 HN51.bam HN51_namesort.bam

samtools fixmate HN51_namesort.bam HN51_namesort_fixmates.bam
```

HN72 Merge

Sample HN72 was split and run over 2 lanes. While the nature of this split is unclear, the 2 splits are now merged into a single BAM file. Fixing mate flags was performed prior to merging due to the fact that mates are produced during sequencing, and thus the process is run-specific. Because the nature of the split is unknown and could be the product of a single library preparation, merging must be done prior to removing duplicates to ensure that any PCR duplicates that could have separated into each split are thus rejoined and considered together.

The Samtools "merge" tool was used to combine the 2 HN72 splits, using the single "-n" option to indicate the the BAM files were name-sorted.

```
samtools merge -n HN72_Merged_namesort_fixmates.bam \
  HN72_AC254KACXX_namesort_fixmates.bam \
  HN72_AH0LENADXX_namesort_fixmates.bam
```

Duplicate Removal

The 3 BAM files were then checked for duplicate reads. These duplicate reads arise from multiple PCR clones binding to the flowcell. While the optimum library prep size and number of PCR rounds should result in a very low probability of multiple clones binding to the flowcell, any duplicates that do occur must be removed to avoid misrepresenting allele counts during variant calling.

To remove duplicates, each BAM was first re-sorted by chromosome location of each read, rather than name, as duplicates have identical alignment coordinates and thus sort together. As before, memory allocation was increased for the same reason of preventing thousands of small temporary files from forming.

After sorting, the "rmdup" tool was then used to fully remove any reads that presented as duplicates, leaving just one copy of each read.

```
samtools sort -m 10000000000 HN51_namesort_fixmates.bam HN51_chrsort_fixmates.bam

samtools rmdup HN51_chrsort_fixmates.bam HN51_chrsort_fixmates_rmdup.bam
```

BAM Indexing and Alignment Statistics

With each sample's BAM file "cleaned" and ready for variant calling, each BAM file was indexed for use. Mapping statistics were also generated for review, showing the general quality of mapping performed.

```
samtools index HN51_chrsort_fixmates_rmdup.bam

samtools flagstat HN51_chrsort_fixmates_rmdup.bam > HN51_mapstats.txt
```

Mapping stats for HN51 after processing follow, showing zero duplicates and a low proportion of improper mating:

```
211881369 + 0 in total (QC-passed reads + QC-failed reads)
0 + 0 duplicates
211855312 + 0 mapped (99.99%:nan%)
211881369 + 0 paired in sequencing
105932142 + 0 read1
105949227 + 0 read2
210604252 + 0 properly paired (99.40%:nan%)
211738437 + 0 with itself and mate mapped
116875 + 0 singletons (0.06%:nan%)
733022 + 0 with mate mapped to a different chr
575952 + 0 with mate mapped to a different chr (mapQ>=5)
```

IGV Visualization

The indexed BAM files were submitted to IGV for visualization (Fig. 1). From the read coverage, it became clear that this is exome data. However, as shown, the coverage for sample HN72 was exceedingly poor, with very poor exome enrichment and purification resulting in reads aligning across the full genome. This resulted in poor coverage depth for actual exon sequences, thus reducing the quality of variants called from HN72. The splits of HN72 are shown separately to additionally show the low levels of coverage from the smaller HN72_AH0LENADXX set, reflecting that the majority of the poor exome enrichment came from the larger HN72_AC254KACXX set.

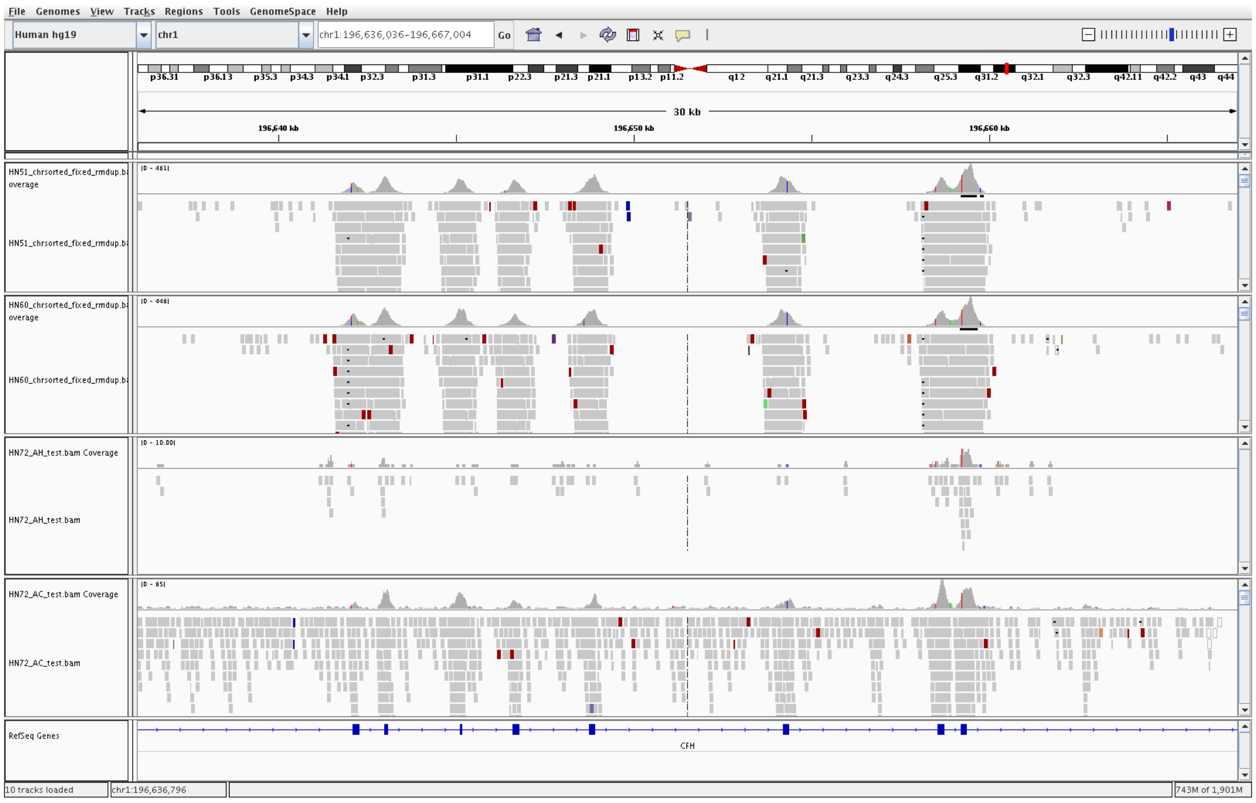

File Genomes View Tracks Regions Tools GenomeSpace Help
Human hg19
chr1
chr1:196,636,036-196,667,004
Go
30 kb
196,640 kb
196,650 kb
196,660 kb
HN51_chrsorted_fixed_rmdup.bam Coverage
HN51_chrsorted_fixed_rmdup.bam
HN60_chrsorted_fixed_rmdup.bam Coverage
HN60_chrsorted_fixed_rmdup.bam
HN72_AH_test.bam Coverage
HN72_AH_test.bam
HN72_AC_test.bam Coverage
HN72_AC_test.bam
RefSeq Genes
CFH
10 tracks loaded
chr1:196,636,796
743M of 1,901M

Figure 1: IGV Visualization of Samtools BAM files

Variant Calling

To prepare for variant calling against the reference genome, Samtools “faidx” tool was used to index the genome again, this time in a Samtools-compatible format.

After indexing, Samtools “mpileup” tool was then used to calculate variant calls. This output was then piped directly to BCFtools to generate a binary variant call format (BCF) file.

Samtools mpileup options used:

- -u: Output an uncompressed file, for piping
- -g: Output genotype likelihoods to a BCF file
- -f: Specifies that the reference genome is faidx-indexed

BCFtools options used:

- -b: Output a BCF file
- -v: Output variant sites only
- -c: Use Bayesian inference for variant calls
- -g: Genotype each variant site

```
samtools faidx hg19.fa

samtools mpileup -ugf hg19.fa HN51_chrsort_fixmates_rmdup.bam | \
  bcftools view -bvcg -> HN51.bcf
```

Variant Filtering

The resulting variants contained in the BCF file were then filtered to remove low-confidence variants with low read depth, using the vcfutils varFilter tool. The final filtered VCF file was then used for later data interpretation using VEP.

```
bcftools view HN51.bcf | vcfutils.pl varFilter -d 60 > HN51_filtered.vcf
```

Excerpts from several variants from HN51_filtered.vcf are shown here. Note that some fields have been removed for readability.

CHROM	POS	ALT	QUAL	FORMAT
chr1	13116	G	14.20	GT:PL:GQ
chr1	13118	G	3.01	GT:PL:GQ
chr1	13302	T	25.00	GT:PL:GQ
chr1	15118	G	205.00	GT:PL:GQ
chr1	15274	T,G	187.00	GT:PL:GQ
chr1	15820	T	30.00	GT:PL:GQ
chr1	16378	C	128.00	GT:PL:GQ
chr1	16534	T	142.00	GT:PL:GQ

GATK Procedure

The following outlines the GATK-adapted workflow for variant calling using the Broad Institute's Picard and GATK software packages. The GATK workflow begins after pre-processing with Trimmomatic and reference genome indexing are complete.

Alignment

Alignment proceeded very similarly to the Samtools workflow, with the addition of the -M tag for Picard compatibility. This tag instructs the aligner to label short split reads as "secondary reads", which are treated differently by Picard tools.

```
bwa mem -t 4 -R "@RG\tID:BD1LYPACXX.3\tSM:HN51\tPL:ILLUMINA\tLB:HN51" -M hg19.fa \
  HN51_P1.fastq HN51_P2.fastq > HN51_GATK.sam
```

Pair Verification

To ensure correct pair mating, Picard's "FixMateInformation" tool was used. Prior to fixing mates, reads are first sorted by name using the "SortSam" tool with SORT_ORDER set as "queryname".

Mate pair verification was then performed using Picard's "FixMateInformation" tool. As an optional argument to save a step, SORT_ORDER is set to "coordinate", instructing the tool to output a chromosome coordinate-sorted BAM file.

```
java -jar picard.jar SortSam \
      I=HN51_GATK.sam \
      O=HN51_GATK_namesort.bam \
      SORT_ORDER=queryname

java -jar picard.jar FixMateInformation \
      I=HN51_GATK_namesort.bam \
      O=HN51_GATK_fixmates.bam \
      SORT_ORDER=coordinate
```

HN72 Merge

As before, HN72 was merged after the mate verification step, prior to treating duplicates. Picard's "Merge-SamFiles" tool was used to accomplish this, taking both splits as an input for a single output BAM.

```
java -jar picard.jar MergeSamFiles \
  I=HN72_AC254KACXX_GATK_fixmates.bam \
  I=HN72_AHOLENADXX_GATK_fixmates.bam \
  O=HN72_MERGED_GATK_fixmates.bam
```

Duplicate Marking

As opposed to the older version of Samtools used, which removes duplicate reads, Picard's "MarkDuplicates" tool instead annotates duplicate reads with an additional SAM bitwise flag, leaving the reads in place in the BAM file. A summary of the actions performed by the tool can be output using the "M" argument and designating an output filename.

```
java -jar picard.jar MarkDuplicates \
      I=HN51_GATK_fixmates.bam \
      O=HN51_GATK_markdup.bam \
      M=HN51_GATK_markdup_metrics.txt
```

Reference Dictionary Creation

The GATK variant calling tool requires a reference genome dictionary file. For this purpose, Picard's "CreateSequenceDictionary" tool was used.

```
java -jar picard.jar CreateSequenceDictionary R=hg19.fa O=hg19.dict
```

BAM Indexing and Visualization

The final step performed prior to variant calling was to create index files for each BAM file, using Picard's "BuildBamIndex".

```
java -jar picard.jar BuildBamIndex I=HN51_GATK_markdup.bam
```

These BAM files and their index files were each uploaded to IGV (Fig. 2), showing nearly identical coverage compared to the Samtools-prepared BAM files, including the poor HN72 exome enrichment.

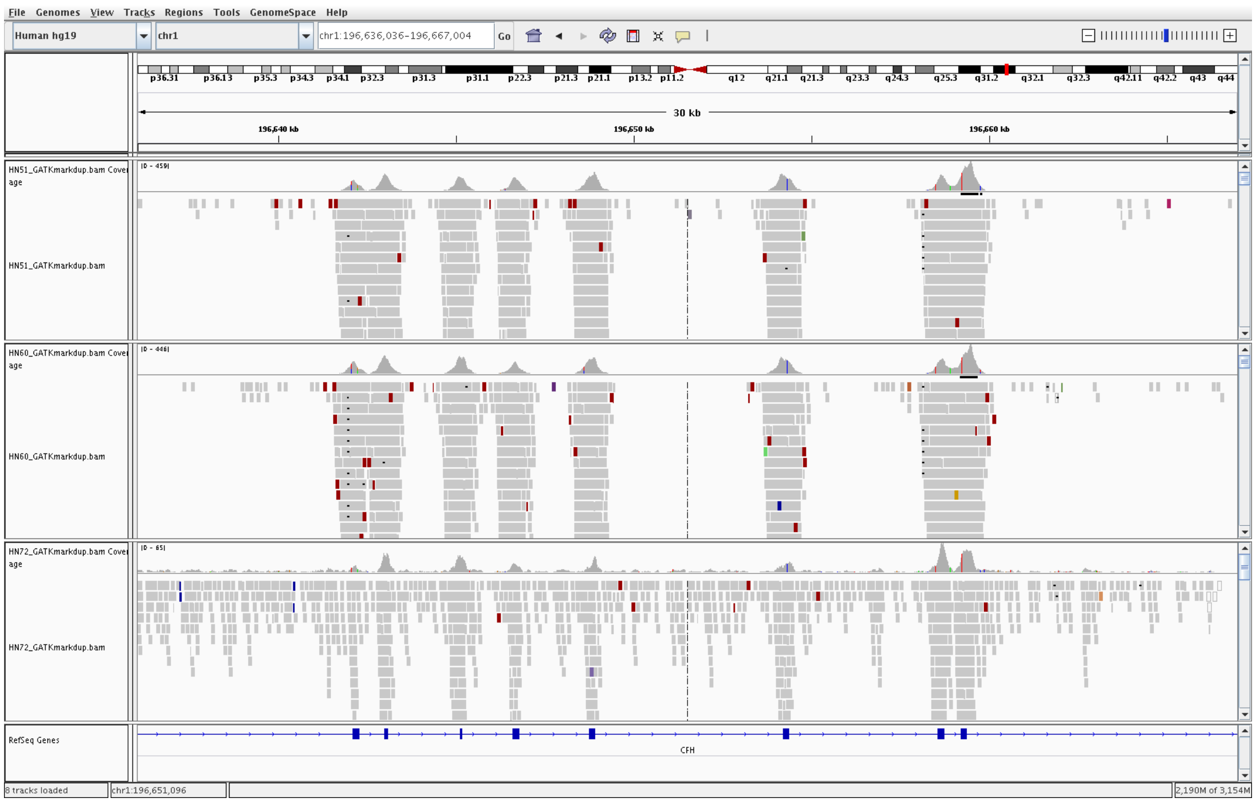


Figure 2: IGV Visualization of Picard BAM files

Haplotype Calling in GVCF Mode

With indexed BAM files and a reference genome dictionary, GATK's "HaplotypeCaller" was then used to generate G.VCF files. HaplotypeCaller was used with the "ERC GVCF" option to run in GVCF mode, preparing the VCF files to be combined across samples for joint genotyping.

```
java -jar gatk.jar HaplotypeCaller \
          -I HN51_GATK_markdup.bam \
          -O HN51_GATK.g.vcf.gz \
          -R hg19.fa \
          -ERC GVCF \
```

GVCF Combination

The 3 VCF files created were then combined using GATK's "CombineGVCFs", forming one cohort GVCF file.

```
java -jar gatk.jar CombineGVCFs \
  -O combine.g.vcf.gz \
  -V HN51_GATK.g.vcf.gz \
  -V HN60_GATK.g.vcf.gz \
  -V HN72_GATK.g.vcf.gz \
  -R hg19.fa
```

GVCF Genotyping

In the final step, the single cohort GVCF file was genotyped using GATK's "GenotypeGVCFs", based off of the genotype likelihoods generated by HaplotypeCaller's GVCF mode.

```
java -jar gatk.jar GenotypeGVCFs \
-O combo.vcf \
-R hg19.fa \
-V combine.g.vcf.gz
```

An excerpt of variants from the output GATK VCF file is shown below. Note that some fields have been removed for readability.

CHROM	POS	ALT	QUAL	FORMAT
chr1	10403	A	24.88	GT:AD:DP:GQ:PGT:PID:PL
chr1	10415	A	193.86	GT:AD:DP:GQ:PGT:PID:PL
chr1	10616	C	191.97	GT:AD:DP:GQ:PL
chr1	12783	A	555.26	GT:AD:DP:GQ:PL
chr1	13012	A	29.14	GT:AD:DP:GQ:PL
chr1	13116	G	1472.92	GT:AD:DP:GQ:PGT:PID:PL
chr1	13118	G	1472.92	GT:AD:DP:GQ:PGT:PID:PL
chr1	13302	T	144.68	GT:AD:DP:GQ:PL

Shared Variants Across Patients

The VCF files output from Samtools and GATK were uploaded to Ensembl's Variant Effect Predictor.

GATK-Processed VCFs

Unfortunately, the extremely long computational time (> 6 days) for the GATK workflow stunted its results. Although a final VCF file was obtained, without the time to add processes such as adaptive variant filtering (such as VQSR) the VCF output proved to be much too large (~ 750 MB) to upload to the VEP browser, which is limited to uploads of less than 50 MB. Thus, it was not possible to compare the variants obtained from the GATK workflow to those of the Samtools workflow.

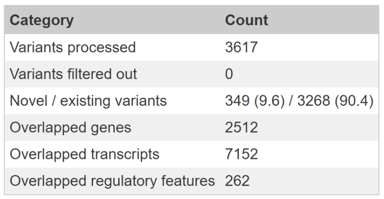
Samtools-Processed VCFs

For Samtools' output, rather than upload the 3 individual VCF files to VEP, a VCF file was produced containing only variants shared by all 3 samples using the following commands, in which each VCF file is first zipped and then indexed using Tabix. The zipped and indexed VCF files were then run through BCFtools "isec" tool to produce VCF files containing only the shared variants between all 3 samples, reducing the total ~200,000 variants down to 3,617.

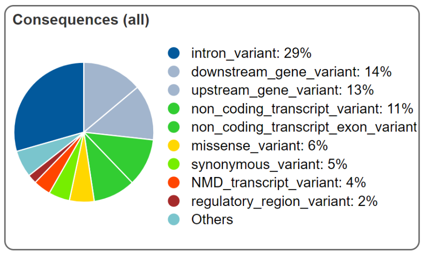
```
bgzip -c > HN51_filtered.vcf.gz && tabix HN51_filtered.vcf.gz
bgzip -c > HN60_filtered.vcf.gz && tabix HN60_filtered.vcf.gz
bgzip -c > HN72_filtered.vcf.gz && tabix HN72_filtered.vcf.gz

bcftools isec -n =3 -p shared_variants HN51_filtered.vcf.gz HN60_filtered.vcf.gz HN72_filtered.vcf.gz
```

VEP of Samtools Variants

The resultant shared-variants VCF was then uploaded to VEP to identify which variants were present in what regions, yielding the following summary statistics:



Category	Count
Variants processed	3617
Variants filtered out	0
Novel / existing variants	349 (9.6) / 3268 (90.4)
Overlapped genes	2512
Overlapped transcripts	7152
Overlapped regulatory features	262



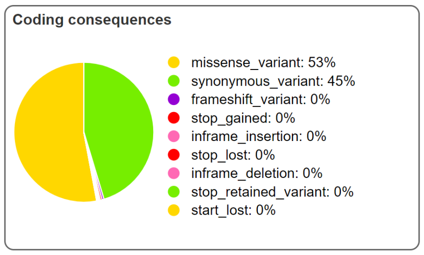


Figure 3: VEP Summary Statistics

To narrow the scope of the variants found, these 3,617 variants were then filtered on VEP to show only variants predicted to be "high impact". These were further filtered to show only one consequence per variant allele, yielding a succinct list of all genes containing these high impact variants. Table 2 on the following page shows a summary of the VEP output of shared, high-impact variants across all 3 samples.

Table 3: VEP High-Impact Shared Variants

Location	SYMBOL	BIOTYPE	Consequence
10:11356093-11356096	CELF2	protein_coding	splice_acceptor_variant,intron_variant
11:48347014-48347014	OR4C3	protein_coding	stop_gained
12:11420454-11420454	PRB3	protein_coding	splice_acceptor_variant
15:20740252-20740252	GOLGA6L6	protein_coding	stop_gained
15:78209436-78209436	RP11-114H24.2	transcribed_unprocessed_pseudogene	splice_acceptor_variant,non_coding_transcript_variant
17:20769899-20769899	CCDC144NL	protein_coding	stop_gained
17:25753587-25753587	TBC1D3P5	processed_transcript	splice_donor_variant,non_coding_transcript_variant
17:45234406-45234407	CDC27	protein_coding	frameshift_variant
19:20807177-20807178	ZNF626	protein_coding	frameshift_variant
19:42092815-42092815	CEACAM21	protein_coding	splice_acceptor_variant
19:43439694-43439694	PSG7	polymorphic_pseudogene	stop_lost
19:58003580-58003580	ZNF419	protein_coding	splice_donor_variant
22:23114327-23114327	IGLV3-12	IG_V_gene	start_lost
6:57467100-57467100	PRIM2	protein_coding	stop_lost
6:132859609-132859609	TAAR9	polymorphic_pseudogene	stop_lost
7:64438667-64438667	ZNF117	protein_coding	stop_gained
7:100552738-100552738	MUC3A	protein_coding	stop_gained
7:142231625-142231625	TRBV10-1	TR_V_gene	stop_gained
9:68455157-68455157	RP11-764K9.4	unprocessed_pseudogene	splice_donor_variant,non_coding_transcript_variant

Analysis of Gene Variants

The 19 genes containing shared high-impact variants were then investigated to observe any possible connections to cancer risk. Of the 19, several stood out as possible candidates:

- Three of the variant genes shown were zinc-finger proteins (ZNF117, ZNF419, and ZNF626), which are involved in gene expression regulation as DNA-binding proteins. Thus, variations in these proteins may lead to altered gene expression in these individuals. However, all three of these variants are known SNPs, and are currently not known to be associated with any phenotype.
- Another possible candidate that appeared was MUC3A. This gene codes for a major mucosal glycoprotein, thought to play a role in epithelial protection and lubrication. This variant contained a stop-gain mutation not associated with a known SNP, and could be hypothesized to be a low or non-functioning variant that results in higher epithelial cell vulnerability to environmental damage, possibly increasing risk of HNSCC cancers.
- CDC27 appeared as a frameshift variant. CDC27 is an important protein for cell-cycle timing, involved in anaphase as part of the APC complex and acting as part of mitotic checkpoints. CDC27 has been shown to be implicated in both cervical and breast cancers, and so may also have implication for other cancer development such as HNSCC, warranting further investigation. In addition, known somatic variants at the same position with CDC27 have also been associated with upper aerodigestive tract tumours, as well as skin and stomach tumours.

Scripts Used

QC, Trimming, and Samtools Procedure

```
#!/bin/bash

# SGE OPTIONS====================================

#$ -N VariantCalling
#$ -q all.q
#$ -cwd
#$ -v PATH
#$ -v LD_LIBRARY_PATH
#$ -v PYTHONPATH
#$ -S /bin/bash

# COMMANDS=======================================
# QC---------------------------------------------
mkdir QC_files
for fastq in samples/*.fastq
do
    fastqc -o QC_files "$fastq"
done
multiqc -o QC_files QC_files


# TRIMMING---------------------------------------
mkdir trimmed
mkdir unpaired
for file in samples/*P1*
do
        name=${file#*/}
    java -jar Trimmomatic-0.36/trimmomatic-0.36.jar PE -phred33 -trimlog trimlog.txt \
      "$file" "${file/P1/P2}" \
      trimmed/"$name" unpaired/"$name".unpaired \
      trimmed/"${name/P1/P2}" unpaired/"${name/P1/P2}".unpaired \
      ILLUMINACLIP:Trimmomatic-0.36/adapters/TruSeq3-PE.fa:2:30:10 \
      LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15 MINLEN:36
done


# QC 2-------------------------------------------
mkdir QC_files_2
for fastq_trim in trimmed/*.fastq
do
        fastqc -o QC_files_2 "$fastq_trim"
done
multiqc -o QC_files_2 QC_files_2


# INDEXING---------------------------------------
cd ref
bwa index -a bwtsw hg19.fa
```

```
samtools faidx hg19.fa
cd ../


# ALIGNMENT-------------------------------------
mkdir sams
for file in trimmed/*P1*
do
    name=${file#*/}
    samfile=${name/.lane*/.sam}
    id1=${name#*.}
    id2=${id1/lane_}
    id=${id2/_*}
    sm=${name:0:4}
    rg="@RG\tID:$id\tSM:$sm\tPL:ILLUMINA\tLB:$sm"
    bwa mem -t 4 -R "$rg" ref/hg19.fa "$file" "${file/P1/P2}" > sams/"$samfile"
done


# SAM TO BAM-------------------------------------
mkdir bams
cd sams
for sam in *.sam
do
    bam=${sam/.sam/.bam}
    samtools view -Sb "$sam" > ../bams/"$bam"
done
cd ../


# SORT BY NAME-----------------------------------
mkdir sorted_bams
cd bams
for bam in *.bam
do
    sorted=${bam/.bam/_namesorted}
    samtools sort -n -m 10000000000 "$bam" ../sorted_bams/"$sorted"
done
cd ../


# FIXMATES---------------------------------------
mkdir fixmates_bams
cd sorted_bams
for bam in *namesorted.bam
do
       samtools fixmate "$bam" ../fixmates_bams/"${bam/.bam/_fixed.bam}"
done
cd ../


# MERGE 72---------------------------------------
cd fixmates_bams
```

```
samtools merge -n HN72_s2_normal.Merged_namesorted_fixed.bam \
  HN72_s2_normal.AC254KACXX_namesorted_fixed.bam \
  HN72_s2_normal.AH0LENADXX_namesorted_fixed.bam
mv HN72_s2_normal.AC254KACXX_namesorted_fixed.bam \
  .HN72_s2_normal.AC254KACXX_namesorted_fixed.bam
mv HN72_s2_normal.AH0LENADXX_namesorted_fixed.bam \
  .HN72_s2_normal.AH0LENADXX_namesorted_fixed.bam
cd ../


# SORT BY CHR AND REMOVE DUPLICATES-------------
mkdir rmdup_bams
cd fixmates_bams
for bam in *namesorted_fixed.bam
do
    chr_sort="${bam/name/chr}"
    samtools sort -m 10000000000 "$bam" "${chr_sort/.bam/}"
    samtools rmdup "$chr_sort" ../rmdup_bams/"${chr_sort/.bam/_rmdup.bam}"
done
cd ../


# INDEX BAMS AND GENERATE MAPPING STATS---------
mkdir bcfs
cd rmdup_bams
for bam in *_rmdup.bam
do
    samtools index "$bam"
    stats=${bam/.bam/_mapstats.txt}
    samtools flagstat "$bam" > ../mapstats/"$stats"


# VARIANT CALLING-------------------------------
    bcf=${bam/_*/_var_raw.bcf}
    samtools mpileup -ugf ../ref/hg19.fa "$bam" | bcftools view -bvcg -> ../bcfs/"$bcf"
done
cd ../


# VARIANT FILTERING-----------------------------
mkdir vcfs
cd bcfs
for bcf in *.bcf
do
    vcf=${bcf/_raw.bcf/_filtered.vcf}
    bcftools view "$bcf" | vcfutils.pl varFilter -d 60 > ../vcfs/"$vcf"
done
```

GATK Procedure

The GATK scripts begins by using the trimmed BAM files output by the trimming procedure run in the previous bash script. Note that most commands were run using the TMP_DIR=./ and MAX_RECORDS_IN_RAM=2000000 options, due to the large size and amount of temporary files created that would fill the system's default tmp directory during any kind of sorting process.

```
#!/bin/bash

# SGE OPTIONS=================================

#$ -N GATKVariantCalling
#$ -q all.q
#$ -cwd
#$ -v PATH
#$ -v LD_LIBRARY_PATH
#$ -v PYTHONPATH
#$ -S /bin/bash

# COMMANDS===================================
# ALIGNMENT--------------------------------
mkdir sams_GATK
for file in ../trimmed/*P1*
do
    name=${file##*/}
    samfile=${name/.lane*/_GATK.sam}
    id1=${name#*.}
    id2=${id1/lane_}
    id=${id2/_*}
    sm=${name:0:4}
    rg="@RG\tID:$id\tSM:$sm\tPL:ILLUMINA\tLB:$sm"
    bwa mem -t 4 -R "$rg" -M ../ref/hg19.fa "$file" "${file/P1/P2}" > sams_GATK/"$samfile"
done


# SORT BY NAME-----------------------------
mkdir name_sorted_bams_GATK
cd sams_GATK
for sam in *.sam
do
    java -jar ../picard.jar SortSam TMP_DIR=./ MAX_RECORDS_IN_RAM=2000000 \
      I="$sam" \
      O=../name_sorted_bams_GATK/"${sam/.sam/_namesorted.bam}" \
      SORT_ORDER=queryname
done
cd ../


# FIXMATES----------------------------------
mkdir fixmate_bams_GATK
cd name_sorted_bams_GATK
for file in *namesorted.bam
do
    java -jar ../picard.jar FixMateInformation TMP_DIR=./ MAX_RECORDS_IN_RAM=2000000 \
      I="${file}" \
      O=../fixmate_bams_GATK/"${file/namesorted.bam/fixed.bam}" \
      SORT_ORDER=coordinate
done
cd ../


# MERGE HN72--------------------------------
cd fixmate_bams_GATK
java -jar ../picard.jar MergeSamFiles TMP_DIR=./ MAX_RECORDS_IN_RAM=2000000 \
  I=HN72_s2_normal.AC254KACXX_GATK_fixed.bam \
  I=HN72_s2_normal.AH0LENADXX_GATK_fixed.bam \
  O=HN72_s2_normal.MERGED_GATK_fixed.bam
mv HN72_s2_normal.AC254KACXX_GATK_fixed.bam .HN72_s2_normal.AC254KACXX_GATK_fixed.bam
mv HN72_s2_normal.AH0LENADXX_GATK_fixed.bam .HN72_s2_normal.AH0LENADXX_GATK_fixed.bam
cd ../


# MARK DUPLICATES---------------------------------
mkdir markdup_bams_GATK
cd fixmate_bams_GATK
for file in *fixed.bam
do
    java -jar ../picard.jar MarkDuplicates TMP_DIR=./ MAX_RECORDS_IN_RAM=2000000 \
      I="$file" \
      O=../markdup_bams_GATK/"${file/_*.bam/_GATKmarkdup.bam}" \
      M=../markdup_bams_GATK/"${file/_*.bam/_GATKmarkdup_metrics.txt}"
done
cd ../


# REFERENCE DICTIONARY CREATION----------------
java -jar picard.jar CreateSequenceDictionary R=../ref/hg19.fa O=../ref/hg19.dict


# VARIANT CALLING---------------------------------
mkdir gvcfs_GATK
cd markdup_bams_GATK

for file in *markdup.bam
do
        java -jar ../picard.jar BuildBamIndex I="$file"
        name="${file:0:4}"
        java -jar ../gatk.jar HaplotypeCaller --TMP_DIR ./ \
          -I "$file" \
          -O ../gvcfs_GATK/"$name".g.vcf.gz \
          -R ../../ref/hg19.fa \
          -ERC GVCF \
          -bamout "$name"_bamout.bam
done

cd ../gvcfs_GATK/
```

```
java -jar ../gatk.jar CombineGVCFs --TMP_DIR ./ \
  -O combine.g.vcf.gz \
  -V HN51.g.vcf.gz \
  -V HN60.g.vcf.gz \
  -V HN72.g.vcf.gz \
  -R ../../ref/hg19.fa
java -jar ../gatk.jar GenotypeGVCFs --TMP_DIR ./ \
-O combo.vcf \
-R ../../ref/hg19.fa \
-V combine.g.vcf.gz
```