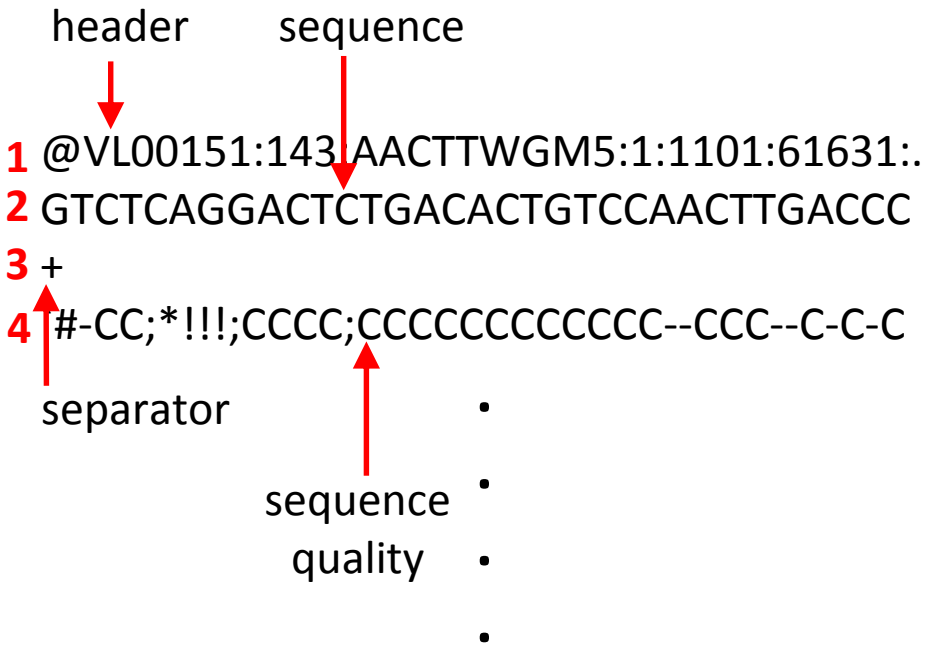
# Pipeline details

Vishram L. Terse, Ph.D.
Scientific Officer D,
Hematopathology department, ACTREC
26th September, 2024

# Fastq is a commonly used format to store the raw sequencing reads




- Fastq, similar to fasta has sequence info. on the 1st line
- Sequence + base quality encoded in ASCII characters

Phred Quality (Q) = - 10 * $\log_{10}$ *p*

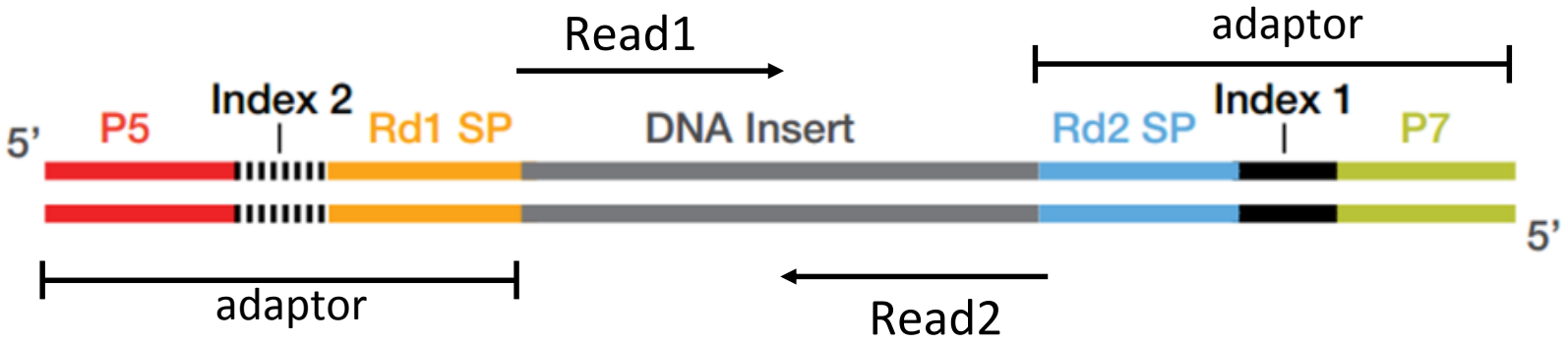
Score

*p* = probability that the base call is incorrect

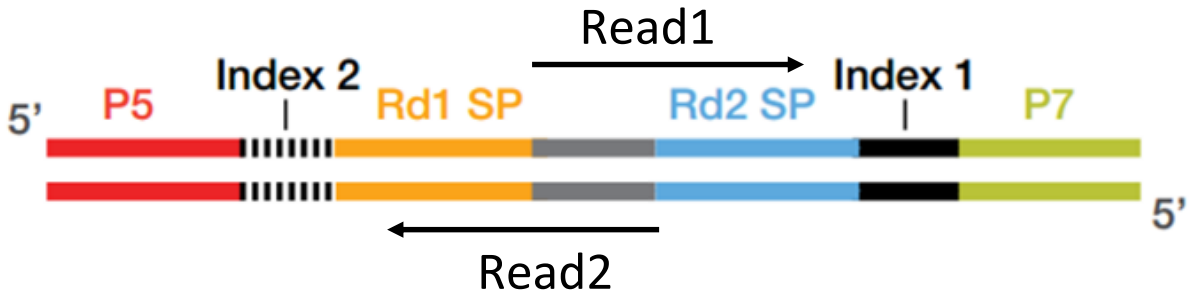
| Phred Quality Score | probability of incorrect base call | Base call accuracy (%) |
|---|---|---|
| 10 | 1 in 10 | 90 |
| 20 | 1 in 100 | 99 |
| 30 | 1 in 1000 | 99.9 |
| 40 | 1 in 10000 | 99.99 |

https://www.ncbi.nlm.nih.gov/sra/docs/submitformats/

# Insert sizes smaller than read length results in adaptor read-through



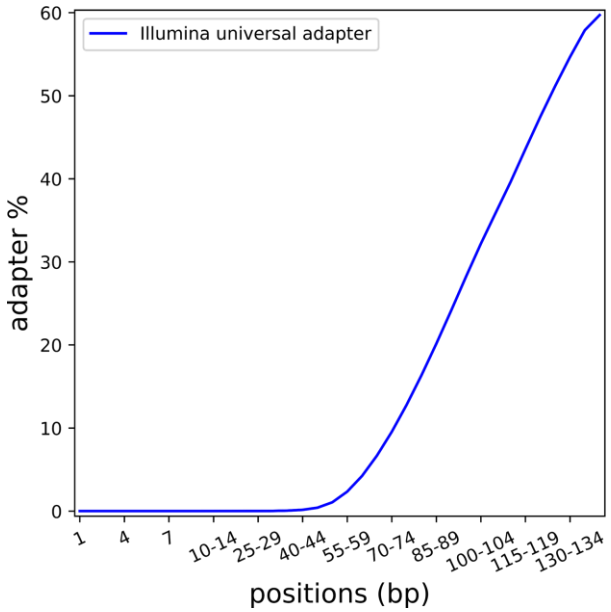

'adaptor read-through'






https://knowledge.illumina.com/library-preparation/general/library-preparation-general-reference_material-

# FastQC provides a fast and efficient way to determine read quality

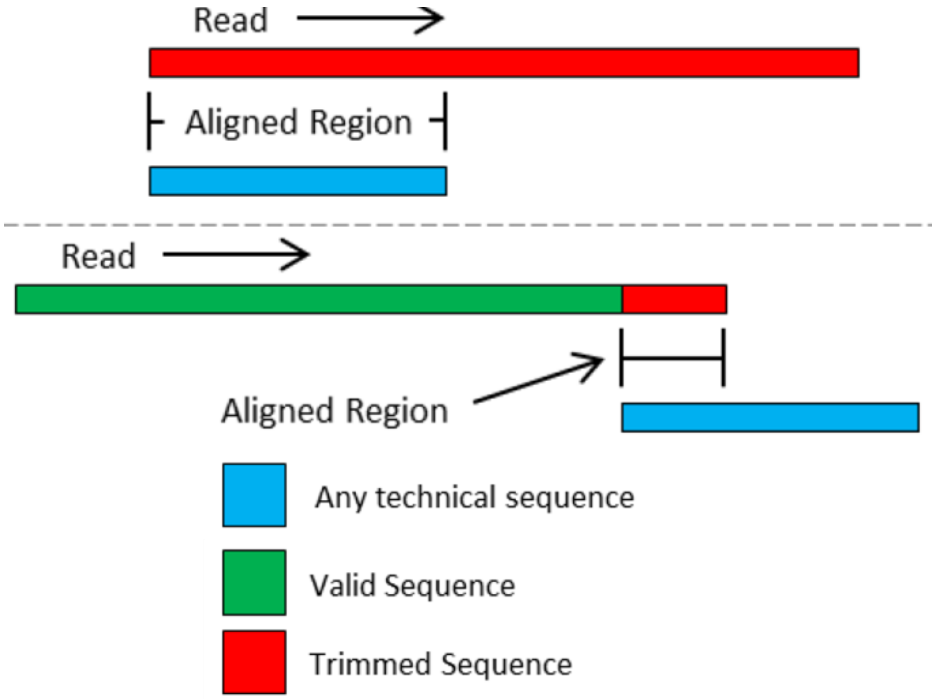




- FastQC (NGS QC, CANGS …), freely available software to perform checks on NGS data
- Problems can result from the sequencing or the input library.
- No. of reads , Average quality / read , Sequence length distribution.
- Results can help modify the experiment / steps in the pipeline.

https://knowledge.illumina.com/software/general/software-general-reference_material-list/000002211, Andrews S. (2010). FastQC, 2012. PLoS ONE 7(2): e30619

# Adapter clipping and quality filtering improves the efficiency of pipeline

- **Trimmomatic**:
  PE : paired end




- ILLUMINACLIP:<fastaWithAdaptersEtc>:<seed mismatches>:<palindrome clip threshold>:<simpleclip threshold>:<minAdapterLength>:<keepBothReads> LEADING:3 SLIDINGWINDOW:4:15 MINLEN:40

1) LEADING: Cut bases off the start of a read, if below a threshold quality
2) SLIDINGWINDOW:<windowSize>:<requiredQuality>
3) MINLEN: Drop the read if it is below a specified length



2014, Bioinformatics. 30(15):2114

# Pair assembly

## Flash:

- flash <trimmed_R1_fastq> <trimmed_R2_Fastq> --cap-mismatch-quals -O -M 250 -o <pair_assembled_fastq>

1) **--cap-mismatch-quals**: Cap quality scores assigned at mismatch locations to 2.
2) **-O**: --allow-outies - Also try combining read pairs in the "outie" orientation,

   e.g, Read 1: <----------

   Read 2: ---------->

   as opposed to,

   Read 1: <----------

   Read 2: ---------->

1) **-M**: –max-overlap - Maximum overlap length expected in approximately 90% of read pairs.
1) **-o**: Prefix of output files

# Mapping

## bwa mem:

- bwa mem -R "@RG\tID:AML\tPL:ILLUMINA\tLB:LIB-MIPS\tSM:24NGS457-B1\tPI:200" -M -t 20 <genomic_fasta>  <pair_assembled_fastq> > <output_sam_file>

1) **-R**: read group header line

   **RG:** read group

   **ID:** read group identifier. (AML)

   **PL:** platform. (ILLUMINA)

   **LB:** DNA preparation library identifier. (LIB-MIPS)

   **SM:** Sample (24NGS457-B1)

   **PI:** Predicted median insert size. (200)

1) **-M**: mark shorter split hits as secondary (In cases where BWA finds multiple possible alignments for a read, it selects the best alignment and marks it as the primary alignment. Shorter split hits i.e. alignments are considered secondary.
2) **-t**: number of threads (20)

# Sam conversion

**samtools：**

- samtools view -b <output_samfile> > <bamfile>
- samtools sort <bamfile> > <sorted_bamfile>
- samtools index <sorted_bamfile> > <sorted_bam_indexfile>

1) **view**: conversion between SAM <-> BAM <-> CRAM formats
2) **sort**: sort the alignment file based on coordinates
3) **index**: index coordinate-sorted SAM, BAM or CRAM files for fast random access

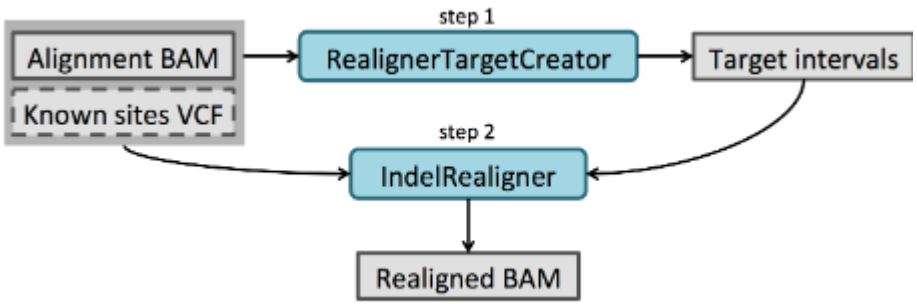
# GATK Best Practices

## RealignerTargetCreator:

- java -Xmx8G -jar <GATK38_path> -T RealignerTargetCreator -R <genomic_fasta> -nt 10 -I <sorted_bamfile> --known <site1> -o <intervals>

RealignerTargetCreator identifies regions in the genome where realignment is needed (called "targets") by finding areas where reads show mismatches, often near indels.

1) **-T**: Analysis type (RealignerTargetCreator)
2) **-R**: Reference genome
3) **-nt**: Number of threads
4) **-I**: Input BAM file
5) **–known**: Known variants file (site1 = in our case, Mills and 1000G Gold Standard Indels: A curated set of high-confidence indels commonly used for realignment.)
6) **-o**: Output Target intervals

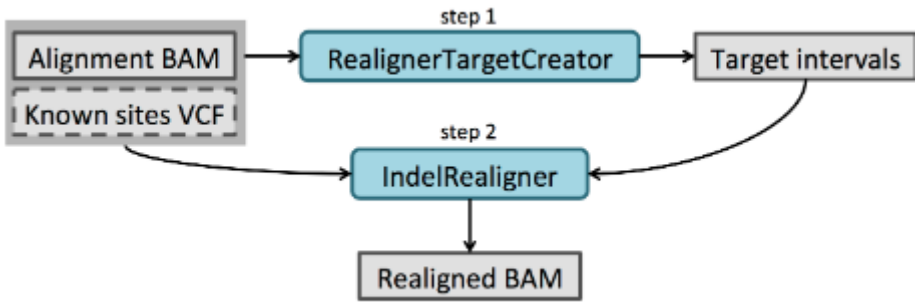


# GATK Best Practices

## IndelRealigner:

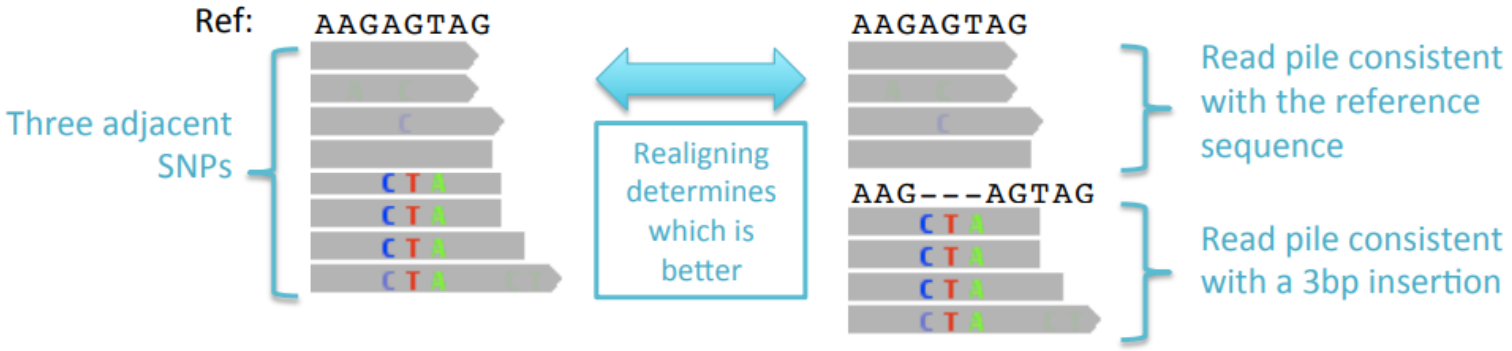
- java -Xmx8G -jar <GATK38_path> -T IndelRealigner -R <genomic_fasta> -I <sorted_bamfile> -known <site1> --targetIntervals <intervals> -o <realigned_bamfile>

1) **-T**: Analysis type (IndelRealigner)
2) **-R**: Reference genome
3) **-I**: Input BAM file
4) **–known**: Known variants file (Mills and 1000G Gold Standard Indels)
5) **--targetIntervals**: Output Target intervals from RealignerTargetCreator
6) **-o**: Output realigned BAM file



# How does the realignment algorithm work?

1. Find the best alternate consensus sequence that, together with the reference, best fits the reads in a pile (maximum of 1 indel)



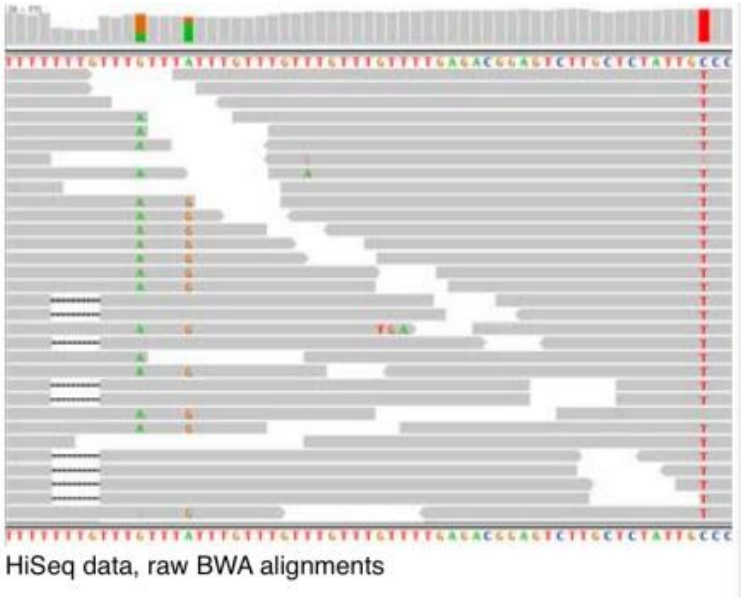

2. Score for alternate consensus = total sum of quality scores of mismatching bases

3. If best alternate consensus is sufficiently better than the original alignments (using LOD score threshold) -> accept proposed realignment

# GATK Best Practices

**RealignerTargetCreator**

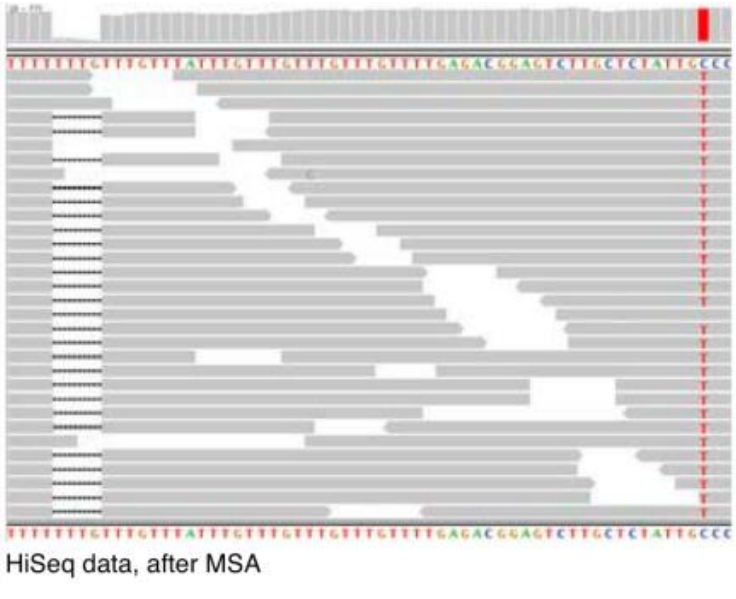
(Identify what regions need to be realigned)



HiSeq data, raw BWA alignments

**IndelRealigner**

Perform the actual realignment



HiSeq data, after MSA

# GATK Best Practices

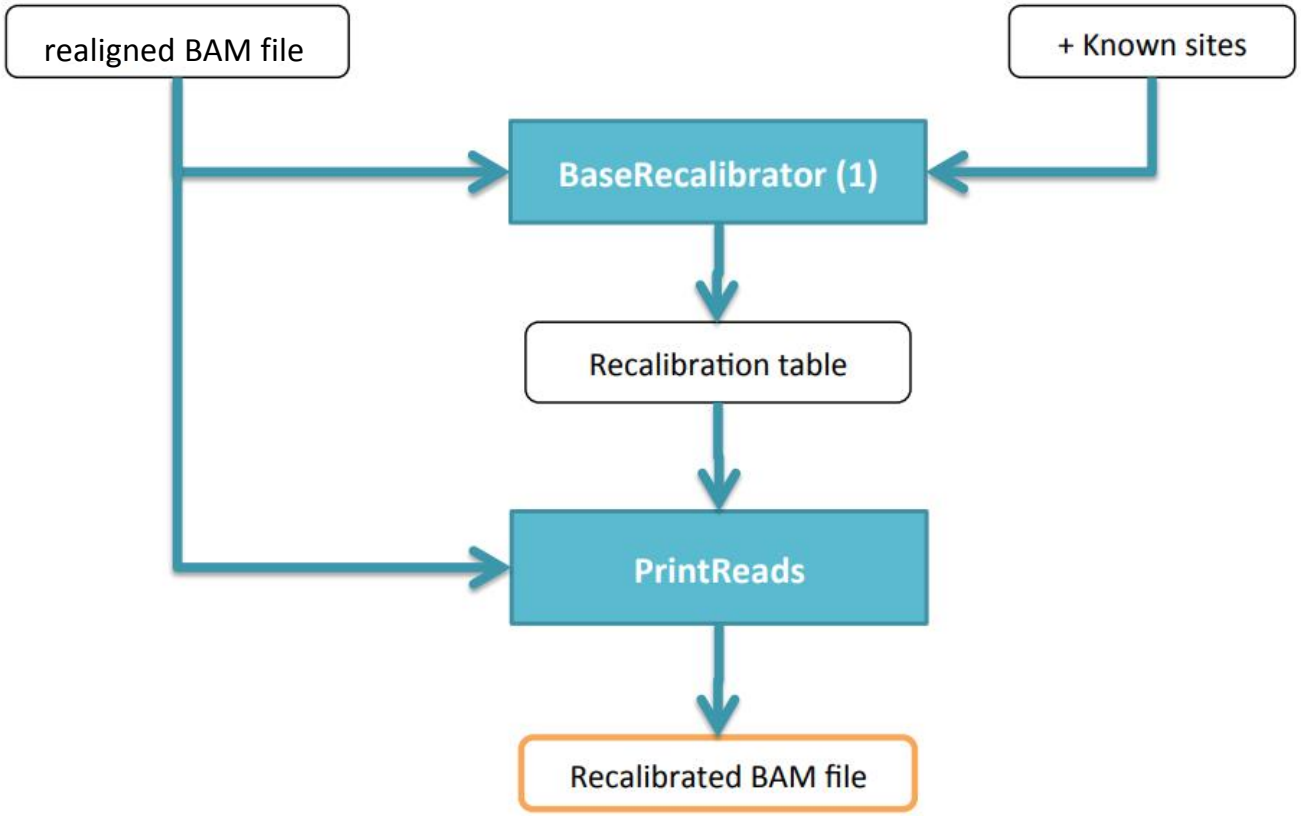
## BaseRecalibrator:

- java -Xmx8G -jar <GATK38_path> -T BaseRecalibrator -R <genomic_fasta> -I <realigned_bamfile> -knownSites <site2> -knownSites <site3> -maxCycle 600 -o <recalibration_table>

It improves the accuracy of the base quality scores assigned by the sequencing machine.

1) **-T**: Analysis type (BaseRecalibrator) **-R**: Reference genome **-I**: Input BAM file (realigned_bamfile)
2) **–knownSites**: One or more databases of known polymorphic sites used to exclude regions around known polymorphisms from analysis. We use
   a) dbSNP: A database of known single nucleotide polymorphisms (SNPs).
   b) 1000 Genomes Project: A comprehensive dataset of human genetic variants.
1) **--maxCycle**: The maximum cycle value permitted for the Cycle covariate
2) **-o**: Output recalibration table (a table of the several covariate values, num observations, num mismatches, empirical quality score).

# GATK Best Practices



# GATK Best Practices

## PrintReads:

- java -Xmx8G -jar <GATK38_path> -T PrintReads -R <genomic_fasta> -I <realigned_bamfile> --BQSR <recalibration_table> -o <final_bam>

After recalibrating base quality scores using the BaseRecalibrator tool, the PrintReads command applies the recalibration to the input BAM file. The recalibrated quality scores are written to a new BAM file, which can be used in downstream analysis like variant calling

1) **-T**: Analysis type (PrintReads)
2) **-R**: Reference genome
3) **-I**: Input BAM file (realigned_bamfile)
4) **–BQSR**: Input Base quality score recalibration table
5) **-o**: Output bam file (final_bam)

# Coverage

## bedtools：

- bedtools bamtobed -i <final_bam> > <final_bed>
- bedtools coverage -counts -a <bedfile> -b <final_bed> > <counts_bed>

1) **bamtobed**: Convert BAM alignments to BED (& other) formats
2) **coverage**: Compute the coverage over defined intervals
   a) **counts**: Only report the count of overlaps, don't compute fraction.
   b) Returns the depth and breadth of coverage of features from **-b** on the intervals in **-a**.

# Variant calling

## MuTect2：

- java -Xmx10G -jar <GATK38_path> -T MuTect2 -R <genomic_fasta> -I:<final_bam> -o <vcf_file> -L <bedfile>

1) **-T**: Analysis type (MuTect2)
2) **-R**: Reference genome
3) **-I**: Input BAM file (final_bam)
4) **-o**: File to which variants should be written (vcf_file)
5) **-L**: One or more genomic intervals over which to operate

# Variant annotation

## ANNOVAR：

- convert2annovar.pl -format vcf4 <vcf_file> --outfile <avinput> --withzyg --includeinfo

  (convert variant call file generated from various software programs into ANNOVAR input format)
- table_annovar.pl <avinput> --out output/24NGS775-B1_final --remove --protocol refGene,cosmic84,exac03 --operation g,f,f --buildver hg19 --nastring '-1' --otherinfo --csvout ${database}

1) a) **-format**: Input format (vcf4)  b) **-outfile**: ANNOVAR input (avinput)

   c) **-withzyg**: print zygosity/coverage/quality

   d) **-includeinfo**:include supporting information in output

1) a) **-out**: output file name prefix  b)**--remove**: remove all temporary files

   c) **–protocol**:  database protocol  d) **--operation**: type of operation

   e) **--buildver**: genome build version

   f) **–nastring**: string to display when a score is not available

   g) **–otherinfo**: print out otherinfo  h) **–csvout**: generate comma-delimited CSV file

# Format output

**Custom script：**

- python3 <formatMutect_script_path> <annovar_output_csv> <sample_name> <outdir>

This gathers following data from the annotated vcf into a csv file.

1) Chr, Start, End, Ref, Alt
2) REF_COUNT, ALT_COUNT, VAF
3) Variant Site, Gene.refGene, GeneDetail.refGene, Variant Function, AAChange.refGene
4) cosmic84, ExAC_ALL, ExAC_AFR, ExAC_AMR, ExAC_EAS, ExAC_FIN, ExAC_NFE, ExAC_OTH, ExAC_SAS,
5) Otherinfo

# KDM mutation database

**Custom script:**

- python3 <KDMdb_script_path> <formatted_mutect_csv_file> <outdir> <sample>

This inserts following columns from the KDM mutation database into our csv file.

1) Mutation - Type of mutation
2) Genomic - genomic coordinates and ref, alt of the mutation
3) Protein - Ensembl ID of the protein and the change occurred in it due to mutation
4) Nucleotide - Ensembl ID of the transcript and the nucleotide changed.
5) Comment- Therapeutic options
6) PMID

# Acknowledgement


- Hematopathology department, ACTREC
- Organisers


Thank you !!