# How to Evaluate Roche Target Enrichment Data for Germline Variant Research

## 1. Overview

Analysis of Roche TE (AVENIO Edge or KAPA Target Enrichment) experimental data sequenced on an Illumina sequencing system is most frequently performed using a variety of publicly available, open-source analysis tools.

The usage examples described here have been used effectively in our hands. *Please note that publicly available, open-source software tools may change and that such change is not under the control of Roche. Therefore Roche does not warrant and cannot be held liable for the results obtained when using the third party tools described herein. Roche does not provide direct analysis support or service for these or any other third party tools. Please refer to the authors of each tool for support and documentation.*

The typical variant calling analysis workflow consists of sequencing read quality assessment, read filtering, mapping against the reference genome, duplicate removal, coverage statistic assessment, variant calling, and variant filtering. At most of these steps, a variety of tools can be utilized. This document shows how to use a selection of the available tools to perform Roche TE data analysis for germline variant applications, but other analysis workflows can also be used.

This document will enable readers with bioinformatics experience to understand the basic analysis workflow in use at Roche to assess capture performance. The reader should carefully consider additional options when deciding the most appropriate workflow for their research.

## 2. Solution

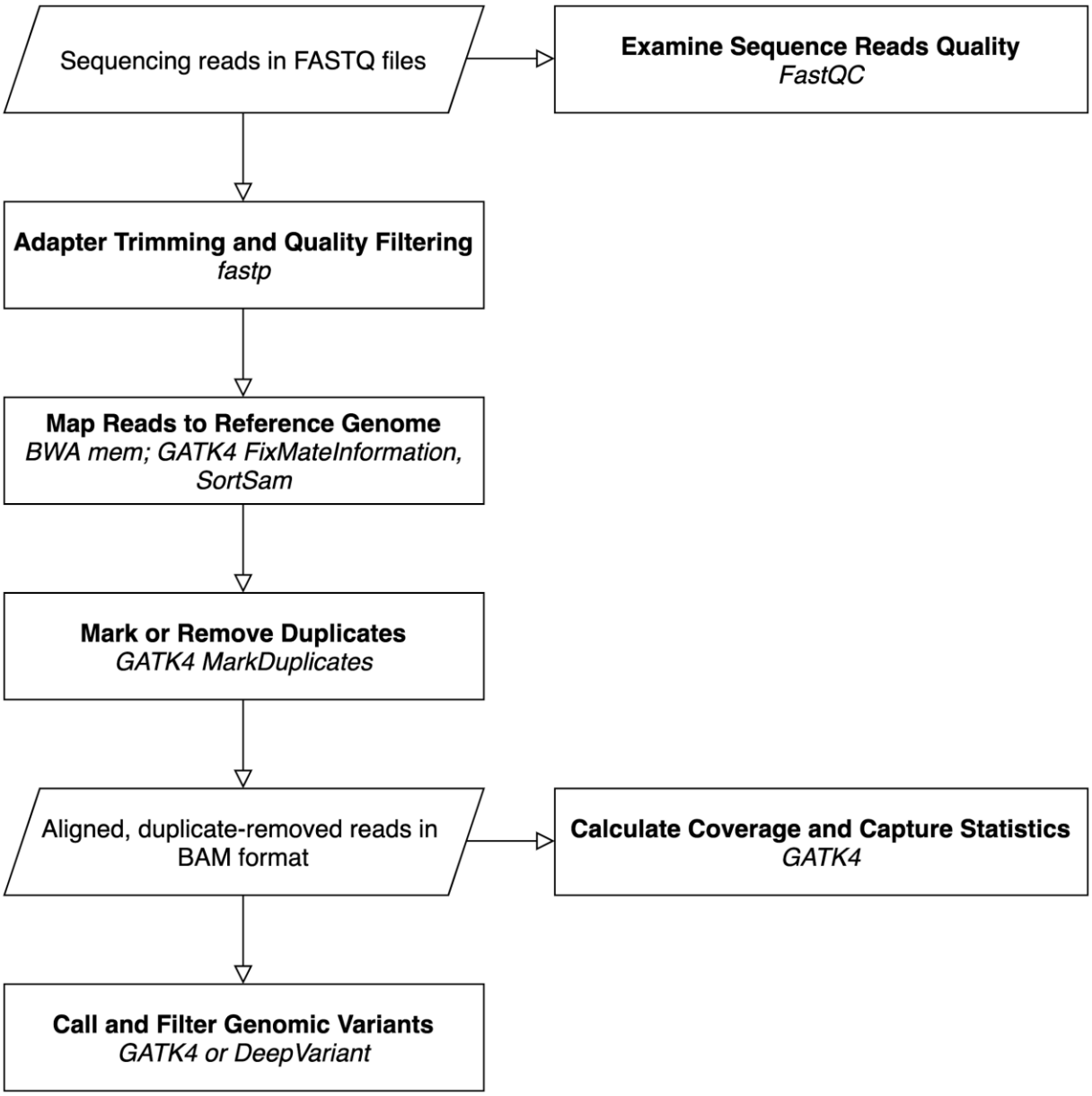


**Figure 1. Schematic of basic analysis workflow.**

Free and open source third-party tools are available for converting raw sequencing data into appropriate file formats, mapping reads to a reference sequence, evaluating sequencing quality, and analyzing variant calls. This white paper describes a number of steps and mini-workflows that use such third-party tools, which can be combined into a variety of data analysis workflows. Many of the tools and steps described here are based on GATK Best Practices with modifications appropriate for the Roche TE analysis workflow. See the GATK website to follow the complete GATK Best Practices (*https://software.broadinstitute.org/gatk/best-practices/*).

Ideally, you should develop a workflow appropriate for your experimental data using benchmark/control samples such as HapMap samples obtained from Coriell. Known variants may be downloaded from the 1000 Genomes

Project (*http://www.internationalgenome.org/*) or in special collections such as the GATK resource bundle (*https://console.cloud.google.com/storage/browser/genomics-public-data/resources/broad/hg38/v0*) and Genome in a Bottle (*https://github.com/genome-in-a-bottle*).

Note that where the text "SAMPLE" appears throughout examples shown here, you should replace it with a unique sample name. Similarly, replace "DESIGN" with the name of the target enrichment design that matches the design files supplied by Roche. "NumProcessors" should be replaced with the number of CPU cores available.

Replace "/path/to/..." in the examples with a valid path on your system. The current directory is assumed to be the location of all input files, and will also be the location of output files and report files. Some of the tools described in this document require execution of a .jar file by calling Java. One exception is GATK, which requires Java but is executed through a wrapper. If Java 1.8 is not the default version on your system, you will need to execute the GATK .jar file using a direct path to Java 1.8 instead.

Type the entire command shown for each step on a single line, despite the way it appears on the printed page. There should be no spaces within a file path, but there must be spaces before and after each option. Due to idiosyncrasies in most if not all PDF viewers, the underscores in command line examples in this document may not display properly at all zoom percentages. One way to confirm whether or not underscores are present is to print the page. Alternatively, try temporarily switching to a very high zoom percentage (*e.g.,* 400%).

Examples included in this document show how to perform the analysis with paired end Illumina sequencing reads. Many of the tools work with single end reads if paired end reads are not available, though input and output formats may vary. Note that using single end reads will artificially increase duplicate rate due to decreased ability to resolve a unique fragment from the library.

## Tools Overview

| Package (version) | Tool | Function as used in this document |
|---|---|---|
| **BWA** (0.7.17) | `index` | Generate an indexed genome from FASTA sequence. |
| | `mem` | Map sequencing reads to an indexed genome. |
| **FastQC** (0.11.9) | `Fastqc` | Assess sequencing read quality (per-base quality plot). |
| **GATK4*** (4.2.0.0) | `BedToIntervalList` | Convert BED file to Genomic Interval List format. |
| | `CollectHsMetrics` | Assess performance of a target enrichment experiment based on mapped reads. |
| | `CollectAlignmentSummaryMetrics` | Report mapping metrics for a BAM file. |
| | `CollectInsertSizeMetrics` | Estimate and plot insert size distribution. |
| | `CountReads` | Count the number of sequencing reads overlapping target regions. |
| | `CountVariants` | Count the number of SNP or Indel calls within target regions. |
| | `CreateSequenceDictionary` | Generate a sequence dictionary (.dict) for the reference genome. |
| | `FixMateInformation` | Clean up paired read information. |

| | | |
|---|---|---|
| | `MarkDuplicates` | Remove or mark duplicate reads and count the number of paired, unpaired, and optical duplicates. |
| | `SamFormatConverter` | Convert between SAM and BAM formats. |
| | `SortSam` | Sort a BAM file. |
| **Java** (≥ 1.8.0_282) | `java` | Required for GATK4. |
| **SAMtools** (1.12) | `faidx` | Generate a FASTA index of the reference genome. |
| | `flagstat` | Counts the number of alignments for each FLAG type in a BAM file. |
| **seqtk** (1.3-r106) | `sample` | Randomly subsample FASTQ file(s). |
| **fastp** (0.20.1) | `fastp` | Trim raw reads for quality and sequenced primer/adapter. |
| **BEDTools** (2.30.0) | `merge` | Combine overlapping or "book-ended" features in a BED file. |
| **DeepVariant** (0.9.0) | `deepvariant` | Call variants from a BAM file. |
| **hap.py** (v0.3.9) | `hap.py` | Compare SNP and indel calls against a gold standard dataset for the sample used. |

**Table 1: Third-party data analysis tools used in this white paper. The examples described in this document were tested using the software versions listed in parentheses, and different software versions may require different function calls and/or flags. See section *Reference Links* for installation instructions and explanations of command options. These tools were tested on a Redhat Linux system. *Many GATK4 tools were originally developed as part of Picard, which maintains detailed documentation referenced throughout this white paper.**

## Index a Reference Genome

Most Next-Generation Sequencing (NGS) mapping algorithms require an indexed genome to be created before mapping. Although algorithms work in different ways, most use the Burrows-Wheeler algorithm for mapping millions of relatively short reads against the reference genome. A genomic index is used to very quickly find the mapping location. Genomic indexing is required only once per genome version. The genomic index files that are created can then be used for all subsequent mapping jobs against that genome assembly version.

We recommend the FASTA formatted genome sequence be indexed with chromosomes in "karyotypic" sort order, *i.e.,* chr1, chr2, ..., chr10, chr11, ... chrX, chrY, chrM, *etc.* In these examples, reference genome files are referred to as "ref.fa", which should be replaced by the actual file name (*e.g.,* "hg38.fa").

| | |
|---|---|
| **Package ➔ Tool(s) Used** | BWA ➔ `index`<br>SAMtools ➔ `faidx`<br>GATK ➔ `CreateSequenceDictionary` |
| **Input(s)** | ref.fa |
| **Output(s)** | ref.fa {indexed}<br>ref.fa = unmodified reference genome<br>ref.fa.amb, ref.fa.ann, ref.fa.bwt, ref.fa.pac, ref.fa.sa = reference genome index files<br>ref.fa.fai = FASTA index<br>ref.dict = reference sequence dictionary |
| **Generate Reference Genome Index**<br>`/path/to/bwa index -a bwtsw /path/to/ref.fa` | |
| **Generate FASTA Index**<br>`/path/to/samtools faidx /path/to/ref.fa` | |
| **Generate Sequence Dictionary**<br>`/path/to/gatk CreateSequenceDictionary --REFERENCE /path/to/ref.fa` | |

The requirement for use of an indexed reference genome in a subsequent step is designated by "ref.fa {indexed}" in the Input(s) section. An indexed reference genome consists of the genome FASTA file and all index files present in the same directory.

Note that there can be multiple versions of the same reference genome available, and selecting the appropriate genome for your analyses is important. Ensure the reference genome build used to generate panel design files (such as primary target or capture target bed files) and used in the analysis pipeline are the same. Chromosome names should match in the two files, otherwise some 3rd party analysis tools will generate errors and fail. If the version of the reference genome contains extra chromosomes or contigs (i.e., ALT contigs in the human reference), consider if these are useful or necessary for your particular analysis. Some regions, such as the pseduo-autosomal regions in the human reference genome, can be represented in different ways that may affect downstream analyses including variant analysis.

## Decompress a FASTQ File

If the FASTQ files have been compressed (with a .gz extension), some tools require them to be decompressed before use.

| **Package ➔ Tool(s) Used** | `gunzip` |
|---|---|
| **Input(s)** | SAMPLE_R1.fastq.gz<br>SAMPLE_R2.fastq.gz |
| **Output(s)** | SAMPLE_R1.fastq<br>SAMPLE_R2.fastq |

```
gunzip -c SAMPLE_R1.fastq.gz > SAMPLE_R1.fastq


gunzip -c SAMPLE_R2.fastq.gz > SAMPLE_R2.fastq
```

## Examine Sequence Read Quality

Before spending time evaluating mapping statistics, use fastqc on raw reads and generate a per-base sequence quality plot and report to evaluate sequencing quality. The fastqc tool can work on both compressed and uncompressed FASTQ files.

| **Package ➔ Tool(s) Used** | FastQC ➔ `fastqc` |
|---|---|
| **Input(s)** | SAMPLE_R1.fastq / SAMPLE_R1.fastq.gz<br>SAMPLE_R2.fastq / SAMPLE_R2.fastq.gz |
| **Output(s)** | SAMPLE_R1_fastqc.zip<br>SAMPLE_R2_fastqc.zip |

```
/path/to/fastqc --nogroup --extract SAMPLE_R1.fastq(.gz) SAMPLE_R2.fastq(.gz)
```

FastQC has a --threads option that allows users to specify the number of files, which can be processed simultaneously. FastQC describes, “Each thread will be allocated 250MB of memory so you shouldn't run more threads than your available memory will cope with, and not more than 6 threads on a 32 bit machine”. In the above example, users can specify `--threads 2` to speed up the calculation. A .zip file is created for each SAMPLE input file. An HTML report named fastqc_report.html is created that is viewable in an internet browser. The authors of FastQC have posted the following examples of the QC report for a good and a bad sequencing run:
*http://www.bioinformatics.babraham.ac.uk/projects/fastqc/good_sequence_short_fastqc.html*
*http://www.bioinformatics.babraham.ac.uk/projects/fastqc/bad_sequence_fastqc.html*

### Select a Subsample of Reads from a FASTQ File

Random subsampling is useful for normalizing the number of reads per set when doing comparisons. With paired end reads, it is important that the two files use the same values for the seed `(-s)` and number of reads. The seqtk application can optionally sample from gzipped FASTQ files, but will write the sampled reads to uncompressed FASTQ files which can be further compressed using gzip command

| **Package ➔ Tool(s) Used** | Seqtk ➔ `sample` |
|---|---|
| **Input(s)** | SAMPLE_R1.fastq / SAMPLE_R1.fastq.gz<br>SAMPLE_R2.fastq / SAMPLE_R2.fastq.gz |
| **Output(s)** | SAMPLE_R1_subset.fastq.gz<br>SAMPLE_R2_subset.fastq.gz |

```
/path/to/seqtk sample -s 12345 SAMPLE_R1.fastq(.gz) 30000000 | gzip - >
SAMPLE_R1_subset.fastq.gz


/path/to/seqtk sample -s 12345 SAMPLE_R2.fastq(.gz) 30000000 | gzip - >
SAMPLE_R2_subset.fastq.gz
```

The commands above will randomly subsample 30 million matched read pairs from the paired end FASTQ files for a total of 60 million reads. Supplying the same random seed value (`-s`) ensures that the FASTQ records will remain in synchronized sort order and can be used for mapping, *etc*. Note that seqtk requires an amount of RAM proportional to the number of reads being subsampled. As you increase the size of the subsampled read set, more RAM is needed.

## Perform Adapter Trimming and Quality Filtering

Before mapping reads to the reference genome, it is advisable to trim off any sequencing adapters found on the reads, and to filter or trim the reads for sequence quality. The fastp application can do both of those steps. Adapter trimming is enabled by default, and adapter sequences are automatically detected. If using subsampled reads, the subsampled FASTQ files should be used as input here.

| **Package ➔ Tool(s) Used** | fastp |
|---|---|
| **Input(s)** | SAMPLE_R1.fastq.gz (shown below) or SAMPLE_R1_subset.fastq.gz if subsampling<br><br>SAMPLE_R2.fastq.gz (shown below) or SAMPLE_R2_subset.fastq.gz if subsampling |
| **Output(s)** | SAMPLE_R1_trimmed.fastq.gz<br><br>SAMPLE_R2_trimmed.fastq.gz<br><br>SAMPLE_fastp.log<br><br>SAMPLE_fastp.html |

**Trim Reads**

```
/path/to/fastp -i SAMPLE_R1.fastq.gz -I SAMPLE_R2.fastq.gz -o
SAMPLE_R1_trimmed.fastq.gz -O SAMPLE_R2_trimmed.fastq.gz -q 20 -u 20 -3 -W 5 -x -g -
l 50 -c -h SAMPLE_fastp.html -w NumProcessors &> SAMPLE_fastp.log
```

The `-3` and `-W 5` options allow trimming from the 3' tail in a sliding window of 5 bp. If the mean quality is below the quality set by `-q`, the bases are trimmed. In addition, -u specifies what percent of bases are allowed to be unqualified before a read is discarded. The `-x` and `-g` options turn on poly X and poly G tail trimming, respectively. The -l option means the length of the trimmed read must be at least 50 bp. The -c option turns on base correction for read pairs where read1 and read2 overlap.

The fastp application will produce two files. The SAMPLE_R1_trimmed.fastq.gz and SAMPLE_R2_trimmed.fastq.gz contain the reads that are still paired after adapter trimming and quality filtering. Unpaired reads can optionally be assigned to output files using the `--unpaired1` and `--unpaired2` options. If you want to increase the percentage of passing reads, the quality and length filters thresholds can be lowered.

For some applications comparing equal numbers of high quality subsampled reads rather than reads of all quality may be preferred (e.g., to reduce sequencing quality bias between compared data sets). In these cases, the order of read subsampling and trimming may be swapped.

## Map Reads to the Reference Genome

Reads are mapped to the indexed reference genome using BWA, and the output is converted to a BAM file with GATK `SamFormatConverter`. GATK `FixMateInformation` ensures consistent information appears for both reads in a pair. GATK `SortSam` is then used to order the output file according to genomic sort order.

| | |
|---|---|
| **Package ➔ Tool(s) Used** | BWA ➔ `mem`<br>GATK ➔ `SamFormatConverter`<br>GATK ➔ `FixMateInformation`<br>GATK ➔ `SortSam` |
| **Input(s)** | ref.fa {indexed}<br>SAMPLE_R1_trimmed.fastq.gz<br>SAMPLE_R2_trimmed.fastq.gz |
| **Output(s)** | SAMPLE_sorted.bam |

**Map Reads and Convert to BAM**

```
/path/to/bwa mem -R "@RG\tID:1\tDS:KAPA_TE\tPL:ILLUMINA\tLB:SAMPLE\tSM:SAMPLE"
/path/to/ref.fa -t NumProcessors -M SAMPLE_R1_trimmed.fastq SAMPLE_R2_trimmed.fastq |
\

/path/to/gatk SamFormatConverter -I /dev/stdin -O SAMPLE_bwa.bam
```

**Clean Up Paired Read Information**

```
/path/to/gatk FixMateInformation -I SAMPLE_bwa.bam -O SAMPLE_fixmate.bam
```

**Sort BAM File**

```
/path/to/gatk SortSam -I SAMPLE_fixmate.bam -O SAMPLE_sorted.bam -SO coordinate --
CREATE_INDEX true
```

In the “Map Reads” step, the -R option defines the read group (“@RG”), which will appear in the BAM header. Within this string is the sample ID (“ID”), description field (“DS”), sequencing platform (“PL”), library name (“LB”), and sample name (“SM”). When a library name, ID and sample name do not separately exist, a SAMPLE descriptor may be used, as shown in the example above. BWA mem output is directly piped into GATK SamFormatConverter to avoid writing and reading the large “.sam” file.

### Basic Mapping Metrics

Basic mapping metrics can be calculated using GATK `CollectAlignmentSummaryMetrics`.

| **Package ➔ Tool(s) Used** | GATK ➔ `CollectAlignmentSummaryMetrics` |
|---|---|
| **Input(s)** | ref.fa {indexed}<br>SAMPLE_sorted.bam |
| **Output(s)** | SAMPLE_alignment_metrics.txt |

```
/path/to/gatk CollectAlignmentSummaryMetrics \
  --METRIC_ACCUMULATION_LEVEL ALL_READS \
  --INPUT SAMPLE_sorted.bam \
  --OUTPUT SAMPLE_alignment_metrics.txt \
  --REFERENCE_SEQUENCE /path/to/ref.fa \
  --VALIDATION_STRINGENCY LENIENT
```

See *https://broadinstitute.github.io/picard/picard-metric-definitions.html#AlignmentSummaryMetrics* for a description of the output metrics.

### Calculate Mapping Rates

Mapping rates (e.g., "% reads mapped" and "% paired reads mapped") can be calculated using SAMtools `flagstat`.

| **Package ➔ Tool(s) Used** | SAMtools ➔ `flagstat` |
|---|---|
| **Input(s)** | SAMPLE_sorted.bam |
| **Output(s)** | SAMPLE_flagstat.txt |

```
/path/to/samtools flagstat SAMPLE_sorted.bam > SAMPLE_flagstat.txt
```

See *http://www.htslib.org/doc/samtools-flagstat.html* for a description of the output metrics.

## Mark or Remove Duplicates

After mapping, GATK `MarkDuplicates` is used to remove or mark PCR duplicates. This is done to avoid allele amplification bias in variant calling.

| | |
|---|---|
| **Package ➔ Tool(s) Used** | GATK ➔ `MarkDuplicates` |
| **Input(s)** | SAMPLE_sorted.bam |
| **Output(s)** | SAMPLE_sorted_rmdups.bam<br>SAMPLE_sorted_rmdups.bai<br>SAMPLE_markduplicates_metrics.txt |

**Mark Duplicates**

```
/path/to/gatk MarkDuplicates \
  --VALIDATION_STRINGENCY LENIENT \
  -I SAMPLE_sorted.bam \
  -O SAMPLE_sorted_rmdups.bam \
  --METRICS_FILE SAMPLE_markduplicates_metrics.txt \
  --REMOVE_DUPLICATES true \
  --ASSUME_SORTED true \
  --CREATE_INDEX true
```

In the "Mark Duplicates" step, `--REMOVE_DUPLICATES` true means that duplicates are removed rather than marked. If you are certain that subsequent tools will appropriately understand the duplicate flag, you may alternatively use `--REMOVE_DUPLICATES` false (a duplicate flag is added for duplicate reads) to keep a record in the BAM file of which reads were considered duplicates. This may also be desirable if the BAM file will be a file retained for long term storage of the sequence data.

View the file SAMPLE_markduplicates_metrics.txt for counts of paired, unpaired, and duplicate reads. See *https://broadinstitute.github.io/picard/picard-metric-definitions.html#DuplicationMetrics* for a description of the output metrics. Note that "optical duplicates" are also reported, based on sequence similarity and sequencing cluster distance. Optical duplicates are a subset of the total duplicate rate and are counted within the paired and unpaired duplicates. For instruments employing patterned flow cells (e.g., HiSeq X and HiSeq 4000), `--OPTICAL_DUPLICATE_PIXEL_DISTANCE` should be changed from the default of 100 to 2500.

The sorted and duplicate-marked SAMPLE.bam file is an input for many of the examples below. A requirement for use of an indexed BAM file in a subsequent step is designated by "SAMPLE.bam {indexed}" in the Input(s) section. The BAM file is indexed if the SAMPLE.bam.bai index file is present in the same directory, which is generated from `MarkDuplicates` by specifying the `--CREATE_INDEX` true option.

## Estimate Insert Size Distribution

The DNA that goes into sequence capture is generated by random fragmentation and later size selected. It is normal to observe a range of fragment sizes, but if skewed too large or too small, the on-target rate and/or percent of bases covered with at least one read can be adversely affected. The size of these fragments can be estimated from paired end sequencing reads (will not work for single end reads).

| | |
|---|---|
| **Package ➔ Tool(s) Used** | GATK ➔ `CollectInsertSizeMetrics` |
| **Input(s)** | SAMPLE_sorted.bam<br>SAMPLE_sorted_rmdups.bam |
| **Output(s)** | SAMPLE_insert_size_metrics_sorted.txt<br>SAMPLE_insert_size_plot_sorted.pdf<br>SAMPLE_insert_size_metrics_sorted_rmdups.txt<br>SAMPLE_insert_size_plot_sorted_rmdups.pdf |

**Estimate Insert Size Before Duplicates Removal**

```
/path/to/gatk CollectInsertSizeMetrics \
  --VALIDATION_STRINGENCY LENIENT \
  -H SAMPLE_insert_size_plot_sorted.pdf \
  -I SAMPLE_sorted.bam \
  -O SAMPLE_insert_size_metrics_sorted.txt
```

**Estimate Insert Size After Duplicates Removal**

```
/path/to/gatk CollectInsertSizeMetrics \
  --VALIDATION_STRINGENCY LENIENT \
  -H SAMPLE_insert_size_plot_sorted_rmdups.pdf \
  -I SAMPLE_sorted_rmdups.bam \
  -O SAMPLE_insert_size_metrics_sorted_rmdups.txt
```

See *https://broadinstitute.github.io/picard/picard-metric-definitions.html#InsertSizeMetrics* for a description of output metrics included in SAMPLE_insert_size_metrics_sorted.txt for all reads and SAMPLE_insert_size_metrics_sorted_rmdups.txt for non-duplicate reads, which can also be used to plot the insert size distributions across samples. As long as R is installed on your system, a PDF plot is also created.

## Count On-Target Reads

Use GATK `CountReads` to calculate the number of reads that overlap a target BED file by at least 1 bp. Calculation of on-target reads is one measure of the success of a Roche TE experiment, though optimal on-target is design-specific. The on-target metric is affected by library insert size, hybridization and wash stringency, and laboratory protocol.

| **Package ➔ Tool(s) Used** | GATK ➔ `CountReads` |
|---|---|
| **Input(s)** | ref.fa {indexed}<br>SAMPLE_sorted.bam<br>SAMPLE_sorted_rmdups.bam<br>DESIGN_capture_targets.bed |
| **Output(s)** | ontarget_reads_sorted.txt<br>ontarget_reads_sorted_rmdups.txt |

**Count Reads Before Duplicates Removal**

```
/path/to/gatk CountReads \
  -R /path/to/ref.fa \
  -I SAMPLE_sorted.bam \
  -L DESIGN_capture_targets.bed \
  --read-filter MappedReadFilter \
  --read-filter NotSecondaryAlignmentReadFilter > ontarget_reads_sorted.txt
```

**Count Reads After Duplicates Removal**

```
/path/to/gatk CountReads \
  -R /path/to/ref.fa \
  -I SAMPLE_sorted_rmdups.bam \
  -L DESIGN_capture_targets.bed \
  --read-filter MappedReadFilter \
  --read-filter NotSecondaryAlignmentReadFilter > ontarget_reads_sorted_rmdups.txt
```

It is necessary to apply filters to include specific reads for analysis. In the example commands above, "MappedReadFilter" and "NotSecondaryAlignmentReadFilter" are used to filter out reads that are unmapped or representing secondary alignments. Note that if `--REMOVE_DUPLICATES` false was set for GATK MarkDuplicates, the BAM file containing duplicate reads that are marked can be used for GATK CountReads by adding --read-filter NotDuplicateReadFilter. See https://gatk.broadinstitute.org/hc/en-us/articles/360057438571--Tool-Documentation-Index#ReadFilters for a full list of available read filters. Divide "the number of on-target reads" found

in ontarget_reads_sorted_rmdups.txt by "the total number of mapped, non-duplicate reads" to get "the percentage of on-target reads after duplicates removal". Similarly, divide "the number of on-target reads" found in ontarget_reads_sorted.txt by "the total number of mapped reads" to get "the percentage of on-target reads before duplicates removal". See *Basic Mapping Metrics* for reporting of the total number of mapped and non-duplicate reads.

Target-adjacent coverage is typical for target enrichment due to the capture of partially on-target DNA library fragments that also extend outside the capture region. To optionally assess the amount of reads that are target adjacent, add `--interval_padding` 100 to the commands above to add 100 bp to both sides of all targets. Although 100 bp is commonly used for this kind of padding, shorter or longer lengths may also be appropriate depending on expected library fragment sizes. Please note: all remaining steps in this document are written to use non-padded targets.

### Create Genomic Interval Lists

Interval lists are genomic interval description files required by GATK `CollectHsMetrics` that contain a SAM-like header describing the reference genome and a set of coordinates with strand and name for each interval. The Roche-provided "primary target" files can be provided as GATK "target interval" inputs, and the Roche-provided "capture target" files can be provided as GATK "bait interval" inputs. However, in the Roche TE application, we focus on evaluating the capture performance on "capture target", and we want to get the --PER_BASE_COVERAGE option in GATK `CollectHsMetrics` applied to "capture target" for detailed outputs of the coverage in each base position. Here we create the interval list for the "capture target" file which will be provided as both the "target interval" input and the "bait interval" input in GATK `CollectHsMetrics`.

Use the GATK `BedToIntervalList` command to create Interval List files from target BED files.

| **Package ➔ Tool(s) Used** | GATK ➔ `BedToIntervalList` |
|---|---|
| **Input(s)** | DESIGN_capture_targets.bed<br><br>ref.dict {one of the files in the indexed genome file set} |
| **Output(s)** | DESIGN_bait.interval_list |

**Create a Genomic Bait Interval List**

```
/path/to/gatk BedToIntervalList \
  --INPUT DESIGN_capture_targets.bed \
  --SEQUENCE_DICTIONARY /path/to/ref.dict \
  --OUTPUT DESIGN_bait.interval_list
```

The GATK `IntervalListTool` command (not described here) can be used to add padding to interval lists.

## Hybrid Selection (HS) Analysis Metrics

The `CollectHsMetrics` command calculates a number of metrics assessing the quality of target enrichment reads, including fold 80 base penalty, which is a metric of sequencing depth uniformity over the targets.

| **Package⇨Tool(s) Used** | GATK➔`CollectHsMetrics` |
|---|---|
| **Input(s)** | ref.fa {indexed}<br>SAMPLE_sorted.bam {indexed}<br>SAMPLE_sorted_rmdups.bam {indexed}<br>DESIGN_bait.interval_list |
| **Output(s)** | SAMPLE_hs_metrics_sorted.txt<br>SAMPLE_per_base_coverage_sorted.txt<br>SAMPLE_hs_metrics_sorted_rmdups.txt<br>SAMPLE_per_base_coverage_sorted_rmdups.txt |

**`CollectHsMetric` Before Duplicates Removal**

```
/path/to/gatk CollectHsMetrics \
  --BAIT_INTERVALS DESIGN_bait.interval_list \
  --BAIT_SET_NAME DESIGN \
  --TARGET_INTERVALS DESIGN_bait.interval_list \
  --INPUT SAMPLE_sorted.bam \
  --OUTPUT SAMPLE_hs_metrics_sorted.txt \
  --METRIC_ACCUMULATION_LEVEL ALL_READS \
  --REFERENCE_SEQUENCE /path/to/ref.fa \
  --VALIDATION_STRINGENCY LENIENT \
  --COVERAGE_CAP 100000 \
  --PER_BASE_COVERAGE SAMPLE_per_base_coverage_sorted.txt
```

**`CollectHsMetric` After Duplicates Removal**

```
/path/to/gatk CollectHsMetrics \
  --BAIT_INTERVALS DESIGN_bait.interval_list \
  --BAIT_SET_NAME DESIGN \
  --TARGET_INTERVALS DESIGN_bait.interval_list \
  --INPUT SAMPLE_sorted_rmdups.bam \
  --OUTPUT SAMPLE_hs_metrics_sorted_rmdups.txt \
  --METRIC_ACCUMULATION_LEVEL ALL_READS \
  --REFERENCE_SEQUENCE /path/to/ref.fa \
  --VALIDATION_STRINGENCY LENIENT \
  --COVERAGE_CAP 100000 \
  --PER_BASE_COVERAGE SAMPLE_per_base_coverage_sorted_rmdups.txt
```

**Additional Levels of Coverage (see note below for calculation)**

```
gawk '$1 != "chrom" && $4 >= N' SAMPLE_per_base_coverage_sorted_rmdups.txt | wc -l
```

Here we supply the same interval file to both `--TARGET_INTERVALS` and `--BAIT_INTERVALS` parameters of GATK `CollectHsMetrics`, as we want to leverage the `--PER_BASE_COVERAGE` option to examine the per base coverages in those capture regions. See *https://broadinstitute.github.io/picard/picard-metric-definitions.html#HsMetrics* for a description of output metrics. Note that some metrics are not directly comparable as some are obtained before read or base filters are applied (e.g., capture bases metrics) while others are calculated after (e.g., target coverage metrics).

**Note**: The CollectHsMetrics tool reports the percent of bases covered at certain sequencing depths (e.g., 1X, 10X, 20X, 30X, 40X, and 50X). To obtain coverage for additional sequencing depths ≥ N use the command "gawk '$1 != "chrom" && $4 >= N' SAMPLE_per_base_coverage_sorted_rmdups.txt | wc -l" to obtain the number of bases with at least N coverage. Divide this number by the "TARGET_TERRITORY" value from SAMPLE_hs_metrics_sorted_rmdups.txt to calculate "% bases ≥ N". The "SAMPLE_per_base_coverage*.txt" files can be quite large for large designs, and can be compressed once sequencing depths have been calculated using gzip as described earlier.

## Call Germline Variants with DeepVariant

After reads are mapped and duplicates are removed, variants are often called against the reference genome. GATK HaplotypeCaller has been a widely used germline caller that calls germline SNPs and indels via local re-assembly of haplotypes. In comparison, DeepVariant has been gaining increasing popularity as it uses a deep neural network which shows high accuracy in SNP and indel calling. Other variant calling software may work as long as it accepts a BAM file as input. Non-diploid, low frequency, and somatic variants must be called using other methods not described in this document (see *https://software.broadinstitute.org/gatk/best-practices/* for more information). Here we describe how to use DeepVariant to call germline variants.

The developers of DeepVariant recommend using the Docker solution for running DeepVariant (*https://github.com/google/deepvariant#how-to-run-deepvariant*). We first need to create a folder /path/to/input where the input files including ref.fa {indexed}, SAMPLE_sorted_rmdups.bam, DESIGN_capture_targets.bed and DESIGN_primary_targets.bed are stored.

| **Package⇨Tool(s) Used** | deepvariant |
|---|---|
| **Input(s)** | input/<br>ref.fa {indexed}<br>SAMPLE_sorted_rmdups.bam {indexed}<br>DESIGN_capture_targets.bed<br>DESIGN_primary_targets.bed |
| **Output(s)** | SAMPLE_deepvariant.vcf.gz<br>SAMPLE_deepvariant.vcf.gz.tbi<br>SAMPLE_deepvariant.gvcf.gz<br>SAMPLE_deepvariant.gvcf.gz.tbi<br>SAMPLE_deepvariant.visual_report.html |

Generate A Union of Capture and Primary Targets

```
cat DESIGN_capture_targets.bed DESIGN_primary_targets.bed | sort -k1,1 -k2,2n | bedtools merge -i - > DESIGN_capture_primary_targets_union.bed
```

Call Variants

```
BIN_VERSION="0.9.0"

docker pull google/deepvariant:"${BIN_VERSION}"

docker run --rm \

  -v "/path/to/input":"/input" \

  -v "/path/to/output:/output" \

  google/deepvariant:"${BIN_VERSION}" \

  /opt/deepvariant/bin/run_deepvariant \

  --model_type=WES \

  --ref="/input/ref.fa" \

  --reads="/input/SAMPLE_sorted_rmdups.bam" \

  --regions="/input/DESIGN_capture_primary_targets_union.bed" \

  --output_vcf="/output/SAMPLE_deepvariant.vcf.gz" \

  --output_gvcf="/output/SAMPLE_deepvariant.gvcf.gz" \

  --num_shards=$(nproc)
```

The variant calling is performed over the union of the capture and primary targets. A 25-50 bp padding to the targets should also be considered for variant calling. Extra variants outside regions of interest can be filtered out later. In the above example, the setting --num_shards=$(nproc) will use all the cores to run the "make_examples" step. It can also be changed to NumProcessors. See https://github.com/google/deepvariant/blob/r0.9/docs/deepvariant-details.md for the detailed usage guide.

While newer versions of DeepVariant are currently available, it's recommended that v0.9.0 be utilized, as later versions show lower recall, precision, and F1 score (see hap.py below) for panels benchmarked with standard reference samples such as NA12878 and NA24385.

Additional downstream variant analysis is not covered in this document, but may consist of comparison of SNP calls against dbSNP, variant classification, and variant effect analysis.

**The variant calling and filtering tools shown in the example here are most appropriate for diploid, homogeneous samples. Calling low frequency variants from heterogeneous samples must be performed using alternate analysis methods. These alternate methods also accept BAM files as input for variant calling but are not described here. Examples of calling genomic somatic variants are described in a separate note.**

### Compare to Known Variants with hap.py

The hap.py program can be used to compare variant calls to known variants for the sample. This is most commonly done to compare variant calls against known variants for Coriell HapMap samples such as NA12878 (female CEU/CEPH), NA12891 (male, CEU/CEPH) or NA24385 (male, Ashkenazim Jewish).

The gold standard VCF file should consist of high confidence known variants for the sample being examined. It is important to note that some sources for known variants are more trustworthy than others. The best gold standard will be built using two or more sources and if possible, two or more technology platforms. This reduces source- and platform-specific systematic errors in your gold standard variant list. Known variants may be downloaded from the 1000 Genomes Project (*http://www.internationalgenome.org/*) or in special collections such as the GATK resource bundle (*https://console.cloud.google.com/storage/browser/genomics-public-data/resources/broad/hg38/v0*) and Genome in a Bottle (*https://github.com/genome-in-a-bottle*).

Note that some gold standard VCF files may need to be modified or normalized before assessing concordance. Confirm that the VCF files have similar notation (e.g., using "|" vs. "/" between alleles) and chromosome names and adjust as needed. Normalization of the gold standard and sample VCF files can be used to reduce differences in genotyping due to the variant caller. The tool bcftools and the command norm is one method for VCF normalization.

| **Package⇨Tool(s) Used** | hap.py |
|---|---|
| **Input(s)** | ref.fa {indexed}<br>SAMPLE_deepvariant.vcf.gz<br>gold_standard.vcf<br>confident.bed<br>DESIGN_capture_primary_targets_union.bed |
| **Output(s)** | SAMPLE_happy.extended.csv<br>SAMPLE_happy.metrics.json.gz<br>SAMPLE_happy.roc.all.csv.gz<br>SAMPLE_happy.roc.Locations.INDEL.csv.gz<br>SAMPLE_happy.roc.Locations.INDEL.PASS.csv.gz<br>SAMPLE_happy.roc.Locations.SNP.csv.gz<br>SAMPLE_happy.roc.Locations.SNP.PASS.csv.gz<br>SAMPLE_happy.runinfo.json<br>SAMPLE_happy.summary.csv<br>SAMPLE_happy.vcf.gz<br>SAMPLE_happy.vcf.gz.tbi |

```
BIN_VERSION="v0.3.9"

docker pull pkrusche/hap.py:"${BIN_VERSION}"

docker run -it \

  -v `pwd`:/data pkrusche/hap.py:"${BIN_VERSION}" /opt/hap.py/bin/hap.py \

  /data/gold_standard.vcf \

  /data/SAMPLE_deepvariant.vcf.gz \

  -o /data/SAMPLE_happy \

  --gender GENDER \

  --threads NumProcessors \

  -f /data/confident.bed \

  -T /data/DESIGN_capture_primary_targets_union.bed \

  -r /data/ref.fa
```

The above example pulls a pre-built Docker image from *https://hub.docker.com/r/pkrusche/hap.py* and then runs hap.py directly. Make sure all the input files are located in the current directory. The option --gender specifies the sex of the sample ("male", "female", "auto", or “None”). It determines how haploid calls on chrX get treated. The option -f specifies truth set high-confidence regions. The latest VCF and BED files with the high-confidence calls and regions can be obtained from the “latest” directory under each genome at the Genome in a Bottle FTP site: *ftp://ftp-trace.ncbi.nlm.nih.gov/ReferenceSamples/giab/release/*. For example, the high-confidence regions for NA12878 can be downloaded from *ftp://ftp-trace.ncbi.nlm.nih.gov/ReferenceSamples/giab/release/NA12878_HG001/latest/GRCh38/HG001_GRCh38_GIAB_highconf_CG-IllFB-IllGATKHC-Ion-10X-SOLID_CHROM1-X_v.3.3.2_highconf_nosomaticdel_noCENorHET7.bed*.

Variants outside the high-confidence regions will be counted as UNK during comparison. The option -T specifies regions to be restricted during comparison. After the hap.py run is finished, the output should appear in the current directory. The output of hap.py consists of multiple files. See *https://github.com/Illumina/hap.py/blob/master/doc/happy.md#full-list-of-output-columns* for details on inputs and outputs. Alternatively, users can build their own Docker image or install the hap.py with the  helper script. See *https://github.com/Illumina/hap.py#installation* for instructions.

The Global Alliance for Genomics and Health (GA4GH) definitions of TP, FP, FN, FP.AL, FP.GT, and UNK are described by Table 2.

| | | Truth | | | | |
|---|---|---|---|---|---|---|
| | Genotype | Ref/Ref | Ref/Var1 | Var1/Var2 | Var1/Var1 | Outside bed |
| Query | Ref/Ref | - | FN | FN | FN | - |
| | Ref/Var1 | FP | TP | FP.GT | FP.GT | UNK |
| | Ref/Var2 | - | FP.AL | FP.GT | FP.AL | - |
| | Ref/Var3 | - | - | FP.AL | - | - |
| | Var1/Var2 | FP | FP.GT | TP | FP.GT | UNK |
| | Var1/Var3 | - | - | FP.GT | - | - |
| | Var2/Var3 | - | FP.AL | FP.GT | FP.AL | - |
| | Var3/Var4 | - | - | FP.AL | - | - |
| | Var1/Var1 | FP | FP.GT | FP.GT | TP | UNK |
| | Var2/Var2 | - | FP.AL | FP.GT | FP.AL | - |
| | Var3/Var3 | - | - | FP.AL | - | - |

Table 2. Contingency table describing the GA4GH definitions of TP, FP, FN, FP.AL, FP.GT, and UNK. TP: true positive, FP: false positive, FN: false negative, FP.AL: allele mismatch, FP.GT: genotype mismatch, UNK: unknown. Query variants outside the high-confidence bed file are counted as UNK. Boxes with dashes are not possible when comparing two VCFs. Matches counted as FP.GT and FP.AL are additionally counted as both FP and FN, since hap.py's default matching stringency requires genotypes to match. This table is adapted from Table 1 of the Nature Biotechnology paper about best practices for benchmarking germline small-variant calls in human genomes. See *https://www.nature.com/articles/s41587-019-0054-x* for details.

## Description of Metrics

The tools used in this document generate output files that contain many metrics. There are some metrics that are frequently monitored to assess capture experiment performance. Table 3 describes many of these metrics, which tool(s) are used to generate the metrics, and additional mathematical or string parsing operations that may be necessary to obtain the final values.

| Metric | Tool(s) used to obtain value (name of output file used) | Description | Metric name in tool's output file and/or calculation method |
|---|---|---|---|
| Total input reads | **fastp (SAMPLE_fastp.log)** | Number of reads prior to fastp processing for quality and adapter trimming | "Read1 before filtering: total reads" + "Read2 before filtering: total read" |
| Total reads after adapter trimming | **fastp (SAMPLE_fastp.log)** | Number of reads after fastp processing for quality and adapter trimming | "Filtering result: reads passed filter" |
| % Input reads after filtering | **fastp (SAMPLE_fastp.log)** | Percentage of total input reads remaining after fastp processing | ("Filtering result: reads passed filter") / ( "Read1 before filtering: total reads" + "Read2 before filtering: total read") * 100 |
| % Reads mapped | **samtools flagstat (SAMPLE_flagstat.txt)** | Percentage of filtered reads that are mapped in the genome | %value in the "mapped" field |
| % Paired reads mapped | **samtools flagstat (SAMPLE_flagstat.txt)** | Percentage of filtered reads that are paired and mapped in the genome | %value in the "properly paired" field |
| Total duplicate rate | **GATK MarkDuplicates (SAMPLE_markduplicates_metrics.txt)** | Percentage of aligned reads identified as PCR duplicates, includes both paired and unpaired reads | PERCENT_DUPLICATION * 100 |
| Optical duplicate rate | **GATK MarkDuplicates (SAMPLE_markduplicates_metrics.txt)** | Percentage of aligned reads identified as optical duplicates, includes both paired and unpaired reads | (READ_PAIR_OPTICAL_DUPLICATES * 2) / ((READ_PAIRS_EXAMINED * 2) + UNPAIRED_READS_EXAMINED) * 100 |

| Metric | Tool(s) used to obtain value (name of output file used) | Description | Metric name in tool's output file and/or calculation method |
|---|---|---|---|
| % reads on-target (before duplicates removal) | **GATK CountReads (ontarget_reads_sorted.txt)**<br><br>**GATK CollectAlignmentSummaryMetrics (SAMPLE_alignment_metrics.txt)** | Percentage of mapped reads overlapping a target region by at least 1 base. No padding/buffering. | ontarget_reads = value ("Tool returned:") in ontarget_reads_sorted.txt<br><br>total_mapped_reads = CollectAlignmentSummaryMetrics PF_READS_ALIGNED in the PAIR column of SAMPLE_alignment_metrics.txt<br><br>% mapped reads on-target = (on_target_reads) / (total_mapped_reads) * 100 |
| % reads on-target (after duplicates removal) | **GATK CountReads (ontarget_reads_sorted_rmdups.txt)**<br><br>**GATK CollectAlignmentSummaryMetrics (SAMPLE_alignment_metrics.txt)**<br><br>**GATK MarkDuplicates (SAMPLE_markduplicates_metrics.txt)** | Percentage of mapped, non-duplicate reads overlapping a target region by at least 1 base. No padding/buffering. | ontarget_reads = value ("Tool returned:") in ontarget_reads_sorted_rmdups.txt<br><br>total_mapped_reads = CollectAlignmentSummaryMetrics PF_READS_ALIGNED in the PAIR column of SAMPLE_alignment_metrics.txt<br><br>unpaired_read_duplicates = MarkDuplicates UNPAIRED_READ_DUPLICATES in SAMPLE_markduplicates_metrics.txt<br><br>read_pair_duplicates = MarkDuplicates READ_PAIR_DUPLICATES in SAMPLE_markduplicates_metrics.txt<br><br>% mapped, non-duplicate reads on-target = (on_target_reads) / (total_mapped_reads - unpaired_read_duplicates - read_pair_duplicates * 2) * 100 |

| Metric | Tool(s) used to obtain value (name of output file used) | Description | Metric name in tool's output file and/or calculation method |
|---|---|---|---|
| Fold enrichment | **GATK CollectHsMetrics (SAMPLE_hs_metrics_sorted_rmdups.txt)** | Fold enrichment of the capture target compared to the whole genome. In terms of the metrics in the CollectHsMetrics output file: ( ON_BAIT_BASES / ( ON_BAIT_BASES + NEAR_BAIT_BASES + OFF_BAIT_BASES ) ) / ( BAIT_TERRITORY / GENOME_SIZE ). In other words, the fraction of sequencing bases in the capture target divided by the fraction of total genomic bases in the capture target. | FOLD_ENRICHMENT |
| Fold 80 base penalty (uniformity) | **GATK CollectHsMetrics (SAMPLE_hs_metrics_sorted_rmdups.txt)** | Fold additional sequencing required to bring 80% of bases to the mean. This is a measure of sequence depth uniformity, lower is better. The best theoretical value is 1. Zero coverage regions are excluded. Another name for this metric is "1/nc80". This metric is sensitive to the total number of reads. | FOLD_80_BASE_PENALTY |
| Median insert size (after duplicates removal) | **GATK CollectInsertSizeMetrics (SAMPLE_insert_size_metrics_sorted_rmdups.txt)** | Median estimated capture fragment insert size | MEDIAN_INSERT_SIZE |
| Mean insert size (after duplicates removal) | **GATK CollectInsertSizeMetrics (SAMPLE_insert_size_metrics_sorted_rmdups.txt)** | Mean estimated capture fragment insert size | MEAN_INSERT_SIZE |
| Insert size std dev (after duplicates removal) | **GATK CollectInsertSizeMetrics (SAMPLE_insert_size_metrics_sorted_rmdups.txt)** | Standard deviation of the estimated capture fragment insert size | STANDARD_DEVIATION |

| Metric | Tool(s) used to obtain value (name of output file used) | Description | Metric name in tool's output file and/or calculation method |
|---|---|---|---|
| Mean target coverage (before duplicates removal) | **GATK CollectHsMetrics (SAMPLE_hs_metrics_sorted.txt)** | Mean depth of coverage over the capture target | MEAN_TARGET_COVERAGE |
| Mean target coverage (after duplicates removal) | **GATK CollectHsMetrics (SAMPLE_hs_metrics_sorted_rmdups.txt)** | Mean depth of coverage over the capture target | MEAN_TARGET_COVERAGE |
| Median target coverage (before duplicates removal) | **GATK CollectHsMetrics (SAMPLE_hs_metrics_sorted.txt)** | Median depth of coverage over the capture target | MEDIAN_TARGET_COVERAGE |
| Median target coverage (after duplicates removal) | **GATK CollectHsMetrics (SAMPLE_hs_metrics_sorted_rmdups.txt)** | Median depth of coverage over the capture target | MEDIAN_TARGET_COVERAGE |
| % bases >= NX (e.g., N=1, 10, 20, 30) | **GATK CollectHsMetrics (SAMPLE_hs_metrics_sorted_rmdups.txt)**<br><br>**OR**<br><br>**"gawk '$1 != "chrom" && $4 >= N' SAMPLE_per_base_coverage_sorted_rmdups.txt \| wc -l"/TARGET_TERRITORY * 100** | Percentage of capture target bases covered by N or more reads | PCT_TARGET_BASES_NX |

**Table 3. Description of important metrics**

## 3. Reference Links

Roche is not responsible for the content of the following third-party websites (accessed on May 2024).

| | |
|---|---|
| BEDTools: *https://github.com/arq5x/bedtools2* | SAMtools: *http://www.htslib.org/* |
| BWA: *https://github.com/lh3/bwa* | hap.py: *https://github.com/Illumina/hap.py* |
| seqtk: *https://github.com/lh3/seqtk* | fastp: *https://github.com/OpenGene/fastp* |
| FastQC: *http://www.bioinformatics.babraham.ac.uk/projects/fastqc/* | DeepVariant: *https://github.com/google/deepvariant* |
| GATK (Broad Institute): *https://software.broadinstitute.org/gatk/* | |

# 4. Glossary

**BAI file** – BAM index file. For tools that require an indexed **BAM file,** the BAI file must be present in the same location as the BAM file.

**Bait interval (GATK)** – See **Capture target**.

**BAM file** – Compressed form of the **SAM file** format.

**BED file** – File format for describing genomic regions/intervals. BED file start coordinates are 0-based.

**bp** – Abbreviation for base pair.

**Capture target** – as defined by Roche, these are the regions covered directly by one or more probes or primer pairs. These are equivalent to the **Bait intervals** referred to by GATK.

**FASTA file** – A standard file format for describing nucleic acid sequences.

**FASTQ file** – A standard file format for describing sequencing reads that also includes base quality information.

**Genomic index** – A form of the reference genome sequence that enables faster comparisons during alignment.

**Interval file** – File format for describing genomic regions/intervals that also contains a header describing the reference genome. Genomic interval file start coordinates are 1-based. See **Bait interval (GATK)** and **Target interval (GATK)**.

**Primary target** – as defined by Roche, these are the regions against which probes or primer pairs were designed. Regions with no probes or primer pairs targeting them are excluded. These are equivalent to the **Target intervals** referred to by GATK.

**SAM file** – Sequence Alignment / Map file; a community standard format for specifying sequencing read alignment to a reference genome.

**Target interval (GATK)** – see **Primary target**.

**Target region** – see **Primary target**.

**VCF file** – Variant call format; a community standard format for specifying variant calls for one or more samples or populations against a reference genome.

---

Published by:
Roche Sequencing Solutions, Inc.
4300 Hacienda Drive
Pleasanton, CA 94588

 MC–08067 05/24