**INTERNATIONAL ORGANISATION FOR STANDARDISATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC 1/SC 29/WG 11**

**CODING OF MOVING PICTURES AND AUDIO**

**AND**

**ISO/TC 276 BIOTECHNOLOGY**

**ISO/IEC JTC 1/SC 29/WG 11 N16324**

**ISO/TC 276/WG 5 N100**

**Geneva, CH – June 2016**

|  |  |
| --- | --- |
| **Source** | **ISO/IEC JTC 1/SC 29/WG 11**  **ISO/TC 276/WG 5** |
| **Status** | **Final** |
| **Title** | **Lossy compression framework for Genome Data** |

# Executive Summary

High-throughput genome sequencing machines produce genomic information in the form of strings of nucleotides (bases) associated to metadata. Among the produced metadata Quality Values (QVs) account for the largest part of the compressed information when lossless compression is adopted. QVs are a vendor-specific expression of the probability each sequenced nucleotide is correct and express the level of approximation of each sequencing process. Due to their nature, their use in downstream analysis is extremely diversified and some tools and applications do not even rely on them to produce results (e.g. BWA [1], one of the most used aligners, does not use QVs).

The focus of efficient compression of QVs is currently shifting from the lossless approach adopted by tools such as SAMtools [2] and other more optimized implementations [3] to lossy schemes recently appeared in literature [4] [5] [6] [7] [8]. In some cases lossy compression of QVs is not only acceptable but seems to actually improve performance of analysis such as genotyping (identification of variants with respect to a reference genome).

The goal of this document is the definition of the characteristics of appropriate methodologies and related sets of genome data and analysis applications (with the related configurations) for the evaluation of the "quality" of genome analysis results. Such evaluation will constitute the base for the comparison of different approaches to lossy compression of genome metadata.

Identification of evaluation methodologies and frameworks requires the definition of:

* The types of data to be analyzed
* The genomic analysis applications in scope
* The specific data to be analyzed (both test sets and references)
* The tools used to perform the analysis
* The metrics to evaluate the impact of lossy compression on analysis

Table of Contents

[1 Executive Summary 1](#_Toc446065324)

[Table of Contents 2](#_Toc446065325)

[2 Type of data 3](#_Toc446065326)

[2.1 Human Genome 3](#_Toc446065327)

[2.2 Metagenomics 3](#_Toc446065328)

[2.3 Cancer genomes 3](#_Toc446065329)

[2.4 Other species 3](#_Toc446065330)

[3 Applications 3](#_Toc446065331)

[3.1 Human genome 3](#_Toc446065332)

[3.1.1 SNPs calling 3](#_Toc446065333)

[3.1.2 Other applications 4](#_Toc446065334)

[3.1.3 De-novo assembly 4](#_Toc446065335)

[3.2 Metagenomics 4](#_Toc446065336)

[3.3 Cancer cells 4](#_Toc446065337)

[3.4 Other species 5](#_Toc446065338)

[4 Data 5](#_Toc446065339)

[4.1 Human Genome 5](#_Toc446065340)

[4.2 Human genome assembly 5](#_Toc446065341)

[4.3 Variant calling “golden reference” 5](#_Toc446065342)

[4.4 Variant calling using simulated data 6](#_Toc446065343)

[4.5 Metagenomics, cancer cells and other species 7](#_Toc446065344)

[5 Tools 7](#_Toc446065345)

[6 Metrics 7](#_Toc446065346)

[6.1 Sensitivity and precision 7](#_Toc446065347)

[6.2 Area under the ROC curve 8](#_Toc446065348)

[7 Caveats 9](#_Toc446065349)

[8 Bibliography 10](#_Toc446065350)

[9 Appendix A 11](#_Toc446065351)

# Type of data

## Human Genome

A wide variety of biomedical applications is currently growing around human genome SNP calling. Such applications include:

* Disease Genetics Studies: study of the relation between gene variations and disease state
* Pharmacogenomic Studies: study of the relation between an individual’s genetic profile and her response to various drugs

Efficient compression of genomic data would enable the biomedical industry to scale these applications to large populations of individuals.

There is a consensus around a small number of tools and pipelines and reference data sets.

## Metagenomics

Metagenomics is the study of genetic material extracted from environmental samples. One of the areas of primary interest for metagenomics in human health is the one related to the human gut as the microbial community contained in the gut plays an important role in protecting against pathogenic microbes, modulating immunity and regulating metabolic processes.

## Cancer genomes

SNPs calling of genetic material extracted from tumor cells can play an important role in oncology with the possibility to define targeted and personalized therapies.

## Other species

Other species such as infectious disease agents are not in the scope of this activity for now as a solid consensus on how to perform SNPs genotyping is not yet present in the scientific community.

# Applications

This section lists the most relevant types of genome analysis applications and processes where today the size of data has an important impact on the time needed to produce results. The level of maturity of each domain is different and in some cases it is not possible to clearly identify best practices and common tools used.

## Human genome

### SNPs calling

Single Nucleotide Variants (SNVs) calling from genomic data refers to a range of methods for identifying the existence of differences between a reference dataset and the results of next generation sequencing (NGS) experiments. Due to the increasing abundance of NGS data, these techniques are becoming increasingly common for performing Single Nucleotide Polymorphism (SNP) calling.

SNP calling refers to the process of identifying SNVs on a population of individuals. Consensus around a few tools exists and the domain is mature enough to define a framework for the evaluation of the impact of lossy compression on the analysis results.

### Other applications

Other applications of human genome analysis that are growing rapidly but are not mature enough to see a clear prevalence of tools or methodologies include:

1. Indel calling

For indel calling the scenario of tools and practices is more diversified and at the moment of editing this document we cannot identify a set of tools to be used for evaluation.

1. Structural variation analysis
2. RNA sequencing
3. ChIP-sequencing

### De-novo assembly

An overview about de-novo assemblers has been presented in previous work [9]; those using quality scores are:

* ALLPATHS (Broad) [9] : Removing and/or correcting erroneous reads based on quality scores (amongst others)
* SHARCGS [10]: Removing erroneous reads based on quality scores (amongst others)

ALLPATHS is still currently maintained by an active group at Broad Institute while SHARCGS is no longer maintained. DISCOVAR (<http://www.broadinstitute.org/software/discovar/blog/>) is currently replacing ALLPATHS at the Broad Institute.

Comparing the impact of lossy compression on de-novo assembly is out of the scope of the first evaluation, but these tools are mentioned in order to enable interested parties to use appropriate tools.

## Metagenomics

In metagenomics there is still no consensus around tools and reference data sets for the most important analysis applications:

1. Abundance analysis
2. Functional analysis
3. Phylogenetic analysis

De-novo assembly is playing an important role in metagenomics as well. The tools are the same as those mentioned in section 3.1.3.

## Cancer cells

The study of cancer cell genomes shows no consensus around tools and the nature of data is still problematic to enable the selection of precise tools and methodologies for evaluations. Relevant applications include:

1. SNP calling
2. Structural variation analysis
3. DNA methylation

## Other species

For other species the scope of analysis is the same as for humans.

# Data

## Human Genome

We are considering individual NA12878

(<https://catalog.coriell.org/0/Sections/Search/Sample_Detail.aspx?Ref=GM12878>).

This individual has been sequenced using different technologies and configurations. In order to make the comparison more relevant we are going to select the following sets:

|  |  |  |
| --- | --- | --- |
| ID | Description | Source |
| 1 | NA12878 from IonTorrent | [SRX517292](ftp://ftp.ddbj.nig.ac.jp/ddbj_database/dra/fastq/SRA096/SRA096885/SRX517292) |
| 2 | NA12878 replicate J – 8bin QS, 30x | [Garvan](http://www.garvan.org.au/research/kinghorn-centre-for-clinical-genomics/clinical-genomics/sequencing-services/sample-data) (robot 2) |
| 3 | NA12878 Illumina, higher coverage not binned | [SMaSH dataset (Berkeley)](http://smash.cs.berkeley.edu/datasets.html) (50x) |
| 4 | Run SRR1231836 of experiment accession SRX514833 stored on the DDBJ repository | [SRX514833](https://trace.ddbj.nig.ac.jp/DRASearch/experiment?acc=SRX514833) |

The dataset shall take into account new generations of machines that might have different behaviors and performance when producing quality scores. The dataset shall be updated accordingly when necessary.

## Human genome assembly

The reference decoy is a pragmatic solution to the “incompleteness” of the reference genome. It integrates the reference sequence from the GRCh37 primary assembly with sequences derived from HuRef, Human Bac and Fosmid clones, and NA12878 (decoy sequences). Such sequences provide additional information that helps in aligning many reads quickly and with high confidence, eliminating excessive mapping depth and mismatches.

The human reference genome GRCh37 published by NCBI can be found here: <http://www.ncbi.nlm.nih.gov/assembly/2758/>

The reference decoy can be found in the GATK bundle via ftp in /bundle/2.8/b37 at:

<ftp://gsapubftp-anonymous@ftp.broadinstitute.org/bundle>

## Variant calling “golden reference”

The reference for variants calling taken into account are:

* Illumina Platinum Genomes High confidence variant calls

<ftp://ussd-ftp.illumina.com/hg19/8.0.1/NA12878/>

* GIAB-NIST

<ftp://ftp-trace.ncbi.nlm.nih.gov/giab/ftp/release/NA12878_HG001/NISTv2.19/>

These data sets contain high confidence regions containing variants for which the probability of error is deemed low enough to represent a benchmark for other tools. It is recommended to separate results related to variants called in high confidence regions from those of variants in other regions of the genome.

## Variant calling using simulated data

In order to assess how a particular bioinformatics pipeline impacts variant calling, a ground truth set of genotypes must be provided. Since ground truth datasets from real individuals are difficult to develop at scale, simulated data is an excellent alternative. Variant calling on simulated data has the advantage of providing a known ground truth. Sequenced reads are simulated by mimicking real sequencing process with empirical error models or quality profiles summarized from large recalibrated sequencing data.

Evaluating variant calls from simulated data requires 1) a simulated ground truth genome with variants that have a known location and 2) simulated sequencing reads from the simulated genome. In order to be useful, the reads should not be error-free, but should instead match the error models and quality profile of the sequencing technology in question.

[ART](http://www.niehs.nih.gov/research/resources/software/biostatistics/art/) [11] is a tool that enables the creation of synthetic sequence reads for a given genome that is documented and maintained by the NIH. The error models and quality profiles for ART are modeled from a large corpus of real sequencing data. ART draws from these distributions to simulate sequence reads.

The input genome is a simple FASTA file created by the user. Ideally, the FASTA file will be a variation of the human reference sequence that includes “spiked in” SNPs, indels, structural variants, etc. These variants should be added in a biologically realistic way (in terms of location and frequency).

For the purpose of this evaluation a synthetic genome has been generated and is available upon request to Claudio Alberti ([claudio.alberti@epfl.ch](mailto:claudio.alberti@epfl.ch)).

To mimic an Illumina sequencer, ART can generate paired-end reads (with sequencing errors), where the mean fragment size and fragment size standard deviation may be given as parameters. The user can also specify the depth of coverage to be generated. The error models and quality profiles for the sequence reads created by ART are modeled after a corpus of real sequencing data.

Sample command structure:

|  |
| --- |
| $art\_illumina -i [simulated\_genome.fa] -p -l [readLength] \  -f [coverage to be generated] -m [meanSizeFragments] \  -s [fragmentSizeStandardDeviation] -o [outFileName] –na |

where

|  |  |
| --- | --- |
| -i | Flag for the input file |
| simulated\_genome.fa | Simulated genome created from a real genome with the insertion of variants such as SNPs, indels, structural variants etc. |
| -p | Flag indicating the use of paired reads |
| -l | Flag for the generated reads length |
| -f | Flag for the coverage |
| -m | Flag for the mean distance between paired reads |
| -s | Flag for the std deviation in the distribution of distances among paired reads |
| -o | Flag for the output file name |
| -na | Do not output alignment file |

For the purpose of this work a simulated genome has been created by Rachel Goldfeder at Stanford University and it can be obtained from Claudio Alberti ([claudio.alberti@epfl.ch](mailto:claudio.alberti@epfl.ch)).

The following command has been used to create the simulated genomic data

|  |
| --- |
| $ art\_illumina -i mpeg\_simulated\_genome\_01.fasta -p -l 100 -f 30 \  -m 300 -s 10 -o mpeg\_01\_l100\_f30\_m300\_s10.fastq -na |

The first set of simulated data can be obtained from Claudio Alberti ([claudio.alberti@epfl.ch](mailto:claudio.alberti@epfl.ch)).

## Metagenomics, cancer cells and other species

The existing MPEG dataset already contains metagenomics and cancer cell data that can be used to evaluate performance of lossy compression on downstream analysis. This is currently not in the scope of the evaluation, but it can provide additional indication of the potential impact of lossy QS compression on downstream analysis.

# Tools

This section lists the toolchains considered for comparison and the related configurations.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Pipeline | Configuration | Source |
| 1 | BWA-MEM + GATK\_HC | See appendix A | [GATK web site](https://www.broadinstitute.org/gatk/guide/best-practices) |
| 2 | Bowtie2 + GATK\_HC | See appendix A | [Bowtie2 web site](http://bowtie-bio.sourceforge.net/bowtie2/index.shtml) |
| 3 | BWA-MEM + SAMtools + BCFtools | See appendix A | [HTSlib web site](http://www.htslib.org/workflow/) |
| 4 | Bowtie2 + SAMtools + BCFtools | See appendix A |  |

# Metrics

## Sensitivity and precision

In literature two metrics are used to assess the correctness of variant calling against a reference:

* ***Sensitivity*** defined as: 

This metric provides a measure of the correctness of positives calls

* ***Precision*** defined as: 

This metric provides a measure of the proportion of the correct calls with respect to the totality of calls

Where

* T.P. = number of variants in the gold standard that have been called and marked as positive;
* F.N. = number of variants in the gold standard that have not been called or have been called but marked as negative;
* F.P. = number of positions that have been called and marked as positive but are not in the gold standard;

These definitions imply some threshold τ to call something “positive”.

The harmonic mean of the sensitivity and precision named *F-score* provides a way to balance the effects of the two metrics and will be used as final measure:



F ranges from 0 to 1 where 0 is the worst score and 1 represents a perfect score.

We recommend the calculation of F-scores at these points of interest (values of τ).

|  |  |
| --- | --- |
| GATK | SAMtools |
| 90 | 10 |
| 99 | 20 |
| 99.9 | 40 |
| 100 | 0 |

## Area under the ROC curve

The ROC curve is defined as the plot of False Positive Rate (FPR=FP/(FP+TN)) versus True Positive rate (TPR=TP/(TP+FN)), where TP, FP, and FN were defined above.

The problem arises with the definition of the negative set, N=FP+TN, as it must agree among all the algorithms to be compared.

Moreover, different algorithms output different sets of calls (i.e., each VCF file contains different calls and a varying number of them). Thus, comparing the different algorithms based on the area under the ROC becomes challenging (far from straightforward). Note that some algorithms might generate short VCF files with mostly TPs, while other algorithms might generate large VCF files with more TPs than the previous algorithms, but at the cost of generating significantly more FPs. Furthermore, different downstream analysis might prefer one type of files over the other.

The approach we follow here is the following. We first set a value of N (see below for details). Given this value, and assuming the FPs of a given VCF file are sorted by the thresholding parameter, we only consider the first N FPs of the VCF file (i.e., if a VCF file contains more FPs than N, we consider only the first N and discard the rest). Varying the value of N, we can report the behavior of the different lossy compressors at different points (i.e., at different thresholding parameters). In other words, small values of N will select the FPs with higher confidence (higher thresholding parameter), while larger values of N will consider FPs with lower confidence as well. Moreover, values of N that are smaller than the number of FPs called by the algorithms (which is equivalent to focus on the left most part of the whole ROC curve) will favor algorithms that call TPs promptly. But as N increases its value, the metric will favor algorithms that call more overall TPs regardless of the number of FPs called.

Based on the aforementioned observations, we recommend the use of the following values of N:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N = 0.1L | N=0.5L | N=L | N=1.1L | N=1.25L | N=1.5L |

where L indicates the number of FPs generated with the original file. Then, each algorithm computes the ROC curve (please see the supplementary data of [6] and [8] for more information regarding the computation of the ROC curve) and calculates the following metric:

M = A/A L

Where

* A is the area under the curve obtained with lossless compression of QS, and
* A L is the area under the curve obtained with lossy compression of QS.

The thresholding scores used to plot the ROC curves will be:

* For GATK the VQSLOD field
* For SAMtools the QD field

Since this metric heavily depends on the value and behavior of N with the changing coverage, we encourage reporting the value(s) of N for each lossy compressor evaluated. This will improve the quality of the comparison by enabling the identification of one or several classes of compression schemes sharing similar values of N.

# Caveats

There are parameters that cannot be easily controlled for and are outside the scope of this document. Namely there can be:

* Individual variations in sequencing equipment of the same model as well as from lab to lab and library to library.
* Lack of guaranteed fidelity to true error model from the sequencer in the simulated data.

# Bibliography

[1] H. Li and R. Durbin, “Fast and accurate short read alignment with Burrows-Wheeler transform,” *Bioinformatics*, vol. 25, no. 14, pp. 1754–1760, 2009.

[2] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, and R. Durbin, “The Sequence Alignment/Map format and SAMtools,” *Bioinformatics*, vol. 25, no. 16, pp. 2078–2079, 2009.

[3] J. K. Bonfield and M. V. Mahoney, “Compression of FASTQ and SAM Format Sequencing Data,” *PLoS One*, vol. 8, no. 3, 2013.

[4] G. Malysa, M. Hernaez, I. Ochoa, M. Rao, K. Ganesan, and T. Weissman, “QVZ: Lossy compression of quality values,” *Bioinformatics*, vol. 31, no. 19, pp. 3122–3129, 2015.

[5] I. Ochoa, H. Asnani, D. Bharadia, M. Chowdhury, T. Weissman, and G. Yona, “QualComp: a new lossy compressor for quality scores based on rate distortion theory.,” *BMC Bioinformatics*, vol. 14, p. 187, 2013.

[6] Y. W. Yu, D. Yorukoglu, J. P. Peng, and B. Berger, “Quality score compression improves genotyping accuracy,” vol. 33, no. 3, pp. 240–243, 2015.

[7] R. Canovas, A. Moffat, and A. Turpin, “Lossy compression of quality scores in genomic data,” *Bioinformatics*, vol. 30, no. 15, pp. 2130–2136, 2014.

[8] I. Ochoa, M. Hernaez, R. Goldfeder, T. Weissman, and E. Ashley, “Effect of lossy compression of quality scores on variant calling,” *bioRxiv*, 2015.

[9] J. R. Miller, S. Koren, and G. Sutton, “Assembly algorithms for next-generation sequencing data,” *Genomics*, vol. 95, no. 6, pp. 315–327, 2010.

[10] J. C. Dohm, C. Lottaz, T. Borodina, and H. Himmelbauer, “SHARCGS, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing,” *Genome Res.*, vol. 17, no. 11, pp. 1697–1706, 2007.

[11] Weichun Huang, “ART: Set of Simulation Tools,” *U.S. Dep. Heal. Hum. Serv.*

# Appendix A

This section provides details on the specific configuration of the pipelines listed in section 5 as representative toolchains to perform variant calling.

The output of these pipelines is the filtered VCF files that will be used to compare against the consensus set of SNPs defined in Section 4.3.

Note: In the following commands, the words proceeded by ‘$’ are user-defined input/output files.

* 1. ***Alignment and sorting***
     1. **BWA-MEM**

First we create a collection of files used by BWA to perform the alignment (\*.fa.amb, \*.fa.ann, \*.fa.bwt, \*.fa.pac, and \*.fa.sa)

|  |
| --- |
| **$ bwa index -a bwtsw** $ref.fa |

Then we generate the FASTA file index of the reference using samtools,

|  |
| --- |
| **$ samtools faidx** $ref |

And generate the sequence dictionary using Picard

|  |
| --- |
| **$ java -jar** $picard **CreateSequenceDictionary** \  **REFERENCE=**$ref.fa \  **OUTPUT=**$ref.dict |

Finally we align the paired-end FASTQ files to the reference. We include the -M option in BWA-MEM for compatibility with Picard tools.

|  |
| --- |
| **$ bwa mem -t 4 -M** $ref.fa $read\_1.fastq $read\_2.fastq **>** $aln.sam |

* + 1. **Bowtie2**

The first step consists in building the reference indexes

|  |
| --- |
| **$ bowtie2-build** $ref.fa $ref\_indexes |

Once completed the current directory will contain new files that all start with ref\_indexes and end with .1.bt2, .2.bt2, .3.bt2, .4.bt2, .rev.1.bt2, and .rev.2.bt2. These files constitute the index.

At this point alignment can take place

|  |
| --- |
| **$bowtie2 -x $ref\_indexes** \  **-1** $reads\_1.fastq **-2** $reads\_2.fastq \  **-S** $aln.sam |

* + 1. **Sorting**

Then we convert the SAM file to BAM using samtools,

|  |
| --- |
| **$ samtools view -b** $aln.sam **>** $aln.bam |

and sort and index the BAM file.

|  |
| --- |
| **$ samtools sort** $aln.bam $aln.sorted  **$ samtools index** $aln.sorted.bam |

* 1. **Preprocessing of the file**

In order to perform the preprocessing of the file, the following additional files are needed:

Mills\_and\_1000G\_gold\_standard.indels.b37.vcf

1000G\_phase1.indels.b37.vcf

dbsnp\_138.b37.vcf

These files can be found in the GATK bundle in /bundle/2.8/b37 at:

<ftp://gsapubftp-anonymous@ftp.broadinstitute.org/bundle>

* + 1. **Duplicate removal**

The duplicates are marked using Picard tools,

|  |
| --- |
| **$ java -jar** $picard **MarkDuplicates** \  **INPUT=**$aln.sorted.bam \  **OUTPUT=**$aln.sorted.prededup.bam \  **METRICS\_FILE=**$metrics.txt |

The following command line removes the duplicates,

|  |
| --- |
| **$ samtools view -hb -F 0xF40** $aln.sorted.dedup.bam **>** $aln.cleaned.bam |

We use Picard to label the bam headers,

|  |
| --- |
| **$java -jar** $picard **AddOrReplaceReadGroups** \  **INPUT=**$aln.cleaned.bam \  **OUTPUT=**$aln.sorted.L.bam \  **RGLB**=Library \  **RGPL=**ILLUMINA\  **RGPU=**PlatformUnit \  **RGSM=**SampleName |

And index the resulting file:

|  |
| --- |
| **$ samtools index** $aln.sorted.L.bam |

* + 1. **Indel Realignment**

Create the target list of intervals:

|  |
| --- |
| **$ java -jar** $gatk **-T RealignerTargetCreator** \  **-R** $ref.fa \  **-I** $aln.sorted.L.bam \  **-known** $Mills\_and\_1000G\_gold\_standard.indels.b37.vcf \  **-o** $target.intervals\_list |

The following command performs the realignment,

|  |
| --- |
| **$ java -jar** $gatk **-T IndelRealigner** \  **-R** $ref.fa \  **-I** $aln.sorted.L.bam \  **-targetIntervals** $target.intervals\_list \  **-o** $bamFile\_realign |

* + 1. **Base Quality Score Recalibration**

A recalibration of the quality scores is performed using the following two commands:

|  |
| --- |
| **$ java -jar** $gatk **-T BaseRecalibrator** \  **-R** $ref.fa \  **-I** $bamFile\_realign \  **-knownSites** $dbsnp\_138.b37.vcf \  **-knownSites** $Mills\_and\_1000G\_gold\_standard.indels.b37.vcf \  **-knownSites** $1000G\_phase1.indels.b37.vcf \  **-o** $recal\_data |

|  |
| --- |
| **$ java -jar** $gatk **-T PrintReads** \  **-R** $ref.fa \  **-I** $bamFile\_realign \  **-BQSR** $recal\_data \  **-o** $bamFile\_recal |

* 1. ***Variant calling and filtering***
     1. **GATK**

In order to perform the variant calling, the following additional files are needed:

hapmap\_3.3.b37.vcf

1000G\_omni2.5.b37.vcf

1000G\_phase1.snps.high\_confidence.b37.vcf

These files can be found in the GATK bundle in /bundle/2.8/b37 at:

<ftp://gsapubftp-anonymous@ftp.broadinstitute.org/bundle>

We consider the Haplotype Caller as the variant caller for the GATK pipeline

|  |
| --- |
| **$ java -jar** $gatk **-T HaplotypeCaller** \  **-R** $ref.fa \  **-I** $bamFile\_recal \  **--dbsnp** $dbsnp\_138.b37.vcf \  **--genotyping\_mode** DISCOVERY \  **-stand\_emit\_conf** 10 \  **-stand\_call\_conf** 30 \  **–o** $file.vcf |

Once the calls are made, SNPs extraction is performed using

|  |
| --- |
| **$ java -jar** $gatk **-T SelectVariants** \  **-R** $ref.fa \  **-V** $file.vcf \  **-selectType** SNP \  **-o** $outputSNPS.vcf |

Call filtering is performed using the VQSR command. First the SNP recalibration model is built:

|  |
| --- |
| **java -jar** $gatk **-T VariantRecalibrator** \  **-R** $ref.fa \  **-input** $outputSNPS.vcf \  **-resource:**hapmap,**known=**false,**training=**true,**truth=**true,**prior=**15.0 $hapmap\_3.3.b37.vcf \  **-resource:**omni,**known=**false,**training=**true,**truth=**true,**prior=**12.0 $1000G\_omni2.5.b37.vcf \  **-resource:**1000G,known=false,**training=**true,**truth=**false,**prior=**10.0 $1000G\_phase1.snps.high\_confidence.b37.vcf \  **-resource:**dbsnp,**known=**true,**training=**false,**truth=**false,**prior**=2.0 $dbsnp\_138.b37.vcf \  **-an DP -an QD -an FS -an SOR -an MQ -an MQRankSum -an ReadPosRankSum** \  **-mode** SNP \  **-tranche** 100.0 **-tranche** 99.9 **-tranche** 99.0 **-tranche** 90.0 \  **-recalFile** $recalibrate\_SNP.recal \  **-tranchesFile** $recalibrate\_SNP.tranches |

Where 100.0, 99.9, 99.0 and 90.0 are the thresholds introduced in section 6.1

Then the desire level of recalibration is applied. Note that the variable $recal\_level should be 100.0, 99.9, 99.0 and 90.0 in accordance with Section 6.1

|  |
| --- |
| **java -jar** $gatk **-T ApplyRecalibration** \  **-R** $ref.fa  **-input** $outputSNPS.vcf \  **-mode** SNP \  **--ts\_filter\_level** $recal\_level \  **-recalFile** recalibrate\_SNP.recal \  **-tranchesFile** $recalibrate\_SNP.tranches \  **-o** $recalibrated\_snps\_raw\_indels.vcf |

* + 1. **HTSlib (SAMtools)**

Variants calling is performed using two commands. First the mpileup command from

Samtools, and then the call command from BCFtools.

|  |
| --- |
| **$ samtools mpileup**  **-ugf** $ref.fa $bamFile\_recal **| bcftools call** \  **-vmO** v \  **-o** $file.vcf |

Once the calls are made, SNPs are extracted

|  |
| --- |
| **$ java -jar** $gatk **-T SelectVariants** \  **-R** $ref.fa \  **-V** $file.vcf \  **-selectType** SNP \  **-o** $outputSNPS |

After calling the SNPs the following filter is applied:

|  |
| --- |
| **$ bcftools filter -O** v \  **-o** $outputFilteredVCF \  **-s** $filterName \  **-i** '%QUAL>20' $outputSNPS \ |