**INTERNATIONAL ORGANISATION FOR STANDARDISATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC1/SC29/WG11**

**CODING OF MOVING PICTURES AND AUDIO**

**ISO/IEC JTC 1/SC 29/WG 11 N16727**

**ISO/TC 276/WG 5 N156**

**Geneva, CH – January 2017**

|  |  |
| --- | --- |
| **Source** | **ISO/IEC JTC 1/SC 29/WG 11****ISO/TC 276/WG 5** |
| **Status** | **Final** |
| **Title** | **Benchmark framework for lossy compression of sequencing quality values** |

**Table of contents**

[1 Executive Summary 2](#_Toc472976121)

[1.1 Notes to the new edition of this document 2](#_Toc472976122)

[2 Types of data 2](#_Toc472976123)

[2.1 Human genome 2](#_Toc472976124)

[2.2 Metagenomics 3](#_Toc472976125)

[2.3 Cancer genomes 3](#_Toc472976126)

[2.4 Other species 3](#_Toc472976127)

[3 Applications and toolchains 3](#_Toc472976128)

[3.1 Genomic variation detection 3](#_Toc472976129)

[3.1.1 Alignment and variant calling 3](#_Toc472976130)

[3.1.2 Copy number variation and large structural variant detection 3](#_Toc472976131)

[3.2 RNA-seq quantification and differential expression evaluation 4](#_Toc472976132)

[3.3 Other applications and toolchains 4](#_Toc472976133)

[4 Data 4](#_Toc472976134)

[4.1 Human genome 4](#_Toc472976135)

[4.2 Human genome assembly 5](#_Toc472976136)

[4.3 Variant calling “golden reference” 5](#_Toc472976137)

[4.4 Metagenomics, cancer genomes, and other species 5](#_Toc472976138)

[5 Metrics 5](#_Toc472976139)

[5.1 Sensitivity/recall and precision 5](#_Toc472976140)

[5.2 Benchmarking tools 6](#_Toc472976141)

[6 References 6](#_Toc472976142)

[7 Appendix A 8](#_Toc472976143)

[7.1 Alignment with Bowtie2 8](#_Toc472976144)

[7.2 Sorting and indexing 8](#_Toc472976145)

[7.3 Duplicate marking 8](#_Toc472976146)

[7.4 INDEL realignment 9](#_Toc472976147)

[7.5 Base quality score recalibration (BQSR) 9](#_Toc472976148)

[7.6 SNP calling with GATK 10](#_Toc472976149)

# Executive Summary

High-throughput 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 analyses is extremely diversified.

The focus of efficient compression of QVs is currently shifting from the lossless approach adopted by tools such as Samtools [1] and other more optimized implementations [2] to lossy schemes recently appeared in literature [3]–[7]. However, because of the specific characteristics of QVs, “traditional” methods of measuring the impact of lossy compression (e.g. mean squared error (MSE)) do not consider that in certain cases errors are acceptable, whereas in other cases a high fidelity of QVs is required. The evaluation of lossy QV compression should take into account how the QVs are actually used in downstream analyses. The goal of this document is the design of a benchmark framework to measure the “perceived” impact of lossy QV compression on the results produced by various processing pipelines. Such framework will constitute the base for the comparison of different approaches to lossy compression of QVs. The identification of evaluation methodologies to form such framework requires the definition of:

* the types of data to be analyzed,
* the genomic analysis applications and toolchains in scope,
	+ methodologies to cope with possible special behavior of the genomic analysis applications and toolchains in scope,
* the specific data to be analyzed,
* the metrics to evaluate the impact of lossy compression on subsequent analyses.

## Notes to the new edition of this document

This is an updated document referring to the output document N16525/N119 with the following amendments:

* Section 7.2: The filtering of unmapped reads has been removed.
* Section 7.3: The filtering of duplicate reads has been removed.

# Types of data

## Human genome

A wide variety of biomedical applications is currently growing around human genomic variation detection. Such applications include:

* disease genetics studies: the study of the relation between gene variations and disease state,
* pharmacogenomic studies: the 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 and other 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. However, there is still not enough bias in the scientific community towards the use of a specific analysis tool or a set of tools.

## Cancer genomes

Genomic variation detection of genetic material extracted from tumor cells can play an important role in oncology with the possibility to define targeted and personalized therapies. There are some tools for the detection of somatic single nucleotide variants (sSNVs) in cancer genome sequencing data [8]–[10]. However, there is still not enough bias in the scientific community towards the use of a specific tool or a set of tools.

## 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 genomic variation detection is not yet present in the scientific community.

# Applications and toolchains

## Genomic variation detection

In general, genomic variations can be split up into the following categories which cover single-base variations up to chromosomal-level alterations [11]:

* single nucleotide variants (SNVs),
* (small) insertions and deletions (INDELs),
* copy number variations (CNVs),
* (large) structural variants (SVs).

SNV and INDEL calling is also often referred to as “variant calling”. While there exist many algorithms for genomic variation detection, they are all geared towards different types of variation. Thus, there is no tool being considered a community standard [12].

### Alignment and variant calling

It has been identified in document N16135, that reference-based alignment tools BWA [13] and Bowtie 2 [14], [15] are widely used. For the task of variant calling, in document N16135, the GATK Haplotype Caller [16] and SAMtools in conjunction with BCFtools [1] were chosen.

### Copy number variation and large structural variant detection

Other types of variations such as CNV and SV calling are not (yet) in the scope of this report, as we were not able to identify a clear bias towards certain tools or pipelines used yet. However, Zhao et al. [17] provide an overview of several tools for CNV detection. Consensus candidates might be the tool DELLY [18] (for CNV calling) and LUMPY [19] (for SV calling).

## RNA-seq quantification and differential expression evaluation

High-throughput mRNA sequencing (RNA-seq) quantification tools count the abundances of RNA transcripts from the raw FASTQ files. Differential expression tools take as input the RNA-seq quantification data (i.e. the “counts”) to look for genes (or transcripts) that have been differentially expressed across conditions. Some well-established tools are e.g. the Cufflinks pipeline [20] and the RSEM pipeline [21]. Newer tools are e.g. kallisto [22] as well as Sailfish and its successor Salmon [23].

We have put together a DGE pipeline consisting of HISAT2, StringTie, and DESeq2 [24]–[26] and have run tests to identify differentially expressed genes between two conditions of the yeast S. Cerevisiae: the wild type and a mutant. We selected six biological replicates per condition from a dataset of highly replicated yeast samples for the purpose of differential gene expression testing [44]. The pipeline was run on two settings: with lossy compression of quality values and without compression. A ranked list of the most up- and down- regulated genes was produced. Preliminary results show that RNA-seq experiments have a limited dynamic range and therefore RNA-seq experiments will now not be considered a part of this benchmark framework. However, such experiments should be taken further into account in the scope of this activity to document and proof that lossy compression does only marginally affect the results.

## Other applications and toolchains

Other applications of genomic information processing that are growing rapidly but are not mature enough to see a clear prevalence of tools or methodologies include:

1. de-novo assembly,
2. ChIP-seq.

# Data

All data mentioned in the scope of this document can be found here:

* <http://www.tnt.uni-hannover.de/cloud/mpeg-omics/>
* <http://www.tnt.uni-hannover.de/cloud/tc276-omics/>

The user names are mpeg-omics and tc726-omics, respectively. The mpeg-omics password is the same as for the MPEG document website. The tc726-omics password can be obtained from the ISO TC 276/WG 5 secretary Björn Hermes (bjoern.hermes@din.de).

## Human genome

We are considering individual NA12878. This individual has been sequenced using different technologies and configurations. To make the comparison more relevant we are going to select the following datasets, where the dataset with ID 11 was sequenced with IonTorrent technology and the dataset with ID 12 with Illumina technology. For the variant calling, we only consider chromosomes 11 and 20.

Table 1: Human whole genome sequencing data

|  |  |  |  |
| --- | --- | --- | --- |
| **Category** | **Name** | **File name** | **Input ID** |
| Human | NA12878-SRX517292 | SRR1238539.fastq | 11 |
| Human | NA12878 Garvan replicate J | NA12878\_V2.5\_Robot\_2\_R1.fastq | 12 |
| NA12878\_V2.5\_Robot\_2\_R2.fastq |

## 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.

## Variant calling “golden reference”

The “golden reference” for variant calling considered are the NIST Genome in a Bottle (GIAB) high confidence variant calls version 3.2.2. This data set contains 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.

## Metagenomics, cancer genomes, and other species

The existing MPEG dataset contains cancer cell data that will be used to evaluate performance of lossy compression on downstream analysis. We are going to select the following datasets.

Table 1: Cancer genome sequencing data

|  |  |  |  |
| --- | --- | --- | --- |
| **Category** | **Name** | **File name** | **Input ID** |
| Cancer cell lines | WGS-cancer-300x | WGS-cancer-300x.bam | 27 |
| Cancer cell lines | WGS-cancer-100x | WGS-cancer-100x.bam | 28 |
| Cancer cell lines | WGS-cancer-30x | WGS-cancer-30x.bam | 29 |

# Metrics

## Sensitivity/recall and precision

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

* *Sensitivity (or Recall)* defined as: $S=\frac{TP}{TP+FN}$

This metric provides a measure of the correctness of positives calls.

* *Precision*defined as: $P=\frac{TP}{TP+FP}$

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

Where:

* TP = number of variants in the gold standard that have been called and marked as positive;
* FN = number of variants in the gold standard that have not been called or have been called but marked as negative;
* FP = 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:

$$F=2∙\frac{S∙P}{S+P}$$

The F-score 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): 90, 99, 99.9, 100.

## Benchmarking tools

We recommend the use of the benchmarking tools proposed by the Global Alliance for Genome and Health (GA4GH). The benchmarking is based mainly on the Haplotype Comparison Tools (hap.py[[1]](#footnote-1)), developed by Illumina. Hap.py needs the following files to run:

* the VCF file containing the “golden reference”,
* the BED file containing the confident regions of the golden reference,
* the VCF file generated after running the variant calling pipeline as defined in document N16324/N100.

Also, the following option is recommended: --roc VQLSOD. This option will allow the creation of a ROC curve-like curve based on Recall and Precision. This plot is more consistent than the traditional ROC curve as it does not use false negatives and true negatives, which for this case are ill-defined. Thus, the recommended command to run hap.py is as following:

|  |
| --- |
| $ **python hap.py** $gt\_vcf $calls\_VCF \ **-f** $gt\_bed \ **-o** $happy\_root \ **-r** $ref \ **--roc** VQLSOD |

where $gt\_vcf and $gt\_bed are the paths to the golden reference and its associated BED file, respectively. $calls\_VCF is the VCF file output by the variant calling pipelines after variant filtering, $happy\_root is the output files root and $ref is the reference genome in FASTA format. The output files can be summarized in an HTML file using the benchmarking tool rep.py[[2]](#footnote-2) from the GA4GH.

# References

[1] 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.

[2] J. K. Bonfield and M. V Mahoney, “Compression of FASTQ and SAM Format Sequencing Data,” *PLoS One*, vol. 8, no. 3, p. e59190, 2013.

[3] 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.

[4] 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, no. 187, 2013.

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

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

[7] I. Ochoa, M. Hernaez, R. Goldfeder, T. Weissman, and E. Ashley, “Effect of lossy compression of quality scores on variant calling,” *Brief. Bioinform.*, Mar. 2016.

[8] Q. Wang, P. Jia, F. Li, H. Chen, H. Ji, D. Hucks, K. B. Dahlman, W. Pao, and Z. Zhao, “Detecting somatic point mutations in cancer genome sequencing data: a comparison of mutation callers,” *Genome Med.*, vol. 5, no. 10, p. 91, Jan. 2013.

[9] K. Cibulskis, M. S. Lawrence, S. L. Carter, A. Sivachenko, D. Jaffe, C. Sougnez, S. Gabriel, M. Meyerson, E. S. Lander, and G. Getz, “Sensitive detection of somatic point mutations in impure and heterogeneous cancer samples,” *Nat. Biotechnol.*, vol. 31, no. 3, pp. 213–9, Mar. 2013.

[10] D. C. Koboldt, Q. Zhang, D. E. Larson, D. Shen, M. D. McLellan, L. Lin, C. A. Miller, E. R. Mardis, L. Ding, and R. K. Wilson, “VarScan 2: Somatic mutation and copy number alteration discovery in cancer by exome sequencing,” *Genome Res.*, vol. 22, no. 3, pp. 568–76, Mar. 2012.

[11] L. Tattini, R. D’Aurizio, and A. Magi, “Detection of genomic structural variants from next-generation sequencing data,” *Front. Bioeng. Biotechnol.*, vol. 3, p. 92, Jan. 2015.

[12] K. Lin, G. Bonnema, G. Sanchez-Perez, and D. De Ridder, “Making the difference: Integrating structural variation detection tools,” *Brief. Bioinform.*, vol. 16, no. 5, pp. 852–64, 2014.

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

[14] B. Langmead, C. Trapnell, M. Pop, and S. L. Salzberg, “Ultrafast and memory-efficient alignment of short DNA sequences to the human genome,” *Genome Biol.*, vol. 10, no. 3, p. R25, 2009.

[15] B. Langmead and S. L. Salzberg, “Fast gapped-read alignment with Bowtie 2,” *Nat. Methods*, vol. 9, no. 4, pp. 357–359, 2012.

[16] A. McKenna, M. Hanna, E. Banks, A. Sivachenko, K. Cibulskis, A. Kernytsky, K. Garimella, D. Altshuler, S. Gabriel, M. Daly, and M. A. DePristo, “The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data,” *Genome Res.*, vol. 20, no. 9, pp. 1297–1303, 2010.

[17] M. Zhao, Q. Wang, Q. Wang, P. Jia, and Z. Zhao, “Computational tools for copy number variation (CNV) detection using next-generation sequencing data: features and perspectives,” *BMC Bioinformatics*, vol. 14, no. 11, p. S1, Jan. 2013.

[18] T. Rausch, T. Zichner, A. Schlattl, A. M. Stütz, V. Benes, and J. O. Korbel, “DELLY: Structural variant discovery by integrated paired-end and split-read analysis,” *Bioinformatics*, vol. 28, no. 18, pp. 333–9, 2012.

[19] R. M. Layer, C. Chiang, A. R. Quinlan, and I. M. Hall, “LUMPY: a probabilistic framework for structural variant discovery,” *Genome Biol.*, vol. 15, no. 6, p. R84, Jan. 2014.

[20] C. Trapnell, B. A. Williams, G. Pertea, A. Mortazavi, G. Kwan, M. J. van Baren, S. L. Salzberg, B. J. Wold, and L. Pachter, “Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation,” *Nat. Biotechnol.*, vol. 28, no. 5, pp. 511–515, May 2010.

[21] B. Li and C. N. Dewey, “RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome.,” *BMC Bioinformatics*, vol. 12, no. 1, p. 323, Jan. 2011.

[22] N. L. Bray, H. Pimentel, P. Melsted, and L. Pachter, “Near-optimal probabilistic RNA-seq quantification,” *Nat. Biotechnol.*, vol. 34, no. 5, pp. 525–527, Apr. 2016.

[23] R. Patro, S. M. Mount, and C. Kingsford, “Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms,” *Nat. Biotechnol.*, vol. 32, no. 5, pp. 462–464, May 2014.

[24] D. Kim, B. Langmead, and S. L. Salzberg, “HISAT: a fast spliced aligner with low memory requirements,” *Nat. Methods*, vol. 12, no. 4, pp. 357–360, Mar. 2015.

[25] M. Pertea, G. M. Pertea, C. M. Antonescu, T.-C. Chang, J. T. Mendell, and S. L. Salzberg, “StringTie enables improved reconstruction of a transcriptome from RNA-seq reads,” *Nat. Biotechnol.*, vol. 33, no. 3, pp. 290–295, 2015.

[26] M. I. Love, W. Huber, and S. Anders, “Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2,” *Genome Biol.*, vol. 15, no. 12, p. 550, 2014.

# Appendix A

This section provides details on the specific configuration of the pipeline listed in Section 3 as representative toolchain to perform genomic variation detection. Note: In the following commands, the words proceeded by “$” are user-defined input/output files.

The following tool versions shall be used:

* Bowtie 2 version 2.2.5
* Picard version 2.4.1
* Samtools version 1.3 (built with HTSlib version 1.3)
* GATK version 3.6

To perform the preprocessing of the file and the variant calling, the following additional files are needed:

* Mills\_and\_1000G\_gold\_standard.indels.b37.vcf
* 1000G\_phase1.indels.b37.vcf
* dbsnp\_138.b37.vcf
* hapmap\_3.3.b37.vcf
* 1000G\_omni2.5.b37.vcf
* 1000G\_phase1.snps.high\_confidence.b37.vcf

## Alignment with Bowtie2

The first step consists in building the reference indexes:

|  |
| --- |
| **$ bowtie2-build** $ref.fa $idx |

Once completed the current directory will contain new files that all start with $idx 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** $idx \ **-1** $reads\_1.fq **-2** $reads\_2.fq \ **-S** $aln.sam |

## Sorting and indexing

Then we convert the SAM file to the BAM format using SAMtools.

|  |
| --- |
| **$ samtools view -@ -bh** $aln.sam **>** $aln.bam |

Then we sort and index the BAM file:

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

## Duplicate marking

The duplicates are marked in the BAM file using Picard tools:

|  |
| --- |
| **$ java -jar** $picard **MarkDuplicates** \ **I=**$sorted.bam \ **O=**$prededup.bam \ **M=**$metrics.txt \ **ASSUME\_SORTED=**true |

We use Picard to label the BAM headers:

|  |
| --- |
| **$ java -jar** $picard **AddOrReplaceReadGroups** \ **I=**dedup.bam \ **O=**$label.bam \ **RGID=**1 **RGLB=**Library \ **RGPL=**Illumna\ **RGPU=**PlatformUnit \ **RGSM=**SampleName |

Then we index the resulting file:

|  |
| --- |
| **$ samtools index** $label.bam |

## INDEL realignment

Create the target list of intervals:

|  |
| --- |
| **$ java -jar** $gatk **-T RealignerTargetCreator** \  **-R** $ref.fa \ **-I** $label.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** $label.bam \ **-targetIntervals** $target.intervals\_list \ **-o** $realign.bam |

## Base quality score recalibration (BQSR)

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

|  |
| --- |
| **$ java -jar** $gatk **-T BaseRecalibrator** \ **-R** $ref.fa \ **-I** $realign.bam \ **-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** $realign.bam \  **-BQSR** $recal\_data \  **-o** $recal.bam  |

## SNP calling with GATK

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

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

Once the calls are made, SNPs extraction is performed using the following command:

|  |
| --- |
| **$ java -jar** $gatk **-T SelectVariants** \ **-R** $ref.fa \  **-V** $calls.vcf \ **-selectType** SNP \ **-o** $snps.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** $snps.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** $snps.recal \ **-tranchesFile** $snps.tranches \ **-rscriptFile** $snps.r |

Where 100.0, 99.9, 99.0 and 90.0 are the thresholds introduced in Section 5.1. Then the desired 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 5.1.

|  |
| --- |
| **java -jar** $gatk **-T ApplyRecalibration** \ **-R** $ref.fa  **-input** $outputSNPS.vcf \ **-mode** SNP \ **--ts\_filter\_level** $recal\_level \ **-recalFile** snps.recal \ **-tranchesFile** $snps.tranches \ **-o** $recal.vcf |

1. https://github.com/Illumina/hap.py [↑](#footnote-ref-1)
2. <https://github.com/ga4gh/benchmarking-tools/blob/master/reporting/basic/bin/rep.py> [↑](#footnote-ref-2)