**INTERNATIONAL ORGANISATION FOR STANDARDISATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

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

**CODING OF MOVING PICTURES AND AUDIO**

**ISO/IEC JTC1/SC29/WG11 N16326**

**Geneva, CH – June 2016**

|  |  |
| --- | --- |
| **Source** | **Requirements** |
| **Status** | **Draft** |
| **Title** | **Benchmark framework for lossy compression of genome sequencing quality values** |

**Table of contents**

[1 Executive summary 2](#_Toc452656924)

[2 Types of data 2](#_Toc452656925)

[2.1 Human Genome 2](#_Toc452656926)

[2.2 Metagenomics 3](#_Toc452656927)

[2.3 Cancer genomes 3](#_Toc452656928)

[2.4 Other species 3](#_Toc452656929)

[3 Applications and toolchains 3](#_Toc452656930)

[3.1 Genomic variation detection 3](#_Toc452656931)

[3.1.1 Alignment and variant calling 3](#_Toc452656932)

[3.1.2 Copy number variation and large structural variant detection 4](#_Toc452656933)

[3.2 RNA-Seq quantification and differential expression evaluation 4](#_Toc452656934)

[3.3 Other applications and toolchains 4](#_Toc452656935)

[4 Noise floor of genomic information processing pipelines 4](#_Toc452656936)

[5 Data 5](#_Toc452656937)

[5.1 Human Genome 5](#_Toc452656938)

[5.2 Human genome assembly 5](#_Toc452656939)

[5.3 Variant calling “golden reference” 6](#_Toc452656940)

[5.4 Variant calling using simulated data 6](#_Toc452656941)

[5.5 Metagenomics, cancer cells and other species 7](#_Toc452656942)

[6 Metrics 7](#_Toc452656943)

[6.1 Sensitivity and precision 7](#_Toc452656944)

[6.2 Area under the ROC curve 8](#_Toc452656945)

[7 References 9](#_Toc452656946)

[8 Appendix A 10](#_Toc452656947)

[8.1 Alignment and variant calling 11](#_Toc452656948)

[8.1.1 BWA-MEM 11](#_Toc452656949)

[8.1.2 Bowtie 2 11](#_Toc452656950)

[8.1.3 Sorting 12](#_Toc452656951)

[8.1.4 Duplicate removal 12](#_Toc452656952)

[8.1.5 Indel realignment 12](#_Toc452656953)

[8.1.6 Base quality value recalibration 13](#_Toc452656954)

[8.1.7 GATK 13](#_Toc452656955)

[8.1.8 SAMtools and BCFtools 14](#_Toc452656956)

[8.2 Copy number variation and large structural variant detection 14](#_Toc452656957)

[8.2.1 LUMPY 14](#_Toc452656958)

[8.3 RNA-Seq quantification and differential expression evaluation 14](#_Toc452656959)

[8.3.1 Cufflinks 14](#_Toc452656960)

[8.3.2 RSEM 14](#_Toc452656961)

# 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.

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]. In some cases, lossy compression of QVs is not only acceptable but seems to actually improve the performance of subsequent analyses. 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 take into account 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 (both test sets and references)
* The metrics to evaluate the impact of lossy compression on subsequent analyses

# 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.

## 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 [9]–[11]. 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 [8]:

* Single nucleotide variants (SNVs)
* (Small) insertions and deletions (indels)
* Copy number variations (CNVs)
* (Large) structural variants (SVs)

SNV and indel calling (i.e. variant calling) together is also often referred to as “variant calling”.

### Alignment and variant calling

The MPEG output document N16324 mentions four pipelines for alignment and variant calling. These pipelines were built using the reference-based alignment tools BWA [12] and Bowtie (2) [13], [14]. Although BWA and Bowtie are very widely used tools, there are other reference-based alignments tools which have been designed with maximum speed in mind: the GEM mapper [15] and mrsFAST [16]. These two tools should be considered as an extension to the benchmark framework, as they would have a minor speed impact on the benchmark performance.

For the task of variant calling, document [1] lists GATK’s Haplotype Caller [17] and SAMtools in conjunction with BCFtools [1]. We propose to add Platypus [18] to the list of variant calling tools.

|  |  |  |
| --- | --- | --- |
| **ID** | **Pipeline** | **Configuration** |
| AVC1 | Bowtie 2 + GATK HC | See Appendix A |
| AVC2 | Bowtie 2 + SAMtools + BCFtools | See Appendix A |
| AVC3 | BWA-MEM + GATK HC | See Appendix A |
| AVC4 | BWA-MEM + SAMtools + BCFtools | See Appendix A |

### Copy number variation and large structural variant detection

We propose to include CNV and SV calling as well, although we were not able to identify a clear bias towards certain tools or pipelines used yet. However, Zhao et al. [19] provide an overview of several tools for CNV detection. A framework for the detection of large SVs is for example LUMPY [20].

|  |  |  |
| --- | --- | --- |
| **ID** | **Pipeline** | **Configuration** |
| CNV1 | To be identified | To be identified |
| LSV1 | LUMPY | To be identified |

## 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”) in order to look for genes (or transcripts) that have been differentially expressed across conditions. Some well-established tools are e.g. the Cufflinks pipeline [21] and the RSEM pipeline [22]. Newer tools are e.g. kallisto [23] as well as Sailfish and its successor Salmon [24]. These tools reduce the quantification computing time by several orders of magnitude. However, kallisto does not use QVs and is therefore not suited for the benchmark. Sailfish and Salmon seem to be under active development. Thus, we selected Cufflinks and RSEM for incorporation into the lossy compression assessment methodology.

|  |  |  |
| --- | --- | --- |
| **ID** | **Pipeline** | **Configuration** |
| RNAS1 | Cufflinks | To be identified |
| RNAS2 | RSEM | To be identified |

## 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

# Noise floor of genomic information processing pipelines

In 2016, Firtina and Alkan [25] showed how mapping and variant calling pipelines are affected when only the read order in a FASTQ file changes. One example illustrating this behavior is that some software relies on the fact that the reads are sorted randomly (as they come off a sequencing machine). For instance, alignment tools might try to align the first N reads in order to derive the distribution of template/insert sizes (assuming a paired-end sequencing protocol) and then start a more thorough alignment armed with that knowledge [12]. Subsequently, the aligned data might be sorted (with e.g. SAMtools [1]). However, taking this data and attempting to realign may lead to very poor results as the computation of the insert size from the first N reads will likely be incorrect (the start of each chromosome is the telomere – usually highly repetitive and very hard to sequence or accurately place reads within).

However, the original read order is the result of a stochastic process and does not carry any information. Nevertheless, compression tools which reorder the reads such as ORCOM [26] or SCALCE [27] may result in changes of mapping loci and variant calls. We call this behavior the “noise level” of a processing pipeline. In a broader sense, this means that tools in a typical processing pipeline might be non-deterministic either because of the nature of the employed algorithm(s) or because of implementation compromises (multi-threading, block-wise processing, etc.).

The benchmark framework has to distinguish between the effects of lossy compression and these non-deterministic effects. One way to do so would be to characterize the statistical properties of such noise so that the effects of lossy compression of QVs could be clearly identified.

# 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 | IonTorrent | SRX517292 |
| 2 | Replicate JIllumina8-binned QVs30x | [Garvan](http://www.garvan.org.au/research/kinghorn-centre-for-clinical-genomics/clinical-genomics/sequencing-services/sample-data) (robot 2) |
| 3 | IlluminaNot binned50x | [SMaSH dataset (Berkeley)](http://smash.cs.berkeley.edu/datasets.html) |
| 4 | Run SRR1231836 of experiment accession SRX514833 stored on the DDBJ repository  | [SRX514833](https://trace.ddbj.nig.ac.jp/DRASearch/experiment?acc=SRX514833)  |
|  |  |  |
| 5 | PacBioNot binned44x | ftp://ftp-trace.ncbi.nlm.nih.gov/giab/ftp/data/NA12878/NA12878\_PacBio\_MtSinai/sorted\_final\_merged.bam |

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 references for variant 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/) 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).

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).

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 |

## 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 QV compression on downstream analysis.

Moreover, 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.

# 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 [5] and [7] 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 QVs, and
* A L is the area under the curve obtained with lossy compression of QVs.

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.

# 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. 3122–9, 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, p. 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. 240–3, 2015.

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

[9] 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.

[10] 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.

[11] 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.

[12] H. Li and R. Durbin, “Fast and accurate short read alignment with Burrows-Wheeler transform,” *Bioinformatics*, vol. 25, no. 14, pp. 1754–60, 2009.

[13] 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, pp. R25.1–10, 2009.

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

[15] S. Marco-Sola, M. Sammeth, R. Guigó, and P. Ribeca, “The GEM mapper: fast, accurate and versatile alignment by filtration,” *Nat. Methods*, vol. 9, no. 12, pp. 1185–1188, Oct. 2012.

[16] F. Hach, F. Hormozdiari, C. Alkan, F. Hormozdiari, I. Birol, E. E. Eichler, and S. C. Sahinalp, “mrsFAST: a cache-oblivious algorithm for short-read mapping,” *Nat. Methods*, vol. 7, no. 8, pp. 576–7, Aug. 2010.

[17] 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. 1297–303, Sep. 2010.

[18] A. Rimmer, H. Phan, I. Mathieson, Z. Iqbal, S. R. F. Twigg, WGS500 Consortium, A. O. M. Wilkie, G. McVean, and G. Lunter, “Integrating mapping-, assembly- and haplotype-based approaches for calling variants in clinical sequencing applications,” *Nat. Genet.*, vol. 46, no. 8, pp. 912–8, Aug. 2014.

[19] 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.

[20] 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.

[21] 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. 511–5, May 2010.

[22] 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.

[23] 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.

[24] 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. 462–4, May 2014.

[25] C. Firtina and C. Alkan, “On genomic repeats and reproducibility,” *Bioinformatics*, vol. Advance Ac, Mar. 2016.

[26] S. Grabowski, S. Deorowicz, and Ł. Roguski, “Disk-based compression of data from genome sequencing,” *Bioinformatics*, vol. 31, no. 9, pp. 1389–95, May 2015.

[27] F. Hach, I. Numanagic, C. Alkan, and S. C. Sahinalp, “SCALCE: boosting sequence compression algorithms using locally consistent encoding,” *Bioinformatics*, vol. 28, no. 23, pp. 3051–7, Dec. 2012.

# Appendix A

This section provides details on the specific configuration of the pipelines listed in Section 3 as representative toolchains to perform genomic variation detection.

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

## Alignment and variant calling

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

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

### 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 |

### Bowtie 2

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 |

### 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 |

### 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 |

### 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 |

### Base quality value 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  |

### GATK

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 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 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 |

### SAMtools and BCFtools

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 \ |

## Copy number variation and large structural variant detection

### LUMPY

The detailed configuration of this pipelines still needs to be identified.

## RNA-Seq quantification and differential expression evaluation

### Cufflinks

The detailed configuration of this pipelines still needs to be identified.

### RSEM

The detailed configuration of this pipelines still needs to be identified.