On the compression of FASTQ quality-scores

Diogo Pratas
pratas@ua.pt
Armando J. Pinho
ap@ua.pt

Signal Processing Lab, IEETA / DETI,
University of Aveiro,
3810–193 Aveiro, Portugal

Abstract

The genomics sequencing centers and the scientific community are being flooded with data. A single sequencing machine can nowadays generate more data in one day than any existing machine could have produced throughout the entire year of 2005. The pressure to find efficient compression algorithms for next-generation sequencing (NGS) data is being felt worldwide.

The NGS most used format, FASTQ, contains three channels of information, namely headers, DNA bases and quality-scores (QS). Together with the DNA bases, the QS are the most demanding components in terms of storage requirements and compressibility. In this paper, we analyze the QS using several compression techniques aiming to find the best compression model that describes this type of data.

1 Introduction

The scientific community is currently facing a shift in concern that was probably not foreseen until recently: Genomic data are been produced at a pace that exceeds the growth of the media capacity to store them. Nowadays, it is common to find genomics sequencing projects having a larger fraction of the budget allocated to the computational infrastructure (including the storage component), than to the biological part.

Modern sequencing instruments are able to generate at least hundreds of millions of short reads of genomic data and to store them in files with a specific format, such as the most used FASTQ [4] format. The FASTQ format, originally developed at the Wellcome Trust Sanger Institute, is a text-based format for storing biological sequences (DNA bases) and its corresponding QS, where both are identified with headers and encoded with ASCII.

Several compression algorithms have been proposed to date. Tembe *et al.* proposed G-SQZ [12] and DSRC [5] was proposed by Deorowicz *et al.*. Both methods split the data into separate channels, namely headers, sequences and QS, and compress them individually using the Lempel-Zip algorithm and Huffman coding.

Kozanitis *et al.* and Hsi-Yang Fritz *et al.* proposed, respectively, SlimGene [9] and mzip [6]. These are reference-based compression methods that exploit the redundant nature of the data by aligning reads to a known reference genome sequence and storing genomic positions instead of nucleotide sequences. Wan *et al.* proposed Q-scores [13], a method based on lossless and lossy transformations for the compression of FASTQ QS. Very recently, Jones *et al.* proposed Quip [7], a lossless compressor based on the Markov property, reference-based compression (for the DNA bases) and arithmetic coding, with fast execution and low memory usage. However, these algorithms seem to have difficulties when compressing the QS. Therefore, this work aims to study the nature of the QS, using compression techniques, in order to serve as a starting point for a future state-of-the-art compression algorithm.

2 Methods and Results

Compression can be seen as a two stage process: modelling and coding. Modelling is the process of studying the characteristics of the data source to estimate the probability distribution of data. Coding involves the transformation of data into a sequence of bits according to the probability distribution acquired in the modelling process. Therefore, compression of QS data is a problem that is tightly related with QS information modelling. In order to study the model that better describes the QS, we have compressed ten sequences from the 1000 Genomes Project[1].

The results presented in Table 1 show better ratios for the compressors based on finite-context modelling (FCM), comparing with BZIP2 (based on the Burrows-Wheeler transform [3] and LZ77 algorithms) and GZIP (based on the DEFLATE algorithm). Moreover, for the most of the files, an order-3 seems to produce the best compression ratios.

[1] http://www.1000genomes.org/

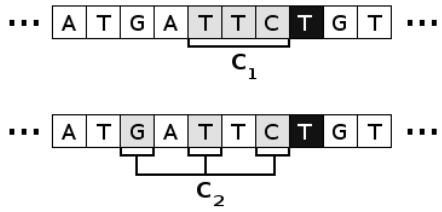

Figure 1: Context templates of finite-context models. First layer shows a regular template with a context size of three. The second layer shows a sparse template using also a context size of three.

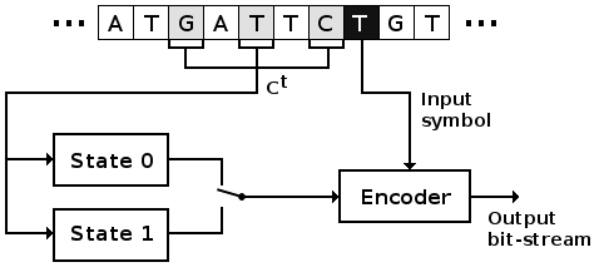

Figure 2: Two-state model using a sparse template with context order of three, exploiting the two-symbol periodicity of the QS.

Thereafter, we have implemented a finite-context model [1] using a sparse template (c135), as can be seen in the second layer of Fig. 1. This implementation improves the compression ratios, if compared with a regular template (presented in the first layer of Fig 1).

Since in a previous work [11] we have explored a three-state model for compressing the coding regions of DNA sequences, characterized by a periodicity of three (codons have three nucleotides), and gathering the information that the QS at a given position is highly correlated with the score at the preceding position [9], we have explored a two-state finite-context model.

A two-state finite-context model is causal, and the decoder is able to reproduce identical probability estimates without side information. Figure 2 shows the two-state finite-context model using a sparse template with context order of three. It differs from a (one-state) finite-context model by the inclusion of two internal states. Each state is selected periodically, according to a two-symbol period (t mod 2). Each state comprises a finite-context model, similar to the one presented in the second layer of Fig. 1.

As it can be seen in Table 1, the two-state finite-context model using an order-3 sparse template (c135s2), on average, attains the best compression ratio.

Most of the recent NGS compression algorithms explore similarities between DNA bases (for instance in reference-based compression), achieving compression ratios better than those that do not use it. Therefore, our intention is to study the QS with the aim of discovering if there are similarities between different QS files.

In order to study the existence (or nonexistence) of similarities, we used the Normalized Compression Distance (NCD). Li *et al.* [10], aligning ideas from Kolomogorov [8] and based on an information distance proposed by Bennett *et al.* [2], defined this practical analog based on

Id	Filename	nα	size	P	M	Bin	GZIP	BZIP2	FCM-1	FCM-2	FCM-3	FCM-4	FCM-5	FCM-6	c135	c13s2	c135s2
S1	DRR000607	32	3,238,398	B	&	5.000	3.515	3.308	3.079	2.989	3.027	3.198	3.489	3.757	3.023	**2.944**	3.043
S2	DRR000617	32	39,931,464	!	&	5.000	1.694	1.478	1.607	1.540	1.523	1.544	2.404	2.456	**1.471**	1.490	1.472
S3	ERR001517	27	472,896	?	@	4.755	3.218	2.957	2.930	2.757	2.804	3.001	3.214	3.431	2.813	**2.736**	2.829
S4	ERR012631	33	92,698,824	#	B	5.044	4.975	4.642	4.502	4.413	4.379	4.468	4.683	4.926	4.232	4.339	**4.214**
S5	ERR020142	32	432,671,250	%	C	5.000	4.701	4.440	4.261	4.183	4.148	4.130	4.059	4.278	3.799	4.126	**3.783**
S6	ERR048940	31	17,468,418	%	C	4.954	4.063	3.779	3.728	3.612	3.592	3.665	3.933	4.065	3.430	3.540	**3.426**
S7	SRR000921	35	92,283,120	F	!	5.129	2.110	1.808	1.655	1.607	1.573	1.576	2.929	2.932	1.531	1.552	**1.513**
S8	SRR001113	40	374,619,140	I	H	5.322	3.897	3.758	3.491	3.414	3.385	3.429	4.442	-	3.370	3.359	**3.348**
S9	SRR004387	35	48,141,342	A	!	5.129	2.612	2.280	2.095	2.015	1.962	1.969	2.675	2.778	1.918	1.945	**1.905**
S10	SRR034542	32	19,940,184	#	B	5.000	4.925	4.604	4.433	4.331	4.336	4.519	4.749	4.929	**4.235**	4.287	4.254

Table 1: Rates (bits per base) for GZIP, BZIP2 and FCM-C compressors over ten sequences. The number of symbols of the alphabet is represented by nα and size stands for the number of quality scores in the file. 'P' and 'M' represent, respectively, the most and least occurring symbols. The Bin identifier expresses the upper bound, given by: $-\log_2 P(n\alpha)$. The '-' indicates an inability to run due to memory constraints.

Id	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
S1	**0.9535**	1.0002	1.0117	1.0027	1.0007	1.0066	1.0127	1.0014	1.0155	1.0057
S2	1.0000	**0.9845**	1.0041	1.0073	1.0029	1.0181	1.0368	1.0037	1.0439	1.0142
S3	1.0116	1.0040	**0.9014**	1.0008	1.0003	1.0035	1.0033	1.0003	1.0045	1.0019
S4	1.0027	1.0077	1.0008	**0.9925**	1.0157	1.0120	1.0290	1.0201	1.0313	**0.9984**
S5	1.0007	1.0027	1.0003	1.0156	**0.9978**	1.0003	1.0123	1.0295	1.0116	1.0051
S6	1.0066	1.0187	1.0035	1.0120	1.0003	**0.9770**	1.0303	1.0062	1.0429	1.0284
S7	1.0124	1.0359	1.0033	1.0287	1.0122	1.0301	**0.9877**	1.0099	1.0094	1.0342
S8	1.0014	1.0038	1.0003	1.0201	1.0301	1.0063	1.0098	**0.9945**	1.0085	1.0069
S9	1.0153	1.0430	1.0046	1.0310	1.0114	1.0427	1.0099	1.0085	**0.9844**	1.0575
S10	1.0058	1.0145	1.0019	**0.9983**	1.0052	1.0284	1.0344	1.0069	1.0578	**0.9800**

Table 2: Normalized Compression Distance (NCD) values over the ten sequences. The bold values (without underlining) represent the NCD performed with the own sequence (diagonal). The bold and underlining values represent the measures below one.

standard compressors as

$$\mathrm{NCD}(A,B) = \frac{C(AB) - \min\{C(A),C(B)\}}{\max\{C(A),C(B)\}}, \qquad (1)$$

where $C(A)$ and $C(B)$ denote, respectively, the number of bits needed by the (lossless) compression program to represent A and B, and $C(AB)$ denotes the number of bits required to compress the concatenation of A and B. Notice that, ideally, $\mathrm{NCD}(A,A) = 0$ (this implies $C(AA) = C(A)$). However, practical compression algorithms usually yield $\mathrm{NCD}(A,A) > 0$. On the other hand, if knowing A does not help reducing the size of B, then we have $C(AB) \approx C(A) + C(B)$ and $\mathrm{NCD}(A,B) \approx 1$. In conclusion, smaller values of $\mathrm{NCD}(A,B)$ indicate more similar sequences.

According to Table 2, the NCD values are all greater or equal to one, with the exception of the diagonals ($C(AA)$ scenario) and the relation of Id10 with Id4 in both directions (still remains close to one). Therefore, overall, there are not similarities between QS files, leading to infer that a reference-based compression should not be the best implementation for this type of information.

3 Conclusions

In this paper, we have analyzed the QS, using several compression techniques, in order to find the best compression model that describes this type of data, which appears to be two-state finite-context modelling using sparse context order of three. Moreover, we have seen that a reference-based compression is not ideal to this type of information, since there are not relevant similarities between QS files. Several hypotheses for testing multiple models will be considered future work.

Funding

Supported by the European Fund for Regional Development (FEDER) through the Operational Program Competitiveness Factors (COMPETE) and by the Portuguese Foundation for Science and Technology (FCT) in the context of the project FCOMP-01-0124-FEDER-022682 (FCT reference PEst-C/EEI/UI0127/2011).

References

[1] T. C. Bell, J. G. Cleary, and I. H. Witten. *Text compression*. Prentice Hall, 1990.

[2] C. H. Bennett, P. Gács, M. Li P. M. B. Vitányi, and W. H. Zurek. Information distance. *IEEE Trans. on Information Theory*, 44(4): 1407–1423, July 1998.

[3] M. Burrows and D. J. Wheeler. *A block-sorting lossless data compression algorithm*. Digital Systems Research Center, May 1994.

[4] P. J. A. Cock, C. J. Fields, N. Goto, M. L. Heuer, and P. M. Rice. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic Acids Research*, 38(6):1767–1771, 2010.

[5] S. Deorowicz and S. Grabowski. Compression of DNA sequence reads in FASTQ format. *Bioinformatics*, 27(6):860–862, 2011. doi: 10.1093/bioinformatics/btr014.

[6] M. H.-Y. Fritz, R. Leinonen, G. Cochrane, and E. Birney. Efficient storage of high throughput DNA sequencing data using reference-based compression. *Genome Research*, 21:734–740, 2011. doi: 10.1101/gr.114819.110.

[7] Daniel C. Jones, Walter L. Ruzzo, Xinxia Peng, and Michael G. Katze. Compression of next-generation sequencing reads aided by highly efficient de novo assembly. *Nucleic Acids Research*, 40(22): e171, 2012. doi: 10.1093/nar/gks754.

[8] A. N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1(1):1–7, 1965.

[9] C. Kozanitis, C. Saunders, S. Kruglyak, V. Bafna, and G. Varghese. Compressing genomic sequence fragments using SlimGene. *Journal of Computational Biology*, 18(3):401–413, 2011.

[10] M. Li, X. Chen, X. Li, B. Ma, and P. M. B. Vitányi. The similarity metric. *IEEE Trans. on Information Theory*, 50(12):3250–3264, December 2004.

[11] A. J. Pinho, A. J. R. Neves, V. Afreixo, Carlos A. C. Bastos, and P. J. S. G. Ferreira. A three-state model for DNA protein-coding regions. *IEEE Trans. on Biomedical Engineering*, 53(11):2148–2155, November 2006.

[12] W. Tembe, J. Lowey, and E. Suh. G-SQZ: compact encoding of genomic sequence and quality data. *Bioinformatics*, 26(17):2192–2194, 2010. doi: 10.1093/bioinformatics/btq346.

[13] R. Wan, V. N. Anh, and K. Asai. Transformations for the compression of FASTQ quality scores of next-generation sequencing data. *Bioinformatics*, 28(5):628–635, 2012.