BioStringCompress: A Lesson Learned

Yannick Bijl & Olga Veth

April 2020

Abstract

Biological sequences have increased in size and as a result, leads to higher demand of data compression methods. However, most existing compression methods do not take all possible symbols within biological sequences into account. With the intention of introducing novel methods within this field, as well as including all symbols, we present three compression methods: 1) *count*, 2) *bin* and 3) *bincount*. *Count* uses the count and symbol, *bin* shapes symbols into 4 bits and *bincount* is a combination of both aforementioned methods. We assessed our methods and gzip on genes and genomes. We observed that *count* as well as *bincount* performed with on average 50% and 30% compression rate, respectively. In contrast, gzip has a more effective average compression rate of 70%. Probably, low compression rate of *count* is due to lack of exact repetitions of bases. With regards to *bincount*, methods using a 2bit approach have been proven to be more effective compared to 4 bit. Overall, even though our three methods under performed against gzip, they may serve as inspiration for new compression concepts.

1 Introduction

With the advent of next- and third-generation sequencing, a wealth of data is being generated. Biological sequences can be categorized into two broad categories; DNA/RNA and protein sequences. Protein sequences are build from the twenty amino-acids. DNA and RNA sequences generally consist of the four nucleotides A, C, G, T, and U replaces T in the case of RNA. In the case of DNA and RNA there are also additional symbols (Table 1[10]) that can be used.

Biological sequencing techniques have advanced to a point that it is almost cheaper to generate the data anew, than to store it.[17] This cost depends of course on the equipment and man-hours needed for operational use. However, data storage can have incredible benefits. These benefits are now often used to further reduce costs in the long-term. A benefit can be the reproducibility of experiments. Most sequencing methods have some margin of error as they do not .[9] Recreating the data could imply generating small changes within it. This could cascade into diverging results. Thus, being able to retrieve the same data is useful when redoing experiments.

Another benefit is the possibility of data sharing. Privacy concerns aside, research benefits incredibly by having a large amount of data publicly available. This means that more research can be based on different sources of data, reducing bias. At the same time, it allows research to be done with less costs by sharing the same data among multiple groups in the same organization. Data sharing could be important for less wealthy institutes, as well as student and open-source projects, to be able to continue research.

Of course, there are more benefits. It would provide a different method of diagnosis. By storing previous sequences, one could monitor a patient over time. Showing a progression in genomic changes, which could be correlated to diseases. Overall, effectively stored data can be used for many purposes.

Different methods have been introduced to improve the efficiency of DNA/RNA sequence data storage.[9] However, these methods only take the symbols A, T, C, G into account. However, other bases except the aforementioned four exist. The symbol N is well represented in genomes, and the symbol . for gaps is commonly found in sequence alignments. Therefore we find that methods should have taken the other symbols alongside the four canonical ones into account. Here we will present three of our own methods, and compare it to the gzip compression method implemented in 7-zip. Gzip is a general compression method that has proven to compress the data without loss, and works well. Thus a successful compression algorithm for DNA and RNA should take all sixteen symbols in Table 1 into account, and perform better than gzip.

Table 1: Symbols and their meaning for DNA and RNA sequences.

Symbol	**Meaning**
A	Adenine
C	Cytosine
G	Guanine
T (or U)	Thymine (or Uracil)
R	A or G
Y	C or T
S	G or C
W	A or T
K	G or T
M	A or C
B	C or G or T
D	A or G or T
H	A or C or T
V	A or C or G
N	Any Nucleotide
.	Gap

2 Methods

2.1 Compressing Biological Sequence Data

Sequence compression can be done in several ways. We will focus on reducing the data size by using repetitions in the data, as well as using compression by reducing the number of needed bits.

A naive method to arrange the nucleotides is by combining it with numbers. Biological sequences are known to have repetitions in them.[16] It should be possible to take advantage of that through counting the repetition. The repetition is then compressed by using a number together with a single symbol. Thus 'AAAA' becomes '4A', as seen in Figure 1A. Thus, compression would occur when the same symbol occurs three times in a row. Due to the nature of this approach, we will further indicate it as *count*.

Another approach to sequence compression is reducing the number of bits in a file. The nucleotides are stored as text in 8-bit Unicode Transformation Format (UTF-8). Thus, for each characters there are 8 bits used. To encode 16 symbols, only 4 bits are needed. Therefore, it is possible to compress the data by storing a character in 4 bits instead of 8, a visual representation can be seen in Figure 1B. We will refer to this method as *bin*. As this method cuts the number of bits in half, a compression rate of 50% is expected to be achieved.

Our third method combines the concepts of the last two. *bincount* first counts the nucleotides in the same manner as the method *count*, with a maximum count of sixteen. Thereafter, each pair of number and symbol can be transformed into 8 bits. The first 4 bits would cover the number, and the last 4 bits would encode the nucleotide symbol. This conversion uses the same concept as the *bin* approach. The whole process is visualized in Figure 1C and clearly shows that the concepts of both *count* and *bin* are used.

These three methods will be compared to the compression method gzip. Gzip will be used in conjunction with tar, to create tar.gz files. This is mostly due to the common use of this format for biological sequences. Gzip uses the DEFLATE algorithm[1] for compression.

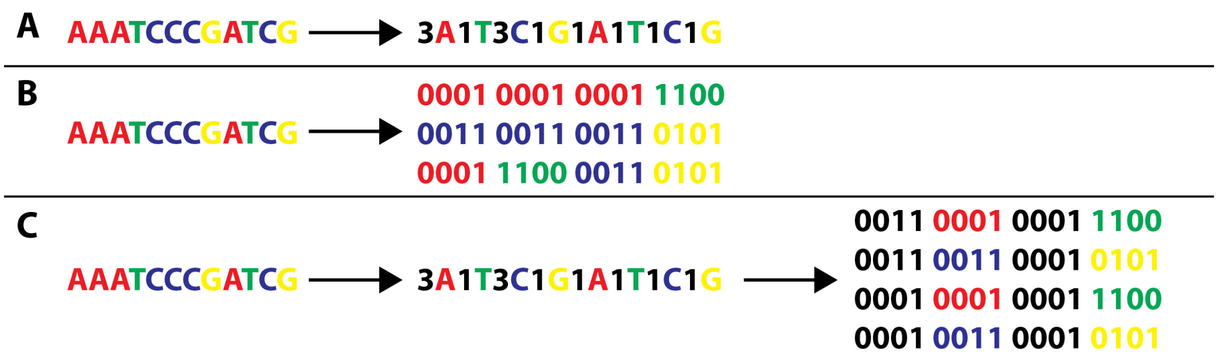

Figure 1: Compression methods. (**A**) *count*, count repeating symbols and restructure the sequence using the count and symbol;(**B**) *bin*, compress the sequence by conversion each symbol into 4 bits;(**C**) *bincount*, count repeating symbols, then compress the data by converting the counts and symbols into 4 bits each.

2.2 Measuring Performance

To measure the performance of our methods we will use these on a set of ten genes and two relatively small genomes. The performance is measured by the size of the files the compression methods generate. These sizes are converted into rates by dividing the filesize of the compression method by the filesize of the original file.

We have used the human variant of the genes AKT1[18], APOE[11], EGFR, ESR1[6], IL6[7], MTHFR[5], TGFB1[3], TNF[14], TP53[12], VEGFA[13]. We have based our test genes list on the work of *Dolgin* (2017)[2], but for the intents of this research it could just as well be random. For the genomes, we have chosen to work with the model organisms *E. coli*[8], and *S. cerevisiae*[4]. These are relatively small in comparison to the human genome, which is a a few gigabits, but serve well as an initial experiment.

3 Results

The performance of the compression methods showcase a distinct difference. Based on the results in Table 2 there is a clear difference between the methods. Ranked by order of worst to best; *count*, *bincount*, *bin*, and at the lead *gzip*. Between the methods *bincount*, *bin*, and *gzip* there is around 20 percent increase in performance. This is a large difference that showcases how small changes in algorithmic design can drastically alter the performance of a methods.

Table 2: Compression rates of genes using different methods. From left to right: The gene name; the original file size in bits; the results of the methods count, bin, bincount, and gzip in percentages.

	size	count	bin	bincount	gzip
AKT1	33874	-35,83	50,71	32,09	70,63
APOE	10764	-37,37	50,71	31,33	69,18
EGFR	248084	-40,42	50,70	29,79	71,6
ESR1	486786	-37,74	50,70	31,14	72,21
IL6	12026	-39,03	50,71	30,49	68,56
MTHFR	27766	-38,70	50,71	30,66	69,64
TGFB1	30449	-38,32	50,70	30,85	71,75
TNF	9763	-38,88	49,99	30,55	69,67
TP53	33241	-38,00	50,71	31,02	72,37
VEGFA	23605	-34,88	50,71	32,56	69,49

The *count* method clearly is the worst, as it inflates the file sizes instead of compressing them. The method *bin* on the other hand behaves as expected, every compression rate is around 50 percent. These results proves that the theory works, but is has an underperformance compared to gzip as it has a compression rate around 70 percent.

The *bincount* approach is of interest as it performs between the methods *count* and *bin*. The method uses in first instance the same approach as *count*, but does not inflate the file size. Thus, we can extrapolate that compression is possible due to the other component of *bincount*. Therefore, *bincount* proves that converting the data to contain less bits works.

Although, both methods *bin* and *bincount* are far off from gzip, it does show an interesting prospect which we will further discuss in the following section. There we will also discuss a possible reason why performance of the method *count* is low. Remarkable is that the results of all four methods are fairly consistent across the ten genes, despite having different file sizes. Therefore, these results should be representative for most genes.

Table 3: Compression rates of genomes using different methods. From left to right: The gene name; the original file size in bits; the results of the methods count, bin, bincount, and gzip in percentages.

	size	count	bin	bincount	gzip
e.coli	5664540	-45,33	50,62	27,33	72,10
s.cerevisiae	12309078	-39,40	50,62	30,31	70,46

It is not possible to conclude that the results for the genomes (Table 3) are representative as the sample size is only two. However, we do see the same trend as with the results of Table 2. Most noteworthy is that the method *count* seems to perform even worse, with *E. coli* having a compression rate of -45.33 percent. Thus, we presume that lessons learned from the gene compression results are transferable to genomes.

4 Discussion

Overall, we have failed to develop an algorithm to compress biological sequence data with higher efficiency than gzip. However, there have been interesting findings in this work. Firstly, repetition is present in biological sequence, but not as repeats of the same symbol. Rather, it is more common to find repetitions of sets of symbols. As an example, it is easier to find multiple repetitions of 'ATTCGA' than of 'AAACCC'. Thus, the method *count* inflates the file-size as there are not many repetitions of three or more identical symbols that occur in the biological sequences tested. This would also explain why methods like MFCompress[15], which uses repetitions in combination with probabilistic models, work so well for sequence compression.

Another finding is that data conversion is useful to create compression by using less bits. This effect is visible with the methods *bin* and *bincount*. There are methods[16, 17] using a 2-bit approach where only the basic symbols A, T, C, and G are compressed into 2 bits each. It would be interesting to see how the performance changes when a 4-bit approach is used for these methods.

Thus, we can conclude that none of the methods proposed here are viable for usage, but have shown avenues for further research. Up to now, gzip (tar.gz files) have been the standard for good reason. This will likely continue for the foreseeable future. Despite this, research such as seen in *Hosseine et al. 2016*[9], and this work showcase some of the available possibilities.

References

[1] P. Deutsch. *DEFLATE Compressed Data Format Specification Version 1.3*. en. Tech. rep. RFC1951. RFC Editor, May 1996, RFC1951. DOI: 10.17487/rfc1951.

[2] Elie Dolgin. "A Tour through the Most Studied Genes in Biology Reveals Some Surprises." en. In: (2017), p. 5.

[3] Anne C Fergusonmith et al. "Regional Localization of the Interferon-@,/B-Cell Stimulatory Factor Z/Hepatocyte Stimulating Factor Gene to Human Chromosome 7˜1 S-P2". en. In: (), p. 6.

[4] Françoise Foury et al. "The Complete Sequence of the Mitochondrial Genome of *Saccharomyces Cerevisiae*". en. In: *FEBS Letters* 440.3 (Dec. 1998), pp. 325–331. ISSN: 00145793. DOI: 10.1016/S0014-5793(98)01467-7.

[5] Philippe Goyette et al. "Human Methylenetetrahydrofolate Reductase: Isolation of cDNA, Mapping and Mutation Identification". In: *Nature Genetics* 7 (1994), pp. 195–200. DOI: 10.1038/ng0694-195.

[6] Stephen Green et al. "Human Oestrogen Receptor cDNA: Sequence, Expression and Homology to-v-Erb-A". In: *Nature* 320 (Mar. 1986), pp. 134–139. DOI: 10.1038/320134a0.

[7] G.G.M. Habets et al. "Sublocalization of an Invasion-Inducing Locus and Other Genes on Human Chromosome 7". en. In: *Cytogenetic and Genome Research* 60.3-4 (1992), pp. 200–205. ISSN: 1424-859X, 1424-8581. DOI: 10.1159/000133336.

[8] T. Hayashi. "Complete Genome Sequence of Enterohemorrhagic Eschelichia Coli O157:H7 and Genomic Comparison with a Laboratory Strain K-12". en. In: *DNA Research* 8.1 (Jan. 2001), pp. 11–22. ISSN: 1340-2838. DOI: 10.1093/dnares/8.1.11.

[9] Morteza Hosseini, Diogo Pratas, and Armando Pinho. "A Survey on Data Compression Methods for Biological Sequences". en. In: *Information* 7.4 (Oct. 2016), p. 56. ISSN: 2078-2489. DOI: 10.3390/info7040056.

[10] *IUPAC Codes.* https://www.bioinformatics.org/sms/iupac.html.

[11] Eden R. Martin et al. "Analysis of Association at Single Nucleotide Polymorphisms in the APOE Region". en. In: *Genomics* 63.1 (Jan. 2000), pp. 7–12. ISSN: 08887543. DOI: 10.1006/geno.1999.6057.

[12] G. Matlashewski et al. "Isolation and Characterization of a Human P53 cDNA Clone: Expression of the Human P53 Gene." en. In: *The EMBO Journal* 3.13 (Dec. 1984), pp. 3257–3262. ISSN: 02614189. DOI: 10.1002/j.1460-2075.1984.tb02287.x.

[13] Marie-Geneviève Mattei et al. "Assignment of Vascular Endothelial Growth Factor (VEGF) and Placenta Growth Factor (PlGF) Genes to Human Chromosome 6p12–P21 and 14q24–Q31 Regions, Respectively". en. In: *Genomics* 32.1 (Feb. 1996), pp. 168–169. ISSN: 08887543. DOI: 10.1006/geno.1996.0098.

[14] Diane Pennica et al. "Human Tumour Necrosis Factor: Precursor Structure, Expression and Homology to Lymphotoxin". In: *Nature* 312.724-729 (1984). DOI: 10.1038/312724a0.

[15] Armando J. Pinho and Diogo Pratas. "MFCompress: A Compression Tool for FASTA and Multi-FASTA Data". en. In: *Bioinformatics* 30.1 (Jan. 2014), pp. 117–118. ISSN: 1460-2059, 1367-4803. DOI: 10.1093/bioinformatics/btt594.

[16] E. Rivals et al. "A Guaranteed Compression Scheme for Repetitive DNA Sequences". In: *Proceedings of Data Compression Conference - DCC '96*. Snowbird, UT, USA: IEEE Comput. Soc. Press, 1996, p. 453. ISBN: 978-0-8186-7358-0. DOI: 10.1109/DCC.1996.488385.

[17] Muhammad Sardaraz et al. "SeqCompress: An Algorithm for Biological Sequence Compression". en. In: *Genomics* 104.4 (Oct. 2014), pp. 225–228. ISSN: 08887543. DOI: 10.1016/j.ygeno.2014.08.007.

[18] Simon M. Schultze et al. "Promiscuous Affairs of PKB/AKT Isoforms in Metabolism". en. In: *Archives of Physiology and Biochemistry* 117.2 (May 2011), pp. 70–77. ISSN: 1381-3455, 1744-4160. DOI: 10.3109/13813455.2010.539236.