**ABRIDGE: An Ultra-Compression Software for sequence alignment files**

Sagnik Banerjee1,2,\*, Carson M. Andorf1,3,4,\*

1Program in Bioinformatics and Computational Biology, Iowa State University, USA

2Department of Statistics, Iowa State University, USA

3Crop Improvement and Genetics Research Unit, USDA-Agricultural Research Service, Ames, USA

4Department of Computer Science, Iowa State University, USA

\*Corresponding author

**Abstract**

Advancement in technology has enabled sequencing machines to produce vast amounts of genomic data, causing an increase in storage demands. Most genomic software utilizes read alignments for several purposes including transcriptome assembly and gene count estimation. Herein we present, ABRIDGE, a state-of-the-art compressor for sequence alignment files offering users both lossless and lossy compression options. This reference-based file compressor achieves the best compression ratio among all compression software ensuring lower space demand and faster file transmission. Central to the software is a novel algorithm that retains non-redundant information. This new approach has allowed ABRIDGE to achieve a compression 16% higher than the second-best compressor for RNA-Seq reads and over 35% for DNA-Seq reads. ABRIDGE also offers users the option to randomly access location without having to decompress the entire file. ABRIDGE is distributed under MIT license and can be obtained from GitHub and docker hub. We anticipate that the user community will adopt ABRIDGE within their existing pipeline encouraging further research in this domain.

**Keywords**

Sequence alignment compression, Random access, Alignment file compression, SAM file compressor

**Introduction**

Next generation sequencing (NGS) has opened up opportunities to study several biosystems from a quantitative viewpoint (Hickman et al., 2017; Erffelinck et al., 2018; Hunt et al., 2019; Elmore et al., 2020). Over the years, numerous sequencing protocols have been designed to probe the modus operandi of number of biological processes (Wang et al., 2009; Buenrostro et al., 2015). Researchers have perfected these protocols - making them more economical and effective. This made sequencing accessible to even underfunded labs leading to a surge in data (Banerjee et al., 2015a). Short read data (generated typically on Illumina platforms) is often mapped to a reference (genomic/transcriptomic) and then used for several purposes – assembling (Haas et al., 2013; Shao and Kingsford, 2017; Kovaka et al., 2019; Song et al., 2019), annotating (Haas et al., 2003; Holt and Yandell, 2011; Bruna et al., 2020; Banerjee et al., 2021; Li et al., 2021), finding differentially expressed genes (Robinson et al., 2010; Love et al., 2014) and for proteomics (Banerjee et al., 2015b, 2016a, 2016b, 2020; Velásquez-Zapata et al., 2020). Several bioinformatics projects utilize a very large set of RNA-Seq or DNA-Seq samples collected from multiple tissue types and conditions. The first step in such experiments is to align the RNA-Seq samples to a reference that generates a sequence alignment map (SAM) (Li et al., 2009) that is stored in either a binary alignment map (BAM) (Li et al., 2009) or compressed alignment file (CRAM) (Fritz et al., 2011) format. Even though these formats offer compression to some extent, the total size of all the aligned files can often exceed the storage capacity that small labs can afford. Hence, better compression techniques are needed that utilize the underlying structure of reference alignment files and offer a multitude of options to cater to a diverse range of user requirements.

Short reads, generated by sequencing platforms like Illumina, need to be mapped to a reference using aligners like STAR (Dobin et al., 2013, 2016), HiSAT2 (Kim et al., 2015) or BWA (Abuín et al., 2015) before further processing. These aligners typically output the result in a SAM format which can be converted to a binary BAM format to achieve better compression. SAM format stores the location, shape ([CIGAR](https://genome.sph.umich.edu/wiki/SAM) string), nucleotide bases, quality scores and tag level information for each aligned read. Since alignments in SAM format are stored for each read, the file size grows linearly with the number of reads in the sample. Hence, there is a need to devise an algorithm that can exploit the underlying structure of SAM files and offer the best possible compression in a reasonable amount of time.

A considerable amount of time and effort has been directed to designing algorithms to compress alignment files to reduce storage demands and facilitate file transfers (Giancarlo et al., 2014; Hosseini et al., 2016; Numanagić et al., 2016). Most approaches achieve compression by eliminating redundant data by accumulating alignment information across multiple reads or alignments. SAM compressors, like NGC (Popitsch and von Haeseler, 2013), DeeZ (Hach et al., 2014) and Genozip (Lan et al., 2021) are reference based while BAM, CRAM, Quip (Jones et al., 2012) and CSAM (Cánovas et al., 2016) are reference free. Reference-based approaches achieve compression by representing an aligned read with a description of how it differs from the reference. This eliminates the need to store the actual read sequence thereby reducing storage demands. Quality scores do not map to any reference and hence cannot be compressed like the read string. Hence some compressors like NGC, CSAM, Genozip and DeeZ offers users the option to map quality values within a range to a single value. While this can lead to better compression, it might remove quality scores of mismatched bases which are essential for detecting single nucleotide polymorphisms (SNPs). Quip implements Markov chains to encode read sequences and quality scores. Samcomp (Bonfield and Mahoney, 2013) compresses SAM alignments in lossless fashion by tokenizing the read identifiers and sorting the reads as a reference difference model. A very similar approach is undertaken by DeeZ where tokenized read names and read sequence are compressed with delta encoding.

To overcome the shortcoming of previous SAM compression approaches, we introduce ABRIDGE. We offer users a plethora of choices to compress SAM files. To optimize space utilization, ABRIDGE accumulates all reads that are mapped onto the same nucleotide on a reference. ABRIDGE modifies the traditional CIGAR string to store soft-clips, mismatches, insertions, deletions, and quality scores thereby removing the need to store the MD string. To further reduce space demand, ABRIDGE modifies the CIGAR information to store the strand on which the read was mapped. ABRIDGE also offers the option to alter quality scores of nucleotide bases that had a perfect match with the reference thereby reducing even more space (\textbf{Supplementary Figure \ref{RLE}}). All features of multi-mapped reads are stored with their individual CIGAR strings. Hence reads mapping to homeologs in polyploid species will retain their alignment profile. Users can choose from three levels of compression offering varying extents of compression with the caveat of the duration of compressing. ABRIDGE offers options of completely lossless compression and selectively lossy conversions. Consequently, decompressions in ABRIDGE can regenerate the entire SAM file with or without modifications depending on the choices made during compression. In this manuscript, we explore the different modes in which ABRIDGE can operate and compare it with other state-of-the-art tools.

**Materials & Methods**

ABRIDGE accepts a single SAM file as input and returns a compressed file that occupies less space than its BAM or CRAM counterpart. Users can choose to retain all the quality scores which would initiate a lossless compression. In several applications, storing the entire quality score is redundant. Hence, ABRIDGE can be configured to preserve only those quality values which for which the corresponding nucleotide base was a mismatch to the reference or an insertion into the read sequence. This option considerably reduces the compressed size while storing the most relevant information which can later be used for analysis that uses quality scores (e.g. variant calling). To further reduce space, users can eliminate quality scores altogether. Some downstream software like transcriptome aligners do not use soft-clips or mismatches, so we designed ABRIDGE to provide options to ignore such information in the SAM file while compressing. ABRIDGE compresses SAM files in two passes – in the first pass, relevant information from the SAM file is rearranged and in the second pass, the file is compressed using generic compressors. ABRIDGE decompresses data by applying the reverse algorithm producing all the requested information to be stored during compression. Once the data is compressed, users can retrieve alignment information from random locations making it very easy to access alignments from anywhere in the genome without having to decompress the entire file.

ABRIDGE achieves a high compression ratio due to the underlying strategies of eliminating redundant data. Instead of storing the entire sequence of reads, ABRIDGE stores the location of the reference to which the read mapped and relevant information about the mismatched and/or inserted base pairs. Instead of storing the exact mapped location, it keeps the difference in mapped position from the previous alignment. This saves a substantial amount of space for both RNA-Seq and DNA-Seq data. ABRIDGE also merges the exact same reads originating from the same nucleotide position of the reference. Read names for uniquely mapped single-ended reads are discarded but are preserved for multi-mapped single ended reads and for paired-ended reads to associate each read with the corresponding fragment. ABRIDGE offers users a multitude of choices for storing quality values. Users can request to store all the quality values without making any changes or allow ABRIDGE to modify the quality scores of some bases to facilitate better and faster compression. Instead of blindly modifying the quality scores, ABRIDGE inspects each base pair and modifies its quality value only if the base pair was aligned perfectly to the reference. Hence, the quality scores of bases which are inserts and/or mismatches are preserved. This provides the users with the opportunity to retain all the relevant information necessary to perform vital downstream analysis. ABRIDGE stores a modified version of the CIGAR string by including soft clipped bases, quality scores of mismatched and inserted bases along with nucleotides that did not match with the reference. Users are also provided with the choice of achieving best compression by eliminating quality scores altogether. This option is helpful for storing alignments files for the purpose of performing transcriptome assemblies where quality scores are not typically used (Song et al., 2019).

Unlike the read sequence, quality scores cannot be “mapped” to any reference. Hence ABRIDGE stores quality values as reported and then compresses those with generic compressors. ABRIDGE can store quality values in four different ways – (1) Discard quality values of reference matched bases and include only the mismatched and inserted bases. For this case, quality values are stored within the enhanced CIGAR, (2) Store all quality scores with altered values for reference matched bases, (3) Store all quality values without making any change in the quality values, and (4) Discard quality scores altogether.

Information about the alignment of each read is typically stored in the CIGAR and the MD string. While CIGAR string can indicate the soft-clips, matches, insertion and deletions, it is not designed to store mismatched nucleotides and read inserts. MD string, on the other hand, reports the mismatched bases. Hence, both the CIGAR string and MD string are needed to accurately reconstruct the appropriate alignment of the read to the reference. Since the CIGAR string and the MD string contain overlapping information, we decided to integrate them and generate a single representation which we call the integrated CIGAR. The integrated CIGAR contains complete information from which the entire alignment can be reconstructed (\textbf{Figure \ref{abridge\_IntegratedCIGARConstructionMF}}). Quality scores are stored within the integrated CIGAR if the user requests for it. Quality scores for only the mismatched bases and the inserts are stored. If the user requests to store quality scores for all the nucleotide bases then the scores will be stored in a separate file. An illustration of how the integrated CIGAR string is constructed has been provided in \textbf{Figure \ref{abridge\_IntegratedCIGARConstructionMF}}. Once each alignment entry is encoded, an index file is generated that can speed up file access in the future. The index contains information about the location of a pile of reads. During random access, the entire index file is read into the memory. According to the request made by the user, a specific portion of the compressed file is read and subsequently decompressed. Consulting the index file eliminates the need to decompress the whole alignment thereby speeding up random access. Finally, Generic compressors are used to compress the index along with the concise alignment file.

ABRIDGE will generate the compressed file in `.abridge` format which is essentially several files compressed using one of the three compressors - Brotli, 7z or ZPAQ. During decompression, a SAM is produced from the compressed files. The decompression step might require substituting dummy quality scores for some cases, depending on how the quality scores were stored during compression. The decompressed file will be sorted and dummy read names will be produced where they were discarded to save space. Some applications, like genome-guided assembling, do not require the nucleotide sequence. Hence, ABRIDGE allows the user to decompress without generating the actual read sequence. This option is faster to execute since it does not require the reference to be used.

**Results**

We tested ABRIDGE on RNA-Seq and DNA-Seq data of various read depths (\textbf{Supplementary Table \ref{abridge\_ListOfNCBI-SRASamplesForExperimentST}}). Programs were executed on a cluster with Intel(R) Xeon(R) CPU E5-2670 v2 processors with 2.50 GHz. CentOS Linux release 7.9.2009 was the operating system. ABRIDGE is entirely written in C and gcc compiler (v4.8.5) was used. We carried out experiments using different parameter settings as described in \textbf{Table \ref{abridge\_ParameterSettingST}}. The first parameter setting produces lossless compression and then we demonstrate how ABRIDGE can be configured to retain the requested information without impacting downstream applications. Details about data acquisition and processing have been mentioned in \textbf{Supplementary document}.

**SAM file format requirements**

Input to ABRIDGE, and to the other programs, need to be provided in SAM format. The file must be sorted by position and should have a proper SAM header. In addition, each alignment must have three tags - NH, MD, and XS. NH tag stores the number of times the read has been mapped which assists ABRIDGE to distinguish between uniquely mapped and multi mapped reads. MD tag contains information about mismatched bases and deletions which are used to generate a field in the compressed file. XS tag stores information about the strand to which the read was aligned.

**ABRIDGE achieves the best lossless compression**

ABRIDGE has two major goals - (1) achieve a high level of lossless compression, and (2) provide users with different modes of compression. Lossless compression is achieved by preserving only non-redundant information from SAM alignment file. Alignment files in SAM format were provided as input to the compression software. ABRIDGE performs the best compression owing to the usage of zpaq compressor (\textbf{Table \ref{abridge\_DifferentSettingsCompression}}). For single-ended reads ABRIDGE discards the read names for uniquely mapped reads. But for paired end reads, ABRIDGE needs to store the read names of both the pairs to enable associating the reads with the same fragment during decompression. This causes a slightly poor compression performance of ABRIDGE (with 7z) (\textbf{Figure \ref{abridge\_CompressionRatioComparisonMF}}). CSAM generates a file which is larger than the CRAM file itself. SAMCOMP attains the second best compression for paired-ended reads and third best for single ended reads. GENOZIP and DEEZ exhibit average performance in terms of ratio of compression.

**ABRIDGE offers multitude of options for Lossy compression**

ABRIDGE offers users with different options of compression as outlined in \textbf{Table \ref{abridge\_ParameterSettingST}}. Instead of blindly compressing quality scores, ABRIDGE offers users to modify quality scores of those nucleotide bases that perfectly match with the reference. This allows the user to retain the exact quality score of mismatched bases and insertions which would be useful for downstream analysis. With parameter setting number 2, ABRIDGE converts the quality score of matched bases to facilitate vertical run-length encoding leading to higher compression resulting in lower file size (\textbf{Supplementary Table \ref{abridge\_ComparisonAmongCompressorsSizeST}}). This is further illustrated in \textbf{Supplementary Figure \ref{abridge\_pie\_chart}} where the space requirement for storing quality scores greatly reduces from parameter setting 1 to parameter setting 2. The next set of parameters discard all quality scores except for the non-matched bases. The complete discard of quality scores leads to a further reduction in space requirements. The fourth parameter setting removes all quality scores, soft-clips, and mismatched bases. Since these did not occupy too much space, their removal did not reduce space significantly. In the final parameter setting, only the position of the mapped reads are preserved leading to the smallest file size. As expected, ZPAQ produces the best compression followed by 7z (\textbf{Supplementary Table \ref{abridge\_ComparisonAmongCompressorsSizeST}}).

Other software also offer the provision of lossy compression. Both DEEZ and GENOZIP were executed with different parameter settings of lossy compression. ABRIDGE lossy compression, with approximates quality scores (parameter setting 2) was able to produce a better compression than all other software operating in lossy mode (\textbf{Table \ref{abridge\_DifferentSettingsCompression}}).

**ABRIDGE quickly compresses data**

We compared the duration required to compress the SAM files. Even though ABRIDGE was not able to compress data the fastest, it was comparatively faster than CSAM and GENOZIP \textbf{Supplementary Figure \ref{abridge\_ComparisonAmongCompressorsTimeST}}. The main bulk of time is taken by the generic compressors (brotli, 7z and ZPAQ) which can be improved by allocating more CPU cores. Compression of a file is performed only once, hence we believe users will not be hesitant to dedicate the time. \textbf{Supplementary Table \ref{abridge\_ComparisonAmongCompressorsTimeST}} lists the duration of compression for the three generic compressors used in ABRIDGE along with different modes of compression. The duration of compression reduces with more lossy compression for both Brotli and 7Z. It is interesting to note that for ZPAQ the duration does not change much.

**ABRIDGE decompresses data faster than other software**

To utilize the alignments, the compressed files by ABRIDGE need to be decompressed. Unlike compression, decompression needs to be done multiple times depending on how often the alignment files are required to be accessed. Hence, we offer users the choice of multiple compressors that can help decompress files quicker. As depicted in \textbf{Supplementary Figure \ref{abridge\_DecompressionTime\_SF}}, 7z can decompress files very quickly. Unfortunately, ZPAQ takes the most time to decompress files even when it offers the best compression. Both brotli and 7z take almost the same time to decompress files which were compressed using different parameter settings (\textbf{Supplementary Table \ref{abridge\_ComparisonAmongCompressorsTimeST}}). ZPAQ, on the other hand, decompresses files faster when the compression was lossy.

**ABRIDGE can retrieve data randomly from any location**

During compression, ABRIDGE creates an index to facilitate random search. We compared the duration of generating the ABRIDGE index with the time taken to generate the BAM or CRAM index. As listed in \textbf{Supplementary Table \ref{abridge\_IndexST}}, CRAM takes the least time to generate indices. BAM and ABRIDGE take almost the same amount of time for single-ended reads. ABRIDGE takes longer for paired-ended reads since it needs to navigate through all the read names to index the file. ABRIDGE consume more memory to generate the indices whereas BAM and CRAM consumes much less memory. Interestingly, CRAM takes the same amount of memory for generating index even when the number of alignments increase.

Random access with ABRIDGE involves decompressing the file and then randomly accessing the requested location. Since ABRIDGE decompresses the entire file, it takes much longer to access random locations than CSAM, GENOZIP, BAM and CRAM. DEEZ takes the longest to access locations randomly since it decompresses the entire file (\textbf{Supplementary Table \ref{abridge\_RandomAccessTimeST}}). Both BAM and CRAM consume the least memory while ABRIDGE consumes the most (\textbf{Supplementary Table \ref{abridge\_RandomAccessMemoryST}})).

**Discussion**

We present ABRIDGE - a state-of-the-art software for compressing short-read alignments. ABRIDGE compresses alignments after retaining only non-redundant information. It achieves superior compression by merging similar reads mapped to the same location of the reference and encoding only those nucleotides that deviate from the provided reference. Strand information is also encoded in such a way that it does not occupy any additional space. ABRIDGE exploits the sorted file order to store the difference between adjoining mapping positions further reducing space demand. It also discards read names for single-ended uniquely mapped reads which improves compression further. Finally, column-wise conversion of quality scores assists in achieving the best compression.

For ABRIDGE to be a viable compression software, the decompression should be achieved at an acceptable pace. ABRIDGE (with 7z compression) outperforms SAMCOMP, GENOZIP and DEEZ in terms of the duration for decompressing a lossless compressed SAM file. While ABRIDGE with ZPAQ attains the best compression it also takes a much higher time to decompress. Although the decompression time is less than downloading the fastq from NCBI and aligning it to the reference.

Our analysis establishes ABRIDGE as the most recent SAM alignment compressor that offers a very high compression ratio. For single-ended DNA-Seq reads, ABRIDGE produced a file which is ~164 MB smaller than the next best compressor. This means that ABRIDGE can achieve an improvement of 15TB with 100K alignment files facilitating both storage and file transmission speed. Additionally, ABRIDGE compressed files can be randomly accessed making it convenient to perform searches without decompressing the entire file.

ABRIDGE provides users the option of choosing either lossless or lossy compression. It is recommended to use lossy compression in conjunction with downstream applications. For example, if the alignment files are produced for transcriptome assembly, then users can do away with quality scores and unmapped reads altogether. But if the downstream application involves SNP calling, then the quality scores (at least for the nucleotides that were a mismatch with the reference) should be preserved. Users are recommended to opt for ZPAQ if they choose to attain ultra-high compression ratio. On the other hand, if decompression time is of essence, then 7z compression would be the best choice. It is important to remember that ABRIDGE uses the reference file both for compression and decompression. Hence ABRIDGE stores a message digest of the reference to ensure that a correct copy is used for decompression.

An interesting future addition would be to expand ABRIDGE to other file types such as VCF, BED etc. Additionally, we would like to explore options to compress quality scores since those occupy the most space. Further, we will offer users the option to generate coverage information from compressed files directly. We are also collaborating with colleagues to adopt ABRIDGE as an acceptable file format to assembly and gene count software. We pledge to continually develop ABRIDGE to cater to a wide variety of file types storing biological information and facilitate its integration into existing pipelines.

**COMMENTS FROM REVIEWERS**

Major:  
1. Although the authors claim that ABRIDGE can handle SAM files from DNA-seq and RNA-seq alignments, it is weird that the authors only used STAR to perform the alignment. STAR was specifically designed for RNA-seq alignment, it might not be accurate in DNA-seq alignment. I noticed that ABRIDGE requires a SAM file with an ‘NH’ tag. However, for the most used DNA-seq aligners, such as BWA and Bowtie2, they do not produce such a tag, which means the adoption of this tool would be limited.

I agree in part with the reviewer. Yes, using BWA and Bowtie2 will not generate NH tags. The recent version of ABRIDGE will generate those if they are not found. It does not matter whether one does the alignment using STAR or bwa or bowtie. But since the reviewers want it, I will align DNA-Seq reads with bwa and bowtie.

2. Based on my Question 1, the authors may need to change the title to avoid misleading.

DONE

3. In ABRIDGE, the authors offer different options to use ABRIDGE in either a lossless or a lossy manner. However, the authors did not state clearly in what kind of situation different options can be selected. In the updated version, the authors may want to give a guide to do so.

It has been mentioned in the text that operations like SNP calling will require preserving of the quality scores so a lossless compression would be the way to go in this context. But I guess, we need to provide them with information (like in the form of a table) to help with the settings required for downstream analysis.

4. According to the specs (<https://samtools.github.io/hts-specs/SAMv1.pdf>), in CIGAR, it seems that sequence mismatches can be documented. Not sure why the authors claim that CIGAR is not designed to store mismatched nucleotides.

I should have been clearer about this. Yes, CIGAR string can indicate the position where the mismatch occurs but it doesn’t store the alternate nucleotide.

5. The authors state that ABRIDGE would help small labs to save storage spaces with a limited budget to buy disks. However, when using ABRIDGE, extra RAM and CPU are needed, would this be paradoxical? As the expense of RAM and CPUs are also high.

Small labs cannot afford large workspaces for a long duration of time. Even though high RAM is needed the job can be performed much faster. But we will need to reword the section to attract small labs as well.

6. During random access, the file needs to be entirely decompressed. Extra storage and time are also needed. I guess this would be one of the concerns for the adoption of this tool, as such as BAM/CRAM formats are good enough to quickly do so without extra storage needed.

Thinking about doing away with random access altogether. The objective of ABRIDGE is long term storage where I am focusing more on a high compression ratio.

7. The authors claim that ‘ABRIDGE produced a file which is 164 Mb smaller than the next best compressor’. So, for how large a file, ABRIDGE can have such an improvement? Is this improvement for all files or it has some association with the number of reads, types of reads and divergence of the alignment?

A Lot of evidence was provided in the manuscript. Not sure what tripped the reviewer.

8. In what kind of system did the authors test the tool? Were there some beta tests before the release? I downloaded the tool and had difficulties in running it which prevented me from assessing the performance. I guess the README file in the GitHub repo needs to document how to set the tool and why is docker or singularity need (I have singularity installed on my platform)?

Also we provided a lot of information explaining the platform and the CPUs on which the tests were performed.

9. In some cases, a BAM file is directly generated from software to save space. However, ABRIDGE needs to convert BAM to SAM first and then perform SAM compression. Would it be possible to avoid such a step to directly compress a BAM file to save the time and storage used in getting a SAM file?

It will not save time, since the bam file needs to be converted to sam anyways. But I will check whether other software can compress both BAM and SAM files.  
  
Minor:  
1. L23, Column 1, Page 1: ‘Most genomic software utilizes read alignments for several purposes’ -- This is not accurate, may reword

Could not understand why???

2. L20, Column 2, Page 1: ‘several purposes – assembling, annotating’ -- May change to ‘several purposes, such as assembly, annotation’

Ok

3. L22, Column 2, Page 1: ‘Most bioinformatics projects utilize …’ -- It depends on what kind of species the researchers working on. 'Most' is not an appropriate word here. May reword

Ok

4. L25, Column2, Page 1: ‘The primary step’ -- May change to ‘The routine step’

Hmmm Sure

5. L36, Column 2, Page 1: ‘need to be mapped’ -- May change to ‘are usually’

Ok sure

6. As there are different options to select the compression level, the authors may need to make this clear when talking about ABRIDGE. For instance, at L33, Column 1, Page 2, the authors say, ‘ABRIDGE modifies the traditional CIGAR’. Is this for all compression or only for lossy compression? If this is for lossy compression, would the modification be for all conditions or some of them?

This is changing completely.

7. L15, Column2, Page 2: ‘ABRIDGE compresses SAM files in two passes – in the first pass, relevant information …’ -- Make this clear that what the relevant information is.

As will this.

8. L17, Column 2, Page 2: ‘using generic compressor’ -- Please clearly list the compressors used in ABRIDGE

Also this

9. L23-24, Column 2, Page 2: ‘ABRIDGE achieves a high compression ratio … redundant data’ -- Is this for all modules in ABRIDGE or some of it? Please make this clear

Sure

10. L25-26, Column 1, Page 3: ‘an index file … in the future’ -- Is this for current usage or only for future?

Will be removed

11. L34, Column 1, Page 3: ‘compressed file in .abridge format’ -- It seems the compression relies on third-party compressors and the actual format is not '.abridge'. It's a rename of the original compression format, right?

Right

12. L38, Column 1, Page 3: ‘The decompression step might … during compression’ -- May clearly state in what kind of situation a dummy quality score is used and this would affect the accuracy of some downstream analyses, such as variant calling.

Exactly, will focus on this more in future versions

13. L5-7, Column 2, Page 6: ‘Although the decompression … to the reference’ -- Not sure what kind of information the authors want to deliver here.

Needs to be clearer

14. L16, Column 2, Page 6: ‘without depressing the entire file’ -- Previously, the authors

mentioned that during a random search, the entire file needs to be decompressed, but here they stated that there is no need to decompress, which confuses me. Please check.

Will be removed

15. Figure 1: There are two integrated CIGARs from ‘Construct the final Integrated CIGAR’ to ‘Exact same mapping of adjoining sequence with different SAM format Flag’. What’s the difference between the two in each section?

Will explain better in supplementary figures.

16. Figure 2: Does the comparison in the same level, for instance, was the file size calculated after a lossless compression or a lossy compression? If the compression is lossy, did they discard the same information? I guess all relevant figures need to state this clearly.

Lossless compression. Will improve the legends

17. Figure 3: From this figure, it seems ABRIDGE can only produce a modified SAM file and a subset of SAM files, is this true? Can the users get the original SAM file after decompression?

Yes, if lossless compression was used. Not if lossy was used during compression.

18. GitHub README ‘samtools calmd -bAr aln.bam > aln\_baq.bam’ -- a reference file is missed here, right?

Yep, my bad.

19. For the usage of ‘abridge’, the ‘-aq’ option says ‘Adjust quality scores for matched bases to achieve better encoding. For more details please check ...’. Please indicate what to ‘check’ here.

Will check thanks.  
  
Reviewer: 2  
  
Comments to the Author  
The authors present a novel compressor for the information stored in a SAM file. The manuscript is well-written, but I do not think it adds valuable contributions to the field.

Completely disagree

- My main concern is that the algorithm itself does not add new ideas, as most of the presented methodological steps have already been applied in other algorithms, except maybe for the extended CIGAR idea. Note however that this steps assumes the availability of the MD field, which is not mandatory and hence most of the available SAM files may not contain it. This should be clearly stated, as it is an important drawback of the algorithm. Moreover, the fields NH and XS are also not compulsory, and expected by ABRIDGE.

At the end of the day, what matters to the user is a high compression ratio. The extended CIGAR is a novel idea and so is the entire pipeline.

- The performance assessment only uses datasets of one species, and all datasets are very small. The method should be tested in large human datasets.

It does not matter at all what species are being compared. Heck, the analysis could have been done only with simulated reads. But to please the reviewers we will add more species.

- The choice of methods for comparison is very limited, note that NGC is from 2013, and DeeZ from 2014.

Most of the recent methods do not work and the authors do not respond when I report those issues on GitHub.

- Following the previous point, important references to recent work are missing, such as CALQ, QVZ, GeneComp, SPRING... Note that although Spring compresses FASTQ files instead of SAM, it has been shown to compare favorable to methods that compress SAM files.

CALQ, QVZ are only quality score compressor. They **DO NOT** compress the entire SAM file. GeneComp did not install properly. SPRING is a fantastic software, but it takes a long time to compress. In fact, I tested it and found that you can download the raw sra files quicker. I am not sure if the recent version has any improvements. What is surprising though is that the reviewers expect to compare ABRIDGE with SPRING. Either we were not clear, or the reviewer is BSing.

- Following previous point, a throughful comparison should include comparison on compression performance for only reads, only QS, only identifiers, only additional information. This way one can really assess where the gain from ABRIDGE comes. currently this info is provided only for ABRIDGE.

I agree, will deliberate

- Supplementary material contains only figures and tables, but the main document says that details about data acquisition are specified in the supplementary.

It is there.

- Regarding the datasets, the link used for download should be provided. I was not able to find the exact files online.

Sure, as if the reviewers will try to rerun the whole analysis!!

Reference:

Abuín, J. M., Pichel, J. C., Pena, T. F., and Amigo, J. (2015). BigBWA: approaching the Burrows–Wheeler aligner to Big Data technologies. *Bioinformatics*, btv506.

Banerjee, S., Basu, S., and Nasipuri, M. (2015a). “Big Data Analytics and Its Prospects in Computational Proteomics,” in *Information Systems Design and Intelligent Applications* (Springer), 591–598. doi:10.1007/978-81-322-2247-7\_60.

Banerjee, S., Bhandary, P., Woodhouse, M., Sen, T. Z., Wise, R. P., and Andorf, C. M. (2021). FINDER: An automated software package to annotate eukaryotic genes from RNA-Seq data and associated protein sequences. *BMC Bioinformatics*, 2021.02.04.429837. doi:10.1186/s12859-021-04120-9.

Banerjee, S., Ghosh, D., Basu, S., and Nasipuri, M. (2016a). *JUPred\_MLP: Prediction of phosphorylation sites using a consensus of MLP classifiers*. doi:10.1007/978-81-322-2695-6\_4.

Banerjee, S., Ghosh, D., Basu, S., and Nasipuri, M. (2016b). JUPred\_SVM : Prediction of Phosphorylation Sites using a consensus of SVM classifiers. in *Proceedings of Fifth International Conference on Soft Computing for Problem Solving* (Springer), 1–8. doi:10.1007/978-981-10-0448-3\_45.

Banerjee, S., Guha, S., Dutta, A., and Dutta, S. (2015b). Improvement of protein disorder prediction by brainstorming consensus. in *Computing and Communication (IEMCON), 2015 International Conference and Workshop on* (IEEE), 1–7. doi:10.1109/IEMCON.2015.7344428.

Banerjee, S., Velásquez-Zapata, V., Fuerst, G., Elmore, J. M., and Wise, R. P. (2020). NGPINT: a next-generation protein–protein interaction software. *Brief. Bioinform.* 2020, 1–14. doi:10.1093/bib/bbaa351.

Bonfield, J. K., and Mahoney, M. V (2013). Compression of FASTQ and SAM format sequencing data. *PLoS One* 8, e59190.

Bruna, T., Hoff, K., Stanke, M., Lomsadze, A., and Borodovsky, M. (2020). BRAKER2: Automatic Eukaryotic Genome Annotation with GeneMark-EP+ and AUGUSTUS Supported by a Protein Database. *bioRxiv*.

Buenrostro, J. D., Wu, B., Chang, H. Y., and Greenleaf, W. J. (2015). ATAC‐seq: a method for assaying chromatin accessibility genome‐wide. *Curr. Protoc. Mol. Biol.* 109, 21–29.

Cánovas, R., Moffat, A., and Turpin, A. (2016). Csam: Compressed sam format. *Bioinformatics* 32, 3709–3716.

Dobin, A., Davis, C. A., Schlesinger, F., Drenkow, J., Zaleski, C., Jha, S., et al. (2013). STAR: Ultrafast universal RNA-seq aligner. *Bioinformatics* 29, 15–21.

Dobin, A., Gingeras, T. R., Spring, C., Flores, R., Sampson, J., Knight, R., et al. (2016). Mapping RNA-seq with STAR. *Curr Protoc Bioinforma.* 51, 586–597. doi:10.1002/0471250953.bi1114s51.Mapping.

Elmore, M. G., Banerjee, S., Pedley, K. F., Ruck, A., and Whitham, S. A. (2020). De novo transcriptome of Phakopsora pachyrhizi uncovers putative effector repertoire during infection. *Physiol. Mol. Plant Pathol.* 110. doi:10.1016/j.pmpp.2020.101464.

Erffelinck, M.-L., Ribeiro, B., Perassolo, M., Pauwels, L., Pollier, J., Storme, V., et al. (2018). A user-friendly platform for yeast two-hybrid library screening using next generation sequencing. *PLoS One* 13, e0201270. doi:10.1371/journal.pone.0201270.

Fritz, M. H.-Y., Leinonen, R., Cochrane, G., and Birney, E. (2011). Efficient storage of high throughput DNA sequencing data using reference-based compression. *Genome Res.* 21, 734–740.

Giancarlo, R., Rombo, S. E., and Utro, F. (2014). Compressive biological sequence analysis and archival in the era of high-throughput sequencing technologies. *Brief. Bioinform.* 15, 390–406.

Haas, B. J., Delcher, A. L., Mount, S. M., Wortman, J. R., Smith Jr, R. K., Hannick, L. I., et al. (2003). Improving the Arabidopsis genome annotation using maximal transcript alignment assemblies. *Nucleic Acids Res.* 31, 5654–5666.

Haas, B. J., Papanicolaou, A., Yassour, M., Grabherr, M., Blood, P. D., Bowden, J., et al. (2013). De novo transcript sequence reconstruction from RNA-seq using the Trinity platform for reference generation and analysis. *Nat. Protoc.* 8, 1494. doi:10.1038/nprot.2013.084.

Hach, F., Numanagic, I., and Sahinalp, S. C. (2014). DeeZ: reference-based compression by local assembly. *Nat. Methods* 11, 1082–1084.

Hickman, R., Van Verk, M. C., Van Dijken, A. J. H., Mendes, M. P., Vroegop-Vos, I. A., Caarls, L., et al. (2017). Architecture and dynamics of the jasmonic acid gene regulatory network. *Plant Cell Online*, tpc-00958.

Holt, C., and Yandell, M. (2011). MAKER2: an annotation pipeline and genome-database management tool for second-generation genome projects. *BMC Bioinformatics* 12, 491. doi:10.1186/1471-2105-12-491.

Hosseini, M., Pratas, D., and Pinho, A. J. (2016). A survey on data compression methods for biological sequences. *Information* 7, 56.

Hunt, M., Banerjee, S., Surana, P., Liu, M., Fuerst, G., Mathioni, S., et al. (2019). Small RNA discovery in the interaction between barley and the powdery mildew pathogen. *BMC Genomics* 20, 610.

Jones, D. C., Ruzzo, W. L., Peng, X., and Katze, M. G. (2012). Compression of next-generation sequencing reads aided by highly efficient de novo assembly. *Nucleic Acids Res.* 40, e171–e171.

Kim, D., Langmead, B., and Salzberg, S. L. (2015). HISAT: a fast spliced aligner with low memory requirements. *Nat. Methods* 12, 357–360. doi:10.1038/nmeth.3317.

Kovaka, S., Zimin, A. V, Pertea, G. M., Razaghi, R., Salzberg, S. L., and Pertea, M. (2019). Transcriptome assembly from long-read RNA-seq alignments with StringTie2. *Genome Biol.* 20, 1–13.

Lan, D., Tobler, R., Souilmi, Y., and Llamas, B. (2021). Genozip-A Universal Extensible Genomic Data Compressor. *Bioinformatics*.

Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., et al. (2009). The sequence alignment/map format and SAMtools. *Bioinformatics* 25, 2078–2079.

Li, J., Singh, U., Bhandary, P., Campbell, J., Arendsee, Z., Seetharam, A. S., et al. (2021). Foster thy young: enhanced prediction of orphan genes in assembled genomes. *Nucleic Acids Res.*, gkab1238. doi:10.1093/nar/gkab1238.

Love, M., Anders, S., and Huber, W. (2014). Differential analysis of count data–the DESeq2 package. *Genome Biol.* 15, 550.

Numanagić, I., Bonfield, J. K., Hach, F., Voges, J., Ostermann, J., Alberti, C., et al. (2016). Comparison of high-throughput sequencing data compression tools. *Nat. Methods* 13, 1005–1008.

Popitsch, N., and von Haeseler, A. (2013). NGC: lossless and lossy compression of aligned high-throughput sequencing data. *Nucleic Acids Res.* 41, e27–e27.

Robinson, M. D., McCarthy, D. J., and Smyth, G. K. (2010). edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics* 26, 139–140.

Shao, M., and Kingsford, C. (2017). Accurate assembly of transcripts through phase-preserving graph decomposition. *Nat. Biotechnol.* 35, 1167–1169. doi:10.1038/nbt.4020.

Song, L., Sabunciyan, S., Yang, G., and Florea, L. (2019). A multi-sample approach increases the accuracy of transcript assembly. *Nat. Commun.* 10, 5000. doi:10.1038/s41467-019-12990-0.

Velásquez-Zapata, V., Elmore, J. M., Banerjee, S., Dorman, K. S., and Wise, R. P. (2020). Y2H-SCORES: A statistical framework to infer protein-protein interactions from next-generation yeast-two-hybrid sequence data. *bioarxiv*. doi:https://doi.org/10.1101/2020.09.08.288365.

Wang, Z., Gerstein, M., and Snyder, M. (2009). RNA-Seq: a revolutionary tool for transcriptomics. *Nat. Rev. Genet.* 10, 57–63.