ABRIDGE: A software to ultra-compress SAM alignment files

ABRIDGE: An ultra-compression software for SAM alignment files

\section{Abstract}

(200 words max)

Advancements in technology has enabled sequencing machines to produce vast amounts of genetic data, causing an increase in storage demands. Most genomic software utilizes read alignments for several purposes like transcriptome assembly or gene count estimation. Herein we present, ABRIDGE, a state-of-the-art compressor for SAM 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 provides users the option to randomly access location without having to decompress the entire file. ABRIDGE is distributed under MIT license and available from GitHub and docker hub. We anticipate that the user community will adopt ABRIDGE within their existing pipeline encouraging further research in this domain.

\section{Keywords}

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

\section{Introduction}

Para 1

Next-generation sequencing (NGS) has opened up opportunities to study several biosystems from a quantitative viewpoint (\cite{Hickman2017ArchitectureNetwork,Erffelinck2018ASequencing,Hunt2019SmallPathogen,Elmore2020DeInfection}). Over the years, numerous sequencing protocols have been designed to probe the modus operandi of a number of biological processes (\cite{Wang2009RNA-Seq:Transcriptomics,Buenrostro2015ATACseq:Genomewide}). Researchers have perfected these protocols - making them more economical and effective. These improvements made sequencing accessible to even underfunded labs leading to a surge in data. Short read data (generated typically on Illumina platforms) is often mapped to a reference (genomic/transcriptomic) and then used for several purposes – assembling (\cite{Haas2013DeAnalysis,Shao2017AccurateDecomposition,Kovaka2019TranscriptomeStringTie2,Song2019AAssembly}), annotating (\cite{Haas2003ImprovingAssemblies,Holt2011MAKER2:Projects,Bruna2020BRAKER2:Database,Banerjee2021FINDER:Sequences}), finding differentially expressed genes (\cite{Robinson2010EdgeR:Data,Love2014DifferentialPackage}) and in proteomics (\cite{Banerjee2015BigProteomics,Banerjee2021NGPINT:Software,Velasquez-Zapata2020Y2H-SCORES:Data}). Most bioinformatics projects utilize a very large set of RNA-Seq or DNA-Seq samples collected from multiple tissue types and conditions. The primary step in such experiments is to align the RNA-Seq samples to a reference that generates a file in either a binary, BAM format (\cite{Li2009TheSAMtools}), or a compressed format, CRAM (\cite{Fritz2011EfficientCompression}) 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.

Para 2

Short reads, generated by sequencing platforms like Illumina, need to be mapped to a reference using aligners like STAR (\cite{Dobin2013STAR:Aligner,Dobin2016MappingSTAR}), HiSAT2 (\cite{Kim2015HISAT:Requirements}) or BWA (\cite{Abuin2015BigBWA:Technologies}) before further processing. These aligners typically output the result in a SAM (\cite{Li2009TheSAMtools}) format which can be converted to a binary BAM format to achieve better compression. SAM format stores the location, shape (CIGAR string) (\href{https://genome.sph.umich.edu/wiki/SAM}{https://genome.sph.umich.edu/wiki/SAM}), 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.

Para 3

A considerable amount of time and effort has been devoted to designing algorithms to compress alignment files to reduce storage demands and facilitate file transfers (\cite{Giancarlo2014CompressiveTechnologies,Hosseini2016ASequences,Numanagic2016ComparisonTools}). Most approaches achieve compression by eliminating redundant data by accumulating alignment information across multiple reads or alignments. SAM compressors, like NGC (\cite{Popitsch2013NGC:Data}), DeeZ (\cite{Hach2014DeeZ:Assembly}) and genozip (\cite{Lan2021Genozip-ACompressor}) are reference based while BAM, CRAM, Quip (\cite{Jones2012CompressionAssembly}) and CSAM (\cite{Canovas2016Csam:Format}) 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 offer users the option to map quality values within a range to a single value. While this approach can lead to a better compression, it might remove quality scores of mismatched bases which are essential for SNP detection. Quip implements Markov chains to encode read sequences and quality scores. Samcomp (\cite{Bonfield2013CompressionData}) compresses SAM alignments in a 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 the read sequence is compressed with delta encoding.

Para 4

To overcome the shortcoming of previous SAM compression approaches, we introduce ABRIDGE, a tool that offers a plethora of choices to compress SAM files. To optimize space utilization, ABRIDGE accumulates all reads 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 increased 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. This manuscript explores the different modes in which ABRIDGE can operate and compares it with other state-of-the-art tools.

\section{Materials \& Methods}

ABRIDGE accepts a single SAM file as input and returns a compressed file that occupies substantially less space than its BAM or CRAM counterpart. Users can choose to retain all the quality scores, which would launch a lossless compression. Several downstream applications do not use quality scores. Storing the entire quality score would be redundant for those cases. Hence, ABRIDGE offers configurations to preserve only those quality values for which the corresponding nucleotide base was a mismatch to the reference or was an insertion into the read sequence. This option considerably reduces the compressed size but stores the most relevant information which is important for downstream analysis that uses quality scores like 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 information that was requested to be stored during compression. Once the data is compressed, users have the option of retrieving 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 owing 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 was mapped and relevant information about the mismatched and/or inserted base pairs. Instead of storing the exact mapped location, it retains 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 matching nucleotide position of the reference. Read names for uniquely mapped single-ended reads are discarded but preserved for multi-mapped single-ended reads and for paired-end reads to associate each read with the corresponding fragment. ABRIDGE offers users a multitude of choices for storing quality values. Users can request to keep all the quality values without making any changes or allow ABRIDGE to modify the quality scores of some bases to accelerate compression. Instead of blindly modifying the quality scores, ABRIDGE inspects each base pair and adjusts its quality value only if the base pair was aligned perfectly to the reference. Hence, the quality scores of bases that are inserts and/or mismatches are preserved. This allows the users to retain all the relevant information necessary to perform vital downstream analysis like variant calling. 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 the best compression by eliminating quality scores altogether. This option helps store alignments files for the purpose of performing transcriptome assemblies where quality scores are not typically used (\cite{Song2019AAssembly}).

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 integrated 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 the 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 (iCIGAR)’. The iCIGAR contains complete information from which the entire alignment can be reconstructed  (\textbf{Figure \ref{abridge\_IntegratedCIGARConstructionMF}}). Quality scores are stored within the iCIGAR if the user requests a lossy compression. Quality scores for only the mismatched bases and the inserts are stored. A separate file is used to store quality scores, if th user launches a lossless compression. An illustration of how the iCIGAR string is constructed has been provided in \textbf{Figure \ref{abridge\_IntegratedCIGARConstructionMF}}. Once each alignment entry is encoded, ABRIDGE generates an index file that speeds 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 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 essentially compresses several files using one of the three compressors - Brotli, 7z or ZPAQ. During decompression, a SAM file is produced from the compressed files. The decompression step might require substituting dummy quality scores for some cases, depending on how the compression step stored the quality scores. ABRIDGE sorts the decompressed file and produces dummy read names 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 read in.

\section{Results}

We tested ABRIDGE on RNA-Seq and DNA-Seq data of various read depths (\textbf{Supplementary Table \ref{abridge\_ListOfNCBI-SRASamplesForExperimentST}}). The programs were executed on a cluster with XX processor with XX GHz. Ubuntu XX-version XX was the operating system. ABRIDGE is entirely written in C and YY compiler 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}

\subsection{SAM file format requirements}

ABRIDGE, and other compression software, accepts input 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, 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.

\subsection{ABRIDGE achieves the best lossless compression}

The main goals of ABRIDGE are two-fold- (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 the 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}}) achieving a compression over 16% compared to SAMCOMP. 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 reduced compression for ABRIDGE (with 7z) (\textbf{Figure \ref{abridge\_CompressionRatioComparisonMF}}) but still manages to compress better with zpaq. CSAM generates a file which is larger than the CRAM file itself. SAMCOMP attains the second-best compression for paired-end reads and third best for single-end reads. GENOZIP and DEEZ exhibit average performance in terms of ratio of compression.

\subsection{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 an option 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 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 improvement 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 discards all quality scores except for the non-matched bases. The complete discard of quality scores leads to a further reduction in space requirement since the quality scores can now be accommodated in the iCIGAR eliminating the need for another file. The fourth parameter setting all removes 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 is 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 offers the provision of lossy compression. Both DEEZ and GENOZIP were executed with different parameter settings of lossy compression. ABRIDGE lossy compression, with approximated 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}}).

\subsection{ABRIDGE compresses data fairly quickly}

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 execution 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 3 generic compressors used in ABRIDGE along with different modes of compression. The duration of compression reduces with more increasing levels of lossy compression for both Brotli and 7z. It is interesting to note that for zpaq the duration remains same for all levels of lossy compression.

\subsection{ABRIDGE decompresses data faster than other software}

Downstream applications for generating assemblies and generating gene counts require sam/bam files. Hence, the files compressed by ABRIDGE need to be decompressed. Unlike compression, decompression is performed multiple times depending on how often the alignment files are 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 that were compressed using different parameter settings (\textbf{Supplementary Table \ref{abridge\_ComparisonAmongCompressorsTimeST}}). zpaq, on the other hand, decompresses files faster when the compression is lossy.

\subsection{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 and 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 a longer time for paired-end reads since it needs to navigate through all the read names to index the file. ABRIDGE consumes more memory to generate the indices, whereas BAM and CRAM consume much less memory. Interestingly, CRAM takes the same amount of memory for generating an index even when the number of alignments increases.

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}})).

\section{Discussion}

We present ABRIDGE - a state-of-the-art software for compressing SAM 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-end 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 decompressed data needs to be restored back 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. However, 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-end DNA-Seq reads, ABRIDGE produced a file ~164 MB smaller than the next best compressor. This result demonstrates 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 an 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 currently 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.