Boiler: a compression tool for RNA-seq alignments in BAM format supporting fast, accurate queries

Mark J Pritt[1], Ben Langmead[1,2]

[1]Johns Hopkins, Department of Computer Science, Baltimore, MD, [2]Johns Hopkins, Department of Biostatistics, Baltimore, MD

Abstract

RNA Sequencing (RNA-Seq) is an increasingly important tool in studying genome structure and gene expression. RNA-Seq reads must first be aligned by a tool such as TopHat, after which downstream tools such as Cufflinks, DESeq, and DER Finder can explore gene structure and expression. With the increasing availability of RNA-seq datasets covering hundreds or thousands of samples, there is a crucial need for methods that store results in a format that is both compact and easy to query.

We will be presenting Boiler, a tool for compressing a set of aligned reads in SAM/BAM format. Boiler partitions reads based on the introns they span and combines them into coverage vectors, compressing a SAM file to less than 1/5 of its original size in our experiments. Reads can be reproduced from the coverage vectors without significantly compromising accuracy. We tested the downstream effects of compression on Cufflinks transcript assembly and found that the assembled transcripts were almost identical before and after compression.

Our tool also supports common queries needed for tools like Cufflinks and DESeq, many of which are not naturally supported by the SAM/BAM format. Boiler supports queries for coverage levels and individual reads within a gene or interval, and gene boundaries. Our compressed file format is designed to return these queries quickly and accurately without the need to fully expand the compressed file.

Introduction

SAM/BAM is a ubiquitous file format used to store sets of read alignments produced by aligners such as TopHat. This format stores alignments line-by-line, including the location, CIGAR string, sequence, quality, and other information for each read. Much of this information, such as the sequence string, is redundant, and other information, such as quality values, is not used by most downstream tools like Cufflinks.

The downside to this format is that file size increases linearly with the number of reads in the dataset, requiring more space to store and longer times to query. With the increasing availability of large RNA-Seq datasets containing millions of reads, it is important to develop a new file format that is both compact and that enables fast queries. A format based on the coverage vector would be linear in the size of the genome, reducing the space dependence on the number of reads.

Several compression tools have been developed for SAM/BAM files, including CRAM, Quip and NGC. However, these formats were not designed to enable fast processing, and must be uncompressed before being queried. The bigWig and bigBed formats, while designed for annotation data and not for sequencing data, are similar to Boiler in that they enable targeted access without expanding the entire vector. Our goal in developing Boiler was to create an efficient compressed format for alignment data that supports fast, targeted queries without the need to recreate the initial BAM file.

Methods

Boiler is a lossy compression tool for alignment storage. The algorithm first converts a set of aligned reads to a vector containing the coverage level at every base. This vector can be run-length encoded for efficient storage. We also store the distribution of read lengths contained in the coverage vector. Since most reads from a single sequencing experiment have the same length, this distribution usually takes very little space. We can recover the original set of reads from this compressed format with high accuracy. While the complete algorithm is NP-hard, we found that a greedy algorithm recovers almost all the reads correctly.

In the case of paired-end reads, we store not only the distribution of paired-end lengths, but also the distributions of fragment lengths from the left and right ends of each paired read.

```
2L:7529-9484W:FBtr0330654:2    97    2L   7761    50   76M   =    7949    264    ...
2L:7529-9484W:FBtr0330654:2    145   2L   7949    50   76M   =    7761    -264   ...
2L:7529-9484W:FBtr0300689:4    153   2L   7996    50   76M   *    0       0      ...
2L:7529-9484W:FBtr0300689:7    97    2L   8463    50   76M   =    8620    233    ...
2L:7529-9484W:FBtr0300689:7    145   2L   8620    50   76M   =    8463    -233   ...
2L:7529-9484W:FBtr0300689:6    97    2L   8633    50   76M   =    8964    407    ...
2L:7529-9484W:FBtr0300689:8    97    2L   8678    50   76M   =    8837    235    ...
2L:7529-9484W:FBtr0300689:5    97    2L   8756    50   76M   =    9022    342    ...
2L:7529-9484W:FBtr0330654:3    153   2L   8775    50   76M   *    0       0      ...
2L:7529-9484W:FBtr0300689:8    145   2L   8837    50   76M   =    8678    -235   ...
2L:7529-9484W:FBtr0300689:2    97    2L   8838    50   76M   =    9176    414    ...
2L:7529-9484W:FBtr0300689:10   99    2L   8873    50   56M   =    8873    -56    ...
2L:7529-9484W:FBtr0300689:10   147   2L   8873    50   56M   =    8873    -56    ...
...
```

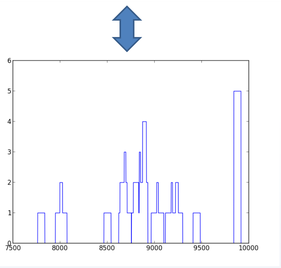

Figure 1. We can accurately convert between a set of aligned reads and their combined coverage vector accompanied by the read length distribution.

Before compressing, we separate the list of reads into spliced reads, which span multiple exons, and unspliced reads, which lie entirely in one exon. We store all the unspliced reads in a single coverage vector spanning the entire genome. Each exon junction containing at least one spliced read is stored as a separate coverage vector and read length distribution.

This format allows us to quickly query the compressed file for coverage levels in any segment of the genome. We can also quickly locate potential gene boundaries by searching for any regions where coverage drops to zero for a sufficiently long segment. Finally, we can recover all the reads from a targeted region by expanding only the appropriate part of the compressed file. All of these queries can be answered without recreating the initial BAM file.

The supported queries for coverage, gene boundaries, and reads should be sufficient for many downstream tools to execute without the need for any further processing of the BAM file. It would be relatively easy to modify downstream tools such as Cufflinks to directly query a Boiler-compressed file. The speed of Boiler's targeted queries should also save time over directly processing a BAM file.

Results

Results were gathered on two datasets generated using Flux Simulator, containing 500,000 and 1,000,000 paired-end reads. Boiler takes less than 300 seconds to compress the dataset of 500,000 paired-end reads and less than 400 seconds to compress the set of 1,000,000.

Space

Boiler can compress alignments into either a readable text format or a smaller binary format. We compared the sizes of both of these files against the original SAM and BAM files, as well as the minimal SAM and BAM files. The minimal files contained the same alignments as the originals, but all extraneous information not stored by Boiler was removed. Removed information included the longer read names, sequences, quality values, and most tags, all of which are not used by Cufflinks.

	500,000 Reads		1,000,000 Reads	
	Text Size (MB)	Binary Size (MB)	Text Size (MB)	Binary Size (MB)
Original	279	63.5	552	121
Minimal	47.6	9.32	95.4	17.9
Compressed	10.4	3.14	18.0	5.44

Table 1. Size comparison for SAM/BAM files and Boiler-compressed files.

Accuracy

We analyzed the accuracy of our algorithm by comparing the Cufflinks results before and after the alignments were compressed. We developed several tests to assess accuracy.

The k-mer compression (KC) score is a measure developed by Dewey et al. as a reference-based accuracy measure for assemblies. The KC score is equal to the sum of the weighted recall of all k-mers (WKR) minus the inverse compression ratio (ICR), which is proportional to the size of the compressed data and serves to penalize large assemblies. Figure 2 compares the KC scores for both the uncompressed and compressed Cufflinks results for various k-mer lengths.

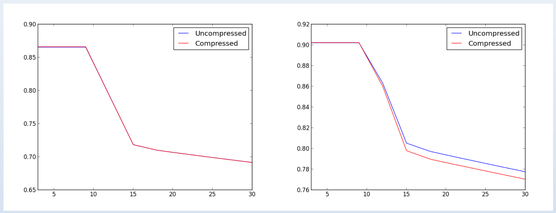

Figure 2. KC score of Cufflinks output across varying k-mer lengths.

We also created our own scoring function that assigns to any two transcripts a score between 0 (indicating no match) and 1 (indicating a perfect match). We use this scoring function to calculate the precision and recall of a set of transcripts compared to the true transcripts recorded by Flux. Precision is weighted by the predicted coverage levels for each transcript, and recall is weighted by the true coverage levels for each.

	500,000 Reads		1,000,000 Reads	
	Precision	Recall	Precision	Recall
Original	0.540	0.728	0.521	0.783
Compressed	0.351	0.725	0.572	0.771

Table 2. Precision and recall of Cufflinks output compared to the true transcripts used by Flux Simulator.

Conclusion

We have shown that we can significantly compress SAM/BAM files to a fraction of their original size by converting alignments to coverage vectors. The size of this file format is not strongly dependent on the number of reads, making it useful for increasingly high-throughput modern sequencing data. We can recover the initial reads quickly and with high accuracy, without significantly sacrificing information.

Moreover, the compressed files we produce are more suited than BAM files to quickly answering common downstream queries. Many downstream tools need to calculate coverage levels and gene boundaries over different regions of the genome. Boiler-compressed files can quickly access this information without recovering the initial reads. When necessary, we can also expand only those reads from targeted sections of the genome, saving time.

We hope that future alignment processing tools can make use of Boiler-compressed files to enable faster computation and smaller space requirements.

Future Work

While these results are promising, we need to test on more datasets as well as on real sequencing data to more accurately analyze Boiler's performance. We have not yet completed speed optimization of the described queries and also plan to fully test query speed and accuracy compared to BAM files.

In addition, we have not yet optimized the binary encoding of compressed files. The compressed binary files compared here were obtained by simply zipping the text version of the files. We plan to implement our own binary encoding to optimize for the specific structure of Boiler files. While the results shown here show a significant space improvement over BAM, we expect to achieve even stronger results after optimization.

Acknowledgements

Special thanks to my adviser Ben Langmead for all of his help.

Contact Information

mpritt4@jhu.edu