Supplementary Data for manuscript: The importance of data modeling for compression of genomic aligned data

1 Data Sets

The data sets used to asses the performance of the proposed compression method and the previously proposed algorithms in the literature is summarized in Table 1. Following the convention of the main manuscript, we divide the data in two ensembles, the low coverage data sets and the high coverage ones.

2 SAM file generation

We generate the SAM files with Bowtie2 [Langmead *et al.*, 2012] and the following parameters:

./bowtie2 −−no-unal −−fast −−threads 2 −−mm −x *index* −U *fastqFile* > *samFile*

The parameters used by bowtie2 are the same as the ones used in [Bonfield *et al.*, 2013]. The only difference is that we add the option "−−no-unal", which suppresses the records that failed to align, as we are interested only in compressing the reads that mapped to the reference.

To generate the index for the H. Sapiens, the C. Elegans and the M. Musculus we use the scripts *make_hg19.sh*, *make_c_elegans.sh*, and *make_m_musculus_ncbi37.sh*, respectively, provided by Bowtie2 in the folder "scripts". We modified the script *make_m_musculus_ncbi37.sh* to use the release *ncbi38* instead of the *ncbi37*, as the latter was no longer available.

Low Coverage Data Sets

Data	Species	Num. Reads	Read Length
SRR062634_1	H. Sapiens	24,148,993	100
SRR027520_1	H. Sapiens	23,063,389	76
SRR027520_2	H. Sapiens	23,063,389	76
SRR043366_2	H. Sapiens	15,575,360	76
SRR013951_2	H. Sapiens	18,212,437	76
SRR005729_1	H. Sapiens	16,831,095	76
SRR032209	M. Musculus	18,828,274	36

High Coverage Data Sets

Data	Species	Num. Reads	Read Length
SRR065390_1	C. Elegans	33,771,758	100
SRR065390_2	C. Elegans	33,771,758	100
ERR262997_2	H. Sapiens	643,097,275	101
ERR262996_1	H. Sapiens	434,794,343	101

Table 1: Data sets used for the simulations.

Finally, to generate the sorted SAM file by position, we run the following commands with samtools [Li *et al.*, 2009].

1. samtools view -bS *samFile* | samtools sort - *sortedBamFile*
2. samtools view -h *sortedBamFile* > *sortedSamFile*

The alignment information for the different data sets is summarized in Table 2.

Low Coverage Data Sets

Data	Reference	Mapped Reads*	Coverage	BAM Size [GB]	SAM Size [GB]
SRR062634_1	hg19	23.3 M	0.26×	1.9	6.9
SRR027520_1	hg19	20.4 M	0.17×	1.7	5.1
SRR027520_2	hg19	20.4 M	0.17×	1.7	5.1
SRR043366_2	hg19	13.5 M	0.11×	0.72	3.4
SRR013951_2	hg19	9.5 M	0.08×	0.84	2.4
SRR005729_1	hg19	9.4 M	0.08×	0.96	2.5
SRR032209	mm GRCm38	13.5 M	0.17×	0.59	2.7

High Coverage Data Sets

Data	Reference	Mapped Reads*	Coverage	BAM Size [GB]	SAM Size [GB]
SRR065390_1	ce ws235	31.6 M	32×	1.6	9.5
SRR065390_2	ce ws235	31.2 M	32×	1.7	9.4
ERR262997_2	hg19	410.5 M	14×	30	122
ERR262996_1	hg19	272.8 M	10×	21	81

Table 2: Alignment information for the different data sets introduced in Table 1. *M stands for millions.

3 Compression programs

For comparison, we used the following programs proposed previously in the literature: Fastqz [Bonfield *et al.*, 2013], SamComp [Bonfield *et al.*, 2013], Quip [Jones *et al.*, 2012], Goby [Campagne *et al.*, 2013] and CRAM [Fritz *et al.*, 2011]. Next we specify the versions and commands used for the simulations by each of the aforementioned algorithms. Also, recall that we are only interested in the compression of the reads that mapped to a reference, and not in the compression of the whole SAM file. Therefore, for each of the algorithms, we also specify how we computed the compression ratio corresponding to the mapped reads.

1. **Fastqz** (version v1.5):

   Recall that Fastqz does its own alignment, and as such, it takes as input a fastq file and a reference file. Thus, we generate the fastq file from the SAM file, so that it only contains the reads that mapped to the reference. For the reference, we first pre-process it with the command

   ./fapacks *indexRef* *reference*.fa

   Once the reference is pre-processed and the fastq file generated, we run the Fastqz compression program with the following command

   ./fastqz c *fastqFile* *outputPrefix* *indexRef*

   Fastqz outputs separate files per data type, and thus we can compute the final size of the compressed reads by summing the size of the files *outputPrefix*.fxb.zpaq and *outputPrefix*.fxa.fxa.zpaq, which are the ones corresponding to the reads.

   To decompress, we use the following command:

   ./fastqz d *inputPrefix* *outputFastqFile* *indexRef*

   Note that the *inputPrefix* is equal to the *outputPrefix* used for the compression command.

2. **SamComp** (version v0.7, corresponding to SamComp1):

   We use SamComp1 rather than SamComp2, since the first one is optimized for sorted SAM files, and thus it produces better compression results than SamComp2 for the case considered in this paper (see [Bonfield *et al.*, 2013]).

   We have modified the algorithm to omit compression of the identifiers and the quality values, so that it only compresses the reads. We run the program with the following command:

   ./sam_comp -r *pathToFolderWithRef* -M -f SAM < *samFile* > *compFile*.zam

   For decompression, we run the following command:

   ./sam_comp -d -r *pathToFolderWithRef* -M -f SAM < *compFile*.zam > *outputSamFile*

3. **Goby** (version v2.3.4):

Goby does not report separate figures upon completion, and we were not able to modify the program so as to omit the compression of the identifiers and the quality values. Thus, to get an approximation of the compressed size corresponding to the reads, we modified the corresponding SAM files. Specifically, we set the identifiers and quality values of all the records to the same value, removed all the auxiliary fields (except the MD field, which is needed for Goby), and set columns 7-9 as "*", "0" and "0", respectively.

We run Goby with the following command:

```
java -jar goby.jar -m sam-to-compact -i SAMfile -o outputFile −−sorted -x
MessageChunksWriter:codec=hybrid-1 -x MessageChunksWriter:template-compression=true
-x AlignmentCollectionHandler:enable-domain-optimizations=true -x
AlignmentWriterImpl:permutate-query-indices=false -x
AlignmentCollectionHandler:ignore-read-origin=true
```

This corresponds to the H+T+D approach, which as demonstrated in the Goby paper produces the best compression results.

To decompress, we run the following command:

```
java -jar goby.jar -m compact-to-sam GobyFile -o outputSamFile -g referenceGenome
```

4. **Quip** (version v1.1.8):

   We run Quip with the following command:

   ```
   quip -i sam -r refInFasta SAMfile
   ```

   To find the compression size corresponding to the reads, we run the following command, which gives the desired information:

   ```
   quip -l -v SAMfile.qp
   ```

   To decompress, we use the following command:

   ```
   quip -i quip -r refInFasta QUIPfile
   ```

5. **CRAM** (version v1.0)

   CRAM does not report separate figures upon completion, and we were not able to modify the program so as to omit the compression of the identifiers and the quality values. Thus, to get an approximation of the compressed size corresponding to the reads, we modified the corresponding SAM files. Specifically, we set the identifiers and quality values of all the records to the same value, removed all the auxiliary fields, and set columns 7-9 as "*", "0" and "0", respectively.

   We run CRAM with the following command:
   java -jar cramtools-1.0.jar cram −−input-is-sam −−input-bam-file *samFile* −−reference-fasta-file *refFileFastaFormat* −−output-cram-file *outputFile*.cram

   To decompress, we run CRAM with the following command:
   java -jar cramtools-1.0.jar bam −−input-cram-file *inputFile*.cram −−reference-fasta-file *refFileFastaFormat* −−output-bam-file *outputFile*.bam

4 Compression results

We divide the compression results in two subsections, the first one for the low coverage data sets and the second one for the high coverage ones.

4.1 Low coverage data sets

In Table 3 we specify the compression results of Fastqz, SamComp, Quip, Goby, CRAM and the proposed method for the low coverage data sets.

In Table 4 we show the compression times of the different algorithms. However, note that Quip, Goby and CRAM are compressing the whole SAM file. Also, Fastqz is performing its own alignment, instead of using the alignment information provided by the SAM file, and compressing not only the reads, but also the identifiers and the quality values. Finally, in Table 5, we show the decompression time of the different algorithms. We do not show the decompression time of CRAM, as we were unable to decompress the files. In this case, the only method that is

Data	Fastqz	SamComp	Proposed Method	Quip	Goby	CRAM
SRR062634_1	93.74 MB	49.07 MB	48.92 MB	76.34 MB	70.37 MB	61.84 MB
SRR027520_1	122.01 MB	55.45 MB	54.67 MB	78.25 MB	81.39 MB	67.60 MB
SRR027520_2	111.66 MB	48.66 MB	48.37 MB	70.42 MB	70.05 MB	59.66 MB
SRR043366_2	59.73 MB	26.07 MB	25.95 MB	43.81 MB	36.73 MB	33.05 MB
SRR013951_2	81.93 MB	39.16 MB	36.06 MB	51.07 MB	58.56 MB	45.20 MB
SRR005729_1	92.05 MB	49.55 MB	48.10 MB	72.60 MB	82.51 MB	62.59 MB
SRR032209	95.68 MB	19.64 MB	19.56 MB	78.15 MB	26.14 MB	24.41 MB

Table 3: Compression results for the different algorithms in the low coverage data sets (1 MB = 10^6 Bytes).

decompressing only the reads is the proposed method. The other methods are decompressing also the quality values and the identifiers.

Data	Fastqz	SamComp	Proposed Method	Quip	Goby	CRAM
SRR062634_1	887	160	163	284	359	574
SRR027520_1	751	129	160	218	322	451
SRR027520_2	797	126	154	224	294	447
SRR043366_2	405	61	124	158	172	290
SRR013951_2	403	45	130	140	182	240
SRR005729_1	416	48	143	162	218	258
SRR032209	190	46	66	136	119	132

Table 4: Compression time, in seconds, employed by the different algorithms.

4.2 High coverage data sets

In Table 6 we present the compression results of Fastqz, SamComp, Quip, Goby, CRAM and the proposed method for the high coverage data sets corresponding to the *C. Elegans* species, whose size is similar to that of the low coverage data sets. Note that as the file size increases,

Data	Fastqz	SamComp	Proposed Method	Quip	Goby	CRAM
SRR062634_1	596	324	225	327	399	-
SRR027520_1	752	256	203	324	288	-
SRR027520_2	709	259	206	327	295	-
SRR043366_2	324	141	172	203	196	-
SRR013951_2	411	121	190	186	186	-
SRR005729_1	423	135	195	185	201	-
SRR032209	211	91	73	152	133	-

Table 5: Decompression time, in seconds, employed by the different algorithms.

the compression result of Goby and CRAM becomes less accurate.

Data	fastqz	SamComp1	Proposed Method	Quip	Goby	CRAM
SRR065390_1	96.14 MB	47.71 MB	40.46 MB	81.49 MB	68.61 MB	53.19 MB
SRR065390_2	109.47 MB	55.20 MB	45.91 MB	87.40 MB	78.71 MB	59.97 MB

Table 6: Compression results of the different algorithms. (1 MB = 10^6 Bytes).

Finally, in Table 7 we show the decompression time employed by the proposed method and SamComp for the high coverage data sets. As for the low coverage datasets, the SamComp algorithm is decompressing not only the reads, but also the quality values and the identifiers, which explains the higher running time.

Data	SamComp	Proposed Method
SRR065390_1	357	103
SRR065390_2	360	112
ERR262996_1	3240	1150
ERR262997_2	4185	1665

Table 7: Decompression time, in seconds, employed by the proposed method and SamComp.

References

[Bonfield *et al.*, 2013] Bonfield, J.K. and Mahoney, M.V. (2013) Compression of FASTQ and SAM Format Sequencing Data, *PLOS ONE*.

[Langmead *et al.*, 2012] Langmead B. and Salzberg S. (2012) Fast gapped-read alignment with Bowtie 2. *Nature Methods*, **9**:357-359.

[Li *et al.*, 2009] Li H., Handsaker B., Wysoker A., Fennell T., Ruan J., Homer N., Marth G., Abecasis G. and Durbin R.(2009) The Sequence Alignment/Map format and SAMtools., *Bioinformatics*, **25**:16.

[Jones *et al.*, 2012] 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 Research*, **40**:22.

[Fritz *et al.*, 2011] 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.

[Campagne *et al.*, 2013] Campagne F, Dorff KC, Chambwe N, Robinson JT, Mesirov JP.(2013) Compression of structured high-throughput sequencing data *Plos One*, in Press.