Subject Section

BioSeqZip: a collapser of NGS redundant reads for the optimisation of sequence analysis

Gianvito Urgese [1]*, Emanuele Parisi[2], Orazio Scicolone[2], Santa Di Cataldo[2], and Elisa Ficarra[2]

[1] Interuniversity Department of Regional and Urban Studies and Planning, Politecnico di Torino, 10129, Torino, Italy. and
[2] Department of Control and Computer Engineering, Politecnico di Torino, 10129, Torino, Italy

*To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: High-Throughput Next-Generation-Sequencing can generate huge sequence files, whose analysis requires alignment algorithms that are typically very demanding in terms of memory and computational resources. This is a significant issue, especially for machines with limited hardware capabilities. As the redundancy of the sequences typically increases with coverage, collapsing such files into compact sets of non-redundant reads has the two-fold advantage of reducing file size and speeding-up the alignment, avoiding to map the same sequence multiple times.
Method: *BioSeqZip* generates compact and sorted lists of alignment-ready non-redundant sequences, keeping track of their occurrences in the raw files as well as of their quality score information. By exploiting a memory-constrained external sorting algorithm, it can be executed on either single or multi-sample datasets even on computers with medium computational capabilities. On request, it can even re-expand the compacted files to their original state.
Results: Our extensive experiments on RNA-seq data show that *BioSeqZip* considerably brings down the computational costs of a standard sequence analysis pipeline, with particular benefits for the alignment procedures that typically have the highest requirements in terms of memory and execution time. In our tests, *BioSeqZip* was able to compact 2.7 billions of reads into 963 millions of unique tags reducing the size of sequence files up to 70% and speeding-up the alignment by 50% at least.
Availability: *BioSeqZip* is available at https://github.com/bioinformatics-polito/BioSeqZip
Contact: gianvito.urgese@polito.it
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

In ultra-high-throughput sequencing, as many as 500k sequencing-by-synthesis operations may be run in parallel, allowing the deep sequencing of DNA and RNA molecules in short time and producing a massive amount of data to be analysed. With the ever-increasing production of new sequence data, and the continuous necessity of re-elaborating old data to extract hidden knowledge, new paradigms for data storage and analysis are becoming more and more critical (Muir *et al.*, 2016).

DNA and RNA reads are collected in the form of formatted files whose size can exceed a terabyte, typically storing two main types of information, both encoded by ASCII strings: i) sequences, representing the bases of the biological molecules (RNA/DNA) and ii) quality scores, representing the reliability level of each sequenced nucleotide.

The analysis of such RNA/DNA sequences starts with preliminary steps that consist of filtering the reads based on quality thresholds. Then, the high-quality reads are trimmed from the adapters and either aligned on a known reference database or assembled to construct new unknown genomes.

To have a more in-depth overview of the analysis flow, we can consider the example of RNA-Seq experiments. RNA-Seq reads, either in single-end or paired-end form, are first filtered based on their quality scores and trimmed from the adapters. Then, the most reliable reads (i.e. the ones with highest quality scores) are analysed by applying one of three main strategies, as described by Conesa *et al.* (2016): i) if a reference genome is available, reads are aligned to the genome with a gaped aligner. This approach allows the identification and quantification of known transcripts as well as the discovery of novel ones. ii) if no novel transcript discovery is needed, reads can be mapped to the reference transcriptome by using an ungapped aligner. iii) when no genome is available, reads are usually first assembled into transcripts. Then, reads are mapped back to the assembled reference transcriptome for quantification and annotation.

As far as the alignment problem is concerned, several tools are available (Mazzoni and Kadarmideen, 2016), mostly making use of suffix arrays to provide faster alignment compared to traditional Dynamic Programming methods (Smith and Waterman, 1981; Urgese *et al.*, 2014). Among the most representative: BWA (Li and Durbin, 2009), Bowtie2 (Langmead and Salzberg, 2012), STAR (Dobin *et al.*, 2013), Rail-RNA (Nellore *et al.*, 2016) and Yara (Siragusa, 2015). Even though the computational time of the alignment is greatly reduced compared to classic algorithms, it still remains prohibitive for most of the medium-low computational systems, like the work-stations that are commonly used in most bio-labs.

A possible solution to this problem is collapsing all the repeated reads into a single one, reducing the number of reads that need to be analysed, and hence the size of the read file. The advantage of this solution is two-fold: i) the memory required for the storage can be reduced and ii) more efficiently, the alignment algorithms can map each unique read to the reference only once, instead of multiple times.

To the best of our knowledge, there exist in the literature a few tools that already provide a collapsing option, but with several significant limitations. First of all, none of the tools to date allows controlling memory consumption. This is a critical lack, as it may make impossible to execute the collapsing programs on mid-low end platforms. Most of the tools are also limited in the type of data that can be processed. *SeqCluster (SC)* (Pantano *et al.*, 2011) can only collapse single-end reads from small RNA-Seq datasets, while *FastUniq (FU)* (Xu *et al.*, 2012) can only deal with paired-end reads. On the other hand, the popular *FASTX-toolkit (FXT)* (Gordon and Hannon, 2010) provides a simple function to collapse identical reads but does not keep any trace of the quality scores provided in the input files. On top of that, it cannot deal with paired-end samples. *SuperDeduper (SD)* (Petersen *et al.*, 2015) is an interesting tool for PCR duplicate removal, as it examines only a small portion of each reads (called key) and automatically discards data containing unknown nucleotides (Ns). However, it is not able to solve the exact read collapsing problem. *ParDRe (PDR)* (González-Domínguez and Schmidt, 2016) has a reasonable run-time, but a significant memory consumption.

The full list of the literature tools, with collapsing functionality, is reported in *Table 1*, together with a summary of their respective features. In the order: the possibility to deal with single-end and paired-end sequences, the way the quality score of the input sequences is processed (i.e. AVG: by averaging the quality scores of the redundant reads per nucleotide, $HIGH$: by retaining the best score per nucleotide, SUM: by summing up base qualities scores), the possibility to set a user-defined memory bound to the alignment, the supported file formats. As it can be easily gathered from the table, none of the tools provides a fully functional collapsing option, as they all lack some essential features.

Collapsing procedures are widely adopted as a preliminary step for the alignment of sRNA-Seq on miRNA and other small non-coding RNA databases such as miRBase (Griffiths-Jones *et al.*, 2006), miRGeneDB (Fromm *et al.*, 2015, 2018) and piRBase (Zhang *et al.*, 2014). Tools designed to quantify miRNA expression levels such as *isomiR-SEA* (Urgese *et al.*, 2016), SeqBuster (Pantano *et al.*, 2009), sRNAbench (Barturen *et al.*, 2014) miRDeep2 (Friedländer *et al.*, 2011), and others (Desvignes *et al.*, 2019) take a simple file[1] as input, with the unique sequence in the first column and its number of occurrences in the second column, to minimise the number of calls to a computationally expensive alignment procedure. In our work, we leverage this concept and take it to a higher level.

Table 1. Features of the available collapsing tools, with our proposed tool BioSeqZip (BSZ) in the last row

Tool	Single End	Paired End	Quality	Memory Bound	File Formats
FU	No	Yes	High	No	.fa .fq
FXT	Yes	No	No	No	.fa
PDR	Yes	Yes	High-AVG	No	.fa .fq
SC	Yes	No	AVG	No	.fa .fq
SD	Yes	Yes	High-Sum	NO	.fa .fq
BSZ	Yes	Yes	AVG	Yes	.fa .fq .sam .bam .tag .tagq

In this paper, we propose *BioSeqZip*; a new approach to collapse redundant reads generated by NGS machines. The functionality of our proposed solution is three-fold: i) A read collapsing technique based on the external sorting algorithm[2] allows limiting memory usage (Knuth, 1998). Thus, making the tool suitable not-only to cluster computers, but even to medium systems with limited hardware capabilities. ii) A multilevel collapsing procedure can be applied to compress even further the read files. Recurrent reads from different samples will be collapsed into a single file, where unique reads will occur only once. iii) An integrated expansion procedure enables an easy restore of the read aligned files produced by mapping tools.

As it can be gathered from *Table 1*, our proposed solution addresses the lack of functionalities of the collapsing tools available to date (see the last row of the table). *BioSeqZip* generates compact files of unique reads storing the number of collapsed sequences, so that the number of detected molecules can be considered in the alignment algorithms, as well as their quality score information. On top of that, it supports seven different file formats, allowing easy integration to several classes of alignment, mapping, and assembly algorithms.

To demonstrate the quality of our solutions, we tested the *BioSeqZip* compression module on 32 RNA-Seq samples from the Human BodyMap 2.0 dataset, collecting 2.34 billions of single and paired-end reads. We assessed *BioSeqZip* performance in terms of run-time, memory consumption, reduction of reads number as well as of collapsed files size and compared our tool against five alternative tools in terms of both performance and usability. On top of that, we compared the performance of four different alignment algorithms on collapsed and raw/uncollapsed input files, respectively, with either single-end and paired-end data-sets.

2 Methods

We propose *BioSeqZip* as a solution to the extensive computational time and memory requirements of NGS data analysis flows involving sequences, such as transcriptome/genome mapping and small RNA-Seq analysis. A common trait of these tasks is that they need to process all the reads of the sequenced samples multiple times before moving to the next steps of the analysis. Hence, these tasks are majorly benefited by the removal of redundant reads.

BioSeqZip is a read collapser that groups and counts the occurrence of the identical reads in the input sequence file, producing a minimal sorted list of unique reads (and the corresponding occurrence counts) ready for the alignment to a reference. The read collapsing can be obtained at two different levels: i) at *single-sample* level (SS), *BioSeqZip* collapses the

[1] Defined as Tag file

[2] External sorting is required when the data being sorted do not fit into the RAM of a computing device and instead they must reside in the slower external memory (HDD or SSD). In our implementation, we use a hybrid sort-merge strategy. In the sorting phase, chunks of data small enough to fit in main memory are read, sorted, and written out to a temporary file. In the merge phase, the sorted sub-files are combined into a single larger file.

redundant reads stored in a single file, producing a compacted output file with unique reads. ii) At *multi-sample* level (MS), *BioSeqZip* aggregates redundant reads from a data-set containing multiple sample files, allowing a supplementary level of aggregation.

In both cases, if the input files provide quality information about the reads, *BioSeqZip* returns the collapsed sequences with a quality score, that is computed as the average consensus quality of the corresponding collapsed reads. On top of that, *BioSeqZip* allows to choose the format of the collapsed file (either Fastq, Fasta, Tagq, and Tag)[3]. By doing so, the user can choose whether to maintain the quality information (and hence, bear the corresponding computational costs) or not, with a pay-for-what-you-use philosophy.

Besides the read collapsing functionality, *BioSeqZip* implements also a re-expanding functionality, that is able to recover the original occurrences of each sequence to update, with the number of read occurrences, the alignment files generated by mapping algorithms.

The functionalities of *BioSeqZip* are implemented into two main modules: a *Collapser* for compacting the redundant reads and an *Expander* for restoring aligned and compacted files. The modules were developed in C++ leveraging several packages of the SeqAn C++ bioinformatics library (Doring *et al.*, 2008; Reinert *et al.*, 2017). Most of our design choices were explicitly made to optimise the memory consumption and the computational performance on systems equipped with medium hardware resources. In the following, we describe the main design aspects, functionality and supported file formats of the *Collapser* and the *Expander* modules.

2.1 *BioSeqZip*_Collapser module

Collapsing the reads is a powerful strategy to reduce the computational time and complexity of the alignment steps of an RNA-Seq or DNA-Seq analysis pipeline. Indeed, in files with billions of reads, the same sequence will likely be analysed multiple times. The collapser module collapses and counts the occurrence of redundant reads, that are detected based on their identical sequence. This new reduced collection of non-redundant sequences can be stored in a much smaller file and used for optimised alignment analysis. *BioSeqZip_Collapser* accepts as input Fasta, Fastq, or SAM file formats and generates compact output with four different file formats: Fasta, Fastq, Tag, and Tagq (*Text S1.1*). All the input/output file formats can be optionally provided in compressed form (gzip).

The most straightforward approach to collapsing reads is the one implemented by the *fastx_collapser* routine offered in *FASTX-Toolkit* (Giardine *et al.*, 2005). This solution requires to first load all the reads into the main memory, then to sort them, and finally to collapse them into unique tags, while counting and storing occurrence of the redundant reads. However, this approach is hugely memory-intensive, as it requires to load in memory the entire file that is typically in the order of tens of GBs.

Indeed, memory issues are always a critical aspect of NGS data analyses that typically exploit big data. As the dimension of the NGS files is generally huge, the amount of memory necessary to load a whole data-set will be overwhelming for medium-small memory systems such as standard workstations and laptops, leading to the necessity of leveraging powerful machines such as clusters of compute nodes and servers. Thus, increasing the costs of the analysis.

BioSeqZip_Collapser overcomes such limitation by implementing a memory-constrained collapsing functionality. For this purpose, the user is asked to set the maximum amount of the memory that can be exploited by the program, based on the capabilities of the available hardware.

The main steps of *BioSeqZip_Collapser* are shown in *Fig. 1*.

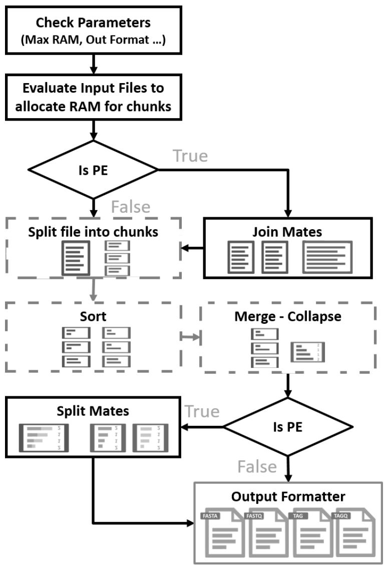

Fig. 1. BioSeqZip_Collapser flow-chart, with external-sorting and collapsing steps highlighted in dashed boxes.

In the first phase, we pass and check the input parameters: *–input* and *–max-ram* respectively for the input file storing the reads to be collapsed (either FASTA, FASTQ, SAM, or BAM) and the maximum amount of RAM allowed. A set of reads (T_{size}) are read from the input file to evaluate their memory occupancy (T_{RAM}). Then, we estimate the maximum number of reads (C_{size}) that can be loaded at the same time into memory without exceeding the memory constraint, set by the user (C_{RAM}), by computing the *Eq. 1* where α is an empirical correction factor.

$$C_{size} = \alpha \cdot \frac{C_{RAM} \cdot T_{size}}{T_{RAM}} \tag{1}$$

The module implements a custom *parallel external sorting* procedure (Knuth, 1998) that sorts the reads in chunks and generates M temporary files of sorted records. For clarity, in *Fig. 2* we show a schematic example of the External-Sort-Collapsing procedure. Each chunk of reads (of size 3, in the example of *Fig. 2*) is loaded into RAM, where it is alphabetically sorted by a multi-threaded function. Then, the sorted set of reads are collapsed and written on a temporary file on the disk (in dark-grey background). In the merge-collapsing phase, the first C_{size}/M sequences from all the temporary files are loaded in M buffers. Then, an iterative selection-sort process is started, for identifying the overall occurrences count of each read. At each iteration, the tool appends a collapsed sequence to the output file, together with the corresponding occurrences count. Empty chunks are refilled with a portion of the temporary file from which the collapsed reads

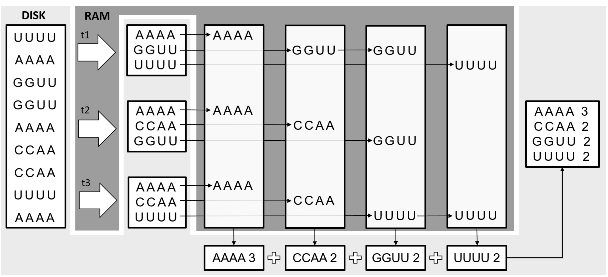

Fig. 2. Example of read collapsing based on external sorting.

[3] See *Text S1* for the file format description

come. At the end of the merge-collapse procedure, the output file will contain all the collapsed reads and their corresponding occurrence count. If requested by the user, in case the input file provided a quality score per read, the procedure computes the average quality score of each block of collapsed reads, base-by-base.

Besides the aforementioned, *BioSeqZip_Collapser* supports two additional cases: i) files with paired-end reads stored with the interleaved format, where mate 1 is in the even reads and mate 2 in the odd reads. ii) files with joined mates and fixed break-point, where the two mates are already in the same read.

As far as file formats are concerned, six different input/output combinations are available, all optimised to minimise the number of required operations. More specifically, with Fastq input, the user can select any of the four output formats. With a Fasta format, on the other hand, the output can either be Tag or Fasta. Optionally, *BioSeqZip_Collapser* can be even forced to generate a collection of output files with constant size set by the user. In this case, the module generates multiple files of the set dimension, each storing a portion of the collapsed reads sorted in alphabetical order. This feature is particularly useful in cases where the read files must be distributed to an HPC cluster. For Fasta and Fastq output formats, the read count is appended to the identifier on the header line. By doing so, the output file can be easily handled by alignment programs without losing track of the original number of redundant reads before collapsing.

If the input is not a single file, but a folder containing multiple files, the *BioSeqZip_Collapser* will implement a multi-sample collapsing strategy. Even in this case, the procedure can generate any of the four possible output files, with a supplementary table reporting the ID of collapsed sequence in the first column, the sequence in the second column, and its per-sample occurrence counts in the following columns. The information reported in this table can be eventually leveraged by the *Expander* module to recover the original occurrence of redundant reads in each sample.

2.2 *BioSeqZip*_Expander module

The files storing the collapsed reads can be given as input to the most common RNA-Seq and DNA-Seq analysis pipelines, thereby considerably reducing demands in terms of storage space and computational time. As already discussed, this is a significant advantage in case of systems with limited hardware resources that might not be powerful enough to host the analysis of redundant data.

Especially for mapping and alignment procedures, the best practice would be to consider the complete available information at the same time (i.e. sequence, sequence occurrence count and average quality score). However, standard alignment tools are generally unable to leverage the occurrence count field of Fasta/Fastq files. Hence, the analyst is left with two options: i) customising the mapping procedure in order to consider the read counts during the alignment algorithm[4]. ii) align the not-redundant reads and then re-expand the compact aligned files, exploiting the occurrence counts to replicate the original entries. The latter functionality is implemented by *BioSeqZip_Expander*. This module re-expands the output files of alignment/mapping tools by replicating each Tag by the occurrence count of each sample. By doing so, it restores an expanded collection of redundant sequences and corresponding mapping locations.

Multi-mapped reads can either be handled with a random assignation strategy, as done by BWA (Li and Durbin, 2009), or with a first position policy, like Yara (Siragusa, 2015), where the additional locus are listed in a custom field of the SAM file.

BioSeqZip_Expander is meant for re-expanding the output of any alignment tool, to obtain the same output that would have been generated by aligning the raw non-collapsed sample. As long as each SAM record stores the original name of the aligned read, *BioSeqZip_Expander* is able to parse it and extract the number of times the sequence was found by the *BioSeqZip_Collapser* in the raw sample. Then, the target SAM record is written that number of time to the final output file, which represents the expanded alignment file. *BioSeqZip_Expander* can deal with the two most common file formats for alignment files: SAM and BAM.

In the case of multi-sample collapsing, *BioSeqZip_Expander* will exploit the supplementary table generated by the multi-sample collapsing strategy to restore the correct number of read sequence and originally detected in each corresponding sample.

3 Results and Discussion

We tested *BioSeqZip* on a set of 32 transcriptomic samples of 16 different human tissues from the Human Body Map 2.0 Project (ERP000546). This set, characterised by the presence of both single-end (SE) and paired-end (PE) reads, was generated with the HiSeq 2000 sequencing technology (Ogasawara *et al.*, 2006).

To assess our methodology, we evaluated four popular alignment tools on both collapsed and not collapsed reads. We decided to evaluate the quality and computational costs of the alignment, because it is the fundamental step of any sequencing data analysis.

The experiments were performed on a Linux machine with 2x8 Intel cores clocked at 2.4 GHz (Xeon E5-2630), 128 GB RAM, 16 TB HDD SAS in RAID 6.

In the following, we will show the performance of our read collapsing algorithm and the benefits of using tags representing collapsed sequences for mapping procedures. We collapsed Fastq files, hence retaining the average quality of the original files. A quality-agnostic compression would be twice as faster and produce collapsed files of half the size of the case reported here. Thus, the time and memory reported in our results should be considered as a worst-case scenario.

3.1 Single and multi-sample collapsing performance

In this experiment, we collapsed 16 SE and 16 PE samples (see full list and details in *Table S1*), containing 1.26 billion 75 bp single-end and 1.28 billion of 50*2 bp paired-end Illumina reads of human individuals with a 20x coverage at a genome size of 3.1 GB. Each sample contains 80 millions of reads on average and occupies 17 GB on the disk for the SE and 26 GB for the PE.

In our test, the raw files were collapsed using *BioSeqZip_Collapser*, first with a single-sample (SS) and then with a multi-sample (MS) strategy. *Table 2* shows the results of the collapsing procedure, in terms of the number of reads, the total size of samples, and run-time. The first column of the table makes explicit whether the considered samples are uncompressed (i.e. RAW), collapsed single-sample (SS) or collapsed multi-sample (MS). The second column indicates whether the samples are single-end or paired-end reads. The following two columns report the number of reads and file size. Concerning RAW and SS samples, the sum of the corresponding sizes for each sample is reported. While for MS samples, the results were produced by the multi-sample collapse procedure. Finally, the last three columns of the table report the overall collapsing time and the gain (computed with *Eq. 2*) of the collapsing, in terms of number of reads to be

[4] We have adapted BWA and Yara aligners to consider the occurrences stored in the header of each read of the collapsed Fastq files, and the modified code is provided in two additional GitHub repositories (links provided in the Supplementary Availability section)

Table 2. Single-sample and multi-sample collapsing performance

SAMPLE		# Reads	Size	Collapse	GAIN	
Status	Type	(G)	(GB)	Time (s)	# Reads	Size
RAW	PE	1.28	390	0	0	0
	SE	1.26	252	0	0	0
SS	PE	0.59	141	9633	53.7%	63.8%
	SE	0.39	64	6106	69.4%	74.6%
MS	PE	0.54	134	12708	57.7%	65.6%
	SE	0.28	47	7763	78.0%	81.4%

aligned and size of the file on the disk.

$$Gain_{\%} = \left(1 - \frac{Collapsed}{Raw}\right) \cdot 100 \tag{2}$$

Some interesting observations can be gathered from *Table 2*. First of all, single-end samples achieved higher collapsing rate than paired-end samples. This is due to the exact collapsing algorithm leveraged by *BioSeqZip*, that has better performance for shorter sequences, assuming similar data-sets sizes. More specifically, collapsing all the single-end samples with a single-sample strategy leads to a reduction in the number of reads to be analysed of approximately 70%, whereas the storage requirements are reduced to a quarter[5] . Multi-Sample collapsing leads to an even better gain in terms of the number of reads and files size, but it comes at the cost of higher run-time (30% higher in the worst case).

Different values of the memory constraints translate into a different number of intermediate files that are generated during the collapsing (the lower the memory limit, the higher the number of disk operations). To clarify the relation between memory limit and performance, we run an additional experiment by collapsing three random files, at 4, 8, 16 and 32 GB limits by using 4 threads. The run-time in seconds was respectively 1062, 1098, 1172 and 840. This analysis highlights that the run-time required for collapsing a sample increases as the size of the buffer used for storing the sequences increases, up to the amount of memory that can contain the full file to be collapsed (more details in the Section S4.3).

3.2 Alignment performance on collapsed samples

To assess the impact on alignment performance of both single-sample and multi-sample collapsing, we proceeded as follows: i) We performed single-sample collapsing of all the samples. Then, each collapsed file was aligned to the Human transcriptome (Cunningham *et al.*, 2014) using four popular read mappers: respectively, BWA, Bowtie2, Yara, and STAR. Alongside, the raw uncollapsed files were also aligned using the same tools, to measure the alignment time on the original number of reads. ii) We performed multi-sample collapsing of the whole data-set, measuring both execution time and RAM requirements (always within the input constraint). Thus, we obtained a single collapsed Fastq output file and a table of read occurrences for the whole data-set. Then, we run the mappers on the Fastq file, assessing the execution time of the alignment as well as the storage size of the mapped reads. In both experiments, we did not run the Expander module, which is conceived as an optional feature in regular downstream analysis.

The obtained results are plotted in *Fig. 3*, where we show the gain of aligning collapsed files (either single-sample SS or multi-sample MS) instead of raw ones, using alignment time as the figure of merit in *Eq. 2*. For the SS files, the collective gains of the individual samples of the data-set are reported in the form of a box-plot, while an asterisk represents the gain of MS. As can be gathered from the plots of *Fig. 3*, the benefits of collapsing were consistently high for all the mapping tools. More specifically, the average alignment time speed-up attested between 50% and 70% in case of single-sample collapsing (SS) and between 36% and 73% for multi-sample collapsing (MS).

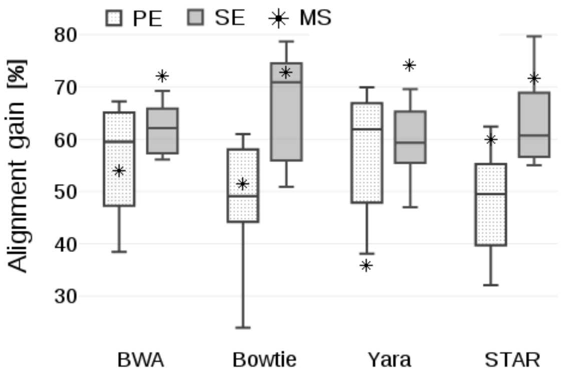

Fig. 3. Gain of aligning collapsed files, using alignment time as the figure of merit.

If we take into account the storage requirements, the impact of collapsing was even more significant, as the mapping file size was reduced by 71% with single-sample and up-to 80% with multi-sample collapsing, respectively. On top of that, we registered a significant reduction in the size of the collapsed alignment files. Indeed, for the overall alignment files the total disk occupancy of raw files aligned was 4.7 TB while SS aligned files occupy 1.9 TB with a reduction of 60%. However, the best size reduction was achieved when aligning MS collapsed files, obtaining a disk space reduction of 66%.

To estimate the run-time benefits of using collapsed files, in *Fig. 4* we show the results obtained by aligning the raw SE and PE files alongside the SS collapsed. In this plot, to have a complete view, we took into account even the additional computational time needed by *BioSeqZip* to collapse the sequence files (blue portion of the bars). As it can be gathered from *Fig. 4*, applying *BioSeqZip* had a significant impact on the computational time required by alignment and mapping procedures. The highest impact was for the tool STAR, for which time reduced from 19 to 7 hours for SE and from 27 to 10 hours for PE, with reductions attesting between 55% and 63%, respectively. Collapsing was very advantageous also for BWA and Bowtie2. The weakest impact was for Yara mapper, for which the alignment time of samples of the size considered in our tests is

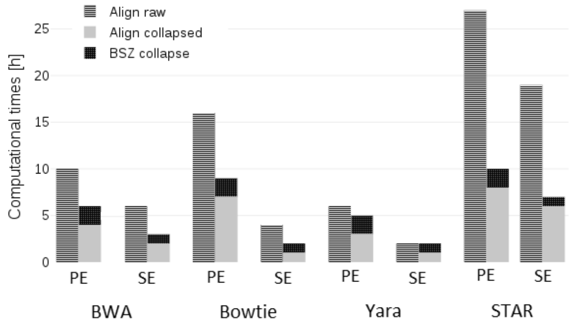

Fig. 4. Alignment performance on raw and collapsed files.

[5] In Section S4.2, we propose a collapsing experiment on a WGS DNA-Seq sample from Scherer *et al.* (2019), showing the benefit of the collapsing strategy on this type of data.

already minimal. Nonetheless, as it is visible from the blue portion of the stacked bars, the additional computational time required for collapsing the sequence file is always a small fraction of the overall time needed to execute the alignment on the raw reads (red bars). Hence, the collapsing is advantageous even in the worst-case scenario.

To avoid possible inconsistencies, in our tests we verified the coherence of the raw and collapsed alignment files, by checking that the same number of reads where mapped by the tools to the same genome location of the alignments produced using the raw Fastq files.

In the last mapping experiment, we analysed single-end samples of BodyMap with either BioSeqZip_Collapser+STAR+BioSeqZip_Expander (BSZ_C+STAR+BSZ_E) and Rail-RNA alone and compared the two runs in terms of both execution time and memory consumption (full analysis in Section S4.4). From this experiment, we obtained that the combination of BSZ_C+STAR+BSZ_E outperformed Rail-RNA by 68% (15.29 vs 47.37 hours). Note that the comparison is limited to time and memory performance of the two pipelines, without in-depth evaluation of the accuracy and consistency of the provided output.

3.3 Comparison with alternative collapsers

For this test, we randomly selected three SE and three PE samples[6] and used them to compare the compression performance achieved by *BioSeqZip* with other tools available in the literature. The SE samples (labelled E890, E894, and E902) contain respectively 64, 77 and 82 millions of 75 bp reads, with a disk space occupancy of approximately 19, 22 and 24 GB. The PE samples (labelled E873, E882, and E886) are characterised by 82, 74 and 83 millions of 50*2 base-pairs reads, with a disk space occupancy of 25, 22 and 25 GB.

In our test, the raw files were collapsed using *BioSeqZip_Collapser (BSZ)*, *Fast Unique (FU)*, *FASTX Toolkit (FXT)*, *ParDRe (PD)*, *SeqClaster (SC)* and *Super Deduper (SD)*, respectively. In case a tool did not support either PE (e.g. *FXT* and *SC*) or SE (e.g. *FU*), we implemented a specific adapter to convert SE into PE and vice-versa. Indeed, some of the tools considered in our test match the collapser requirements only partially, because they were not specifically designed for this task (see *Table 1*). All the tools were executed with default parameters, killing the process if running for more than 5000 seconds.

Fig. 5 reports six groups of bars, one per tool, showing the performance in terms of total computational time spent for collapsing the three SE and the three PE samples, respectively. As it can be observed from *Fig. 5*, *BioSeqZip* was faster than all the other tools except for *Super Deduper*. On the other hand, *Super Deduper* collapses the reads by only looking at the first bases of the sequence. Thus, it generates over-collapsed files where reads with a different sequence are often grouped together (more details in *Fig. S1*, where the collapsing efficiency of each tool is duly reported). *FASTX* was close to our tool when dealing with SE samples, but on the other hand it completely overlooks quality information and generates only Fasta files as output format. Similarly, *ParDRe* obtained a performance comparable to *BioSeqZip* on PE samples, but even in this case not all the identical reads were detected and collapsed (details in *Fig. S1*).

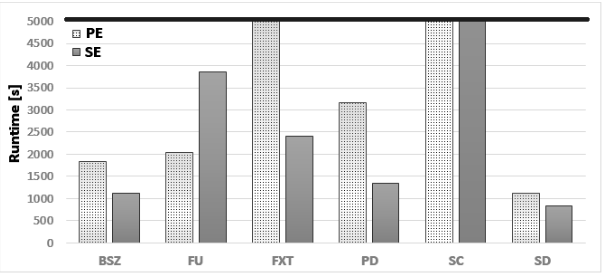

Fig. 5. Collapsing performance of different collapsing tools. Total computational time for collapsing SE and PE samples, respectively.

As already pointed out, a major distinguishing point of our implementation is the possibility to impose a maximum limit of memory that can never be exceeded. In *Fig. 6*, we report the memory consumption of *BioSeqZip* with an imposed memory limit of 8 GB[7] and show the memory consumption of all the other tools for comparison. As can be gathered from *Fig. 6*, three tools out of six reached peeks of 78 GB when dealing with PE samples. Only *FXT* and *SD* were close to the 8 GB limit, even though, unlike our solution, there is no specific guarantee that this constraint is always respected. In our tests, *FXT* tool reached a peak of RAM consumption close to 13 GB, but without keeping tracks of quality information. As already discussed, *Super Deduper* obtained the best performance only in theory, as it over-collapsed non-identical reads.

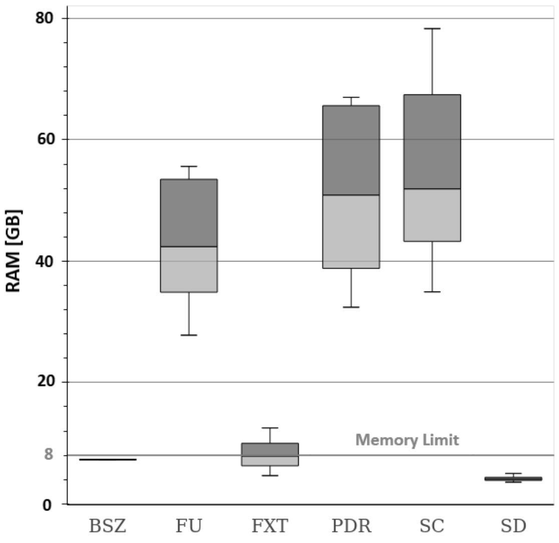

Fig. 6. Memory usage of collapser tools.

In summary, *BioSeqZip* provides the best advantages in terms of run-time and memory usage. Moreover, it generates alphabetically sorted files, which is very useful to reduce the complexity of down-stream analysis.

3.4 Overall impact of *BioSeqZip* on alignment

In this last section, we aggregate the outcome of the previous tests to estimate the overall advantages of *BioSeqZip* in terms of reduction of the file size on disk and of computational time spent for the mapping.

As far as disk usage is concerned, the overall size of the collapsed files, plus mapped files obtained using collapsed files as input, was 2.1 TB for SS and 1.8 TB for MS, against 5.4 TB of the RAW files. More specifically, *BioSeqZip* was able to reduce 2.54 billions of reads into 980 millions of unique tags, in less than 4.4 hours of computations.

[6] Listed with blue background in *Table S3*.

[7] The amount of RAM used impacts on the speed of *BioSeqZip*. The more RAM we use, the lesser reading/writing operations we perform on the disk. Since the disk operations are the bottleneck of our algorithm, reducing the number of these operations has a positive impact on the overall run-time of the *BioSeqZip* collapsing procedure.

In *Table 3* we show details about alignment time of BWA, Bowtie2, Yara, and STAR on raw and SS collapsed reads, respectively. The last column of the table reports the gain, using the execution time of the alignment procedures on raw and SS collapsed files as the figure of merit in *Eq.* (2). As it can be gathered from the table, when using Yara on collapsed files we observed a gain of 48%. The alignment time reduction was even more significant for the other mapping tools: 57% for BWA, 55% for Bowtie2, and 68% for STAR.

Based on the values of *Table 3*, we can estimate a total saved time equal to 57.3 hours, which provides the overall figure of merit of the impact of *BioSeqZip* on alignment time.

4 Conclusion

In this paper, we presented *BioSeqZip*, a read collapsing tool that can reduce the huge sequence files generated by high-throughput NGS machines to compact sets of non-redundant reads.

As we extensively demonstrated in our experiments, *BioSeqZip* brings down the computational requirements of sequence analysis pipelines, with particular benefits for the alignment techniques, that are typically the ones with the highest computational costs.

In summary, the virtues of our approach are manifold:

- it is hardware-adaptive, as it is able to constrain RAM utilisation based on a user-defined threshold depending on the hardware capabilities of the system.
- it is flexible, as it supports all the main sequencing file formats, and operates either at a single-sample or a multi-sample level.
- it is exhaustive, as it maintains track of read quality information and read occurrence counts while collapsing the redundant sequences, allowing easy restoration of the original data.

Besides alignment, we firmly believe that the complete analysis pipeline will significantly benefit from the application of our collapsing strategy. With minor modifications, mapping and alignment tools may even directly leverage the occurrence counts information provided by the compressed files to skip the re-expansion phase and speed-up the mapping process (BWA and Yara are already available). Exploiting sorting and merging operations, performed by *BioSeqZip* during the collapsing phase, the mapped files (in SAM/BAM formats) will be ready for further analysis as-is, without needing any supplementary manipulations. Thus, in perspective, the computational costs of the overall analysis can be reduced even further. Sure indeed, the collapsing provided by *BioSeqZip* comes at a possible cost of losing the read-specific quality. However, in case of need the original quality value can be easily recovered from the full uncollapsed Fastq.

Table 3. Overall impact of BioSeqZip on the alignment time

	Alignment Time (hours)		
Mapper Tool	**Raw Read File**	**Collapsed Tag File**	**Gain**
Yara	9.1	4.7	48%
BWA	16.3	7.05	57%
Bowtie2	21.2	9.5	55%
STAR	47.2	15.2	68%

Acknowledgements

We would like to heartily thank Marta Malandra for her precious help in the graphical editing, Francesco Barchi for his valuable support in solving algorithmic challenges and Elena Grassi for her useful suggestions.

Funding

This work was not supported by any organization.
Conflict of interest: None declared.

References

Barturen, G., Rueda, A., Hamberg, M., Alganza, A., Lebron, R., Kotsyfakis, M., Shi, B.-J., Koppers-Lalic, D., and Hackenberg, M. (2014). srnabench: profiling of small rnas and its sequence variants in single or multi-species high-throughput experiments. *Methods in Next Generation Sequencing*, **1**(1).

Conesa, A., Madrigal, P., Tarazona, S., Gomez-Cabrero, D., Cervera, A., McPherson, A., Szcześniak, M. W., Gaffney, D. J., Elo, L. L., Zhang, X., *et al.* (2016). A survey of best practices for rna-seq data analysis. *Genome biology*, **17**(1), 1.

Cunningham, F., Amode, M. R., Barrell, D., Beal, K., Billis, K., Brent, S., Carvalho-Silva, D., Clapham, P., Coates, G., Fitzgerald, S., *et al.* (2014). Ensembl 2015. *Nucleic acids research*, **43**(D1), D662–D669.

Desvignes, T., Loher, P., Eilbeck, K., Ma, J., Urgese, G., Fromm, B., Sydes, J., Aparicio-Puerta, E., Barrera, V., Espín, R., Thibord, F., Ros, X. B.-D., Londin, E., Telonis, A. G., Ficarra, E., Friedländer, M. R., Postlethwait, J. H., Rigoutsos, I., Hackenberg, M., Vlachos, I. S., Halushka, M. K., and Pantano, L. (2019). Unification of miRNA and isomiR research: the mirGFF3 format and the mirtop API. *Bioinformatics*.

Dobin, A., Davis, C. A., Schlesinger, F., Drenkow, J., Zaleski, C., Jha, S., Batut, P., Chaisson, M., and Gingeras, T. R. (2013). Star: ultrafast universal rna-seq aligner. *Bioinformatics*, **29**(1), 15–21.

Doring, A., Weese, D., Rausch, T., and Reinert, K. (2008). Seqan an efficient, generic c++ library for sequence analysis. *BMC bioinformatics*, **9**(1), 11.

Friedländer, M. R., Mackowiak, S. D., Li, N., Chen, W., and Rajewsky, N. (2011). mirdeep2 accurately identifies known and hundreds of novel microrna genes in seven animal clades. *Nucleic acids research*, **40**(1), 37–52.

Fromm, B., Billipp, T., Peck, L. E., Johansen, M., Tarver, J. E., King, B. L., Newcomb, J. M., Sempere, L. F., Flatmark, K., Hovig, E., *et al.* (2015). A uniform system for the annotation of vertebrate microrna genes and the evolution of the human micrornaome. *Annual review of genetics*, **49**, 213–242.

Fromm, B., Domanska, D., Hackenberg, M., Mathelier, A., Hoye, E., Johansen, M., Hovig, E., Flatmark, K., and Peterson, K. J. (2018). Mirgenedb2. 0: the curated microrna gene database. *bioRxiv*, page 258749.

Giardine, B., Riemer, C., Hardison, R. C., Burhans, R., Elnitski, L., Shah, P., Zhang, Y., Blankenberg, D., Albert, I., Taylor, J., *et al.* (2005). Galaxy: a platform for interactive large-scale genome analysis. *Genome research*, **15**(10), 1451–1455.

González-Domínguez, J. and Schmidt, B. (2016). Pardre: faster parallel duplicated reads removal tool for sequencing studies. *Bioinformatics*, **32**(10), 1562–1564.

Gordon, A. and Hannon, G. (2010). Fastx-toolkit. *FASTQ/A short-reads preprocessing tools (unpublished)*, **5**.

Griffiths-Jones, S., Grocock, R. J., van Dongen, S., Bateman, A., and Enright, A. J. (2006). mirbase: microrna sequences, targets and gene nomenclature. *Nucleic Acids Research*, **34**(suppl 1), D140–D144.

Knuth, D. E. (1998). *The art of computer programming: sorting and searching*, volume 3. Pearson Education.

Langmead, B. and Salzberg, S. L. (2012). Fast gapped-read alignment with bowtie 2. *Nature methods*, **9**(4), 357–359.

Li, H. and Durbin, R. (2009). Fast and accurate short read alignment with burrows–wheeler transform. *Bioinformatics*, **25**(14), 1754–1760.

Mazzoni, G. and Kadarmideen, H. N. (2016). Computational methods for quality check, preprocessing and normalization of rna-seq data for systems biology and analysis. In *Systems Biology in Animal Production and Health, Vol. 2*, pages 61–77. Springer.

Muir, P., Li, S., Lou, S., Wang, D., Spakowicz, D. J., Salichos, L., Zhang, J., Weinstock, G. M., Isaacs, F., Rozowsky, J., *et al.* (2016). The real cost of sequencing: scaling computation to keep pace with data generation. *Genome biology*, **17**(1), 53.

Nellore, A., Collado-Torres, L., Jaffe, A. E., Alquicira-Hernández, J., Wilks, C., Pritt, J., Morton, J., Leek, J. T., and Langmead, B. (2016). Rail-rna: scalable analysis of rna-seq splicing and coverage. *Bioinformatics*, **33**(24), 4033–4040.

Ogasawara, O., Otsuji, M., Watanabe, K., Iizuka, T., Tamura, T., Hishiki, T., Kawamoto, S., and Okubo, K. (2006). Bodymap-xs: anatomical breakdown of 17 million animal ests for cross-species comparison of gene expression. *Nucleic acids research*, **34**(suppl_1), D628–D631.

Pantano, L., Estivill, X., and Martí, E. (2009). Seqbuster, a bioinformatic tool for the processing and analysis of small rnas datasets, reveals ubiquitous mirna modifications in human embryonic cells. *Nucleic acids research*, **38**(5), e34–e34.

Pantano, L., Estivill, X., and Martí, E. (2011). A non-biased framework for the annotation and classification of the non-mirna small rna transcriptome. *Bioinformatics*, **27**(22), 3202–3203.

Petersen, K. R., Streett, D. A., Gerritsen, A. T., Hunter, S. S., and Settles, M. L. (2015). Super deduper, fast pcr duplicate detection in fastq files. In *Proceedings of the 6th ACM Conference on Bioinformatics, Computational Biology and Health Informatics*, pages 491–492. ACM.

Reinert, K., Dadi, T. H., Ehrhardt, M., Hauswedell, H., Mehringer, S., Rahn, R., Kim, J., Pockrandt, C., Winkler, J., Siragusa, E., Urgese, G., and Weese, D. (2017). The seqan c++ template library for efficient sequence analysis: a resource for programmers. *Journal of biotechnology*, **261**, 157–168.

Scherer, A. *et al.* (2019). Towards best practice in cancer mutation detection with wholegenome and whole-exome sequencing. *Nature Biotechnology*.

Siragusa, E. (2015). *Approximate string matching for high-throughput sequencing*. Ph.D. thesis, Freie Universität Berlin.

Smith, T. F. and Waterman, M. S. (1981). Comparison of biosequences. *Advances in applied mathematics*, **2**(4), 482–489.

Urgese, G., Paciello, G., Acquaviva, A., Ficarra, E., Graziano, M., and Zamboni, M. (2014). Dynamic gap selector: A smith waterman sequence alignment algorithm with affine gap model optimisation. In *In Proc. 2nd Int. Work-Conf. Bioinform. Biomed. Eng.(IWBBIO): 7-9 April 2014; Granada*, pages 1347–1358. Copicentro Granada SL.

Urgese, G., Paciello, G., Acquaviva, A., and Ficarra, E. (2016). isomir-sea: an rna-seq analysis tool for mirnas/isomirs expression level profiling and mirna-mrna interaction sites evaluation. *BMC bioinformatics*, **17**(1), 148.

Xu, H., Luo, X., Qian, J., Pang, X., Song, J., Qian, G., Chen, J., and Chen, S. (2012). Fastuniq: a fast de novo duplicates removal tool for paired short reads. *PloS one*, **7**(12), e52249.

Zhang, P., Si, X., Skogerbø, G., Wang, J., Cui, D., Li, Y., Sun, X., Liu, L., Sun, B., Chen, R., *et al.* (2014). pirbase: a web resource assisting pirna functional study. *Database*, **2014**.