Sorting Next Generation Sequencing Data Improves Compression Effectiveness

Raymond Wan and Kiyoshi Asai
University of Tokyo and CBRC, AIST, Japan

December 20, 2010

Introduction

As interest in next generation sequencing (NGS) grows, the problem of effectively storing and transmitting such data will need to be addressed.

Current NGS repositories include NCBI's Sequence Read Archive (SRA) and the mirrors at EBI and DDBJ. They use:

1. the FASTQ data format and
2. GZIP (a general-purpose compression system).

Overview

Aim: To improve compression of FASTQ data using GZIP by pre-sorting the reads.

Results: On a collection of six data sets, we achieve

1. at most 12% when only the sequences are considered and
2. up to 6% when the entire FASTQ data is processed.

(*Note: Due to time constraints, I will talk about only one data set in this talk. Results with all six are presented in the Workshop paper.*)

NGS Data Format and Compression

Next generation sequencing data

Currently, repositories use the FASTQ format to archive data:

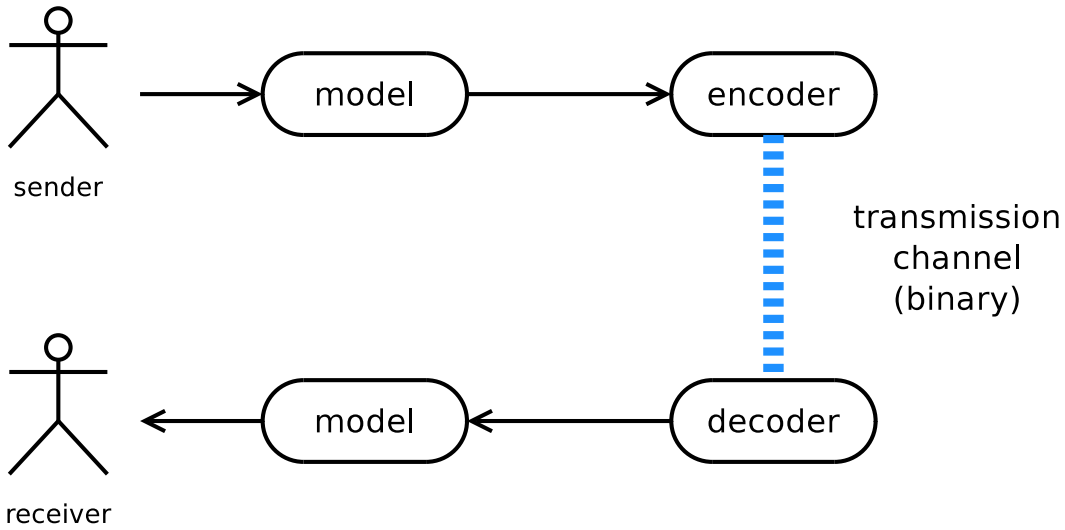
#	FASTQ Data
1	`@SRR005489.1 :5:1:380:1628 length=48`
2	`TACTCATACAATTATTCATATTATAATT`
3	`+SRR005489.1 :5:1:380:1628 length=48`
4	`II$IIII"IIIIIIIIIIIII=IIIII`
5	`@SRR005489.2 :5:1:962:1174 length=48`
6	`TTCCTTTTTCAAAAATTGAATTAAAGTG`
7	`+SRR005489.2 :5:1:962:1174 length=48`
8	`II$IIII"IIIIIIIIIIIIIIIIIII`

Key:

1,5	identifier	2,6	sequence
3,7	'+' and optional information	4,8	quality scores

Compression

Purpose is to model and transform data into a more economical form.



GZIP *compression*

Model based on the LZ77 algorithm [Ziv and Lempel, 1977]. Slides a window over the data, representing new data as a back-pointer and a copy-length.

$\langle 6, 6, \text{i} \rangle$ `woodchuck␣chuck␣if`

In this example, if "`chuck␣if`" is being modeled and everything before it has been processed using an 8-character window, then we can model *part* of it as:

> "Go back 6 characters and copy 6 and then add an '`i`'."

(*Note:* BZIP2 *compression is also mentioned in the results and discussed in more detail in the Workshop paper.*)

Our idea...

Since the order of reads in NGS data do not matter, compression should improve if similar reads are brought closer together (ideally, within the sliding window).

Lossy transformation – after the reads are moved around, we cannot recover the original order. But none of the data itself is lost (none of the reads are deleted and no part of each read is changed).

The most natural (and repeatable) way to bring similar reads closer together is sorting.

Sorting

Sorting algorithms

We apply two basic, well-known sorting algorithms:

	Running time	Stability	Purpose
radix sort	$O(n)$	stable	Fixed-length items
quicksort	$O(n \lg n)$, [$O(n^2)$, worst]	non-stable	General algorithm

We used these algorithms as-is [Cormen et al., 2009] but add parameters to terminate early to take "snapshots" of the data at various points during sorting.

Radix sort

Sorts n items by examining each position, starting from the least significant character.

Usually, most of the items are not in their final positions until the first character is processed.

Items should be equal in length. Since 454 GS FLX reads are not, we *artificially* pad them to the longest one...

Radix sort example

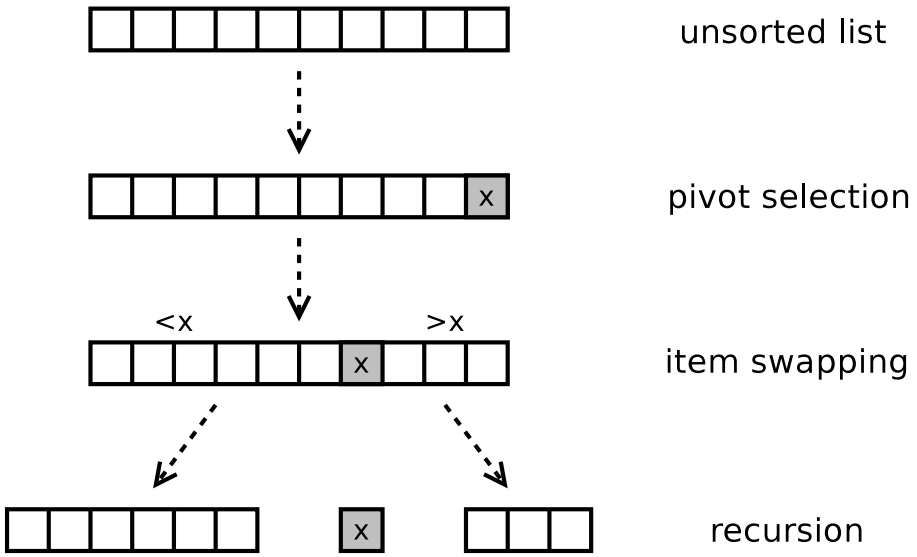
Original list	Column position			
	$k=1$	$k=2$	$k=3$	$k=4$
deck	data	deck	data	data
late	late	data	late	date
days	date	late	date	days
date	deck	date	days	deck
data	days	days	deck	late

We define p_r as the percentage of least significant positions to examine such that:

- $p_r = 0\%$ means original order and
- $p_r = 100\%$ means fully sorted.

Quicksort

Sorts n items by recursively partitioning a list around a pivot.



Once a pivot is selected and moved, it is in its **final** position. Everything to the left of it is less than it; everything to the right is greater.

We used randomized pivot selection to avoid the worst case running time of $O(n^2)$ [Cormen et al., 2009].

Quicksort example

Always choosing the last item as the pivot.

Original list	Fixed pivot				
	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
deck	data	data	data	data	data
late	late	days	date	date	date
days	days	date	days	days	days
date	date	deck	deck	deck	deck
data	deck	late	late	late	late

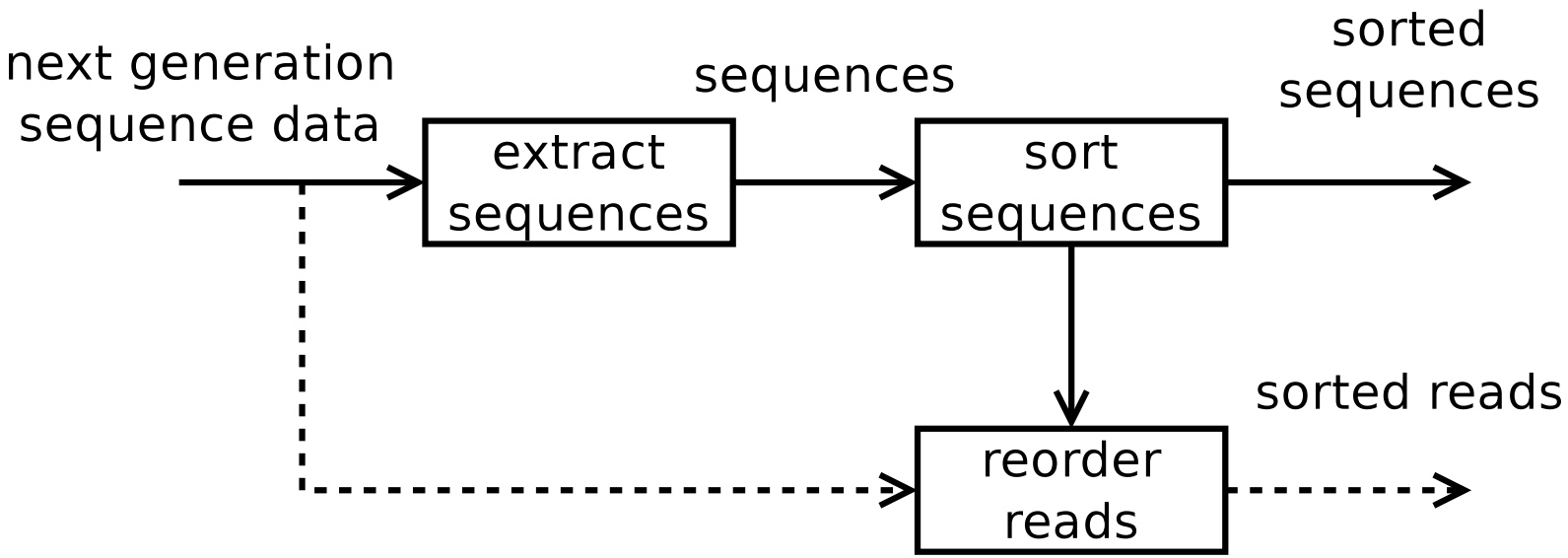
We define p_q as the percentage of items in their final positions such that:

- $p_q = 0\%$ means original order and
- $p_q = 100\%$ means fully sorted.

Framework

Experiment framework

We still need to use the sorted order to deal with the original FASTQ data...



How about the FASTQ *data?*

4 schemes considered:

Original	Use the sorted sequences to reorder the reads.
Independent	Split the data file into sequences and quality scores, compress each independently *with* their identifiers.

How about the FASTQ *data?*

4 schemes considered:

Original	Use the sorted sequences to reorder the reads.
Independent	Split the data file into sequences and quality scores, compress each independently *with* their identifiers.

File #1:

```
@SRR005489.1 :5:1:380:1628 length=48
TACTCATACAATTATTCATATTATAATT
```

File #2:

```
@SRR005489.1 :5:1:380:1628 length=48
II$IIII"IIIIIIIIIIIII=IIIII
```

How about the FASTQ *data? (cont.)*

Identifier Removed	Replace the third line with just '+' and use the sorted sequences to re-order the reads.

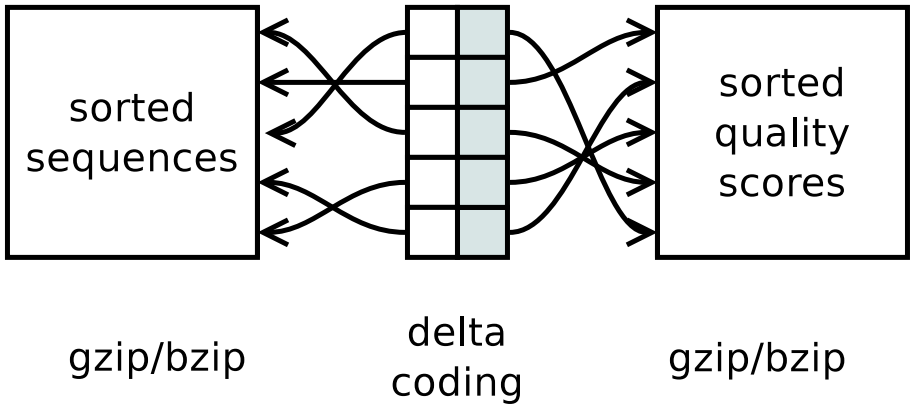
How about the FASTQ *data? (cont.)*

Identifier Removed	Replace the third line with just '+' and use the sorted sequences to re-order the reads.

```
@SRR005489.1 :5:1:380:1628 length=48
TACTCATACAATTATTCATATTATAATT
+
II$IIII"IIIIIIIIIIIIII=IIIII
```

How about the FASTQ *data? (cont.)*

Indexed Split the data file into 3 parts (identifiers, sequences, and quality scores), sort each one, and compress. Construct an index (ordered by the identifiers) to reassemble the data.



Intuitively, should compress well, but does the gain in compression offset the extra cost of the index?

Results

Data sets

Experiments with NCBI's SRA data:

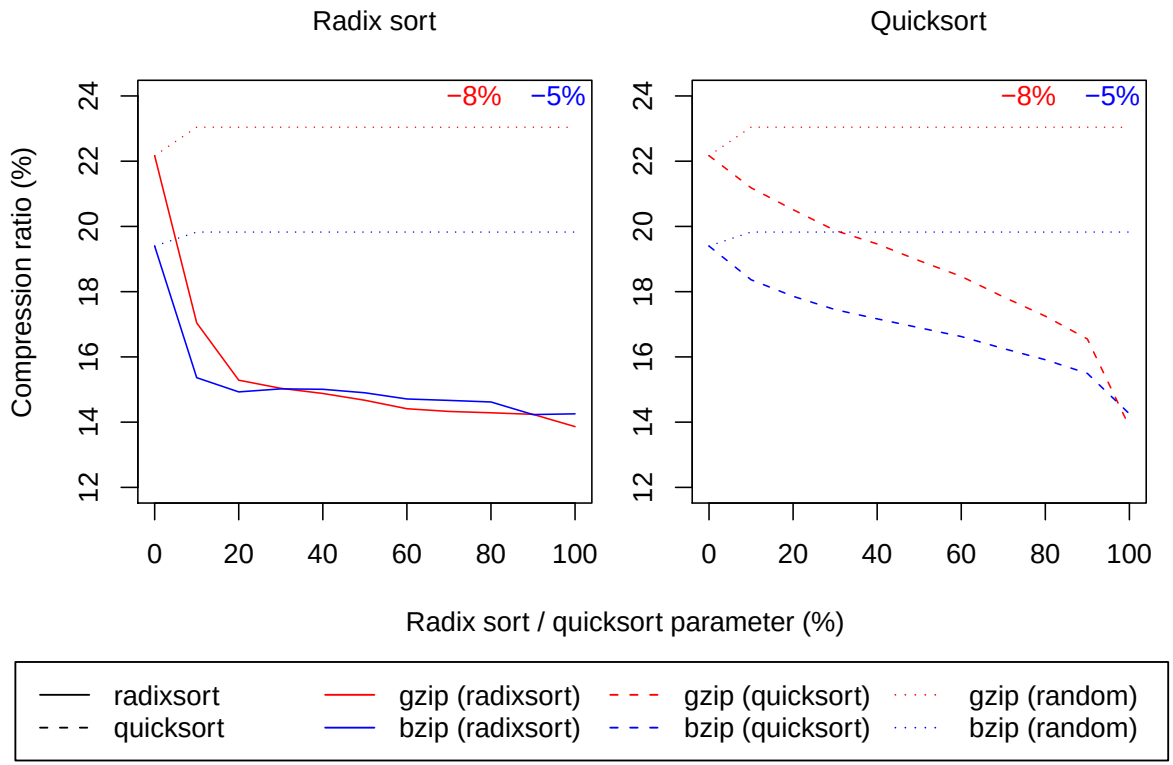
Accession		SRR005489
Species		*P. falciparum*
Platform		Solexa
Identifier %		46.9%
Uncompressed sizes	(Sequences) (FASTQ)	128.4 MiB 483.9 MiB
# reads		2.7×10^6
Read lengths	(Range) (Median)	[48, 48] 48

Observations:

- Almost half of the data file is read identifier information.

SRR005489 (sequences)

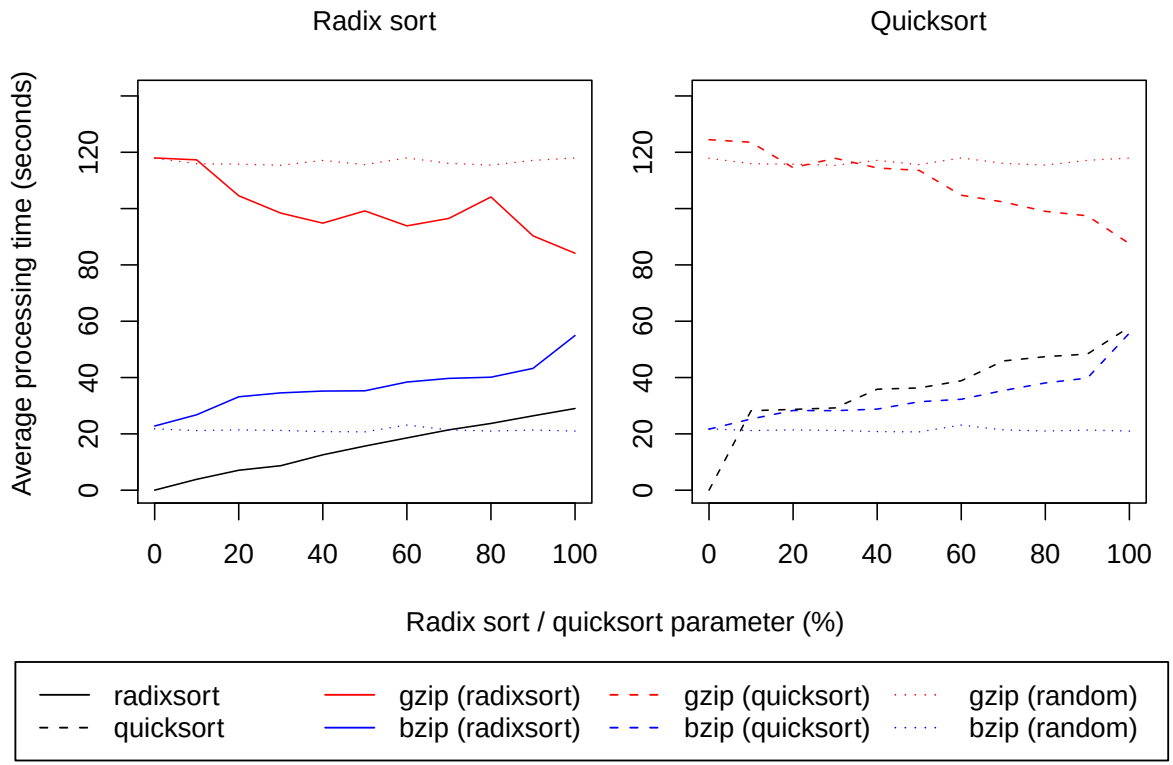
Compression ratio of just the sequences:



Decrease of 5% to 8% compared to unsorted ordering; random permutation is noticeably worse.

SRR005489 (sequences)

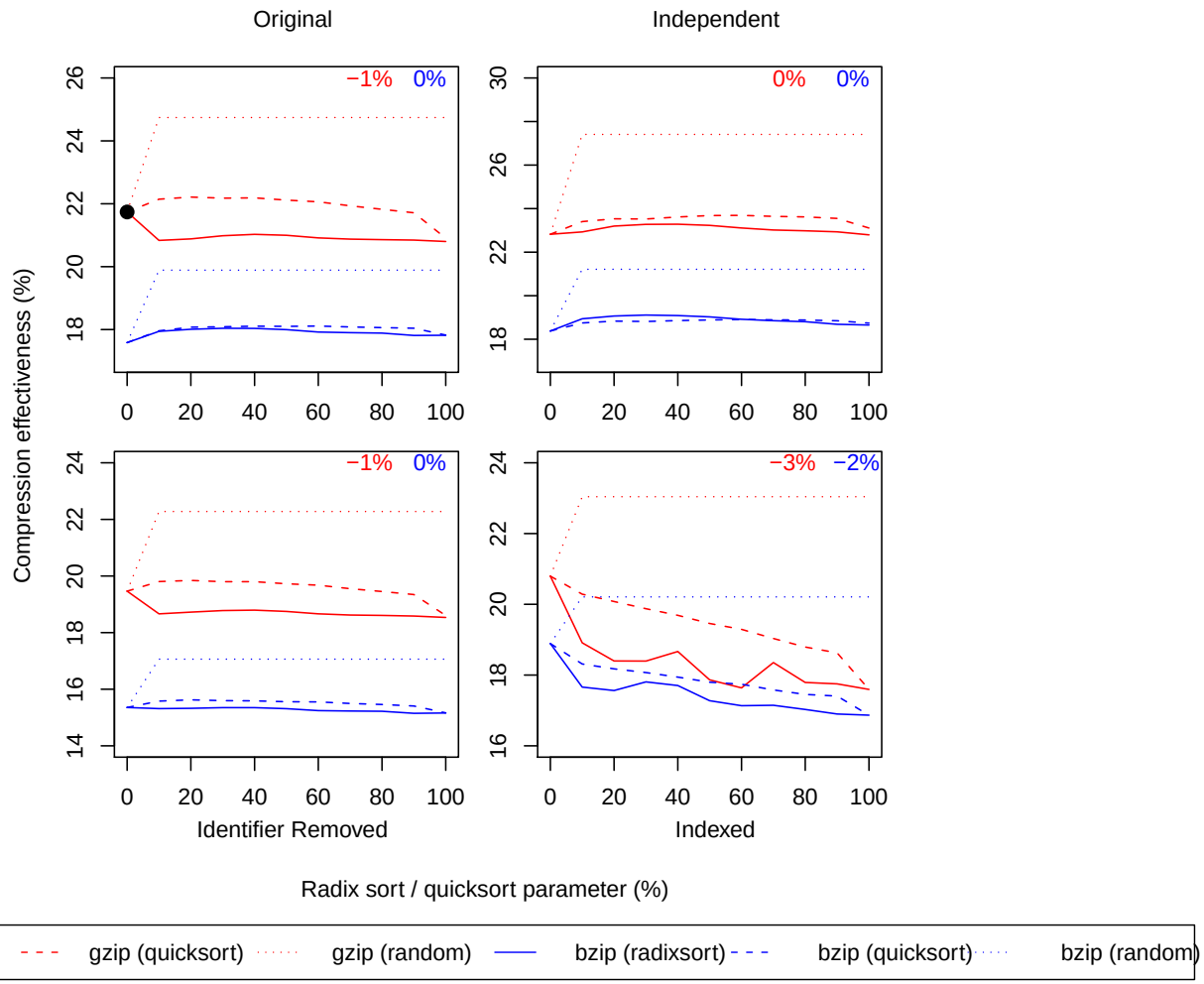
Compression time (user seconds) of just the sequences:



Averaged over 3 trials (2.53 GHz Intel Xeon E5540 with 6 GB of RAM).

SRR005489 (FASTQ)

Compression ratio of the FASTQ data:



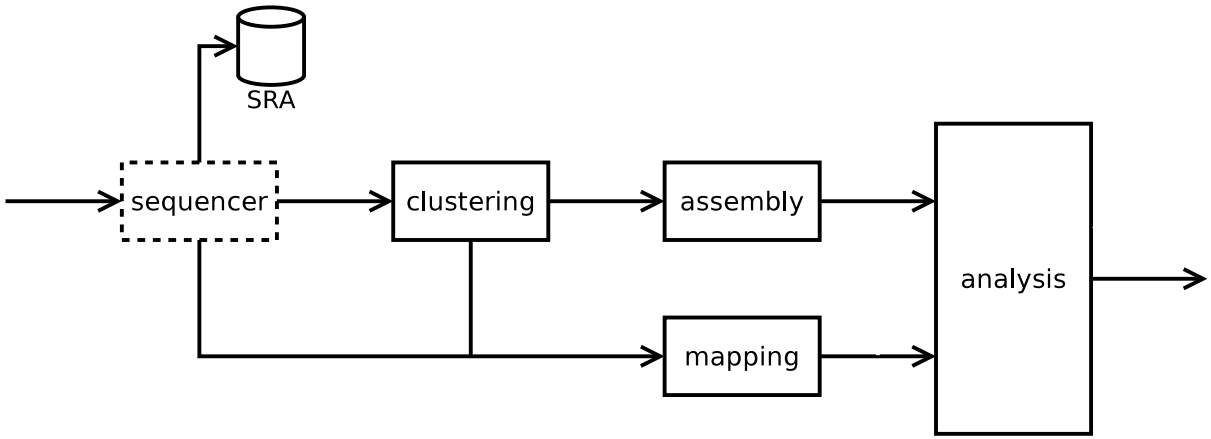
Conclusion

Summary

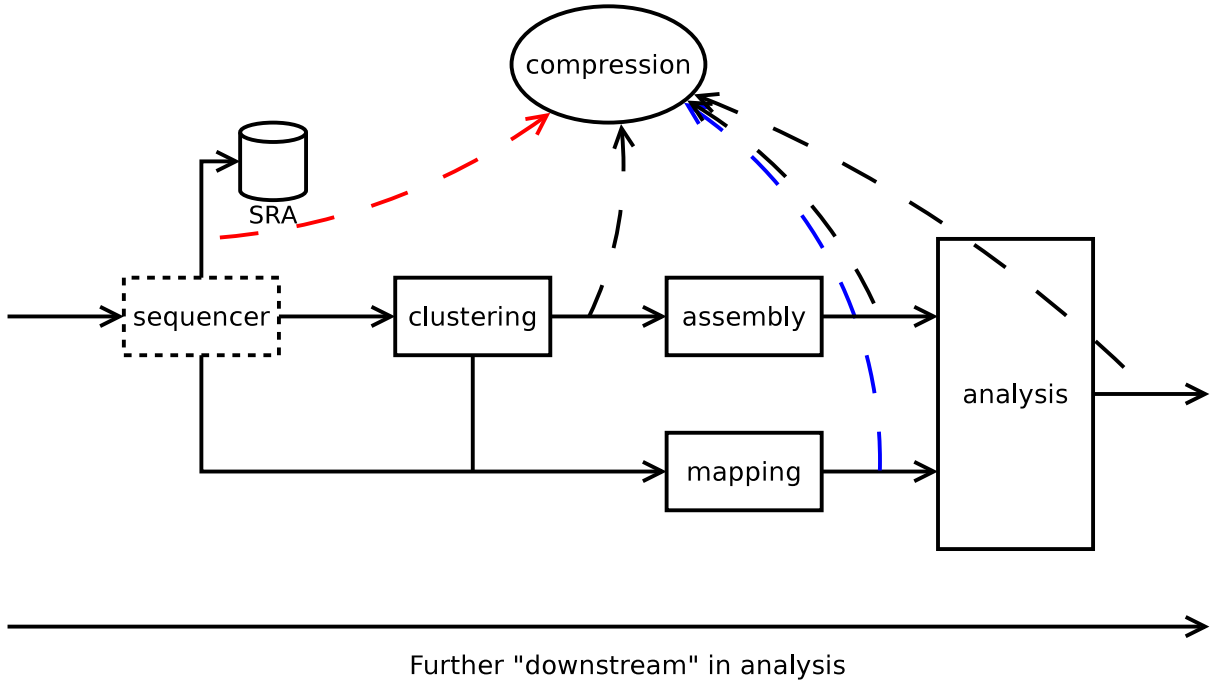
Workshop paper includes:

- Results for the remaining 5 data sets (covering a range of species and both Illumina and 454 GS FLX data) are in the Workshop paper.
- Complete results with BZIP2 with some discussion.
- Decrease in ratio of up to 12% for just sequences (SRR014437 using GZIP); and 6% for FASTQ data (SRR033869 and SRR034845 using **Indexed**).

Summary



Summary



Keypoint: While the gain in compression ratio is small, the advantage is the simplicity of the method – anyone can easily incorporate sorting into existing GZIP-pipelines.

Final words

Contact: Raymond Wan (r-wan@cb.k.u-tokyo.ac.jp)
Kiyoshi Asai (asai@k.u-tokyo.ac.jp)

C++ source code and Perl scripts used in these experiments are available from http://www.cbrc.jp/~rwan/.

Completely unrelated...but, poster #P281:

"Mitochondrial Protein Cleavage Site Predictor"
Yoshinori Fukasawa, Raymond Wan, Szu-Chin Fu, Junko Tsuji, Noriyuki Sakiyama, Kenichiro Imai, and **Paul Horton**

Describes our ongoing work on developing an HMM-based method for determining the cleavage sites of proteins cleaved by mitochondrial peptidases.

References

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, USA, third edition, 2009

P. Elias. Universal codeword sets and representations of the integers. *IEEE Transactions on Information Theory*, 21(2):194–203, March 1975

H. Liefke and D. Suciu. XMill: an efficient compressor for XML data. In W. Chen, J. F. Naughton, and P. A. Bernstein, editors, *Proc. ACM SIGMOD 2000 International Conference on Management of Data*, pages 153–164, New York, May 2000. ACM Press

R. Wan. *Browsing and Searching Compressed Documents*. PhD thesis, University of Melbourne, Australia, December 2003

I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes*. Morgan Kaufmann, second edition, 1999

J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, May 1977

Appendix

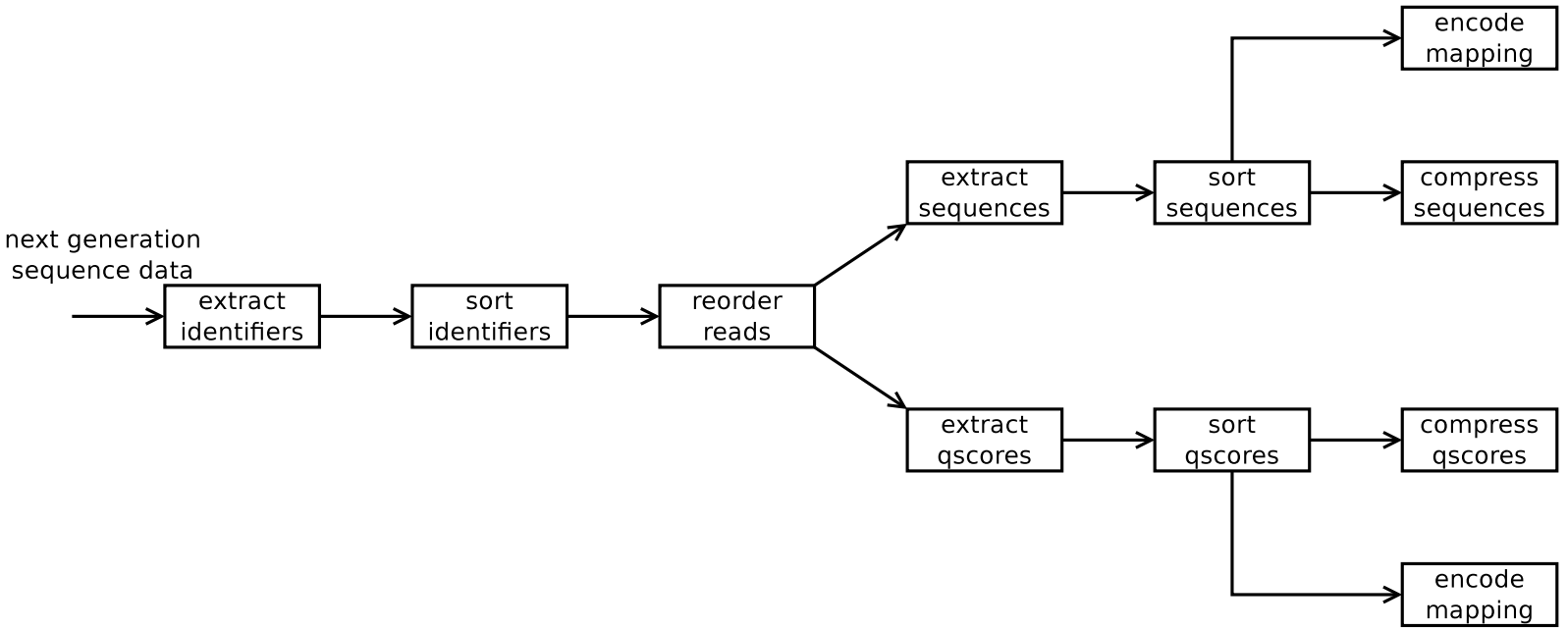
- Gamma and delta coding
- Indexed scheme (workflow)
- About all of the data
- Results – sequences (compression)
- Results – sequences (time)
- Results – FASTQ

Gamma and delta coding

- Method for coding integers (i.e., converting a decimal number to bits).
- Binary coding is well-known: 9 $\rightarrow$ 1001.
- Unary coding: 9 $\rightarrow$ 111111110.
- Gamma coding [Elias, 1975] represents x as $1 + \lfloor \log x \rfloor$ in unary and then $x - 2^{\lfloor \log x \rfloor}$ in binary. i.e., 9 $\rightarrow$ 1110 001.
- Delta coding [Elias, 1975] represents x as $1 + \lfloor \log x \rfloor$ in gamma and then $x - 2^{\lfloor \log x \rfloor}$ in binary. i.e., 9 $\rightarrow$ 11000 001.

Back

Pipeline for Indexed



Back

Data sets

Experiments with six data sets from NCBI's SRA.

Accession	Species	Platform	Identifier percentage
SRR005489	*P. falciparum*	Solexa	46.9%
SRR014437	*S. cerevisiae*	Solexa	60.0%
SRR014835_1	*H. sapiens*	Illumina	40.6%
SRR020180	*M. musculus*	454	15.6%
SRR033869	*Hordeum vulgare L.* (barley)	454	18.0%
SRR034845	*D. melanogaster*	454	10.2%

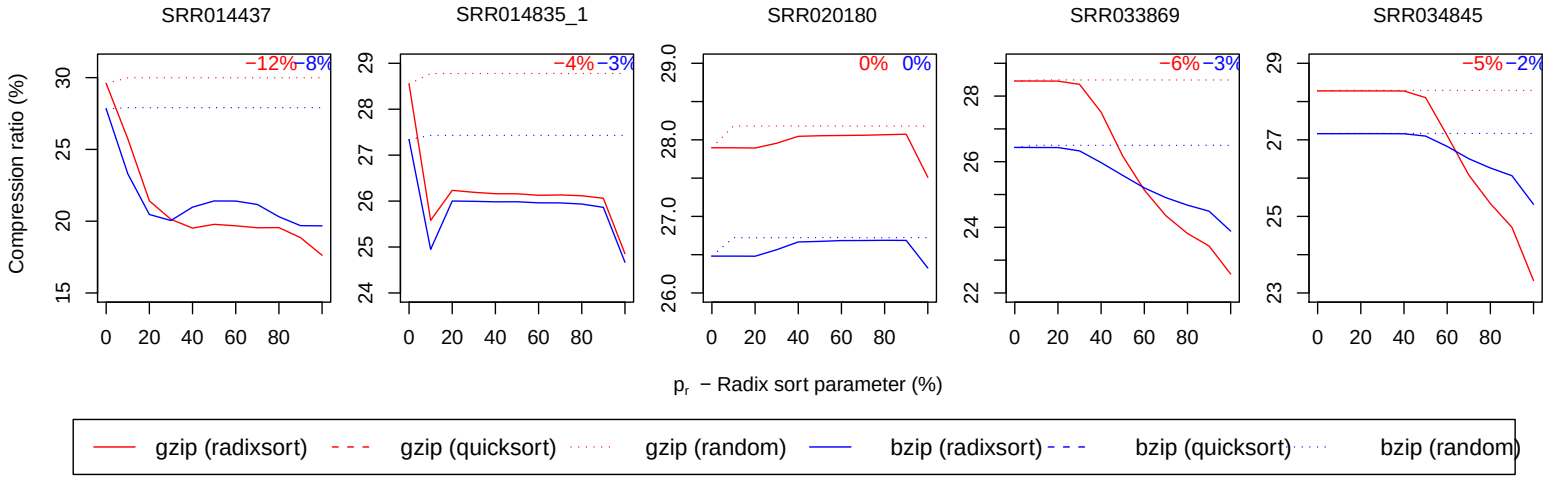
Accession	Uncompressed sizes		# reads ($\times 10^6$)	Read lengths	
	sequences	FASTQ		Range	Median
SRR005489	128.4 MiB	483.9 MiB	2.7	[48, 48]	48
SRR014437	326.0 MiB	1,630.1 MiB	11.8	[28, 28]	28
SRR014835_1	674.2 MiB	2,269.9 MiB	9.2	[76, 76]	76
SRR020180	56.0 MiB	132.5 MiB	0.2	[36, 351]	247
SRR033869	88.9 MiB	216.9 MiB	0.5	[32, 412]	217
SRR034845	455.8 MiB	1015.4 MiB	1.2	[23, 1009]	423

- Illumina and Solexa reads are short; identifiers long.
- 454 data files are noticeably smaller.

Back

All sequences (ratio)

Compression ratio of radix sort and all remaining sequences (exclude SRR005489).



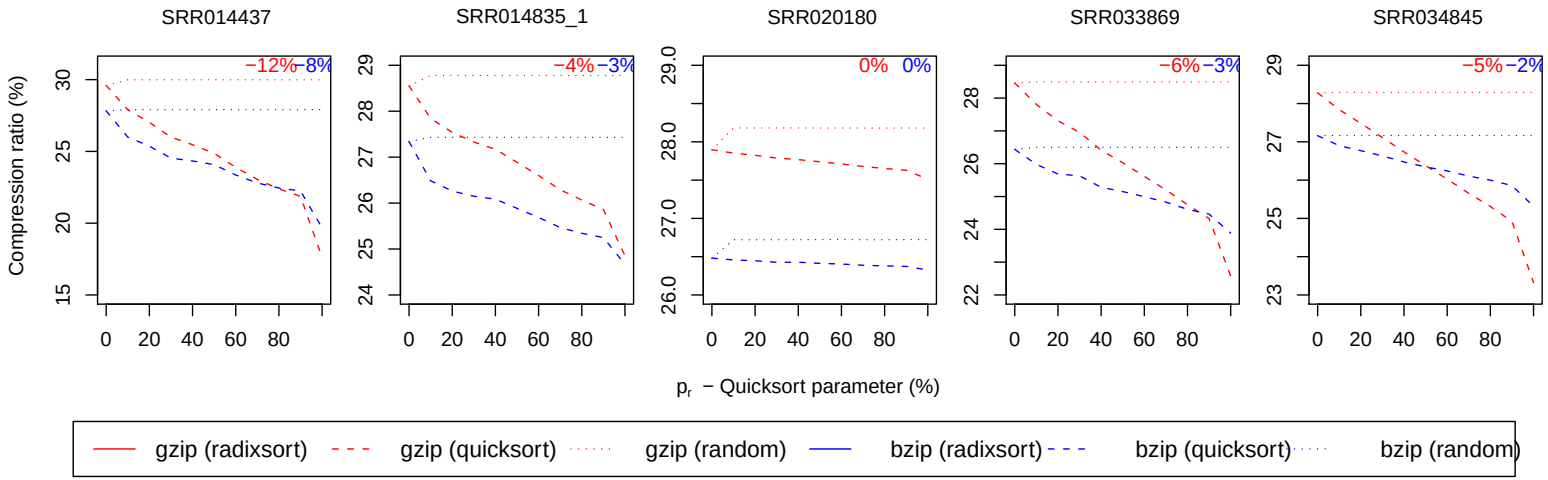
Decrease of 0% to 12% for GZIP; 0% to 8% for BZIP2.

Neglible change for SRR20180.

Random permutation performs poorly throughout.

All sequences (ratio)

Compression ratio of quicksort and all remaining sequences (exclude SRR005489).

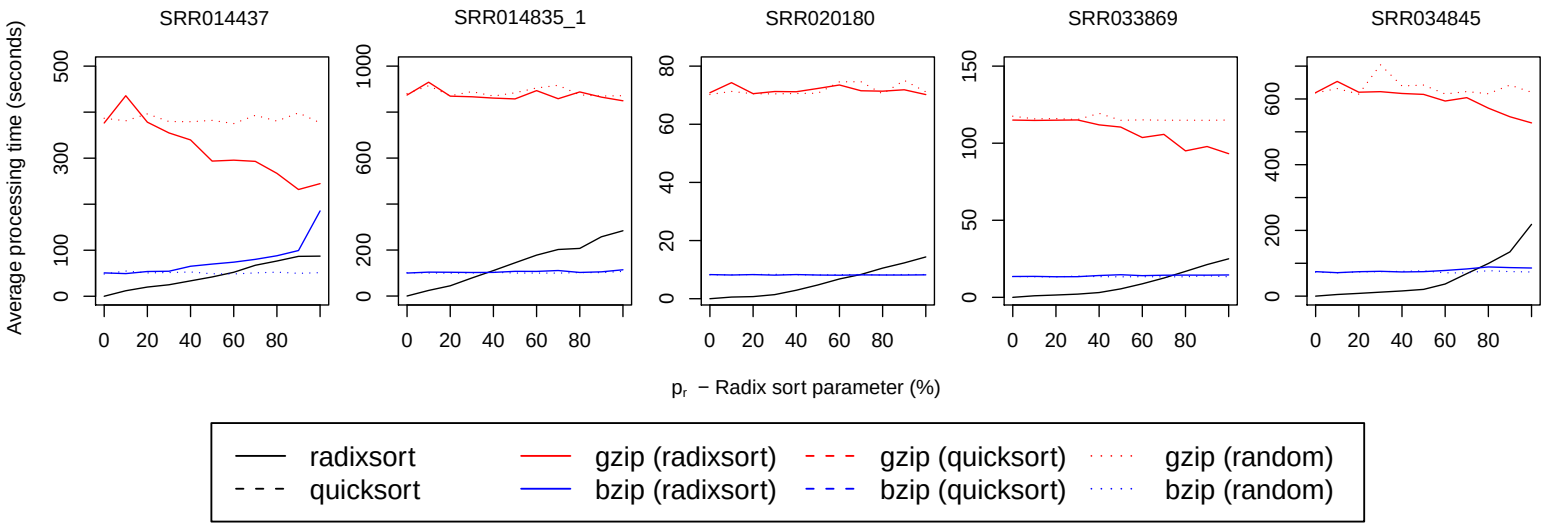


Lines are much more gradual for quicksort (as expected).

Back

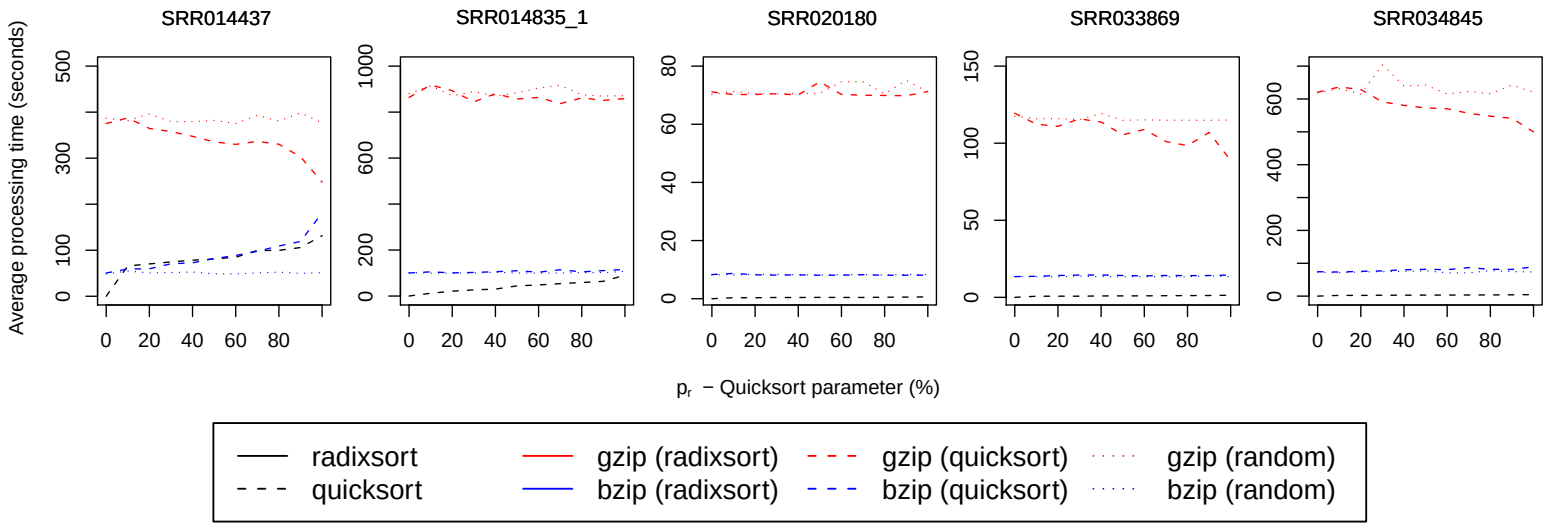
All sequences (time)

Compression time of radix sort and all remaining sequences (exclude SRR005489).



All sequences (time)

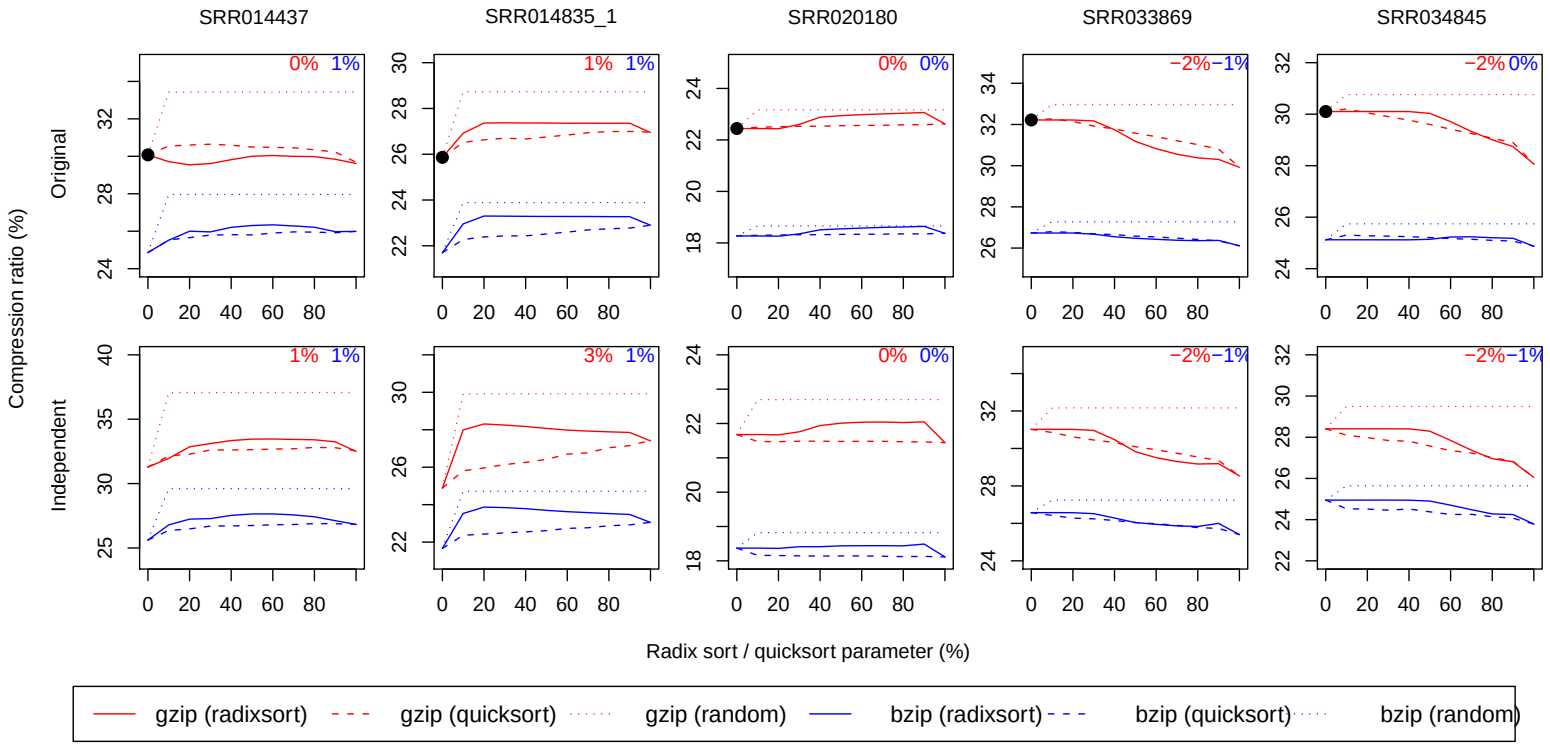
Compression time of quicksort and all remaining sequences (exclude SRR005489).



Back

All sequences (FASTQ)

Compression ratio of the FASTQ data:

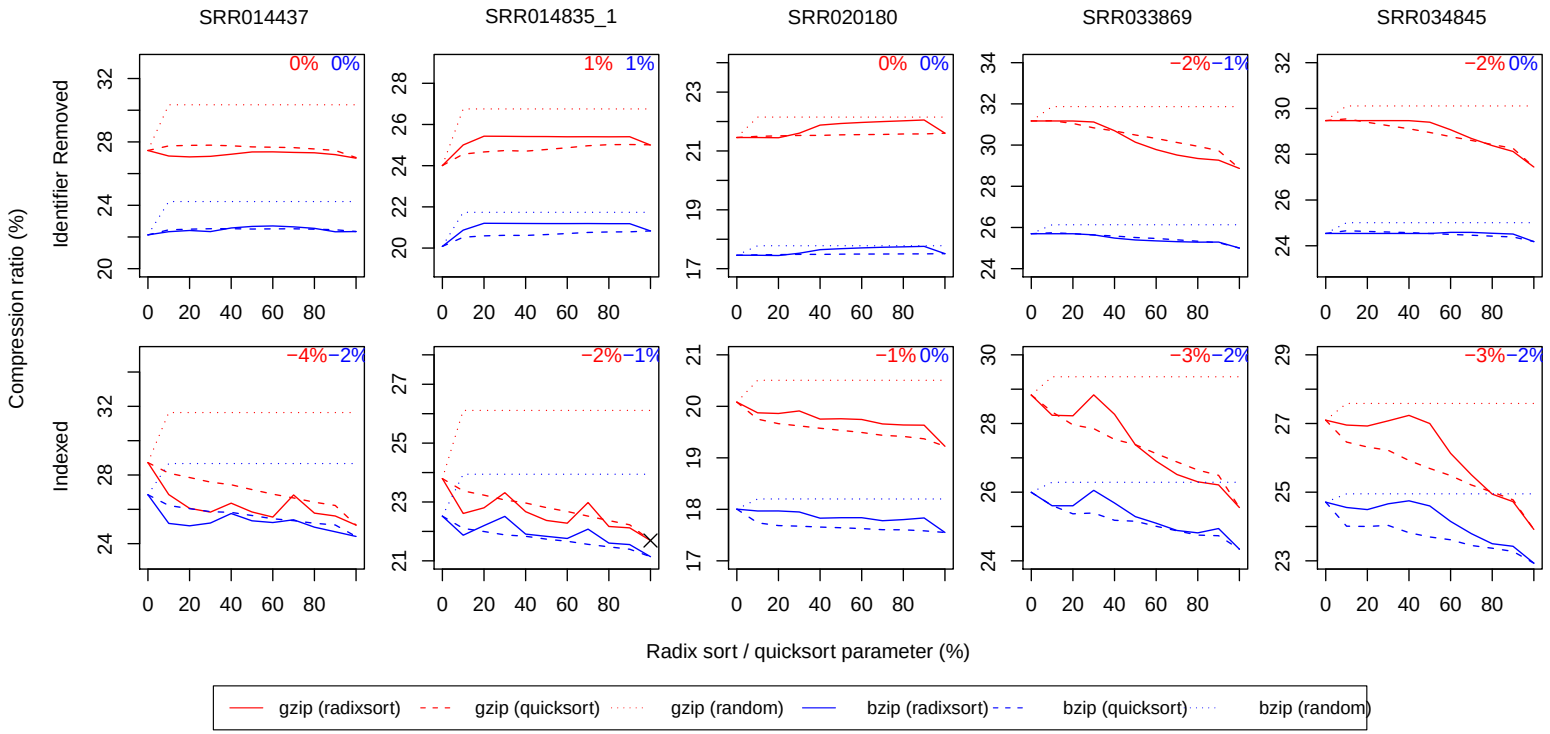


Sorting helps only for the last two 454 data sets.

Independent does worse than **Original**; clearly separating the data into two files has a negative effect.

All sequences (FASTQ)

Compression ratio of the FASTQ data:



Original and unsorted: 30%, 26%, 22%, 32%, and 30%.
Identifier Removed slightly better than **Independent**;
Indexed the best: 26%, 22%, 19.5%, 26%, and 24%.

Back