SPRING: A next-generation compressor for FASTQ data

Shubham Chandak[1], Kedar Tatwawadi[1], Idoia Ochoa[2], Mikel Hernaez[2] & Tsachy Weissman[1]
[1]Stanford University, [2]University of Illinois at Urbana-Champaign
Download: https://github.com/shubhamchandak94/SPRING/

Introduction

- High-Throughput Sequencing (HTS) experiments generate FASTQ files consisting of unaligned reads along with read identifiers and quality scores.
- A typical experiment generates 100s of GBs of uncompressed data.

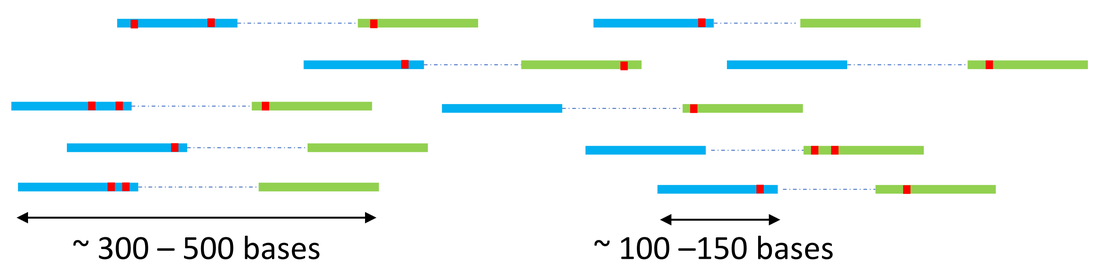

Fig. 1: Noisy paired-end reads from genome sequencing experiment.

Fig. 2: Sample FASTQ file.

Entropy of reads

Notation: m: Genome length, n: Number of reads, $H(X)$: entropy of X

Entropy of reads with $(*)$ exact order preserved & $(**)$ only pairing preserved:

$$H(reads) \lesssim \underbrace{H(genome)}_{\text{Store genome}} + \underbrace{\left\{\begin{matrix}(*) & \frac{n}{2}\log_2 m \\ (**) & \log_2 \binom{m+\frac{n}{2}-1}{m-1}\end{matrix}\right\}}_{\text{Store positions of read pairs in genome}} + \underbrace{\frac{n}{2}(H(insert\ size)+1)}_{\text{Store insert size \& orientation}} + \underbrace{nH(noise)}_{\text{Store noisy bases}}$$

SPRING

- SPRING - practical compressor for FASTQ files.
  - → Support for wide variety of modes.
  - → Support for variable length short reads.
  - → Substantially better compression than existing tools.
  - → Available at https://github.com/shubhamchandak94/SPRING/

Results

- **Perfectly lossless mode** - Entire FASTQ file stored as it is.
- **Information-preserving mode** - Store only information needed for downstream applications:
  - → Read identifiers not retained.
  - → Only read pairing information retained, pairs reordered arbitrarily.
  - → Quality scores binned (for older datasets).

Compression of human FASTQ datasets

Technology	Cvg.	Read length	Size	Perfectly Lossless			Information-Preserving	
				Gzip	FaStore	SPRING	FaStore	SPRING
HiSeq 2000	28x	101	227	74	35.8	**28.0**	17.5	**13.2**
NovaSeq	25x	147	196	36	11.2	**6.9**	10.0	**5.6**
NovaSeq	100x	147	788	145	34.2	**25.5**	29.4	**20.1**

Fig. 3: All sizes are in GB.

Read Compression Algorithm

Stage I - Reordering: We try to reorder reads according to genome position using a dictionary-based greedy scheme.

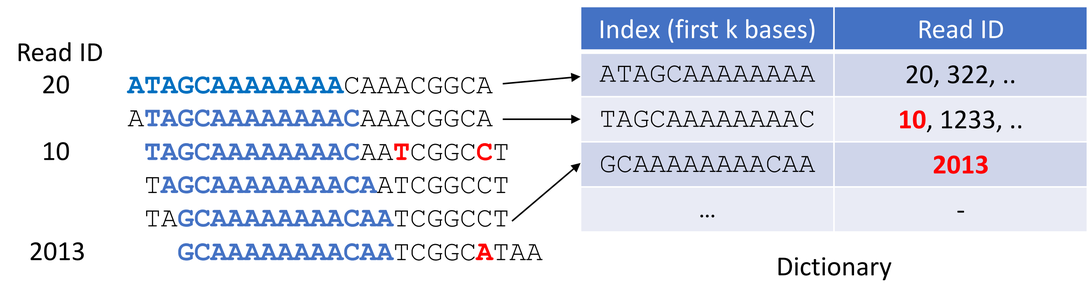

Fig. 4: Finding next read using dictionary indexed by read prefix.

Stage II - Encoding: We encode the reordered reads to remove the redundancy in consecutive reads.

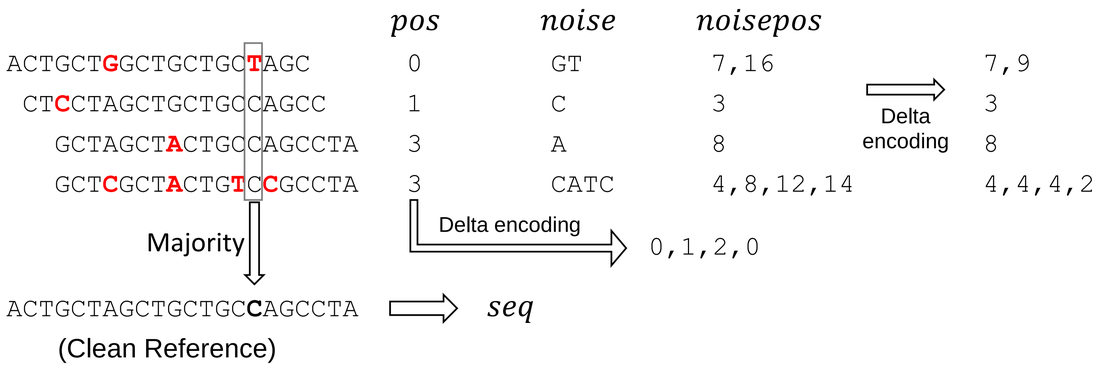

Fig. 5: Encoding reordered reads into four streams.

We also transform the original order of reads to exploit read pairing.

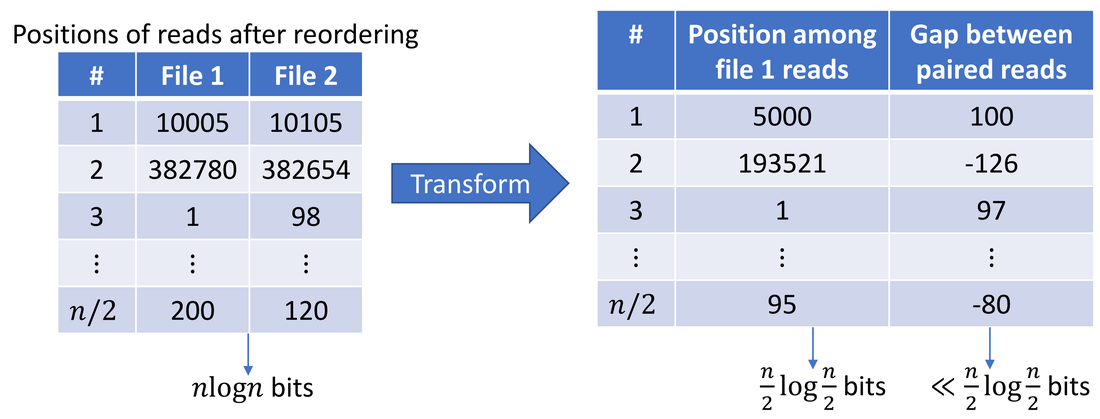

Fig. 6: Encoding of read order. Only second column (gaps) needed for preserving read pairing.

Stage III - Compression: We compress the encoded streams using 7-zip and BSC.

Analysis

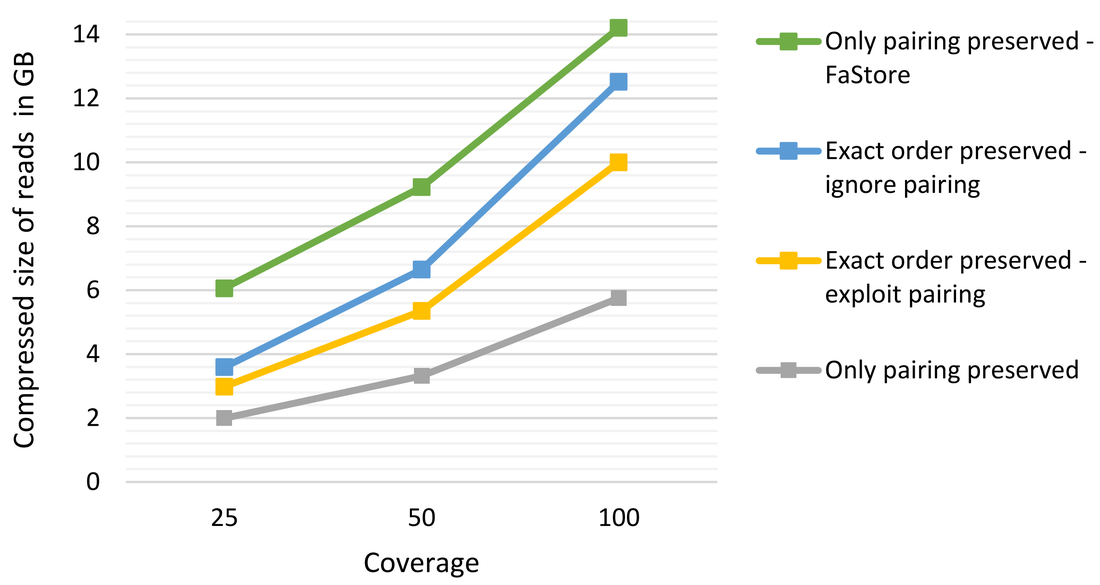

Fig. 7: Comparison of read compression modes for human NovaSeq data.

Future Work

- Integration into MPEG-G standard for genomic information representation.
- Extension to long-read technologies.