SPRING: A next generation compressor for FASTQ data

Shubham Chandak

Stanford University

Allerton Conference, 3rd October 2018

Joint work with

- Kedar Tatwawadi, Stanford University
- Idoia Ochoa, UIUC
- Mikel Hernaez, UIUC
- Tsachy Weissman, Stanford University

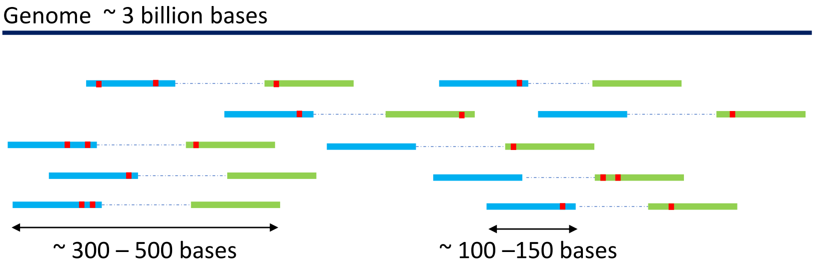
Outline

Introduction
- High-Throughput Sequencing
- Entropy of reads

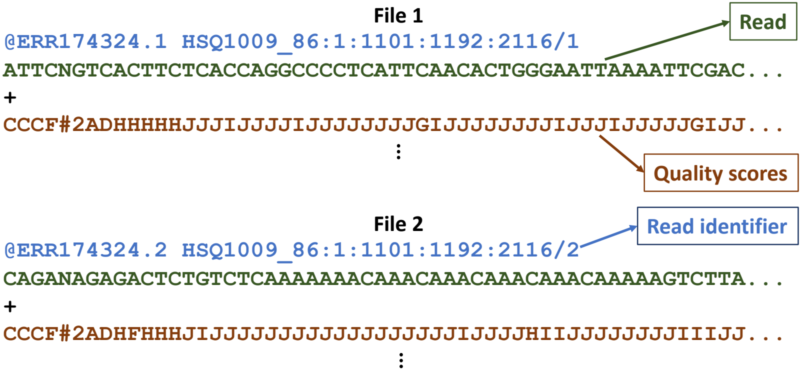
Methods

Results

High-Throughput Sequencing



FASTQ format



File 1

```
@ERR174324.1 HSQ1009_86:1:1101:1192:2116/1
ATTCNGTCACTTCTCACCAGGCCCCTCATTCAACACTGGGAATTAAAATTCGAC...
+
CCCF#2ADHHHHHJJJIJJJJIJJJJJJJJGIJJJJJJJJIJJJIJJJJJGIJJ...
                              ⋮
```

Read

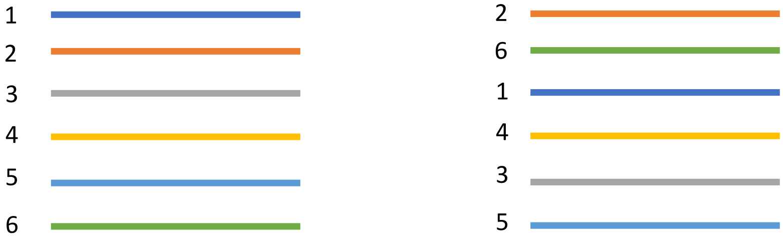
Quality scores

File 2

```
@ERR174324.2 HSQ1009_86:1:1101:1192:2116/2
CAGANAGAGACTCTGTCTCAAAAAAACAAACAAACAAACAAACAAAAAGTCTTA...
+
CCCF#2ADHFHHHJIJJJJJJJJJJJJJJJJIJJJJHIIJJJJJJJJIIIJJ...
                              ⋮
```

Read identifier

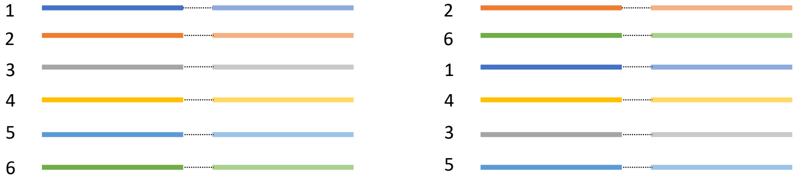
Read order - unpaired



Original order in FASTQ

New order (arbitrary)

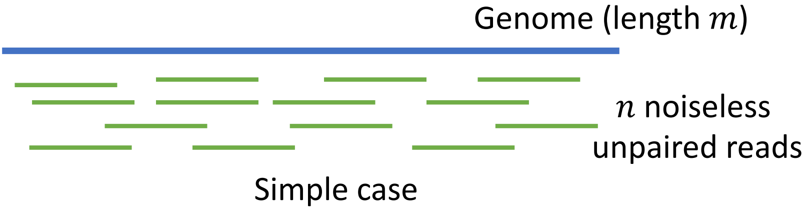
Read order - paired



Original order in FASTQ

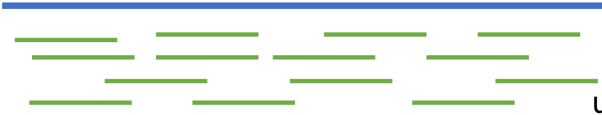
New order (preserves read pairing but pairs ordered arbitrarily)

Entropy of reads (ordered)



$$H(ordered\ \ reads) = H(genome) + H(ordered\ \ reads|genome)$$
$$-H(genome|ordered\ \ reads)$$

Entropy of reads (ordered)



Genome (length m)

n noiseless unpaired reads

Simple case

$$H(\textit{ordered reads}) = H(\textit{genome}) + H(\textit{ordered reads}|\textit{genome})$$
$$-H(\textit{genome}|\textit{ordered reads})$$

For typical datasets, last term is negligible:

$$H(\textit{ordered reads}) \lesssim \underbrace{H(\textit{genome})}_{\text{Store genome}} + \underbrace{n \log_2 m}_{\substack{\text{Store positions of} \\ \text{reads in genome}}}$$

Entropy of reads (unordered)

$$H(unordered\ \ reads) \lesssim \underbrace{H(genome)}_{\text{Store genome}} + \underbrace{\log_2 \binom{m+n-1}{m-1}}_{\substack{\text{Store positions of}\\ \text{reads in genome}}}$$

- $\binom{m+n-1}{m-1}$ = number of ways to distribute n indistinguishable balls into m distinguishable boxes.
- Achievability - sort reads by genome position and entropy code differences of read positions.

Entropy of reads (example)

Example: For human genome and read length 100,

Coverage	Entropy of ordered reads	Entropy of unordered reads
50x	6.7 GB	1.1 GB
100x	12.8 GB	1.4 GB

Table 1: Coverage = average number of reads covering a base in the genome

Entropy of reads (general)

In general, entropy of reads with
$(*)$ exact order preserved &
$(**)$ only pairing preserved (ordering of read pairs discarded):

$$H(reads) \lesssim \underbrace{H(genome)}_{\text{Store genome}} + \underbrace{\left\{ \begin{array}{ll} (*) & \frac{n}{2}\log_2 m \\ (**) & \log_2 \binom{m+\frac{n}{2}-1}{m-1} \end{array} \right\}}_{\substack{\text{Store positions of} \\ \text{read pairs in genome}}}$$

$$+ \underbrace{\frac{n}{2}\left(H(insert\ size) + 1\right)}_{\substack{\text{Store insert size \&} \\ \text{orientation}}} + \underbrace{nH(noise)}_{\text{Store noisy bases}}$$

Upper bound suggests compression scheme

Outline

Introduction
- High-Throughput Sequencing
- Entropy of reads

Methods

Results

Read compression

1. Find “genome”
    - Reorder reads
    - Find consensus
2. Encode reads
3. Compress streams

Reorder reads (simplified)

- Index reads by specific substrings using hash tables

Reorder reads (simplified)

- Index reads by specific substrings using hash tables
- For the current read, try to find an overlapping read within small Hamming distance

Reorder reads (simplified)

- Index reads by specific substrings using hash tables
- For the current read, try to find an overlapping read within small Hamming distance
- **Example (reads indexed by prefix):**

  `ACGATCGTACGTACGATCGTCAG`

  No similar read with highlighted index found → shift

Reorder reads (simplified)

- Index reads by specific substrings using hash tables
- For the current read, try to find an overlapping read within small Hamming distance
- **Example (reads indexed by prefix):**

  `ACGATCGTACGTACGATCGTCAG`

  No similar read with highlighted index found → shift

Reorder reads (simplified)

- Index reads by specific substrings using hash tables
- For the current read, try to find an overlapping read within small Hamming distance
- **Example (reads indexed by prefix):**

```
ACGATCGTACGTACGATCGTCAG
  GATCGTACGTATGATGGTCAGTA
```

Next read found!

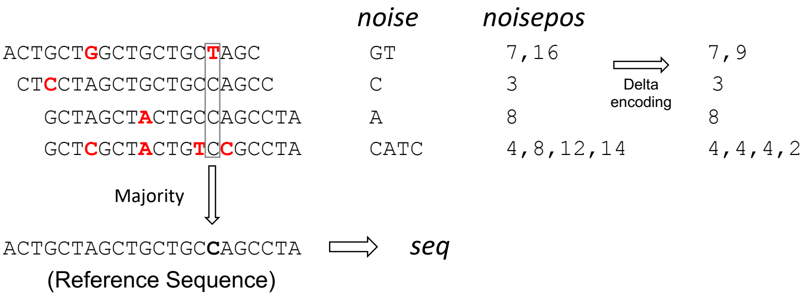
Reorder reads (simplified)

- Index reads by specific substrings using hash tables
- For the current read, try to find an overlapping read within small Hamming distance
- **Example (reads indexed by prefix):**

  ```
  ACGATCGTACGTACGATCGTCAG
    GATCGTACGTATGATGGTCAGTA
  ```

  Next read found!
- Repeat process with the new read

Encode reads



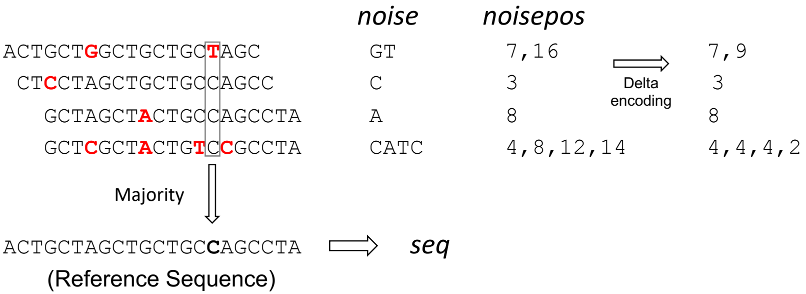
	noise	*noisepos*		
ACTGCT**G**GCTGCTGC**T**AGC	GT	7,16		7,9
CT**C**CTAGCTGCTGCCAGCC	C	3	Delta encoding	3
GCTAGCT**A**CTGCCAGCCTA	A	8		8
GCT**C**GCT**A**CTG**T**C**C**GCCTA	CATC	4,8,12,14		4,4,4,2

Majority

ACTGCTAGCTGCTGC**C**AGCCTA ⟹ *seq*

(Reference Sequence)

Encode reads



	noise	*noisepos*		
ACTGCT**G**GCTGCTGC**T**AGC	GT	7,16	Delta encoding	7,9
CT**C**CTAGCTGCTGCCAGCC	C	3		3
GCTAGCT**A**CTGCCAGCCTA	A	8		8
GCT**C**GCT**A**CTG**T**C**C**GCCTA	CATC	4,8,12,14		4,4,4,2

Majority

ACTGCTAGCTGCTGC**C**AGCCTA ⟹ *seq*

(Reference Sequence)

- Read positions and insert sizes encoded based on the mode (order preserving or not)
- All streams compressed with BSC, a BWT-based compressor

Quality value and read identifier compression

- If read order not preserved, sort quality values and read identifiers according to new read order

Quality value and read identifier compression

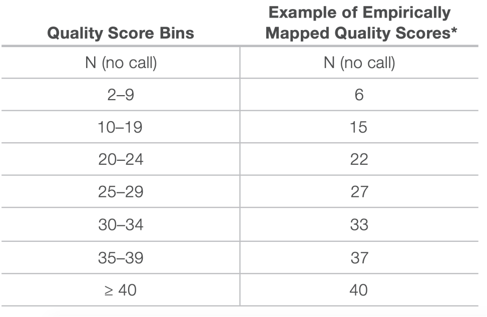
- If read order not preserved, sort quality values and read identifiers according to new read order
- Standard techniques used for compression

Modes

- Lossless (default)

Modes

- Lossless (default)
- Recommended lossy
  - Read order discarded (read pairing still preserved)
  - Quality values quantized using Illumina 8-level binning
  - Read identifiers discarded



Quality Score Bins	Example of Empirically Mapped Quality Scores*
N (no call)	N (no call)
2–9	6
10–19	15
20–24	22
25–29	27
30–34	33
35–39	37
≥ 40	40

Outline

Introduction

- High-Throughput Sequencing
- Entropy of reads

Methods

Results

Results

Organism	Cvg.	FASTQ	Gzip	FaStore	SPRING
P. aeruginosa	50	768 MB	279 MB	145 MB	**115 MB**
Metagenomic	-	19.3 GB	6.9 GB	3.6 GB	**3.2 GB**
H. sapiens	28	227 GB	74 GB	36 GB	**29 GB**
*H. sapiens**	25	196 GB	36 GB	11 GB	**7 GB**
*H. sapiens**	100	788 GB	145 GB	34 GB	**26 GB**

- * sequenced with NovaSeq technology with only 4 quality levels (40 levels for others).

Results

Organism	Cvg.	FASTQ	Gzip	FaStore	SPRING
P. aeruginosa	50	768 MB	279 MB	145 MB	**115 MB**
Metagenomic	-	19.3 GB	6.9 GB	3.6 GB	**3.2 GB**
H. sapiens	28	227 GB	74 GB	36 GB	**29 GB**
*H. sapiens**	25	196 GB	36 GB	11 GB	**7 GB**
*H. sapiens**	100	788 GB	145 GB	34 GB	**26 GB**

- * sequenced with NovaSeq technology with only 4 quality levels (40 levels for others).
- Similar improvements in recommended lossy mode with 20%-50% compression gains over lossless mode.

Results - read compression

Results for read compression of human NovaSeq datasets:

Tool	Mode	Coverage	
		25x	100x
SPRING	order preserving	3.0 GB	10.1 GB
SPRING	pairing preserving	2.0 GB	5.7 GB
FaStore	pairing preserving	6.1 GB	13.7 GB

Conclusion

- SPRING: FASTQ compressor
  - Compression improvements of 1.2x-1.8x on human data
  - Practical computational requirements
  - Several other features: random access, long read compression ...
  - Github: `https://github.com/shubhamchandak94/SPRING/`

Conclusion

- SPRING: FASTQ compressor
  - Compression improvements of 1.2x-1.8x on human data
  - Practical computational requirements
  - Several other features: random access, long read compression ...
  - Github: `https://github.com/shubhamchandak94/SPRING/`
- Future work: integrate with MPEG-G standard for genomic information representation (`https://mpeg-g.org/`)

Thank You!

References

- S. Chandak, K. Tatwawadi, I. Ochoa, M. Hernaez and T. Weissman; SPRING: A next-generation compressor for FASTQ data, *Submitted*.
- S. Chandak, K. Tatwawadi, T. Weissman; Compression of genomic sequencing reads via hash-based reordering: algorithm and analysis, *Bioinformatics*, Volume 34, Issue 4, 15 February 2018, Pages 558–567
- Ł. Roguski, I. Ochoa, M. Hernaez, S. Deorowicz; FaStore: a space-saving solution for raw sequencing data, *Bioinformatics*, Volume 34, Issue 16, 15 August 2018, Pages 2748–2756