

# Traversing the $k$ -mer Landscape of NGS Read Datasets for Quality Score Sparsification

Jacopo Notarstefano  
`jacopo.notarstefano [at] gmail.com`

May 19, 2014

# Next-generation sequencing (NGS)

NGS is the name given to several different sequencing technologies that were developed in the past decade.

These methods have in common a much higher throughput and a much lower cost compared to Sanger sequencing, which was used to sequence the Human Genome [LLL<sup>+</sup>12].

Sequencers	HiSeq 2000	SOLiDv4	Sanger 3730xl
Output data/run	600 Gb	120 Gb	1.9~84 Kb
Cost/million bases	\$0.07	\$0.13	\$2400

# The problem with NGS

*"In the past two decades, genomic sequencing capabilities have increased exponentially, outstripping advances in computing power and storage."*

- Moore's law predicts that the number of transistors on integrated circuits doubles every 24 months [Moo65].
- Kryder's law predicts that hard drive density doubles approximately every 13 months [Wal05].
- Sequencing output has doubled every 9 months [Kah11].

# How to deal with storage

	Lossless	Lossy
General purpose	<b>GZIP</b> <b>BZIP2</b> <b>7zip</b>	
Specific for NGS		<b>RQS</b>

# The FASTQ format

# Phred quality score

## Definition (Phred quality score)

Let  $P_e$  be the estimated probability that a base call is incorrect. Then we define its *Phred quality score*  $Q$  as

$$Q = -10 \log_{10} P_e.$$

This quantity is encoded in the Sanger FASTQ format as a single byte, where the character '!' represents the lowest quality while '~' is the highest.

# The RQS algorithm in brief

Yu, Yorukoglu and Berger [YYB14] propose the following:

- 1 Generate a dictionary  $D$  of all  $k$ -mers that appear with high multiplicity in a representative collection of reads.
- 2 Let  $\gamma$  be a read. Its  $k$ -mers close to  $D$  in Hamming distance have nearly all of their quality score discarded.
- 3 Compress the resulting high redundancy sequence with a general purpose lossless compression algorithm.

# The DICT algorithm

---

**Algorithm 1** DICT

---

**Input:**  $C, k, r$

**Output:**  $D$

```
1:  $D \leftarrow \{\}$ 
2:  $A \leftarrow [0, \dots, 0] \in \mathbb{N}^{4^k}$ 
3: for  $x \in C_k$  do
4:    $A[x]++$ 
5: for  $x \in [4^k]$  do
6:   if  $A[x] \geq r$  then
7:      $D.append(x)$ 
8: return  $D$ 
```

---



# The MARKMER algorithm

---

**Algorithm 2** MARKMER

---

**Input:**  $x, D$

**Output:**  $M$

```
1: if  $\Delta(x, D) > 1$  then
2:    $M \leftarrow [\text{false}, \dots, \text{false}] \in \{\text{true}, \text{false}\}^k$ 
3: else
4:    $M \leftarrow [\text{true}, \dots, \text{true}] \in \{\text{true}, \text{false}\}^k$ 
5:   for  $y \in D \mid \Delta(x, y) = 1$  do
6:     for  $i \in [k]$  do
7:       if  $x_i \neq y_i$  then
8:          $M_i \leftarrow \text{false}$ 
9: return  $M$ 
```

---

# The MARKREAD algorithm

---

## Algorithm 3 MARKREAD

---

**Input:**  $\gamma, D$

**Output:**  $\mathcal{M}$

- 1: // Let  $x^a$  be the  $k$ -mer in  $\gamma$  starting at  $a$ .
  - 2: // Cover  $\gamma$  by  $k$ -mers  $\{x^{a_1}, \dots, x^{a_n}\}$ .
  - 3: **for**  $i \in [n]$  **do**
  - 4:    $M^i \leftarrow \text{MARKKMER}(x^{a_i}, D)$
  - 5:    $\overline{M}^i \leftarrow [\text{false}, \dots, \text{false}] \in \{\text{true}, \text{false}\}^{\text{length}(\gamma)}$
  - 6:   **for**  $j \in [k]$  **do**
  - 7:      $\overline{M}_{j+a_i-1}^i \leftarrow M_j^i$
  - 8:  $\mathcal{M} \leftarrow \overline{M}^1 \vee \dots \vee \overline{M}^n$
  - 9: **return**  $\mathcal{M}$
-

# The SPARSIFYRQ algorithm

---

**Algorithm 4** SPARSIFYRQ

---

**Input:**  $\gamma, Q, D, Q_{\text{threshold}}$

**Output:**  $Q'$

- 1:  $Q' \leftarrow Q$
  - 2:  $\mathcal{M} \leftarrow \text{MARKREAD}(\gamma, D)$
  - 3: **for**  $i \in [\text{length}(\gamma)]$  **do**
  - 4:     **if**  $(Q_i > Q_{\text{threshold}})$  **or**  $(\mathcal{M}_i = \text{true})$  **then**
  - 5:          $Q'_i \leftarrow Q_{\text{threshold}}$
  - 6: **return**  $Q'$
-

# An example

# Results





Results of compressing HG02215, chromosome 21 with  $r = 50$ ,  $k = 32$  and  $Q_{\text{threshold}} = 40$ :

Method	Size	Bits/Q	F1
Uncompressed	273 MiB	8.0000	1
GZIP	143 MiB	4.1923	1
BZIP2	133 MiB	3.8791	1
<b>7zip (PPMd)</b>	<b>124 MiB</b>	<b>3.6269</b>	<b>1</b>
RQS + GZIP	14 MiB	0.3825	0.9914
<b>RQS + BZIP2</b>	<b>8.7 MiB</b>	<b>0.2540</b>	<b>0.9914</b>
RQS + 7zip (PPMd)	11 MiB	0.2953	0.9914

# The implementation of DICT

# An alternative implementation

# Bibliography I

-  Scott D. Kahn, *On the future of genomic data*, Science **331** (2011), 728–729.
-  Lin Lui, Yinhu Li, Siliang Li, Ni Hu, Yimin He, Ray Pong, Danni Lin, Lihua Lu, and Maggie Law, *Comparison of next-generation sequencing systems*, Journal of Biomedicine and Biotechnology **2012** (2012).
-  Gordon E. Moore, *Cramming more components onto integrated circuits*, Electronics (1965), 114–117.
-  Chip Walter, *Kryder's law*, Scientific American (2005).



# Bibliography II



Y. William Yu, Deniz Yorukoglu, and Bonnie Berger, *Traversing the  $k$ -mer landscape of ngs read datasets for quality score sparsification*, Research in Computational Molecular Biology, Lecture Notes in Computer Science, vol. 8394, 2014, pp. 385–399.