# 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+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 | **GZIP**<br>**BZIP2**<br>**7zip** | |
| Specific for NGS | | **RQS** |

# 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:** $D$

```
D ← {}
A ← [0, ..., 0] ∈ ℕ^{4^k}
for x ∈ C_k do
    A[x]++
for x ∈ [4^k] do
    if A[x] ≥ r then
        D.append(x)
return D
```

# The MarkKmer algorithm

**Algorithm 2** MarkKmer

**Input:** $x, D$

**Output:** $M$

```
if Δ(x, D) > 1 then
    M ← [false, ..., false] ∈ {true, false}^k
else
    M ← [true, ..., true] ∈ {true, false}^k
    for y ∈ D | Δ(x, y) = 1 do
        for i ∈ [k] do
            if x_i ≠ y_i then
                M_i ← false
return M
```

# The MarkRead algorithm

**Algorithm 3** MarkRead

**Input:** $\gamma, \mathsf{D}$

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

```
// Let x^a be the k-mer in γ starting at a.
// Cover γ by k-mers {x^{a_1}, ..., x^{a_n}}.
for i ∈ [n] do
    M^i ← MarkKmer(x^{a_i}, D)
    M̄^i ← [false, ..., false] ∈ {true, false}^{length(γ)}
    for j ∈ [k] do
        M̄^i_{j+a_i−1} ← M^i_j
M ← M̄^1 ∨ ··· ∨ M̄^n
return M
```

# The SparsifyRQ algorithm

**Algorithm 4** SparsifyRQ

**Input:** $\gamma, \mathsf{Q}, \mathsf{D}, \mathsf{Q}_{\mathsf{threshold}}$

**Output:** $\mathsf{Q}'$

1: $\mathsf{Q}' \leftarrow \mathsf{Q}$
2: $\mathcal{M} \leftarrow \textsc{MarkRead}(\gamma, \mathsf{D})$
3: **for** $\mathsf{i} \in [\mathsf{length}(\gamma)]$ **do**
4: **if** $(\mathsf{Q}_\mathsf{i} > \mathsf{Q}_{\mathsf{threshold}})$ **or** $(\mathcal{M}_\mathsf{i} = \mathsf{true})$ **then**
5: $\mathsf{Q}'_\mathsf{i} \leftarrow \mathsf{Q}_{\mathsf{threshold}}$
6: **return** $\mathsf{Q}'$

# An example

# Results

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

| Method | Size | Bits/Q | F1 |
|---|---|---|---|
| Uncompressed | 273 MiB | 8.0000 | 1 |
| GZIP | 143 MiB | 4.1923 | 1 |
| BZIP2 | 133 MiB | 3.8791 | 1 |
| **7zip (PPMd)** | **124 MiB** | **3.6269** | **1** |
| RQS + GZIP | 14 MiB | 0.3825 | 0.9914 |
| **RQS + BZIP2** | **8.7 MiB** | **0.2540** | **0.9914** |
| 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.