HammerIT: homopolymer-space Hamming clustering for IonTorrent read error correction

Anton Korobeynikov, Artem Tarasov
St. Petersburg State University, Russia

Introduction

FIXME: a few words about Ion Torrent sequencing technology

IonTorrent error profile

Corrected flow signal intensities are available in BAM files produced by versions of Ion Torrent Suite prior to 3.4. Called homopolymer length is obtained as corrected flow signal intensity rounded to the nearest integer.
We have studied flow signal intensity distributions around insertion/deletion sites. File B7-295.bam, downloaded from Ion Community website, contained 4.6M insertions, 5.0M deletions, and 1.5M mismatches.
Overwhelming majority of errors turned out to be insertions/deletions of length 1, occuring when flow signal intensity is approximately halfway between two adjacent integers.

Figure 1: Flow signal intensities at insertion sites

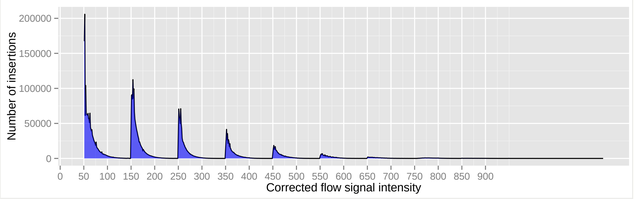
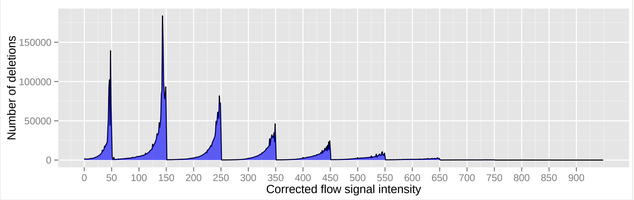


Figure 2: Flow signal intensities at deletion sites



More detailed analysis of errors in IonTorrent data can be found in the article "Shining a Light on Dark Sequencing: Characterising Errors in Ion Torrent PGM Data" *(PLoS Comput Biol 9(4))*

General approach

The core idea is the same as in Hammer (*Bioinformatics* (2011), 27 (13); i137-i141)

1. Count and cluster homopolymer-space k-mers (we used $k = 16$)
2. Fill consensus array for each read, using centers of clusters containing k-mers from the read
3. Compute consensus for each homopolymer run in the read

There are several substantial differences, however. Firstly, the basic unit of operation is homopolymer run, i.e. a pair $(\text{nucleotide}, \text{length})$. Second, using Hamming distance in clustering is out of question due to abundance of indels in Ion Torrent data. Therefore, those tricks for fast clustering, that are based on Hamming distance properties, are, in the best case, only partially applicable for homopolymer-space k-mers.

Results

We evaluated HammerIT on 6 publicly available datasets, using the same pipeline as the authors of the recently published article "Updating benchtop sequencing performance comparison" *(Nature Biotechnology, v. 31, no. 4)*. In that article, error rate in four Ion Torrent datasets has been assessed. We used the same data plus two extra datasets from 314v2 chip, which recently became available on Ion Community Portal.
Indel/mismatch error rates were calculated for uniquely mapped reads before and after correction. For each dataset, correction was done in two ways. In the first setup, trimming was done for read ends that couldn't be corrected due to lack of good k-mers, while in the second one such read ends were preserved in the output. Relative change in read coverage after correction stayed within 0.4% in all cases.

Figure 3: Error rates before and after correction

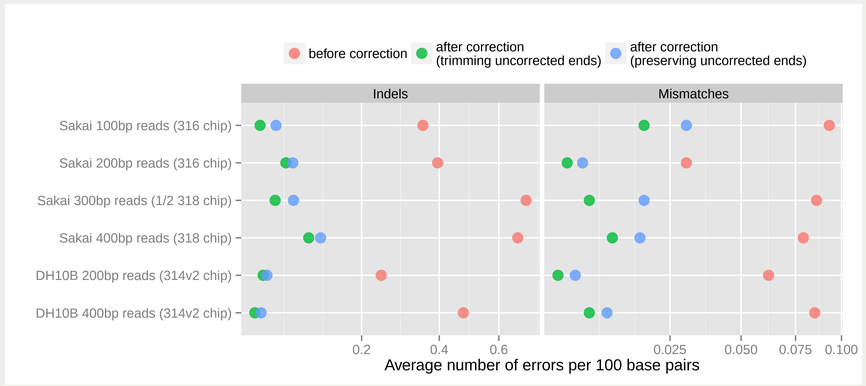
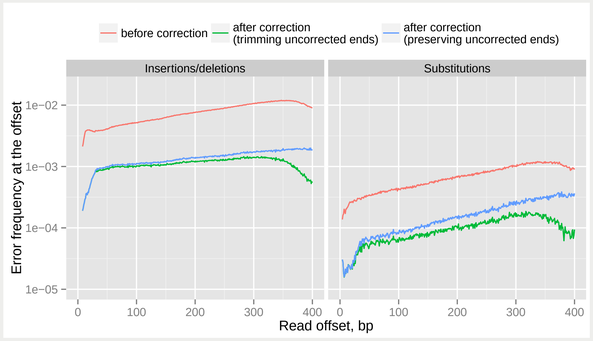


Figure 4: Error reduction by read position for Sakai 400bp reads



Pairwise distance calculation

We use 5-base lookahead to compute distance between homopolymer-space k-mers. Helper table stores precomputed values for the functions

$$H_k : \{\mathrm{A,C,G,T}\}^k \times \{\mathrm{A,C,G,T}\}^k \to \{\mathrm{Insertion, Deletion, Mismatch}\}\,, \quad k = 1, 2, 3, 4, 5\,.$$

The chosen value of 5 is a trade-off between accuracy and speed.
Mapped reads from B7-295.bam dataset were used for training.

G A G T A C A C T G T C G T C G
G T G T A C A T G T C G A T G C

$$\begin{aligned} H_5(\mathrm{AGTAC, TGTAC}) &= \mathrm{Mismatch} \\ H_5(\mathrm{CTGTC, TGTCG}) &= \mathrm{Deletion} \\ H_3(\mathrm{TCG, ATG}) &= \mathrm{Mismatch} \\ H_2(\mathrm{CG, TG}) &= \mathrm{Mismatch} \end{aligned}$$

Algorithm

Input: x, y — nucleotide sequences
Output: $dist$ — approximate Levenshtein distance between x and y

$pos.x \leftarrow 0$; $pos.y \leftarrow 0$; $dist \leftarrow 0$;
while $pos.x < \mathrm{length}(x)$ and $pos.y < \mathrm{length}(y)$ **do**
  if $x[pos.x] = y[pos.y]$ **then**
    $pos.x \leftarrow pos.x + 1$; $pos.y \leftarrow pos.y + 1$;
  else
    $k \leftarrow \min(5, \mathrm{length}(x) - pos.x, \mathrm{length}(y) - pos.y)$;
    adjust $pos.x$ and $pos.y$ according to
      $H_k(x[pos.x \,..\, pos.x + k - 1], y[pos.y \,..\, pos.y + k - 1])$;
    $dist \leftarrow dist + 1$;
  end if
end while

K-mer clustering

FIXME: here goes description of clustering algorithm, maybe also a plot showing cluster size distribution.

Typical cluster

16-mer	n	qual.
GTGTACATGTCGATGC	113	1.00
GTGTACATGTCGATGT	23	0.90
GTGTACATGCGATGCT	7	0.80
GTGTACATGTCGTGCT	6	0.33
GTGTACATGTCATGCT	6	0.31
CTTGTACATGTCGATGC	6	0.26
GTGTACATGTCTGCTC	4	0.28
GTGTACATGTCGATTGC	4	0.12
GTGTACATGTGGATGCT	4	0.09
GTGTACATGTCCGATGC	3	0.41
GTGTACATGTCGAGCT	3	0.14
GTGTACATGTCGAATGC	3	0.12
GTGTACATGTCGTGTC	3	0.10
GTGTACATGCGATGCA	2	0.28
GAGTACACTGTCGCTG	2	0.04
GTGTACATGTCGATGA	2	0.03
GTGTATACATGTCGTG	1	0.37
GTGTACATGCGATGTG	1	0.32
GTGTACACTGTCGCTA	1	0.25
GTGTACACTGTCCGCTA	1	0.13
GTGTACATGTCAATGTC	1	0.10
GTGTACATGTCATGCCT	1	0.09
CAGTACACTGTCGCTA	1	0.08
GTGTACATGCGATGCCT	1	0.08
GTGTACATGTCGAGCCG	1	0.08
GTGTACATGTCGAGGTG	1	0.08
GAGTACACACCACTCTG	1	0.06
GTGTACATGTCATGCGG	1	0.05
GTGTACATGTCGTGCA	1	0.04
GTGTACATGTCGGATGC	1	0.04
GTGTACATGCGCTGTC	1	0.04
GTGTACATGTCGGATGCC	1	0.03
GTGTACATGTCATGTA	1	0.03
CAGTACACTGTAGCTA	1	0.03
GTGTACAATGTCGTGCT	1	0.03
ATGTACGCTGTCGCTA	1	0.03
GAGTATACATGTCGTG	1	0.02
GTGTACATGTCTGCTA	1	0.02
GTGTACATGTCGAATGT	1	0.02
GTTGTTACATGTCGATGC	1	0.02
GTGTATACATGTCATG	1	0.01
CAGTACACTGTCGCTG	1	0.01
GTGTACATGTCGACTA	1	0.01
GTGTACATGTCGATTCT	1	0.01
GTGTACATGTCGAGTA	1	0.01
GTGTTACATGTCGATGC	1	0.01
ATGTACGCTGTAGCTG	1	0.01
ATGTACGCTGTAGCTGG	1	0.01
GTGTTACATGTGGATGCT	1	0.01
GTGTACATGTCGACAGG	1	0.00

Error correction algorithm

for each *read* **do**
  (Producing contiguous corrected parts of read)
  for each homopolymer-space *kmer* from the read **do**
    center ← center of the cluster to which *kmer* belongs;
    if *center* quality is more than user-specified threshold **and**
      center bases agree with the previous center **then**
        include *center* into consensus score calculation;
    else
      yield new corrected part from current consensus;
      estimate its position on the read;
      reset consensus table and start new part;
    end if
  end for
  (Combining corrected parts)
  for each corrected *part* **do**
    trim homopolymer runs with low consensus score from both ends;
    align last 8 homopolymer runs against the read sequence;
    align first 8 runs of the next part;
    if there is a gap on the read between the two parts **then**
      copy read homopolymer runs as is;
    else
      select homopolymer runs with higher consensus score
        from the intersection of the two parts;
    end if
  end for
  (Optionally, attaching uncorrected end)
  align last 8 runs of the last chunk against the read sequence;
  append read homopolymer runs after the last aligned run.
end for