RESEARCH

BisPin and BFAST-Gap: Mapping Bisulfite-Treated Reads

Jacob Porter* and Liqing Zhang

*Correspondence: jsporter@vt.edu
Department of Computer Science, Virginia Tech, 24061 Blacksburg, VA, USA
Full list of author information is available at the end of the article

Abstract

Background: BisPin is a new multiprocess bisulfite-treated short DNA read mapper written in Python 2.7. It performs alignments using BFAST, leveraging its multithreading functionality and thorough hash-based indexing strategy. BisPin is feature rich and supports directional, nondirectional, PBAT, and hairpin construction strategies. BisPin approaches read mapping by converting the Cs to Ts and the Gs to As in both the reads and the reference genome. BisPin uses fast rescoring to disambiguate ambiguously aligned reads for a superior amount of uniquely mapped reads compared to other mappers. The performance of BisPin was evaluated on both real and simulated data in comparison to other read mappers.

BFAST-Gap is a modified version of BFAST meant for Ion Torrent reads. It uses a parameterized logistic function to determine the weights of the gap open and extension penalties based on the homopolymer run length of the DNA read. This is because the Ion Torrent sequencing technology can overcall and undercall homopolymer runs. BisPin works with both BFAST-Gap and BFAST. BFAST-Gap is compatible with indexes built with BFAST. There are few mappers that specifically address Ion Torrent data. BFAST-Gap works with Illumina reads as well.

Results: BisPin with BFAST consistently had a higher amount of uniquely mapped reads compared to other mappers on real data using a variety of construction strategies. Using a hairpin validation strategy, BisPin was superior using the maximum score, and it mapped 73% of reads correctly.

BisPin with BFAST-Gap on Ion Torrent reads with a logistic gap open penalty function improved mapping accuracy with real and simulated data. On simulated bisulfite Ion Torrent data, the area under the curve was improved by approximately seven, and on one real data set, the uniquely mapped percent was improved by seven percent. BFAST-Gap performed better than TMAP on simulated regular Ion Torrent reads, and TMAP is designed for Ion Torrent reads. Other read mappers had worse performance.

Conclusions: BisPin and BFAST-Gap have consistently good accuracy with a variety of data. BisPin is feature-rich. This makes BisPin and BFAST-Gap useful additions to read mapping software.

Keywords: bisulfite; short DNA reads; Ion Torrent; logistic; hairpin sequencing

Background

Short DNA reads are treated with bisulfite to study epigenetic methylation, and these reads are mapped with software to a reference genome for epigenetic methylation discovery. Epigenetic methylation is a phenomenon where cytosine nucleic acids in DNA have a covalently bonded methyl group (CH_3) attached to the 5 car-

bon of the cytosine ring. Epigenetic methylation is inheritable, and it plays a role in disease and development [1, 2].

Bisulfite treatment converts unmethylated cytosines into thymines while leaving methylcytosine unchanged. The DNA sequencing process may sequence the reverse complement so that adenine corresponds to unmethylated cytosine and guanine corresponds to methylcytosine. A read mapper compares these bisulfite converted reads to a normal reference genome to discover methylcytosine since methylcytosine should align with a cytosine in the reference genome, or in the case of the reverse complement, with a guanine. Read mapping of this sort is challenging since the bisulfite treatment introduces differences in addition to natural variation and sequencing error between the reference genome and the reads. Bisulfite read mapping has been characterized by low mapping efficiency [3] that has been shown to be correlated to reduced sequence complexity [4]. Read mapping can be time consuming with millions of short reads.

Software that maps bisulfite treated DNA reads to a reference genome includes Bismark [5], BWA-Meth [6], Walt [7], and others. All of these programs use three phases: (1) reference genome index creation, (2) seeding, and (3) extension with alignment. A string index is usually created with either the Burrows-Wheeler transform and FM-Index or a hash table [8]. Both Bismark, which calls Bowtie2 [9] for read mapping, and BWA-Meth, which calls BWA [8] for read mapping, use the former approach. Walt uses the latter hash table approach. Seeding is accomplished by taking short subsequences of the DNA read and matching them with the string index of the reference genome. This procedure gives candidate hits, which are possible locations where the read can be mapped. Finally, an extension step is performed where the entire read is aligned to the reference genome at the candidate locations. This is normally accomplished with the Smith-Waterman local alignment algorithm. The alignment algorithm scores each location so that the best location can be reported. Differences between the reference genome and the DNA read revealed by the alignment can be either the result of genetic variation, sequencing error, or unmethylated Cs converting to Ts (or Gs to As on the complementary strand), and therefore, methylation at single nucleotide resolution can be determined.

Other bisulfite read mappers include BatMeth, BRAT-nova, BSMAP, BS-Seeker2, and BSmooth. BatMeth filters out reads with low entropy, but does not produce output in the standard SAM file format [10]. BRAT-nova is reported to be fast, but could not be made to work at the time of this writing [11]. BSMAP is outdated and cannot map reads longer than 144 base pairs, making it unsuitable for many modern sequencing projects [12]. BSMAP adds all possible combinations of C to T conversions to its index. BS-Seeker2 is similar to Bismark in that it uses Bowtie2 as a subprocess [13] for read mapping. BSmooth can call methylation marks [14]. These read mappers do nothing special to resolve ambiguously mapped reads, which are multiple high scoring alignments for a read. Their support for multiple protocols is limited to conventional sequencing protocols. They do nothing special for Ion Torrent reads.

To address these problems, this study presents BisPin, a bisulfite read mapper that deploys BFAST for read mapping. BFAST was chosen because its indexing strategy is very thorough and supports mapping with multiple indexes and spaced seeds. It is

feature-rich with informative output and multithreading. More importantly, it has shown superior performance over tools such as Bowtie2 and BWA in the presence of indels over 10bp long [15].

This advantage becomes especially important when considering some sequencing platforms such as Ion Torrent where sequencing errors tend to introduce indels to reads [16]. Ion Torrent technology has two advantages over Illumina technology: longer reads and less expensive machines. Ion Torrent technology can produce 400bp reads [17] while Illumina MiSeq can produce 300bp reads [18]. Ion Torrent machines have a price around $80k, and Illumina machines have a price around $120k [19].

BisPin with BFAST has good performance on Ion Torrent data. Similar to Bismark and BWA-Meth, BisPin calls BFAST for read mapping, but BisPin adds special processing to accommodate short reads generated from the whole genome hairpin bisulfite construction strategy [20, 2]. A forked version of BFAST, called BFAST-Gap, was developed with special processing for Ion Torrent reads. BFAST-Gap is backwards compatible with BFAST. BFAST-Gap has an implementation of the Smith-Waterman alignment algorithm that reduces the gap open and gap extension penalties depending on the length of the homopolymer run. This should improve performance on Ion Torrent reads where gaps occur more frequently in longer homopolymer runs. Tabsat is another read mapper that is specifically designed to map bisulfite-treated Ion torrent reads [21]. Tabsat modifies the Bismark Perl code to call the Ion Torrent mapping program TMAP [21, 22].

BisPin enables BFAST multithreading for alignment, multiprocessing for post processing, and read partitioning for deployment on compute clusters.

Methods

BisPin Features

BisPin is a Python program that calls BFAST, a C++ program, to perform the three phases of read alignment and mapping. In this aspect, BisPin is similar to Bismark, which uses Bowtie2 to perform read alignment and mapping. BisPin supports directional, nondirectional, post bisulfite adapter tagging (PBAT), and hairpin read construction strategies. It maps both single end and paired end reads. For all reads, BisPin reports a methylation calling string in the style of Bismark. The output is given as a SAM file [23]. BisPin has an alignment result summary report, which includes mapping efficiency, methylation calling statistics, timing profile information, and command line arguments. Table 1 summarizes BisPin's features, as compared to several programs including Bismark, Walt, and BWA-Meth. The BisPin software is freely available at `https://github.com/JacobPorter/`.

In bisulfite PCR amplification, four sequences are possible: the original forward strand, the original reverse strand, the reverse complement to the forward strand, and the reverse complement to the reverse strand. The directional construction strategy sequences the original forward and reverse strands [24]. Post-Bisulfite Adapter Tagging (PBAT) sequences the reverse complements to the original forward and reverse strands [25], and the nondirectional strategy sequences all strands [26]. Read mappers such as Bismark and Walt support mapping data with these construction strategies, and data generated with each of these methods is common. The hairpin construction strategy uses the Illumina paired end layout to sequence

Feature	BisPin	Bismark	Walt	BWA-Meth
Paired end and single end support	✔	✔	✔	✔
Directional support	✔	✔	✔	✔
PBAT support	✔	✔	✔	✗
Nondirectional	✔	✔	✗	✗
Hairpin support	✔	✗	✗	✗
Methylation calling string	✔	✔	✗	✗
Ambiguously mapped resolution	✔	✗	✗	✗
Parallelization	✔	✔	✔	✔
Customizable alignment scoring	✔	✔	✗	✗
Read file partitioning	✔	✗	✗	✗

Table 1: Comparison of BisPin features

the forward strand as well as the matching reverse strand with a hairpin adapter that connects the Watson and Crick strands [20]. The technique is especially powerful for bisulfite sequencing as it allows for the recovery of the original untreated strand before bisulfite treatment. This is called "hairpin recovery." This strategy was shown in previous research to improve mapping efficiency by 10% [4]. No known read mappers support special processing for the hairpin construction strategy except BisPin. BisPin has special processing for methylation calling with hairpin data as the hairpin recovery allows the distinction between single nucleotide variants and an unmethylated cytosine.

BisPin's postprocessing of the BFAST raw SAM files can be done in parallel with multiprocessing, which involves six processes. Postprocessing involves choosing the highest scoring alignment for each read, rescoring ambiguously mapped reads, calling the methylation string, updating the SAM record fields, and printing a summary report.

BisPin produces a methylation calling string by matching the aligned read base to the reference base. This is done similarly to Bismark [5]. For hairpin recovered data, the read sequence is compared to the recovered untreated read if available instead of the reference.

Rescoring Ambiguously Mapped Reads

Compared to regular short read mapping, bisulfite short read mapping produces much more ambiguously mapped reads, which are reads mapped to multiple locations [3]. Thus, it is important to have a strategy to distinguish the ambiguously mapped reads.

BisPin employs a simple and fast rescoring technique to disambiguate reads mapped to multiple locations. The rescoring technique scores different mismatches between the read and the reference differently. The matrix that represents this function can be completely specified by the user. More complicated and slower methods

for doing this exist, such as Bayesian inference [27], and few read mappers integrate any such disambiguation except for random assignment.

The rescoring technique is a linear function in the mismatches, matches, and gaps. BisPin examines the alignment of a read to the reference genome as determined by BFAST. If the nucleotide bases match, a positive value is assigned based on the identity of the matching nucleotide base. If the bases do not match, a negative score is assigned based on the identity of the bases. Gaps are scored as they usually are. There is a score for the length of the gap and for the existence of the gap (a gap open). Different scores can be assigned for insertions and deletions. The matrix used to determine the score is the HOXD matrix in [28]. The gap open score is -400, and the gap extenstion score is -30. The gap function and the scoring matrix are the same as in Blastz [29]. The maximum scoring location, if one exists, is used to assign the read to a uniquely mapped location. A read is mapped to a unique location if there is only one location with a maximum score. This increases the uniquely mapped number of reads. The rescoring algorithm is given with Algorithm 1.

Algorithm 1 The BisPin rescoring algorithm.

A the alignment of a read to the reference; R the rescoring matrix
O the gap open penalty; E the gap extension penalty
procedure RESCORE(A, R, O, E)
    $s \leftarrow 0$
    for $a_i \in A$ **do**
        if a_i is a methylation call (CT or GA mismatch) **then**
            $s \leftarrow s$ + the average of all matching scores
        else if a_i is another mismatch **then**
            $a_i^g \leftarrow$ the genome base of a_i
            $a_i^r \leftarrow$ the read base of a_i
            $s \leftarrow s + R[a_i^g][a_i^r]$
        else if a_i is a match **then**
            $a_i^r \leftarrow$ the read base of a_i
            $s \leftarrow s + R[a_i^r][a_i^r]$
        else if a_i is a gap open **then**
            $a_i^d \leftarrow$ indicator for an insertion or deletion
            $s \leftarrow s + O[a_i^d] + E[a_i^d]$
        else if a_i is a gap extension **then**
            $a_i^d \leftarrow$ indicator for an insertion or deletion
            $s \leftarrow s + E[a_i^d]$
    return s

The rescoring matrix and the default BisPin alignment function are appropriate for bisuflite data and mammalian genomes. The rescoring matrix represents ratios of aligned nucleotide frequencies from non-coding mouse and human genomic regions [28]. The average matching score over all bases is used for C to T matches (alternatively, G to A matches) for the bisulfite conversion case. Bases other than C and T (G and A) should be unaffected by bisulfite treatment, so the matching and

mismatching scores for those regions are the same as for regular untreated reads; thus, the functions chosen for scoring are appropriate for bisulfite data.

If the default rescoring matrix and the alignment function are inappropriate for the data, then the user can set another function. Furthermore, the exact score for the C to T (or G to A) match for bisulfite converted reads can be set with an alternative rescoring function that uses two matrices in the format given in [30]; however, the matrices in that paper were based on a uniform distribution of bases, and this does little to disambiguate ambiguous reads. In real genomes, DNA bases are not uniformly distributed [31]. On one extreme there are some Actinobacteria with GC content as high as 70% [32], and on the other extreme there is *Plasmodium falciparum* with 80% AT content [33]. The alignment function can affect the quality of results [34], so a biologically motivated scoring function, as BisPin uses, makes sense. Bismark appears to use an arbitrarily chosen alignment scoring function. A representation of the rescoring function is given in Figure 1.

Hairpin Recovery

BisPin includes special processing for the hairpin construction strategy. This data uses a hairpin connector to connect the Watson and Crick strands, and then Illumina paired end sequencing is performed to sequence the two strands. The strands can be matched together to allow a recovery of the original strand untreated by bisulfite. This strategy is called "hairpin recovery." A thorough description can be found in the paper [4]. For this data, BisPin recovers the original strand and uses BFAST to align it. However, not all strands will perfectly match due to sequencing error, so BisPin aligns these as regular bisulfite treated reads either in a paired end layout or in a single end layout.

BFAST-Gap Implementation

BFAST-Gap implements an adaptive weight to the gap open and gap extension penalties of the Smith-Waterman algorithm based on the length of the homopolymer run. There are four function options for determining the weight: constant, logistic, exponential, and piecewise constant. The exponential model opened gaps too frequently in early tests, and the piecewise constant function was thought to be too simple. For these reasons, these functions were not thoroughly examined, but are provided as is. The logistic function is the most versatile since it has portions that resemble exponential growth, exponential decay, and linear growth. Since read length is effectively bounded, the logistic function can be used to approximate exponential growth, exponential decay, and linear growth with appropriate arguments. The exponential portions of the function can approximate low order polynomials.

In order to score alignments between the read r_1 and the reference r_2, the following constants must be defined. Suppose that the initial gap open score is given as g_o, and the initial gap extension score is given as g_e. The score for two matching characters is m_s, and the score for two mismatched characters is m_n. The length of the homopolymer run at position i in read r_1 is given as $D[i]$. The constants g_o, g_e, and m_n are negative integers, and the constant m_s is a positive integer.

The logistic gap open function G_o is a function of the homopolymer run length $D[i]$ with slope s_o and center c_o. It is given by the following.

$$\text{Rescore} = \begin{array}{c} \\ A \\ C \\ G \\ T \\ N \end{array} \begin{array}{c} \begin{array}{ccccc} A & C & G & T & N \end{array} \\ \begin{bmatrix} 91 & -114 & -31 & -123 & -100 \\ -114 & 100 & -125 & -31 & -100 \\ -31 & -125 & 100 & -114 & -100 \\ -123 & -31 & -114 & 91 & -100 \\ -100 & -100 & -100 & -100 & 100 \end{bmatrix} \end{array}$$

gap open [I, D] = $[-400, -400]$
gap extension [I, D] = $[-30, -30]$

Figure 1: This describes the default rescoring function used to disambiguate ambiguously mapped reads. It is taken from Blastz [29, 28]. The rows of the matrix refer to the reference genome, and the columns refer to the read string. For gap opening and extension, different values can be used for insertions (I) and deletions (D). BisPin uses the average matching and mismatching scores from this matrix as default values for doing the initial alignments with BFAST.

$$G_o(D[i], g_o, m_n, s_o, c_o) = \frac{g_o + m_n}{1 + e^{s_o * (D[i] - c_o)}} + m_n$$

The constant g_o, the constant gap open penalty, gives the maximum value that the function can give, and the constant m_n, the mismatch penalty, gives the minimum value.

The logistic gap extension function G_e, a function of the homopolymer run length $D[i]$, has slope s_e, center c_e, and minimum value $z = -1.0$, and it is given by the following.

$$G_e(D[i], g_e, z, s_e, c_e) = \frac{g_e + z}{1 + e^{s_e * (D[i] - c_e)}} + z$$

The maximum value that the function can give is g_e, the constant gap extension penalty.

These functions are pre-computed once for every execution of BFAST-Gap by storing the function values in a lookup table since the run length parameter, $D[i]$, is discrete and effectively bounded by the maximum read length.

The run-length array is calculated once for every read by Algorithm 2. This algorithm scans through a read r and initially records in array D, at line 9, the length of a homopolymer run length up to position i. When a new homopolymer run is detected by detecting a change in the DNA base stored in b, the algorithm updates all the values in D associated with the homopolymer run with the total length of the homopolymer run with the **for** loop at line 11. The algorithm returns the array D when the outer **for** loop terminates. This algorithm has time in $\Theta(|r|)$ since the outer **for** loop at line 6 iterates over every base in r and the inner loop at line 11 iterates over every base in every homopolymer run, which comprises every base of the read.

Algorithm 2 An algorithm to calculate homopolymer run lengths for each position of a read.

```
1:  r ← a string over Σ_N representing a read
2:  procedure HomoRuns(r)
3:      D ← an array of length |r| with d[0] = 1. Used to store the lengths of the runs.
4:      s ← 0, a variable to keep track of the start of a run
5:      b ← r[0], a variable to keep track of the DNA base of a run
6:      for i ← 1, |r| do
7:          t ← r[i], a base for testing if a run continues or stops
8:          if b = t then                                   ▷ The run continues.
9:              D[i] ← D[i − 1] + 1
10:         else                                            ▷ The run has ended.
11:             for j ← s, i do          ▷ Update all values in the run to the same thing
12:                 D[j] ← D[i − 1]
13:             b ← t                ▷ Update the values to indicate the start of a new run
14:             D[i] ← 1
15:             s ← i
16:     return D after updating the final values by repeating the inner loop of the
    preceding.
```

Once the run-length array D is computed, the alignment score and backtrace and score matrices are computed with Algorithm 3, the Run Length Smith-Waterman al-

gorithm. This algorithm is a similar dynamic programming algorithm to the canonical Smith-Waterman algorithm except that the gap penalties are determined by the gap functions G_o and G_e. The Run Length Smith-Waterman algorithm allows for negative alignment scores and an affine gap penalty. This algorithm maintains four matrices. Matrix H is for deletions, and matrix V is for insertions. Matrix S keeps track of the alignment score, and B stores the backtrace as in the canonical Smith-Waterman algorithm. The backtrace is used to compute the alignment of the read to the reference genome.

The first column and row of each matrix is initialized so that an insertion may start an alignment but not a deletion by initializing the H matrix row and column to negative infinity and initializing the column for the score matrix S and the insertion matrix H to the value of the gap penalty for the appropriate length of the gap. The loops at lines 21 and 22 update the interior of these matrices with the given recurrence. Finally, the position with maximum value from the bottom row is returned along with the matrices S and B. From these elements, the alignment of the entire read locally aligned to the reference genome can be computed as with the canonical Smith-Waterman algorithm using the B matrix. The algorithm has time complexity in $\Theta(|r_1||r_2|)$ since the nested loops at lines 21 and 22 iterate over both the read r_1 and the reference r_2. The algorithm has space complexity in $\Theta(|r_1||r_2|)$ because there are a constant number of arrays of size $|r_1|$ by $|r_2|$.

Data Analysis Methods

To assess BisPin (with BFAST-v0.7.0a), BisPin was compared to Bismark (v0.16.3 with bowtie2-2.2.9), BWA-Meth (downloaded Jan 2017 with BWA-0.7.12-r1039), and Walt (1.0) using the Genome Research Consortium's primary assemblies of the mouse genome (GRCm38.p5) and the Arabidopsis Information Resource (TAIR) version 10 of the *A. thaliana* genome. All tests were run on Virginia Tech's bioinformatics machine, mnemosyne2, running Red Hat Linux 4.8.5-4 with 132 GB of RAM and 16 cores comprising the Intel(R) Xeon(R) CPU E5620 @ 2.40GHz. Timing results were calculated with the Linux `time` command except for BisPin's postprocessing, which used Python's `datetime` module. For Ion Torrent reads only, Tabsat version 1.0.1 with TMAP version 3.4.1 was used. BFAST-Gap on regular Ion Torrent reads was compared with TMAP, Soap2 (2.21), BWA, and Bowtie2 on default settings. TMAP used the map4 algorithm.

BFAST can be run with multiple indexes that are divided into two categories: primary and secondary indexes. Whenever BFAST or BFAST-Gap were run with multiple indexes, only a single index was used as a primary index to increase run-time. Secondary indexes are used only if a match for a read cannot be found in the primary index. Unless indicated otherwise, the primary index had the mask "1111111111111111111", and a secondary index had the mask "1111111100111111111."

Simulated Data

Two data sets were generated to simulate Illumina reads. Simulations were performed with Sherman v0.1.7 from Babraham Bioinformatics on the mouse genome with realistic settings of error 2, CG context 20, and CHH context 98 [2]. Different

Algorithm 3 The Smith-Waterman algorithm with gap open and gap extension weights determined by the homopolymer run length.

$r_1 \leftarrow$ a string over $\Sigma_{\mathcal{N}}$ representing a read
$r_2 \leftarrow$ a stringover $\Sigma_{\mathcal{N}}$ representing the reference genome
$D \leftarrow$ an array representing run lengths for each position in the read
$A_f \leftarrow \{g_o, g_e, m_s, m_n, s_o, c_o, s_e, c_e, z\}$, parameters to determine the alignment function.
$G_o \leftarrow$ the gap open penalty function
$G_e \leftarrow$ the gap extension penatly function
procedure RLSW(r_1, r_2, D, A_f, G_o, G_e)
    $H \leftarrow$ a $|r_1|$ by $|r_2|$ numeric matrix for deletions
    $V \leftarrow$ a $|r_1|$ by $|r_2|$ numeric matrix for insertions
    $S \leftarrow$ a $|r_1|$ by $|r_2|$ numeric matrix for the alignment score
    $B \leftarrow$ a $|r_1|$ by $|r_2|$ matrix for the backtrace
    for $i \leftarrow 0, |r_2|$ **do**
        $H[0, i] = -\infty$
        $V[0, i] = -\infty$
        $S[0, i] = 0$
    for $j \leftarrow 1, |r_1|$ **do** ▷ Initialize the column so that an insertion is possible
        $H[j, 0] = -\infty$
        $V[j, 0] = G_o(D[j-1], g_o, m_n, s_o, c_o) + (j-1)G_e(D[j-1], g_e, z, s_e, c_e)$
        $S[j, 0] = G_o(D[j-1], g_o, m_n, s_o, c_o) + (j-1)G_e(D[j-1], g_e, z, s_e, c_e)$
    Update the backtrace matrix B
    for $i \leftarrow 1, |r_1|$ **do**
        for $j \leftarrow 1, |r_2|$ **do**

$$H[i,j] = \max \begin{Bmatrix} H[i, j-1] + G_e(D[i-1], g_e, z, s_e, c_e) \\ S[i, j-1] + G_o(D[i-1], g_o, m_n, s_o, c_o) \end{Bmatrix}$$

$$V[i,j] = \max \begin{Bmatrix} V[i-1, j] + G_e(D[i-1], g_e, z, s_e, c_e) \\ S[i-1, j] + G_o(D[i-1], g_o, m_n, s_o, c_o) \end{Bmatrix}$$

$$S[i,j] = \max \begin{Bmatrix} S[i-1, j-1] = + \begin{Bmatrix} m_s \text{ if } r_1[i] = r_2[j] \\ m_n \text{ if } r_1[i] \neq r_2[j] \end{Bmatrix} \\ H[i,j] \\ V[i,j] \end{Bmatrix}$$

        Update the backtrace matrix B
    $l \leftarrow$ the location of the position in S along the bottom row that has maximum value.
    return S, l, B

read lengths were used to simulate variety in read mapping tasks. The first data set consisted of ten sets of one million paired end 75bp reads. The second data set consisted of ten sets of one million single end 150bp reads.

All simulated data was aligned in the same fashion as the real data. A Python script was developed to check the accuracy of read mappers. Only uniquely mapped reads were checked, and they were considered accurately mapped if the starting

location was within three bases of the real location. Average precision and recall were computed. Precision is the proportion of the uniquely mapped reads that are correctly mapped of the total uniquely mapped reads, and recall is the proportion of the uniquely mapped reads that are correctly mapped of the total number of reads.

A test to compare the rescoring function against a random choice was performed to see if the rescored read was more likely to align correctly than random chance. This test used 100,000 150bp single end reads. The set of rescored ambiguously mapped reads was extracted and for each read, a random alignment location was chosen. The percentage of these that were correctly mapped was compared to the percentage of the rescored reads that were correctly mapped. This process was repeated 32 times to calculate a p-value.

The preceding Sherman simulated data was used to test BisPin on simulated bisulfite-treated Illumina reads. BisPin with BFAST and BFAST-Gap were tested on simulated Ion Torrent reads generated by DWGSIM since it simulates the undercalling and overcalling of homopolymer runs found in Ion Torrent reads [35]. For DWGSIM, the bisulfite treatment was simulated on the reference genome with the CpG rate as 0.215, the CH rate 0.995, and the over conversion and under conversion rates of 0.0025 [3, 2]. This was done with custom Python scripts provided by another lab member. Custom Python scripts were used to simulate single end data from DWGSIM. DWGSIM was used with the following realistic settings: "dwgsim -e 0.012 -E 0.012 -d 250 -s 30 -S 0 -N 1000000 -c 2 -1 200 -2 200 -f TACGTACGTCTGAGCATCGATCGATGTACAGC" [36].

These settings were used to produce two data sets, one data set for simulated bisulfite-treated reads, and one data set for regular Ion Torrent reads. The simulated bisulfite-treated Ion Torrent reads were divided into a 10k read training set and a one million read test set in order to tune the logistic gap penalty functions.

The logistic gap open function for BFAST-Gap was trained on the 10k read training set by looking at slopes settings from 1.2 to 0.5 and center settings from -20 to 150. A single index was used with mask "11111111100111111111." The alignment function had values of 96 for a match, -90 for a mismatch, -600 for a gap open penalty, and -50 for a gap extension penalty. The optimal slope ($s_o = 1.0$) and center ($c_o = -15$) were chosen based on which values maximized the F1-score and the area under the curve for all BisPin filter values from 0 to 96.

Once the logistic gap open function was tuned, the settings for the logistic gap open function were set, and the logistic gap extension function was tuned in a similar manner. This resulted in a slope of $s_e = 0.1$ and a center of $c_e = 90$. The tuned logistic gap open and extension functions are considered the default settings for BFAST-Gap. Figure 2 gives the tuned logistic gap extension penalty function. This function is approximately constant until a run length of approximately 50 base pairs.

The whole one million test set was mapped with BisPin calling BFAST-Gap with these settings. One execution was run with constant open and extension penalties, another execution was run with the logistic open penalty function and a constant extension penalty function, and a third execution was run with logistic open and extension penalty functions with the tuned parameters. The F1-score was calculated

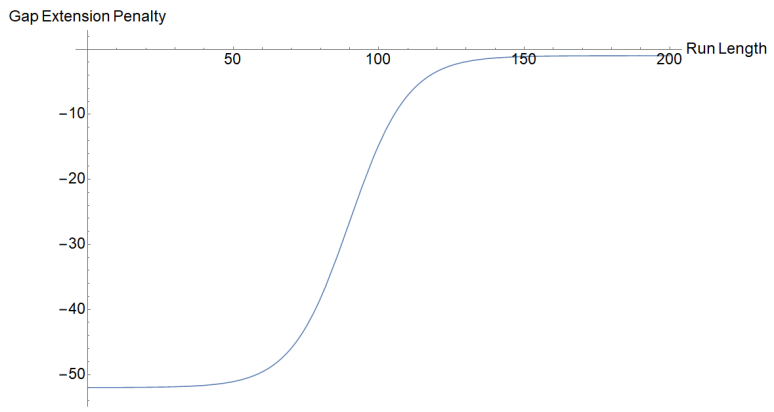


Figure 2: An example of a logistic function for the gap extension penalty for BFAST-Gap. This function has $g_e + z = -51$, $m_n = -50$, $s_e = 0.1$, and $c_e = 90$.

such that an alignment was considered correct if it was within three bases of the real location on the reference genome. BisPin used two indexes, and rescoring for Ion Torrent reads was turned off since the floating point logistic function tends to resolve ambiguously mapped reads. For the simulated regular Ion Torrent data, the $F_{0.5}$ score was calcualted in a similar manner.

Real Data

Variegated real data was downloaded from the SRA (Sequence Read Archive). Each Illumina data set had reads of approximately 100bp length. The mouse data used 100k reads, and the plant data used 500k reads. Only SRR4295457 data was paired end. The rest were single end. Hairpin 101bp mouse data was constructed with the procedure given in [2]. Table 2 gives a summary of the real Illumina read data used in this study. The mapping efficiency was calculated with each mapper's self report except for BWA-Meth since it did not give such a report. For BWA-Meth, a Python script was created to determine the mapping efficiency. If a read had only one location, it was uniquely mapped. If it had several locations including locations given in the `XA:Z` tag, the read was classified as ambiguously mapped. If the alignment had the unmapped or filtered flag set, it was classified as unmapped. The default settings of all mappers were used. A read mapper was not run on a data set if it did not support its construction strategy.

Organism	SRA #	Read Len.	Read Amt.	Layout	Construction	Sequencer
Mus *musculus*	SRR921759	101	100k	Single	Directional	HiSeq 2000
Mus *musculus*	DRR053271	96	100k	Single	PBAT	-
A. *thaliana*	SRR5014638	100	500k	Single	Nondirectional	HiSeq 4000
A. *thaliana*	SRR4295457	101	500k	Paired	Directional	HiSeq 2500

Table 2: A summary of the real Illumina read data features. All data was obtained from the Sequence Read Archive (SRA) at `https://www.ncbi.nlm.nih.gov/sra`.

BisPin used one primary index and two secondary indexes for mouse data but only one secondary index for A. *thaliana* data. These indexes work such that if

a candidate location cannot be found in the primary index, then the secondary indexes are tried. The primary indexes searched for a seed with a length 20 exact match, and the secondary indexes used spaced seeds. Rescoring was on.

To test the logistic gap open function, the optimal slope and center from the simulated data was used and BisPin with BFAST-Gap was run on three real data sets without an alignment quality filter using both a constant and a logistic gap open penalty function. The quality filter for BisPin was turned off since the logistic gap open function changes the alignment score making the alignment scores between the two alignments incomparable. The amount of uniquely mapped reads between these data sets indicates which alignment scoring function performs better. The data was downloaded from the SRA trace archive. The real Ion Torrent data used one million reads of mouse data from SRA numbers SRR1534391 and SRR1534392, and it used one million reads of human data from SRA numbers SRR2842546 and SRR2842547. A small human data set was used, consisting of 251,374 reads from SRA number SRR3305017. The data was aligned with one index when testing BisPin with BFAST-Gap and was aligned with two indexes when testing BisPin with BFAST. Rescoring was turned off.

Real mouse and human data was used to test BFAST-Gap with tuned logistic gap open and extension penalty functions on regular Ion Torrent reads. The SRA numbers were mouse ERR699568, human SRR2734774, and human SRR611141. Each data set had one million reads. The regular mapper programs TMAP, Bowtie2, BWA, and Soap2 were used for comparison. The uniquely mapped, ambiguously mapped, and unmapped percentages were calculated based on the programs self-report for Bowtie2 and Soap2. Custom Python scripts were used to calculate these statistics for BFAST-Gap, TMAP, and BWA.

Results

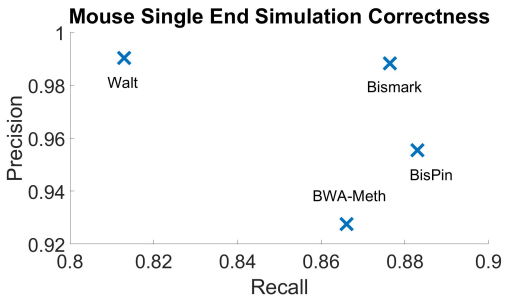
Simulated Data

BisPin had the highest recall for the single end data and the highest precision for the paired end data as shown in Figure 3. Both BisPin and Bismark correctly called the approximate amount of methylation in each context, 80% for CpG and 2% otherwise.

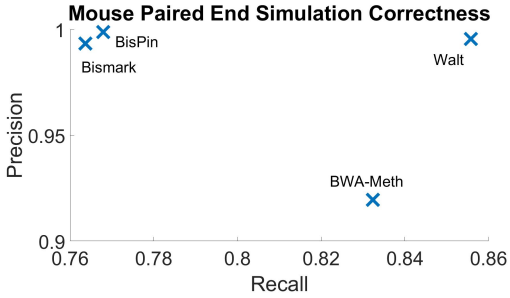
Interestingly, the recall generally decreased by approximately 0.10 for the one million paired end 75bp reads for all the read mappers except for BWA-Meth. The F1-score, a balanced average of precision and recall, for this data was the following, 0.79 (BisPin), 0.80 (Bismark), 0.91 (BWA-Meth), and 0.75 (Walt).

Data consisting of 32 replicates of 100k 150bp single end reads was used to assess the rescoring functionality. Choosing a random alignment for a read from the set of rescored reads mapped 19.1% correctly on average while the rescoring functionality gave an average of 23.5% correct alignments. the rescoring approach always returned more correctly aligned reads than the random choice with an average difference of 4.4%. This suggests a p-value of approximately zero and that there is a statistically significant difference in these distributions such that the rescoring functionality is better than random assignment for uniquely aligning some reads.

Using BisPin with BFAST on default settings and with BFAST-Gap, the one million DWGSIM reads were mapped, and precision, recall, and F1-score were calculated as shown in Figure 4. BWA-Meth and Bismark had moderate performance



(a) Single end directional 150bp data



(b) Paired end directional 75bp data

Figure 3: The average of precision and recall over ten replicates of one million reads of uniquely mapped reads on Sherman mouse read simulations with error 2, CG conversion 20, and CH conversion 98. The standard deviation for all statistics for all data was below 0.0006.

while Walt performed poorly. These read mappers appear ill-suited for mapping Ion Torrent reads. BisPin and Tabsat performed the best. BisPin with BFAST-Gap had slightly higher precision and slightly lower recall but with a higher F1-score than BisPin with BFAST.

The proportion of reads in each mapping category, uniquely mapped, ambiguously mapped, and unmapped or filtered, on the same simulated data for each read map-

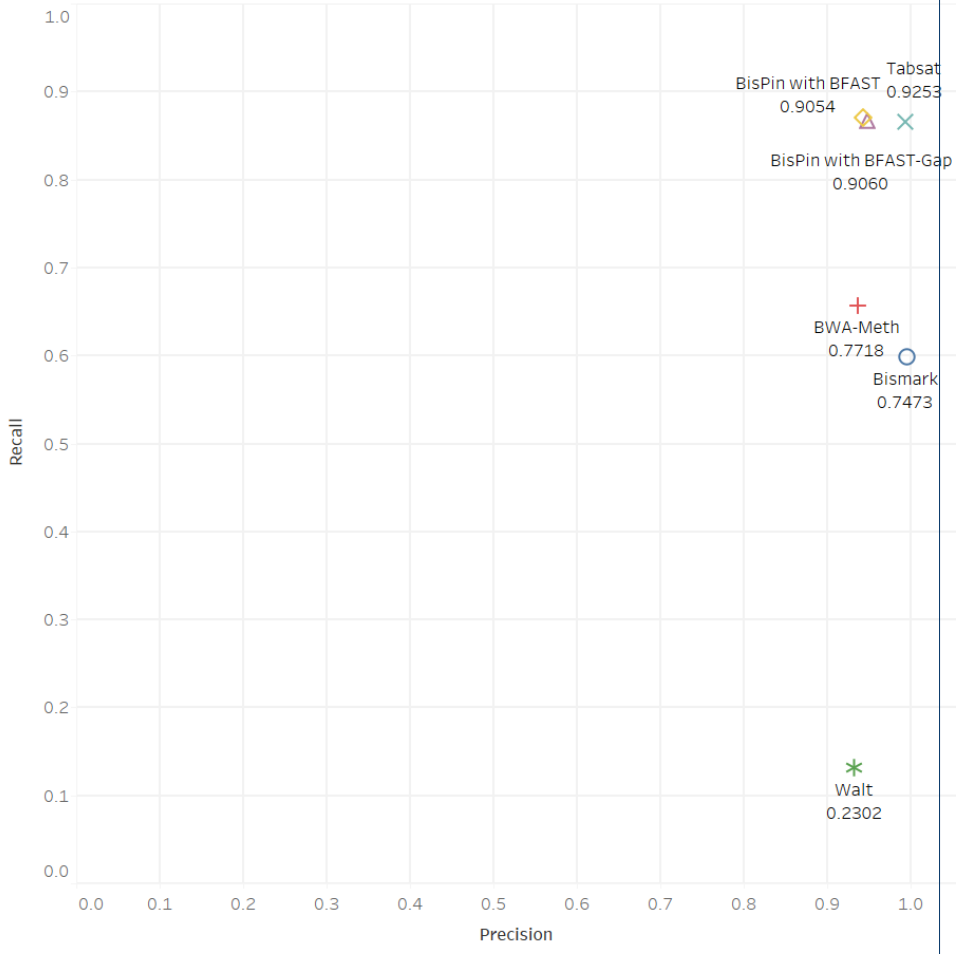


Figure 4: Precision versus recall on simulated human bisulfite-treated Ion Torrent reads. The numeric label is the F1-score.

per is shown in Figure 5. BisPin and Tabsat performed the best with BisPin having about five percent more reads uniquely mapped with few ambiguously mapped. Bismark and BWA-Meth were less good, and Walt performed poorly.

Tuning the logistic gap open function on BFAST-Gap on 10k simulated Ion Torrent reads revealed interesting results as shown in Figure 6. Settings with low slopes tended to perform the worst with low centers performing the best among these. Settings with slopes closer to 1.0 performed the best with centers around 0 to -30; however, with centers much larger than this, performance was worsened. From this analysis, a slope of 1.0 and a center of -15.0 was chosen.

After tuning, BisPin with BFAST-Gap was run on the one million simulated reads test set, and the alignment with the tuned logistic gap open function achieved superior area under the curve and F1-score as can be seen in Figure 7. Executions with two indexes and one index was performed, and using two indexes noticeably improved the performance. The maximizing filter value for the constant penalty function was 65, and for the executions involving the logistic function, it was 85.

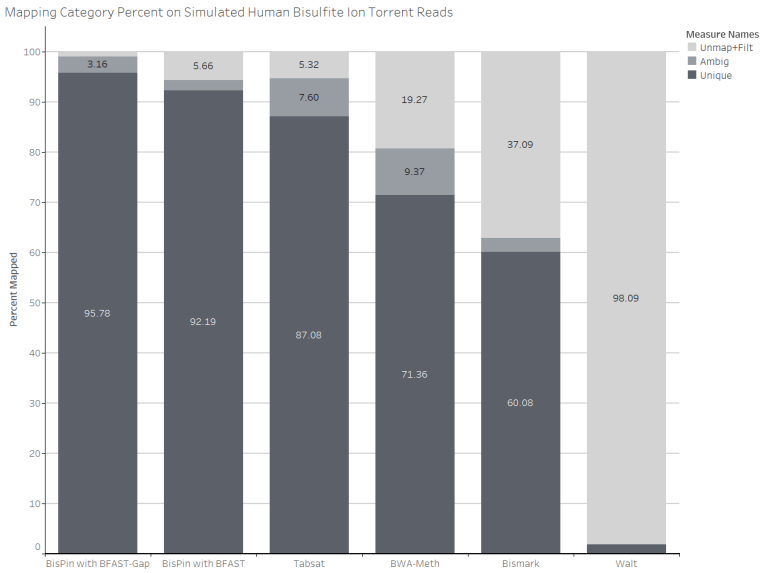


Figure 5: Mapping performance by category on one million DWGSIM simulated bisulfite Ion Torrent reads.

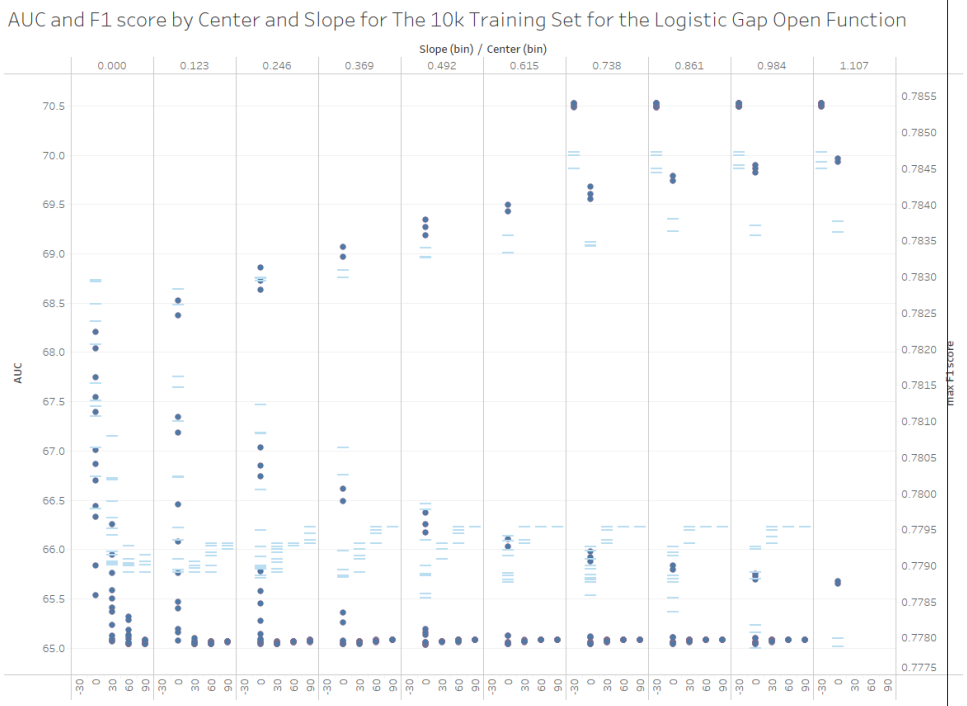


Figure 6: AUC and F1-Score by Slope and Center for the Logistic Gap Open Penalty Function on 10k Simulated BS Ion Torrent Reads. The circles represent AUC scores and the bars represent maximum F1-scores for alignment quality filter values ranging from 0 to 96.

The logistic function tends to reduce alignment scores, so a higher filter value is needed to compensate. The AUC was improved by approximately 7, for the two index case, and the F1-score was improved by 0.003. This indicates that using the logistic gap open function and a constant gap extension penalty function improved mapper performance on this simulation. The executions where both the gap open and extension functions were tuned logistic functions performed nearly identically to the executions with the logistic gap open penalty function and a constant gap extension function but with slightly worse performance.

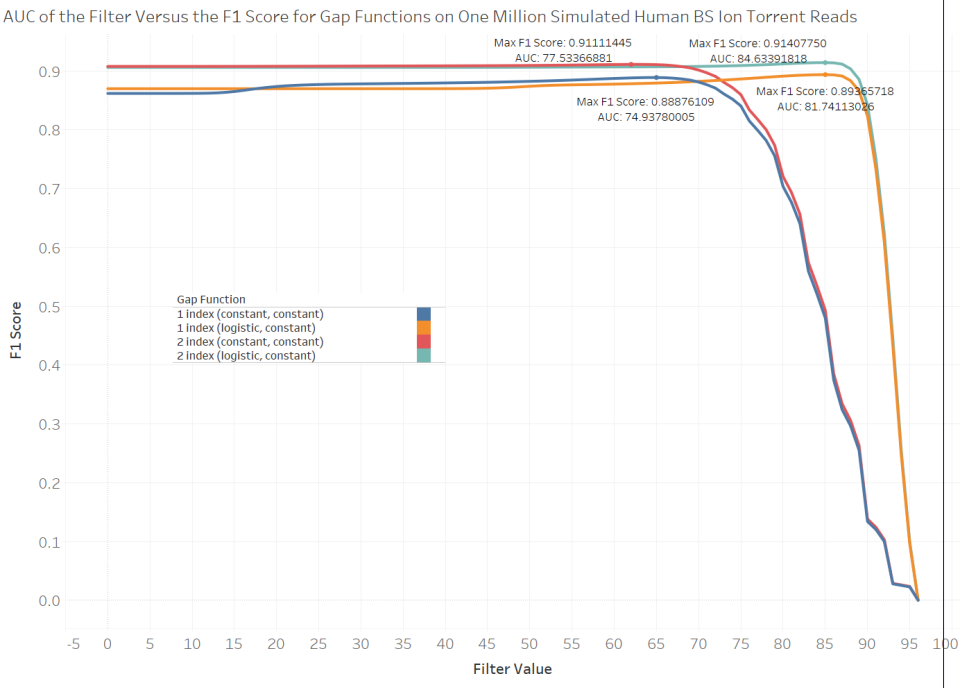


Figure 7: Performance of BisPin with the tuned BFAST-Gap logistic gap open function on one million simulated test reads as measured by area under the curve and F1-score using one index and two indexes.

Figure 8 shows the precision, recall, and $F_{0.5}$-score for mappers on the simulated DWGSIM regular Ion Torrent data. BFAST-Gap had the highest overall score. TMAP performed the worst, which is surprising since it is designed for Ion Torrent reads.

Real Data

The percentage of uniquely mapped reads gives a reasonable test of read mapper performance, and recent papers use this method [11, 7]. The simulated data shows that the percentage of uniquely mapped reads that are correctly mapped is very high. Although the simulated data does not include indels, indels are an order of magnitude more rare than SNPs, so they shouldn't affect mapper performance as much [37]. This suggests that many of the uniquely mapped reads are probably aligned correctly.

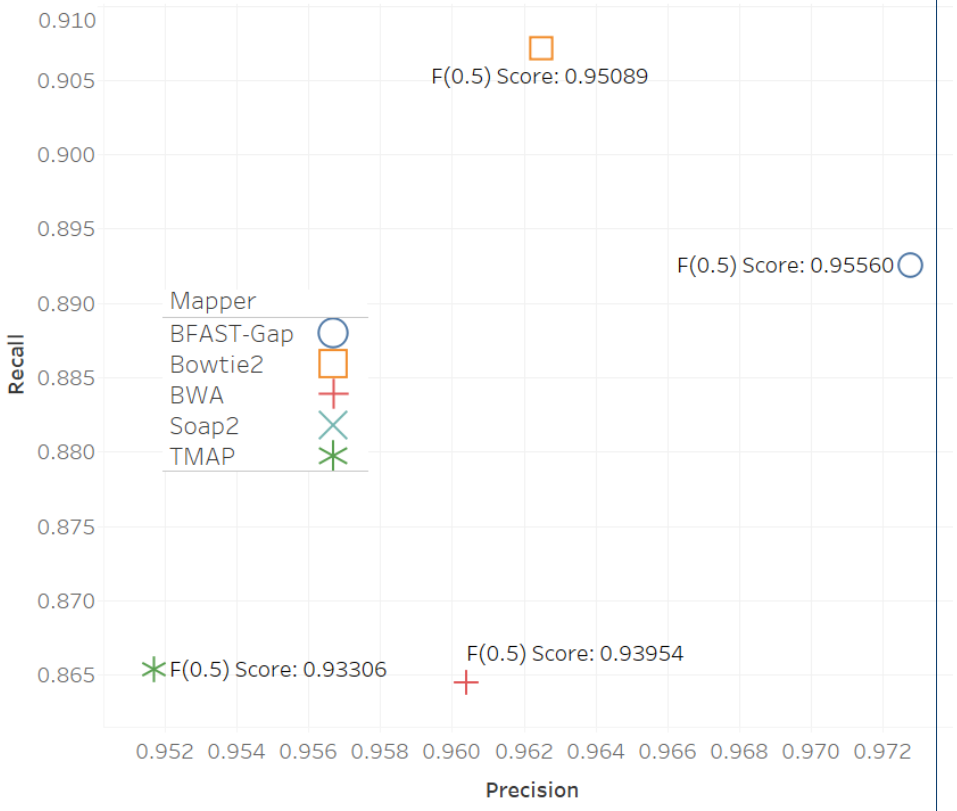
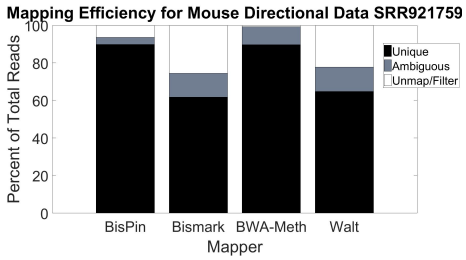


Figure 8: The precision, recall, and $F_{0.5}$-score for regular Ion Torrent DWGSIM 1 million simulated 200bp single-end reads with settings e=1.2 and E=1.2. BFAST-Gap has the best $F_{0.5}$ score and the highest precision. Default settings are used throughout. Soap2 is not shown since it did not produce SAM file output.

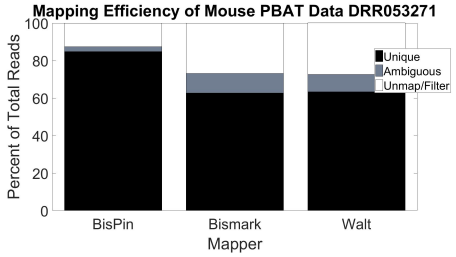
Figure 9 summarizes the mapping efficiency of the read mappers on the four real data sets. In all cases, BisPin had the most uniquely mapped reads. Without rescoring, BisPin would have reported approximately 10% ambiguously mapped reads, but the rescoring moved from $\frac{1}{3}$ to $\frac{2}{3}$ of these to uniquely mapped. BisPin and Bismark comported on the percentage of methylation in all contexts.

On 100k reads of hairpin mouse embryonic data, BisPin did hairpin recovery on 62% of the reads. BisPin mapped this data as regular paired end data with 81.7% uniquely mapped, but other read mappers mapped few reads uniquely in paired end mode. This is probably because paired end mapping usually assumes that one end is upstream of another rather than overlapping as with hairpin data.

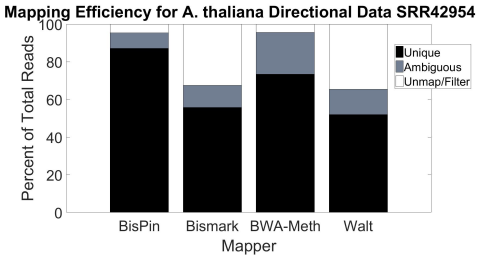
A method of validating the correctness of the read mappers was performed with the hairpin data. One million reads were sampled, and 604431 reads were hairpin recovered. These reads were mapped with BFAST, with BisPin's defaults, and with Bowtie2 and BWA on their default settings. BisPin, Bismark, and BWA-Meth were run on the one million bisulfite reads, and the uniquely mapped reads were compared with the uniquely mapped (hairpin recovered) original reads from each regular read mapper. If the starting location of the bisulfite read was within three bases of the



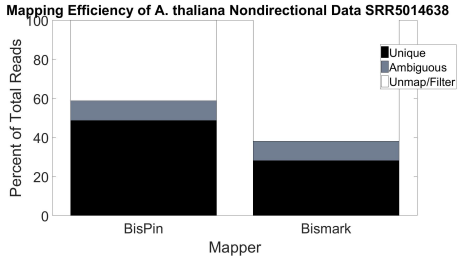
(a) Mouse directional data efficiency



(b) Mouse PBAT data efficiency



(c) A. *thaliana* directional data efficiency



(d) A. *thaliana* nondirectional data efficiency

Figure 9: BisPin mapping efficiency on real data compared to other mappers. BisPin always has the highest number of reads uniquely mapped. Figure 9c is paired-end data, but all other data is single end.

mapped original read, then it was considered correctly mapped. The motivation for this is that the original read should more often map correctly because bisulfite treatment has a tendency to reduce sequence complexity, which can adversely affect mapping quality[4]. These results are summarized in Table 3. BisPin did pretty well with the highest correct using the maximum score.

To test the performance of BisPin with BFAST on bisulfite Ion Torrent data, it was run on default settings along with other mappers with performance results

Bisulfite Mapper	BFAST	Bowtie2	BWA	Average
BisPin	**73.8161%**	60.5174%	60.1448%	64.8261%
Bismark	**63.3881%**	53.0212%	53.8516%	56.75363%
BWA-Meth	**66.3428%**	56.926%	61.8104%	61.693067%
Walt	**72.2992%**	62.6082%	62.0344%	65.647267%

Table 3: Hairpin validation. BFAST, Bowtie2, and BWA were used to map the 604,431 original recovered reads from one million bisulfite reads, and the percent of these recovered reads that were uniquely mapped to the same location with each bisulfite read mapper on the one million bisulfite reads was reported. The maximum value is in bold, and BisPin had the highest value. Walt had the highest average, but BisPin was second with a difference of 0.8%.

indicated in Table 4. BisPin had the most uniquely mapped for all five data sets, and Tabsat was second best. For the mouse data, BisPin mapped more than ten percent of the reads uniquely compared to Tabsat. Walt was the worst with very low performance.

Uniquely Mapped % on Mouse BS Ion Torrent Reads

SRA Mouse #	BisPin	Bismark	BWA-Meth	Tabsat	Walt
SRR1534391	91.79	21.20	45.17	80.65	3.79
SRR1534392	91.38	16.60	38.79	80.05	3.77

Uniquely Mapped % on Human BS Ion Torrent Reads

SRA Human #	BisPin	Bismark	BWA-Meth	Tabsat	Walt
SRR2842546	90.00	60.60	70.42	84.43	19.19
SRR2842547	89.33	60.70	68.28	83.93	18.34
SRR3305017	92.63	35.70	65.32	86.67	23.26

Table 4: The percent of real bisulfite-treated Ion Torrent reads uniquely mapped on five data sets. BisPin performs the best.

Since Ion Torrent reads can vary in length, a test of mapper performance in relation to read length from the SRR2842547 and SRR2842546 data was performed. Figure 10 gives a histogram of read lengths from the SRR1534392 data. This distribution has a long tail with a significant mode at the high end of the distribution indicating that many reads have a large length, but a substantial amount have short lengths.

Buckets, each consisting of 500k reads, were created at assorted read lengths from the SRR2842547 and SRR2842546 data, and the uniquely mapped percent was calculated for each mapper with the results visualized in Figure 11. Except for Walt, mapper performance tended to increase with higher read length; however, for BisPin, BWA-Meth, and Tabsat, mapper performance suffered slightly with the longest reads. Bismark had the most dramatic improvement with increasing read length. Both Tabsat and BisPin performed similarily, but BisPin performed the best.

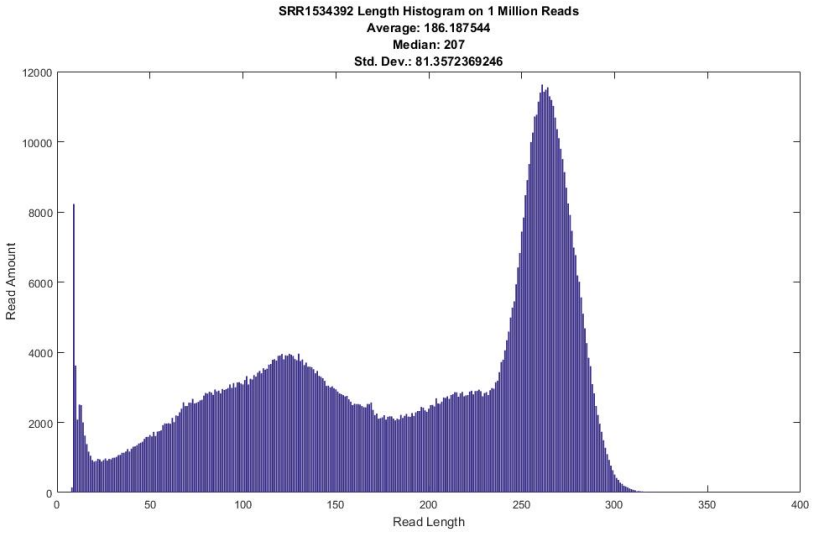


Figure 10: A read length histogram for Ion Torrent reads from data set SRR1534392.

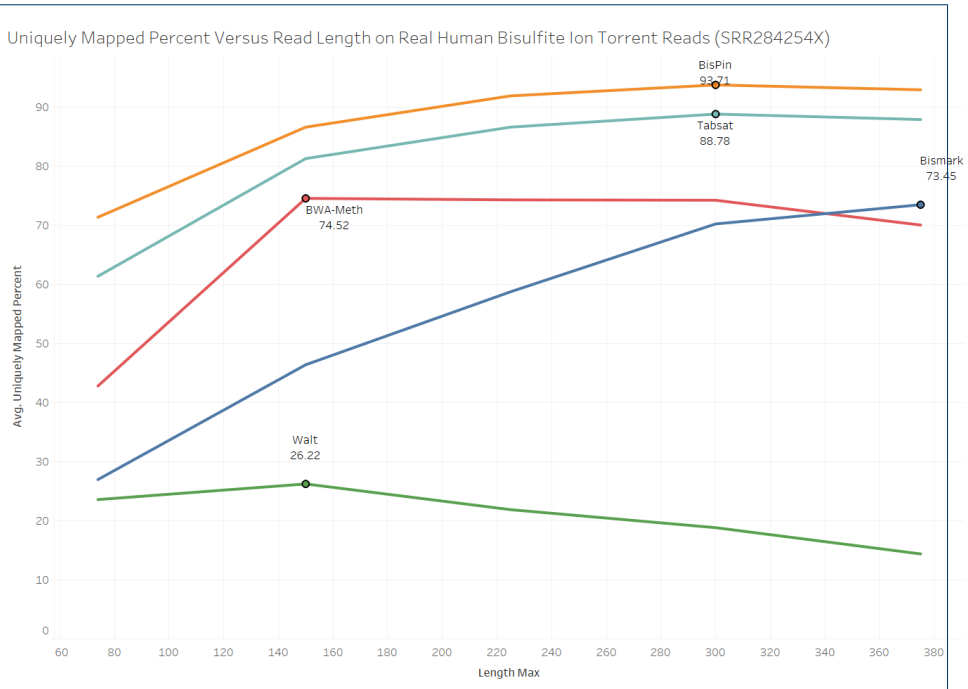


Figure 11: Mapper performance from uniquely mapped percent versus read length on real data. The percent is an average of results from data sets SRR2842547 and SRR2842546.

To see if using extra indexes could improve mapper performance for the smallest reads ranging from 9 to 75 base pairs, six indexes of assorted mask lengths with spaced seeds were created and used to align 500k of these reads, but the results were identical to using two indexes. The ineluctable conclusion is that this strategy will have no effect, so improving the indexing strategy was not pursued further.

For the one million bisulfite-treated Ion Torrent mouse SRR1534391 reads, a test of how similar BisPin with BFAST-Gap results were to Tabsat and BWA-Meth was performed by calculating how many reads there were that mapped to the same location on the genome. A read was considered mapped to the same location for two or more read mappers if it was uniquely aligned and if the genome location differed by no more than three bases. The genome location is given in the SAM alignment file in the RNAME and POS fields. A Venn diagram shows the results in Figure 12.

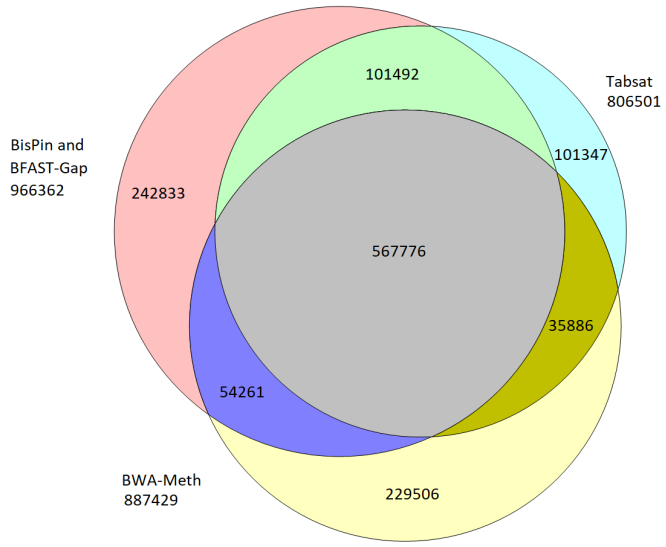


Figure 12: A Venn diagram showing how the uniquely mapped reads from SRR1534391 overlap using mappers BisPin with BFAST-Gap, Tabsat, and BWA-Meth. A read was considered to be in an overlapping set if it was mapped to the same location. The diagram was created with Venn Diagram Plotter available at `https://omics.pnl.gov/software/venn-diagram-plotter`.

The diagram shows that the different read mappers have fairly overlapping results with more than half of the alignments for each read mapper mapping to the same location. This set was 58.758% of BisPin's uniquely mapped reads, and about one quarter of BisPin's uniquely mapped reads had a location unique to BisPin. This shows that BisPin's results are reasonable and similar to other mappers. The set where all read mappers agreed on the location likely has reads correctly mapped. There were 242,833 reads that were uniquely mapped by BFAST-Gap where other read mappers did not agree on the mapped position. Of these reads, 90% were uniquely mapped to other locations by other read mappers. Because of this and since the locations reported by the read mappers disagreed on approximately $25\% - 40\%$ of the reads, this suggests that some kind of ensemble method combining the results of the read mappers could be useful. Perhaps a probabilistic or machine learning model could be employed to combine the results of read mappers.

With the tuned logistic gap open function, BisPin with BFAST-Gap was run on one million real and simulated Ion Torrent bisulfite-treated data with no filter. The execution with a logistic gap open penalty and a constant gap extension penalty had

slightly more uniquely mapped reads indicating better performance as can be seen in Table 5. Using the logistic function for both gap open and gap extension penalty degraded performance a bit for the simulated data and the SRR1534391 data, but it improved performance by six percent on the SRR2842546 data. Two indexes were used for these executions. This table can be compared directly with Table 4 to compare BisPin using BFAST-Gap with other mappers. BisPin performed the best with this data.

Gap Function Type	DWGSIM	SRR1534391	SRR2842546
constant	96.31859%	92.25470%	82.59100%
logistic gap open, constant gap extension	96.46848%	92.87040%	83.24190%
logistic gap open, logistic gap extension	95.80231%	92.4691%	89.6092%

Table 5: Logistic gap open and extension penalty function performance on real and simulated data as measured by uniquely mapped percent without a filter. Compare with Table 4.

Table 6 shows mapper performance on three regular Ion Torrent read data sets. BFAST-Gap generally did the best with consistently high amounts of uniquely mapped reads. The other mappers were more variable with Soap2 generally performing poorly. Although TMAP, Bowtie2, and BWA did fairly well on the human data, Bowtie2 performed very poorly on the mouse data, and TMAP did not do as well. On the simulated data (Figure 8), all of the mappers performed pretty well, so the results on the real data suggest that the simulation is not highly accurate since mapper performance was more varied on real data.

Timing

For alignment time only, on simulated paired end 250k 100bp reads data, BisPin took 103.5 minutes, Bismark took 6.5 minutes, BWA-Meth 3.5 minutes, and Walt 5.5 minutes. BisPin is approximately 20 times slower because BFAST is slow. For one million reads, multiprocessing improved BisPin postprocessing speed, excluding the reference genome load time, by approximately 1.29 times.

On one million reads from the mouse regular Ion Torrent data (SRA#: ERR699568) using 30 threads on a machine with 32 processing cores (E5-2660 @ 2.20GHz with 198 GB of memory), BFAST-Gap took about one hour. Bowtie2 took 5 minutes, and TMAP took 2 minutes. BWA and Soap2 each took approximately half a minute. BFAST-Gap is substantially slower.

Conclusion

Mapping bisulfite reads is challenging. BisPin employs several strategies to address these difficulties. It uses rescoring to disambiguate some ambiguously mapped reads, and it can perform extra processing with additional indexes to attempt to align unmapped reads. It uses multiprocessing and multithreading for improved run-time efficiency, and it can align a selected partition of reads from a file for deployment on a compute cluster. BisPin supports a variety of popular read constructions and layouts including the hairpin construction strategy, which is eminently useful for

Mapper	Unique	Ambiguous	Unmapped or Filtered
Mouse ERR699568			
BFAST-Gap	91.9665%	0.0%	8.0335%
TMAP	60.8798%	12.4475%	26.6727%
Bowtie2	0.53%	0.81%	98.66%
BWA	6.3224%	1.751%	91.9266%
Soap2	0.1%	-	99.2%
Human SRR2734774			
BFAST-Gap	91.374%	0.0%	8.626%
TMAP	91.1326%	8.8664%	0.0%
Bowtie2	77.55%	20.21%	2.23%
BWA	92.4334%	5.8566%	1.71%
Soap2	39.88%	-	60.12%
Human SRR611141			
BFAST-Gap	90.7502%	0.0%	9.2498%
TMAP	89.3589%	9.2704%	1.3707%
Bowtie2	75.06%	21.18%	3.76%
BWA	86.1396%	5.9415%	7.9189%
Soap2	19.7%	-	80.3%

Table 6: Regular Ion Torrent mapper performance on three real data sets of one million reads each. The percentage of reads for each category is shown. BFAST-Gap used a tuned logistic function for the gap open and extension penalty functions, and it generally performed well. Soap2 does not make a distinction between uniquely mapped and ambiguously mapped reads in its reported statistics.

bisulfite-treated reads. BisPin performed well with a variety of real and simulated data compared to other read mappers.

BisPin with BFAST-Gap improves upon read mapping with bisulfite-treated Ion Torrent reads, and BFAST-Gap improves upon read mapping for regular Ion Torrent reads compared to other read mappers including TMAP, which was designed for Ion Torrent reads, in some ways.

For improving run-time, BFAST could be modified to include a highly optimized SIMD implementation of the Smith-Waterman algorithm as in Bowtie2, and it could dispense with intermediate files, which are slow to create. Perhaps BisPin's hairpin sequencing approaches could be applied to Oxford Nanopore data because it uses a hairpin connector. Ranking the alignments with a probability measure could be more precise and informative as is done in [38], but it could be less time efficient as computing a probability may require more calculations.

Competing interests

There are no conflicts of interest to declare.

Author's contributions

Jacob Porter created BisPin, modified BFAST to produce BFAST-Gap, performed the data analysis, and wrote the paper, and Liqing Zhang consulted on the data analysis and edited the manuscript.

Acknowledgements

The hairpin data was provided by Hehuang Xie of the Biocomplexity Institute of Virginia Tech. Hong Tran provided the Python software to simulate bisulfite treatment.

Availability of data and materials

BisPin (Python 2.7) and BFAST-Gap (C++) are available at `https://www.github.com/JacobPorter` under a GNU General Public License.

References

1. Allis, C.D., Jenuwein, T., Reinberg, D., Caparros, M.-L.: Epigenetics. Cold Spring Harbor Laboratory Press Cold Spring Harbor, NY:, ??? (2007)
2. Zhao, L., Sun, M.-a., Li, Z., Bai, X., Yu, M., Wang, M., Liang, L., Shao, X., Arnovitz, S., Wang, Q., *et al.*: The dynamics of DNA methylation fidelity during mouse embryonic stem cell self-renewal and differentiation. Genome Research **24**(8), 1296–1307 (2014)
3. Tran, H., Porter, J., Sun, M.-a., Xie, H., Zhang, L.: Objective and comprehensive evaluation of bisulfite short read mapping tools. Advances in Bioinformatics **2014** (2014)
4. Porter, J., Sun, M.-a., Xie, H., Zhang, L.: Investigating bisulfite short-read mapping failure with hairpin bisulfite sequencing data. BMC Genomics **16**(11), 2 (2015)
5. Krueger, F., Andrews, S.R.: Bismark: a flexible aligner and methylation caller for bisulfite-seq applications. Bioinformatics **27**(11), 1571–1572 (2011)
6. Pedersen, B.S., Eyring, K., De, S., Yang, I.V., Schwartz, D.A.: Fast and accurate alignment of long bisulfite-seq reads. arXiv preprint arXiv:1401.1129 (2014)
7. Chen, H., Smith, A.D., Chen, T.: WALT: fast and accurate read mapping for bisulfite sequencing. Bioinformatics **32**(22), 3507–3509 (2016)
8. Li, H., Durbin, R.: Fast and accurate long-read alignment with Burrows–Wheeler transform. Bioinformatics **26**(5), 589–595 (2010)
9. Langmead, B., Salzberg, S.L.: Fast gapped-read alignment with Bowtie 2. Nature Methods **9**(4), 357–359 (2012)
10. Lim, J.-Q., Tennakoon, C., Li, G., Wong, E., Ruan, Y., Wei, C.-L., Sung, W.-K.: BatMeth: improved mapper for bisulfite sequencing reads on DNA methylation. Genome Biology **13**(10), 82 (2012)
11. Harris, E.Y., Ounit, R., Lonardi, S.: BRAT-nova – Fast and accurate mapping of bi-sulfite-treated reads. Bioinformatics, 226 (2016)
12. Xi, Y., Li, W.: BSMAP: whole genome bisulfite sequence MAPping program. BMC Bioinformatics **10**(1), 232 (2009)
13. Guo, W., Fiziev, P., Yan, W., Cokus, S., Sun, X., Zhang, M.Q., Chen, P.-Y., Pellegrini, M.: BS-Seeker2: a versatile aligning pipeline for bisulfite sequencing data. BMC Genomics **14**(1), 774 (2013)
14. Hansen, K.D., Langmead, B., Irizarry, R.A.: BSmooth: from whole genome bisulfite sequencing reads to differentially methylated regions. Genome Biology **13**(10), 83 (2012)
15. Porter, J., Berkhahn, J., Zhang, L.: A comparative analysis of computational indel calling pipelines for next generation sequencing data. In: Proceedings of the International Conference on Bioinformatics & Computational Biology (BIOCOMP), p. 1 (2014). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp)
16. Bragg, L.M., Stone, G., Butler, M.K., Hugenholtz, P., Tyson, G.W.: Shining a light on dark sequencing: characterising errors in Ion Torrent PGM data. PLoS Computational Biology **9**(4), 1003031 (2013)
17. Perkel, J.: Making contact with sequencing's fourth generation. BioTechniques **50**(2), 93–95 (2011)
18. Schirmer, M., D'Amore, R., Ijaz, U.Z., Hall, N., Quince, C.: Illumina error profiles: resolving fine-scale variation in metagenomic sequencing data. BMC Bioinformatics **17**(1), 125 (2016)
19. Loman, N.J., Misra, R.V., Dallman, T.J., Constantinidou, C., Gharbia, S.E., Wain, J., Pallen, M.J.: Performance comparison of benchtop high-throughput sequencing platforms. Nature Biotechnology **30**(5), 434–439 (2012)
20. Laird, C.D., Pleasant, N.D., Clark, A.D., Sneeden, J.L., Hassan, K.M.A., Manley, N.C., Vary, J.C., Morgan, T., Hansen, R.S., Stöger, R.: Hairpin-bisulfite PCR: assessing epigenetic methylation patterns on complementary strands of individual DNA molecules. Proceedings of the National Academy of Sciences **101**(1), 204–209 (2004)
21. Pabinger, S., Ernst, K., Pulverer, W., Kallmeyer, R., Valdes, A.M., Metrustry, S., Katic, D., Nuzzo, A., Kriegner, A., Vierlinger, K., *et al.*: Analysis and visualization tool for targeted amplicon bisulfite sequencing on Ion Torrent sequencers. PloS One **11**(7), 0160227 (2016)
22. Homer, N.: TMAP: The torrent mapping program (2011). https://github.com/iontorrent/TMAP/blob/master/doc/tmap-book.pdf Accessed 2017-08-30
23. Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., Durbin, R., *et al.*: The sequence alignment/map format and SAMtools. Bioinformatics **25**(16), 2078–2079 (2009)
24. Lister, R., Pelizzola, M., Dowen, R.H., Hawkins, R.D., Hon, G., Tonti-Filippini, J., Nery, J.R., Lee, L., Ye, Z., Ngo, Q.-M., *et al.*: Human DNA methylomes at base resolution show widespread epigenomic differences. Nature **462**(7271), 315–322 (2009)
25. Miura, F., Enomoto, Y., Dairiki, R., Ito, T.: Amplification-free whole-genome bisulfite sequencing by post-bisulfite adaptor tagging. Nucleic Acids Research **40**(17), 136–136 (2012)
26. Cokus, S.J., Feng, S., Zhang, X., Chen, Z., Merriman, B., Haudenschild, C.D., Pradhan, S., Nelson, S.F., Pellegrini, M., Jacobsen, S.E.: Shotgun bisulphite sequencing of the arabidopsis genome reveals DNA methylation patterning. Nature **452**(7184), 215–219 (2008)
27. Tran, H., Wu, X., Tithi, S., Sun, M.-a., Xie, H., Zhang, L.: A Bayesian assignment method for ambiguous bisulfite short reads. PloS One **11**(3), 0151826 (2016)

28. Yap, V.B., Miller, W.: Scoring pairwise genomic sequence alignments. In: Pacific Symposium on Biocomputing 2002: Kauai, Hawaii, 3-7 January 2002, p. 115 (2001). World Scientific
29. Schwartz, S., Kent, W.J., Smit, A., Zhang, Z., Baertsch, R., Hardison, R.C., Haussler, D., Miller, W.: Human–mouse alignments with BLASTZ. Genome Research **13**(1), 103–107 (2003)
30. Frith, M.C., Mori, R., Asai, K.: A mostly traditional approach improves alignment of bisulfite-converted DNA. Nucleic Acids Research, 275 (2012)
31. Birdsell, J.A.: Integrating genomics, bioinformatics, and classical genetics to study the effects of recombination on genome evolution. Molecular Biology and Evolution **19**(7), 1181–1197 (2002)
32. Ventura, M., Canchaya, C., Tauch, A., Chandra, G., Fitzgerald, G.F., Chater, K.F., van Sinderen, D.: Genomics of Actinobacteria: tracing the evolutionary history of an ancient phylum. Microbiology and Molecular Biology Reviews **71**(3), 495–548 (2007)
33. Hamilton, W.L., Claessens, A., Otto, T.D., Kekre, M., Fairhurst, R.M., Rayner, J.C., Kwiatkowski, D.: Extreme mutation bias and high AT content in Plasmodium falciparum. Nucleic Acids Research, 1259 (2016)
34. Smolka, M., Rescheneder, P., Schatz, M.C., von Haeseler, A., Sedlazeck, F.J.: Teaser: Individualized benchmarking and optimization of read mapping results for NGS data. Genome Biology **16**(1), 235 (2015)
35. Homer, N.: DWGSIM (2017). https://github.com/nh13/DWGSIM Accessed 2017-01-01
36. Bhd., N.T.S.: Benchmarking ION Torrent PGM Aligners (2017). http://www.novocraft.com/documentation/other-sequencing-platforms/benchmarking-ion-torrent-pgm-aligners/ Accessed 2017-08-11
37. Mullaney, J.M., Mills, R.E., Pittard, W.S., Devine, S.E.: Small insertions and deletions (indels) in human genomes. Human Molecular Genetics **19**(R2), 131–136 (2010)
38. Kerpedjiev, P., Frellsen, J., Lindgreen, S., Krogh, A.: Adaptable probabilistic mapping of short reads using position specific scoring matrices. BMC Bioinformatics **15**(1), 100 (2014)