BIOINFORMATICS APPLICATIONS NOTE

Sequence analysis

BRAT: Bisulfite-treated Reads Analysis Tool (Supplementary Methods)

Elena Y. Harris[1,*], Nadia Ponts[2], Aleksandr Levchuk[3], Karine Le Roch[2] and Stefano Lonardi[1]

[1]Department of Computer Science, University of California, Riverside, CA 92521

[2]Department of Cell Biology and Neuroscience, University of California, Riverside, CA 92521

[3]Institute for Integrative Genome Biology, University of California, Riverside, CA 92521

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: XXXXXXX

1 BASIC CONCEPTS AND NOTATIONS

The 'gold-standard' method to study genome-wide DNA methylation takes advantage of the effect of sodium bisulfite (BS) conversion on DNA. After BS treatment, several steps of PCR amplification are applied, and the resulting DNA is sequenced. Figure 1 illustrates the effect of BS conversion and subsequent PCR amplification. The protocol for next-generation sequencing instruments (e.g., Illumina Genome Analyzer) requires adding special methylated adaptors before BS treatment. The presence of the adaptors controls which DNA strands are sequenced: although there are four strands of PCR product (PCR1+, PCR1–, PCR2+ and PCR2− in Figure 1), the actual sequencing is carried out only for PCR1+ and PCR2−, which are the original genomic strands (see Figure 2). Methylated Cs in the genome remain Cs after BS conversion, while unmethylated Cs are transformed. Unmethylated Cs in the positive strand of the genome (DS+) are converted into Ts in PCR1+ and stay Cs in PCR2+. The same is true for unmethylated Cs in DS–, which stay Cs in PCR1– and turn into Ts in PCR2–. Observe that since most Cs are expected to be unmethylated in the genome, PCR1+ and PCR2+ are C-poor, while PCR1− and PCR2− are G-poor.

Figure 2 illustrates the effect of sequencing (following PCR) and the resulting paired-end reads, which are the input to BS-mapping. The order of mates in paired-end reads is essential for BS-mapping. When the 5' mate maps to the positive strand of the genome (DS+), then only T-C mismatches are legal. If the reverse-complement of the 5' mate maps to DS+, then only A-G mismatches can be allowed. Similarly, if the 3' mate maps to DS+, then only A-G mismatches are legal, and if the reverse-complement of the 3' mate maps to DS+ then only T-C mismatches can be allowed. Similar rules apply to single reads sequencing: these reads must follow the same rules as for 5' mate for paired-ends (in Figure 2, the 5' mate is called read 1 and the 3' mate is called read 2).

BS-mapping refers to the computational process of mapping short reads obtained after bisulfite treatment to a reference genome. Due to the effect of BS conversion, out of the sixteen possible mappings between a symbol in a read and a symbol in a reference genome we allow six, namely A-A, A-G, C-C, G-G, T-C and T-T. Mappings A-G and T-C are called *BS-mismatches*. The other ten mappings are considered true mismatches (hereafter they are called *non-BS-mismatches)*.

We say that a paired-end (or single) read is *unique* if it maps with any number of BS-mismatches (including zero) to a unique location in the reference genome. If it maps to multiple locations under the same conditions, the read is called *ambiguous*. When we allow non-BS-mismatches, a single read is *unique* if it maps to a unique location with the smallest number of non-BS-mismatches (a BS-mismatch is equivalent to zero non-BS-mismatches); a paired-end read is *unique* if it maps to a unique location with the smallest number of non-BS-mismatches in both mates. If it maps to multiple locations (under the same conditions), that read/pair is called *ambiguous*. We define the *mapping accuracy* as the ratio between the number of correctly mapped unique pairs/reads and the total number of mapped unique pairs/reads.

Let us illustrate the definition of uniqueness with a few examples. Consider a single read that maps with one BS-mismatch to one location in the reference genome and with two BS-mismatches to another location. This read is ambiguous because it maps to more than one location with the allowed BS-mismatches. Let us now consider a single read that maps to one location with three BS-mismatches and to two additional locations with one non-BS-mismatch. This read is unique because it maps with the smallest number of non-BS-mismatches (zero in this case) to a single location. Finally, let us consider an example with paired-end reads. Assume that a paired-end read maps to two distinct locations: in the first, the mapping has two BS-mismatches for the left mate and five BS-mismatches for the right mate; in the second location, the mapping is a perfect match for the left mate and has one non-BS-mismatch for the right mate: in this case the smallest number of non-BS-mismatches is zero (mapping to the first location has only BS-mismatches that are counted as zero non-BS-mismatches) and the mapping is unique.

We define a *k-mer* as any word (or substring) of k consecutive bases obtained from the reference genome.

2 BRAT: THE ALGORITHM

Each read, its reverse complement and the reference genome are represented in BRAT according to their *ta*- and *cg*-binary representations. In the *ta*-representation, all Cs and Ts are converted to ones, and As and Gs are converted to zeroes, which can be interpreted that all Cs are converted to Ts and, similarly, that all Gs are converted to As. In other words, the *ta*-representation is the reduced genome over two letters, namely T and A, which is reflected in the chosen name. In the *cg*-representation, Cs and Gs are converted to ones, and As and Ts are converted to zeroes. This representation is solely used to verify C-C and G-G mappings described below, and its name reflects this functional role. In both cases, each base is represented by one bit. Hereafter, we call these two induced representations for the reference genome as the *ta-genome* and the *cg-genome.*

During the preprocessing step, BRAT builds a hash table on the reference genome. The algorithm first identifies the shortest read in the input reads (let w be its length). The hashing function uses w consecutive bases of the forward strand of the *ta*-genome and the *cg*-genome as *seeds* to calculate the corresponding key for the entry in the hash table. The entry is updated with the corresponding reference name and position within the reference where the seeds come from. Similarly, each read of length $k \geq w$ is hashed on the first w bases to find the corresponding key for that read. We used variants of a hashing function designed by Bob Jenkins (see http://burtleburtle.net/bob/c/lookup8.c). There are four hashing functions used in BRAT: one for normal mapping, one for BS-mapping with large genomes and two for BS-mapping with small genomes (details of the hashing functions are given in Figure 4). After the entry corresponding to a read has been identified, that read is aligned to all k-mers of the genome whose starting positions are stored at this entry (hereafter, in order to distinguish between hashing and mapping, we use w for hashing, and k for mapping, where w is the length of the shortest read and k is the length of a read with $k \geq w$).

The alignment process is carried out on the binary representations in three steps: *ta*-check, *cg*-check and final verification. First, we describe the procedure for zero non-BS-mismatches. We use *ta*-check to verify that Ts in sequenced reads map only to Ts in the *ta*-genome and similarly that As in reads map to As in *ta*-genome; the corresponding mappings on the original genome are: T-C, T-T, A-G and A-A (which are allowable mappings) and C-T and G-A (which are erroneous mappings that should not be allowed, but the *cg*-check takes care of these). During the *ta*-check, the *ta*-representation of a read (hereafter called *ta*-read) is compared with k-mer from the *ta*-genome: if the bits in the *ta*-read and the k-mer are equal, we proceed to the next step. We use *cg*-check to ensure that Cs in sequenced reads map only to Cs in the reference genome and similarly that Gs in reads map exclusively to Gs in the reference. This is done by checking the Boolean condition [*cg*-read == (*cg*-read AND (k-mer in the *cg*-genome))], where AND is the normal bitwise-AND operation. In the final check, we verify the correct combination of a read orientation and BS-mismatches types using the rules described in Section 1 and illustrated in Figure 2.

Next we describe the mapping procedure when the number of non-BS-mismatches is greater than zero. The mapping procedure when the number of non-BS-mismatches is greater than zero is similar to the one for zero non-BS-mismatches. The difference lies in using XOR instead of equality operation in *ta*-check and *cg*-check (where XOR is the normal bitwise exclusive-or operation). The number of non-BS-mismatches is counted as the number of bits set to 1 by XOR. To allow for non-BS-mismatches, BRAT as RMAP-BS uses spaced (or masked) seeds. We still use seeds

of *w* consecutive bases, but mask some of the bases with zeros to allow for non-BS-mismatches. BRAT uses a total of four seeds: the first seed is for the reads that could be mapped with zero non-BS-mismatches (these reads are not considered in further mapping steps) and the other three seeds are used to find all mappings with one non-BS-mismatch. To allow one non-BS-mismatch, it would be sufficient to use just two masked (or spaced) seeds, but using three masked seeds retain more bit information for hashing. Examples of seeds are given in Figure 5.

Figure 3 demonstrates that *ta*-check followed by *cg*-check induces only BS-mismatches and perfect matches.. The two input strings are chosen so that all possible combinations of alignments of bases in a read to bases in a reference are obtained.

3 BRAT: THE SOFTWARE SUITE

We offer users two versions of BRAT, namely BRAT and BRAT-LARGE. Both programs run on 64-bit architecture under Linux: we made sure that they can compile on the five major Linux variants, namely Ubuntu, CentOS (RedHat), Debian, Suse, and Fedora. Input reads have to be at least 24 bases long. There is no upper limit on reads lengths. Users can specify any number of non-BS-mismatches, but BRAT guarantees to find all read-genome mappings with up to one non-BS-mismatch in the first 36 bases of reads. BRAT-LARGE guarantees to find all read-genome mappings as long as the first 24 bases of each read can be mapped either perfectly or with any number of BS-mismatches.

BRAT accepts up to 65,536 reference sequences (genomes/chromosomes), where the maximum size of the cumulative reference sequence is 4.2GB. BRAT-LARGE also allows up to 65,536 references where the size of the largest reference sequence is 4.2GB .

BRAT works best with relatively small genomes because it requires significantly more memory than BRAT-LARGE. However, BRAT is faster than BRAT-LARGE because the latter uses only one hashing function and maps the reads to the reference genome sequentially: first it maps all reads to the first reference, and then all reads to the second reference and so on. The memory space used by BRAT-LARGE depends on the size of the largest reference in the set of input references, whereas the space used by BRAT depends on the total size of all input references (measured in base pairs). If we define N to be the total size of a genome in base pairs, P to be the size of the largest reference in base pairs and R to be the total number of reads (each read is counted, *i.e.* a pair has 2 reads), the space required by the two programs is bounded by

$$\text{Space}_{\text{BRAT}} = 269 \cdot 10^6 + (2 \cdot 4 + 3/8)N + 25R \text{ Bytes}$$

$$\text{Space}_{\text{BRAT-large}} = 269 \cdot 10^6 + (2 \cdot 4 + 3/8)P + 25R \text{ Bytes (with option } S\text{)}$$
$$\text{Space}_{\text{BRAT-large}} = 269 \cdot 10^6 + (4 + 3/8)P + 25R \text{ Bytes}$$

where 269MB accounts for the space required supporting the data structure for the hash table. A total of 4 bytes is used to store a reference position in the hash table for BRAT and BRAT-LARGE. In BS-mapping when non-BS-mismatches are not allowed, BRAT uses two different hashing functions: one for a read and one for its reverse-complement; therefore, each genome position can be hashed to at most two different entries of the hash table. With BS-mapping and non-BS-mismatches, BRAT uses a single hashing function (*hash-ta* shown in Suppl. Fig. 4), so each genome position can be hashed to at most two different entries of the hash table (one for each mate).

When the memory available is limited, a user has a choice to use BRAT-LARGE. For example, with paired-end BS-mapping and non-BS-mismatches, BRAT uses 2.5GB of memory to index human chromosome 1, whereas BRAT-LARGE needs just 1.7GB (or 2.7GB when option *S* is used) of memory for the entire human genome.

The package includes two additional tools: TRIM and ACGT-COUNT. The tool TRIM accepts FASTQ files with reads/pairs as input and trims the ends of reads whose base quality scores are lower than user specified threshold and finally filters reads/pairs, which contain internal Ns. The other tool, ACGT-COUNT, aligns mapped reads to the ge-

nome and counts the number of occurrences of A, C, G and T at each base in the forward and the reverse strands separately.

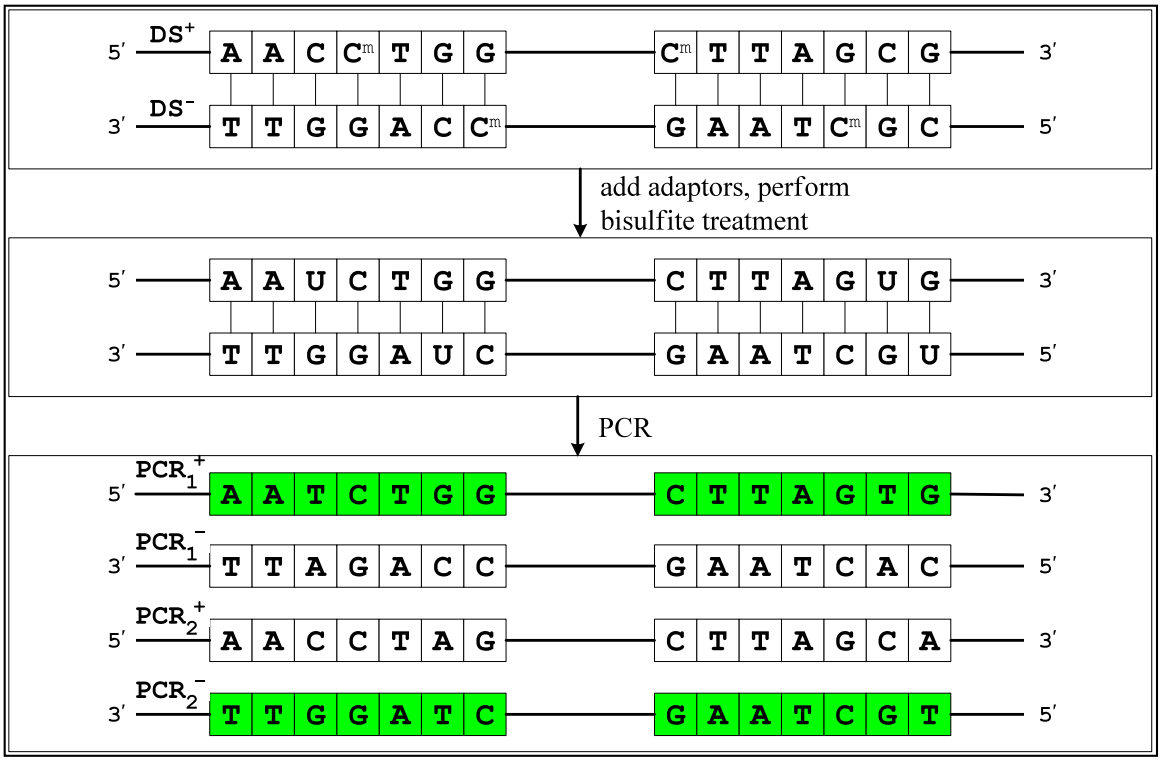
4 DATASETS AND PARAMETERS USED IN EXPERIMENTS

All tests were carried out on a cluster of 4x Quadcore Intel Xeon CPU running at 2.40GHz (16 CPU cores), with a total of 64GB of RAM.. We used *P. falciparum*'s genome version is PlasmoDB 6.0 (downloaded from http://plasmodb.org/plasmo/) and *H. sapiens* genome version hg18 (downloaded from http://hgdownload.cse.ucsc.edu/goldenPath/hg18/bigZips/).

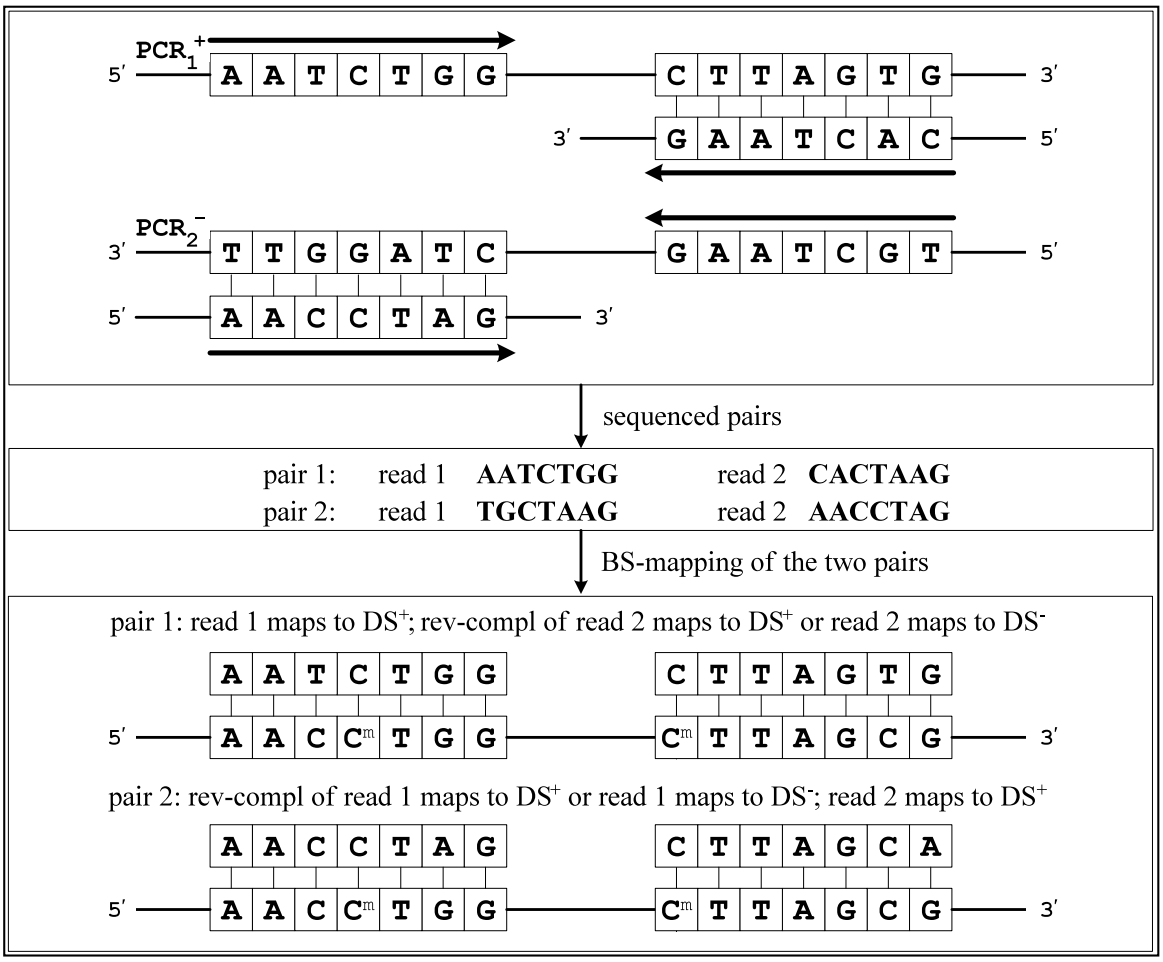
The parameters used in the experiments were RMAP-BS (m 0, S 1, h 26), BSMAP (s 9, v 0, r 0, m 106, x 306), and MRSFAST (e 0, n 2, min 106, max 306). For BSMAP, we used the largest seed allowed by the program (parameter *s*).

Here we describe details for the experiments shown in Figure 1 of the manuscript. Sequencing errors were introduced at random positions: first a read was chosen randomly, then a base within the read was chosen uniformly at random, and finally, the chosen base was substituted by a letter randomly chosen out of the three remaining letters. The parameters used for this experiments were: MRSFAST (*n* 2; *e*: 0, 1, 2; *min* 136; *max* 324) and BRAT (*i* 113; *a* 301; *m*: 0, 1, 2). The minimum and maximum values for insert size were taken different than the corresponding values for generated *in silico* pairs (MRSFAST: *min* and *max* and BRAT: *i* and *a*). Here the values for min/max insert size for both programs are different due to different definitions for these values used by the programs: BRAT counts an insert size as the distance between the leftmost points of the alignments of the two mates on the forward strand, and MRSFAST as the distance between the outmost end points of the alignments of the two mates. The options *e* and *m* set the number of non-BS-mismatches in MRSFAST and BRAT respectively. The two programs differ in the way they output the results: BRAT outputs only unique alignments and MRSFAST outputs a user-specified number of alignments (parameter *n*). Therefore, to distinguish between unique and ambiguous reads/pairs, users must use the option *n* in MRSFAST. Unfortunately, this option does not guarantee to find necessarily unique reads or the best alignments. For example, we found cases when MRSFAST reported mapped locations for pairs with a larger number of non-BS-mismatches than another location that was missed (in this case, a read is counted as ambiguous even though it is unique). Another source of poorer mapping accuracy in MRSFAST is due to missing ambiguous reads, i.e., reads for which only one result is found (with *n* 2) even though there exists an equally good alignment somewhere else in the genome. We computed the number of correctly mapped unique reads for MRSFAST by excluding all the ambiguous reads as identified by MRSFAST (occurring twice in the output) and all ambiguous identified by BRAT (occurring in at least two valid and equally good alignments). For total unique pairs, we counted the reads occurring once in the resulting output.

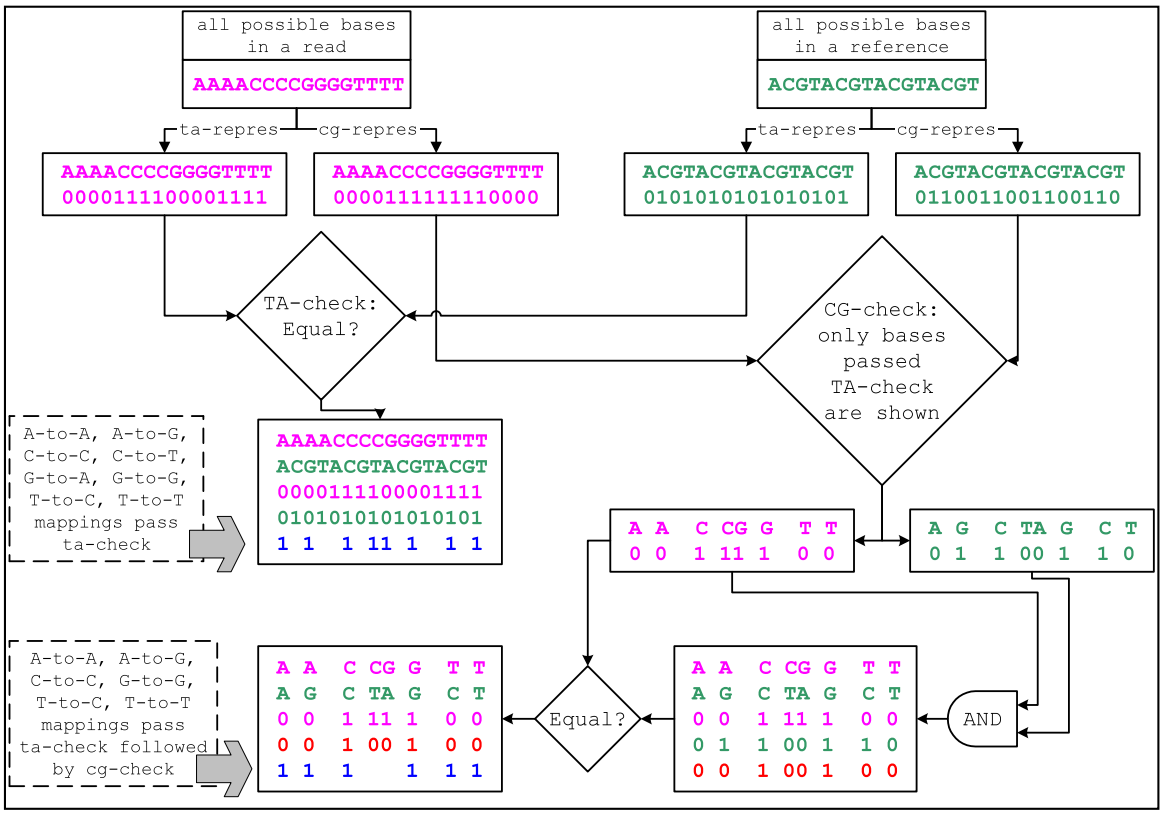
Supplemental Fig. 1. The effect of BS-conversion followed by PCR.



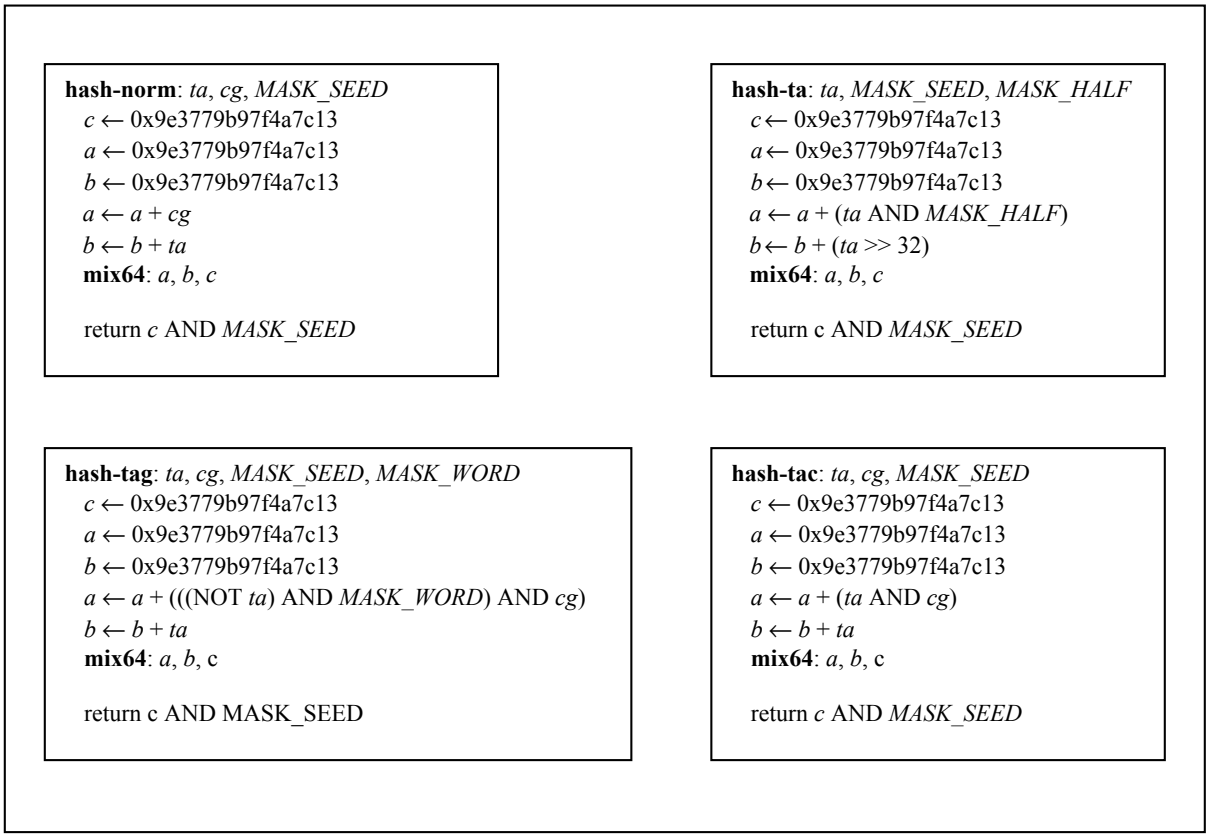
Supp. Fig. 2. BS-sequencing and BS-mapping. The order of mates in paired-end reads is essential for BS-mapping. When the 5' mate maps to the positive strand of the genome (DS+), then only T-C mismatches are legal. If the reverse-complement of the 5' mate maps to DS+, then only A-G mismatches can be allowed. Similarly, if the 3' mate maps to DS+, then only A-G mismatches are legal, and if the reverse-complement of the 3' mate maps to DS+ then only T-C mismatches can be allowed. Similar rules apply to single reads mapping: these reads must follow the same rules as for 5' mate for paired-ends.



Supp. Fig. 3. The effect of *ta*-check followed by *cg*-check on all possible combinations of symbols between a read and *k*-mer from the reference genome.



Supp. Fig. 4. Hashing functions used in mapping (based on the hashing functions by Bob Jenkins): BRAT and BRAT-LARGE use **hash-norm** for normal mapping; BRAT-LARGE uses **hash-ta** for BS-mapping and BRAT uses **hash-tac** and **hash-tag** for BS-mapping. The subroutine **mix64** was designed by Bob Jenkins (see website given above). All variables here are 64-bit unsigned long integers. MASK_SEED masks 24 least significant bits with ones, MASK_HALF masks 32 least significant bits with ones, MASK_WORD masks *w* consecutive bits with ones (*w* is the length of the shortest read), and *ta* and *cg* are seeds of *ta*- and *cg*- binary representations of reads and *k*-mers from the reference genome. The AND operation here is a standard bitwise operation, and (*x*>>*y*) is a shift of the binary number *x, y* bits to right.



Supp. Fig. 5. Example of a seed set used in mapping. If the shortest read has length w, a third of w is taken as a measure by which we subdivide a seed. The figure shows an examples of seeds for $w = 32$ (bits with values 1 here correspond to the bits not masked). These seeds are used with single-end reads and with each mate of paired-end reads. The first seed finds all alignments with zero non-BS-mismatches, while the rest of the seeds guarantee to find all alignments with one non-BS-mismatch. To guarantee to find all alignments with one non-BS-mismatch in each mate of a pair, the mapping runs in rounds. There is a total of ten pairs of seeds for paired-end alignment of reads with BRAT (one pair for zero non-BS-mismatches: both seeds are the first seeds in the sets below; the rest of pairs are the combinations of the three seeds, each of which is chosen from the rest of the seeds below).

```
00000000000000000000000000000001111111111111111111111111111111
00000000000000000000000000000000000000000001111111111111111111
00000000000000000000000000000001111111111111111110000000000
00000000000000000000000000000001111111111000000000001111111111
```