USING BRAT-BW-2.0.1

BRAT-bw is a tool for BS-seq reads mapping, i.e. mapping of bisulfite-treated sequenced reads. BRAT-bw is a part of BRAT's suit. Therefore, input and output formats for BRAT-bw are the same as for BRAT. BRAT-bw supports two distinct bisulfite libraries: type I yields sequenced reads that are bisulfite-converted versions of *two* original genomic strands (Lister *et al.*, 2009); type II produces reads that correspond to *four* possible strands, as a byproduct of PCR steps (Cokus *et al.*, 2008). BRAT-bw supports single and paired-end reads, guarantees to find all matches with up to 1 mismatch in the first 32…64bp, has limitation on the minimum read length (32bp), and no limitation on the maximum read length. The number of references is unlimited with total number of base-pairs limited to 2^{32}. BRAT-bw currently does not support insertions or deletions. Feature comparison of BRAT-bw and other current tools that use Farragina-Manzini (FM) index based on Burrows-Wheeler transform for short reads mapping is provided in Table 1 (features of brat-large, the mapping tool for large genomes of BRAT, are provided for comparison too).

Table 1. Feature comparison of BRAT-bw, BRAT-large, Bismark and BS Seeker (as of on March, 2012)

Feature	**BRAT-bw**	**BRAT-large**	**Bismark**	**BS-seeker**
Number of FM-instances (typeI/typeII bisulfite libraries)	2	NA	4	2/4
Paired-end (PE) support	Yes	Yes	Yes	No
Variable read length	Yes	Yes	Yes	Yes*
Adjustable insert size (PE)	Yes	Yes	Yes	NA
Uses basecall qualities for FastQ mapping	No	No	Yes	No
Supports typeI/typeII bisulfite libraries	Yes/Yes	Yes/No	Yes/Yes	Yes/Yes
Number of mismatches per read	Unlimited	Unlimited	Unlimited	3
Number of mismatches in a seed (bowtie-1/bowtie-2 if applicable)	1	0	3/1	3/NA
Supports indels	No	No	Yes**	No

* user has to provide the maximum read length with option *e*
** with bowtie-2 (we did not test this feature)

BRAT-bw takes 6.4GB to map reads to a human genome (hg18 was tested).
BRAT-bw uses the Burrows-Wheeler indexing together with the opportunistic data structure proposed by Ferragina and Manzini for exact-search of a query. To use BRAT-bw, one has to build two indexes with the following two commands:

```
./build_bw -P path_to_index -G 1 -r file_with_references.txt
./build_bw -P path_to_index -G 2 -r file_with_references.txt
```

Option ***P*** provides absolute path to the directory with the resulting index, option ***G*** denotes the index to be built (see below), and option ***r*** provides a file containing the names of FASTA files, each of which is with a single reference.

NOTE: a user must to build two indexes, one with option *G* set to 1 and another with option *G* set to 2, and path provided with option *P* must be the same for both commands.

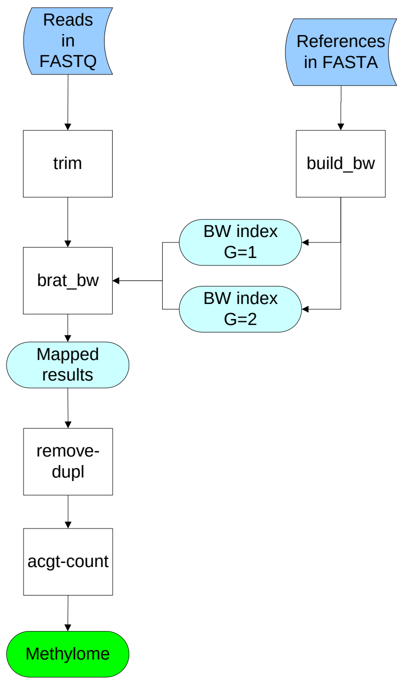
This builds two indexes on positive genomic strand: in the first index all Cs are converted to Ts and in the second all Gs are converted to As. An additional option ***S*** with ***build_bw*** sets the total number of explicitly stored genomic positions (***S***= 1, 2, 4, 8, 16 or 32, with default value of 16: each 16-th genomic position is stored). Setting ***S*** to 2 uses 16GB of memory on human genome, hg18, and maps reads 2 times faster than with ***S*** = 16. It takes about 6 hours to build genome indexes (using two processors: one for each index and building indexes simultaneously). Maximum space required for building an index is 3.5GB.

Once the indexes are built, a user re-uses these indexes to map reads (as with Bowtie).

BRAT-bw supports two distinct bisulfite libraries: type-1, containing bisulfite-converted original genomic strands; and type-2, containing four strands by-product of PCR (use option **-W** for four strands)

1 ANALYSIS PIPELINE

Currently BRAT-bw includes the following tools: **build_bw**, **brat_bw**, **acgt-count**, **trim**, **remove-dupl** and **convert_to_sam**. The flow of the analysis pipeline is given in the figure below.



First, tool **trim** takes reads in FASTQ format, trims low-quality bases from both ends as well as Ns and outputs reads in raw reads format that are accepted by **brat_bw**. Two BW-indexes are built using **build_bw**. After indexes have been built and after using **trim**, **brat_bw** is used to map the reads to the reference genome. Next, mapped results are used as input for **remove-dupl** that removes copy-duplicates keeping only a randomly chosen one, where copy-duplicates are the reads that are mapped to the same start position in the reference. Finally **acgt-count** processes all the mapped results to produce a methylome, a map with methylation status of each cytocine. Tool **convert_to_sam** converts BRAT's output format to SAM format.

Starting with version 2.0.1, the output of singled-end mapped reads includes reads as given in the input with the following exception. Reads that were sequenced from strands complementary to the original genomic strands are taken reverse complement of and the sign of mapping is changed. In other words, in the output, only reads from two original genomic strands are present (even for four-strands mapping with option **W**). This simplifies the followed up analysis. Option -2 for singled-end reads with programs acgt-count, remove-dupl is deprecated. To ensure correct methylation level count, users should follow all steps of the pipeline and do not add their own steps.

2 COMMANDS AND INPUT

To uncompress run:

```
tar zxvf brat_bw-2.0.*.tar.gz
```

To build:

```
cd brat-2.0.*
make
```

This will create executable programs: build_bw, brat_bw, acgt-count, trim, remove-dupl and convert_to_sam.

Input format of the reads for BRAT-bw is raw reads:

- Read < string >: a read after using trim;

- Start <int>: the number of bases trimmed at the beginning of the original read;
- End <int>: the number of bases trimmed at the end of the original read.

To convert reads from FASTQ format to raw reads, one should run trim. If a user does not wish to trim reads' low quality score bases, then he/she should omit the option for the base quality score threshold: the default threshold equals to zero, so all reads will be in the output without change in lengths (except for reads having Ns at the ends). If reads have Ns at the ends, trim trims Ns at the ends and outputs only those reads whose length after trimming is greater or equal to 24 bases.

A command to run **trim**

This program trims low-quality bases (lower than a threshold given with option -*q*) and Ns from each end of a read: bases are trimmed one at a time from both ends of a read until a base with quality score greater or equal than *q* is encountered (similarly, all consecutive Ns from both ends of a read are trimmed). This tool outputs only those reads whose length is at least 24 after trimming and that have at most *m* internal Ns: the number of allowable internal Ns is set by option -*m*.

To trim single-end reads in the file *reads.fastq* in FASTQ format and output trimmed raw reads into a file with name *prefix_reads1.txt*, run the command:

```
./trim -s reads.fastq -P prefix -q 20 -L 64 -m 2
```

This will trim bases whose base quality scores are lower than 20 from the ends of reads. The option *L* specifies the smallest value of the range of base quality scores in ASCII representation (please see Commands Options for details). To learn more about Phred scores, please visit http://www.phrap.com/phred/. Option -*m* allows each read having at most 2 internal *N*s. Option -P provides prefix to the output file names (it might contain a path for an output file: -P /home/directory/prefix).

If the user does not wish to trim ends with low base quality scores, the -q option is not specified. For single-end reads, there is a single output file with trimmed reads.

To trim paired-end reads in the files *reads1.fastq* and *reads2.fastq* in FASTQ format, run the command:

```
./trim -1 reads1.fastq -2 reads2.fastq -P prefix -q 20 -L 64 -m 2
```

Here we assume, that *reads1.fastq* contains sequenced 5` mates, and *reads2.fastq* contains sequenced 3` mates.

The output will be in four files with raw reads: *prefix_reads1.txt*, *prefix_reads2.txt*, *prefix_mates1.txt* and *prefix_mates2.txt*. To further map paired-end reads, use *prefix_reads1.txt* and *prefix_reads2.txt* as input files for paired-end mapping with brat or brat-large. The file *prefix_mates1.txt* contains reads from the file *reads1.fastq* whose mates have shorter length than 24 bases after trimming. Similarly, the file *prefix_mates2.txt* contains reads from the file *reads2.fastq* whose mates are shorter than 24 bases. The user can further map these files, *prefix_mates1.txt* and *prefix_mates2.txt*, as single-end reads: for BS-mapping of the reads in *prefix_mates2.txt*, the user must specify **-A** option for mapping to work correctly (the same is true if a user wishes to map the reads in *prefix_reads2.txt* as single reads).

Additional output files are *prefix_pair1.fastq*, *prefix_pair2.fastq*, *prefix_mates1.fastq* and *prefix_mates2.fastq*. These files have the same reads as do files *prefix_reads1.txt*, *prefix_reads2.txt*, *prefix_mates1.txt* and *prefix_mates2.txt* respectively, except the files *prefix_pair1.fastq*, *prefix_pair2.fastq*, *prefix_mates1.fastq* and *prefix_mates2.fastq* are in FASTQ format. NOTE: current version of BRAT and BRAT-large do NOT support FASTQ format. These additional files are for users to track original reads' names and corresponding base quality scores.

Commands to run **brat**_bw

BRAT-bw maps raw reads (output from trim) using pre-built indexes (with **build_bw**). BRAT-bw accepts an absolute path to the indexes previously built with build_bw. To map bisulfite single-end reads, run either of the commands:

```
./brat_bw -P path_to_index  -s  prefix_reads1.txt -o output_results.txt   [Options]

./brat_bw  -P path_to_index  -s  prefix_reads2.txt -o output_results.txt  -A   [Options]
```

Option **P** accepts absolute path to the pre-built with **build_bw** indexes. The file *output_results.txt* contains the results of the mapping: only uniquely mapped reads are in this file. The option -bs of BRAT is deprecated. BRAT-bw does not support normal reads mapping:

To map bisulfite paired-end reads, run the following commands:

```
./brat_bw  -P path_to_index -1  prefix_reads1.txt -2 prefix_reads2.txt -pe -o output_results.txt  [Options]
```

The option -pe specifies paired-end mapping. The results of the mapping will be in *output_results.txt*. BRAT-bw does not have in the output mates/pairs if a pair could not be mapped because of the wrong insert size, or wrong mates' orientation, or mates mapped to different chromosomes. However, if one mate is mapped ambiguously, and another is unique, then the uniquely mapped mates are provided in the output file *output_results.txt.**single_mates*** (also, if one mate is unmapped, but the other is mapped uniquely, then the mapped mates are in this files). If users wish to process this **.single_mates* file, they should submit this file to acgt-count with option -s (see below how to run **acgt-count**).

To map single-end or paired-end reads containing all four strands, by-product of PCR, use option ***W***. Note, that BRAT-bw takes care of counting methylation level correctly, and users should not use any extra steps besides those included in BRAT-bw pipeline.

Commands Options:

-A specifies 3`mates (in our examples above, either of *prefix_reads2.txt* or *prefix_mates2.txt* files must be used with this option). If the user does not specify this option, and provides either of the files, *prefix_mates2.txt* or *prefix_reads2.txt*, as input reads for single-end mapping, the mapping will NOT be correct;
-W specifies mapping reads containing all four strands, by-product PCR. Uses the same space (RAM) as mapping reads containing two bisulfite converted original strands.
-s <single-end reads file>: to specify the file with input reads for single-end reads mapping;
-1 <paired-end reads file>: to specify the file with 5` mates for paired-end reads mapping (in our example, *prefix_reads1.txt*). This option is also used with acgt-count;
-2 <paired-end reads file>: to specify the file with 3` mates for paired-end reads mapping (in our example above, *prefix_reads2.txt*); This option is also used with acgt-count;
-pe to specify paired-end reads mapping (default is false, *i.e*. single-end mapping);
-i <positive integer>: to specify minimum insert size for paired-end mapping, the minimum distance allowed between the leftmost ends of the mapped mates on forward strand (default is 100);
-a <positive integer>: to specify maximum insert size for paired-end mapping, the maximum distance allowed between the leftmost ends of the mapped mates on forward strand (default is 300);
-o <string>: to specify the file with the results of mapping;
-m <integer>: the maximum number of non-BS-mismatches allowed by a user (default is 0).
-f <integer>: the number of the first bases of a read, where the restriction on the number of non-BS-mismatches applies: for BRAT-bw, only one non-BS-mismatch is allowed in the first <integer> bases. This option is deprecated: the program sets the best-performing values dependent on the read length (this is convenient since BRAT-bw accepts reads of variable length).
-F <int>: When set to 1, it enforces mapping with 1 mismatch in the first ***f*** bases (option above). In this mapping mode, each base within ***f*** first bases (option above) is substituted with 2 other possible bases. For short reads, to find mapping with ***m*** mismatches and 1 mism in the first ***f*** bases, set -F to 1, for longer reads it is more time efficient to keep a default value, 0 (this will not affect mapping accuracy with longer reads) (default 0).
-P <directory name> Absolute path to the pre-built index.
-K <int> for longer reads: maximum number of shifts in multi-seed mapping (default is 10)
-C sets mapping mode, when C in a read is ALLOWED to be mapped to T in a genome without punishment (without this option C in a read to T in a genome is considered as a mismatch);
-L <integer>: the smallest value of the range of base quality scores in ASCII representation (default is 33).

The table below gives examples of different quality scores and their range in ASCII representation (from Wikipedia). The option ***L*** uses the values in the "Smallest Value in ASCII representation" column.

Type	Smallest Score	Largest Score	Smallest Value in ASCII representation	Largest Value in ASCII representation
Phred quality score (Sanger format)	0	93	33	126
Solexa/Illumina, 1.0	-5	62	59	126
Soloexa/Illumina, 1.3+	0	62	64	126

-B: specifies the second option for output with acgt-count (please read **Output format for acgt-count**).

A command to run **remove-dupl**

This program processes the mapping results and removes copy-duplicates: it outputs all reads that are mapped to a unique genomic location and only a randomly chosen one out of copy-duplicates (the reads mapped to the same location).

```
./remove-dupl -r references_names.txt -p pairs_results.txt -1 single_results.txt
```

NOTE: the file *pairs_results.txt* does not contain the actual results, it contains the names of the files with the results for paired-end mapping, and similarly, *single_results.txt* file contains the names of the files with the actual results for single-end mapping.
For example, the content of *pairs_results.txt* is:
output_pairs_lane1.txt
output_pairs_lane2.txt

and the content of *single_results_mates1.txt* is:
output_singles_mates1_lane1.txt
output_singles_mates1_lane2.txt
output_singles_mates2_lane1.txt
output_singles_mates2_lane2.txt

The output of remove-dupl are the files with the same names as before with additional extension "*.nodupl*":
output_pairs_lane1.txt.nodupl
output_pairs_lane2.txt.nodupl
output_singles_mates1_lane1.txt.nodupl
output_singles_mates1_lane2.txt.nodupl
output_singles_mates2_lane1.txt.nodupl
output_singles_mates2_lane2.txt.nodupl

For single-end mapping, run the following command:

```
./remove-dupl -r references_names.txt -s single_results.txt
```

The file *single_results.txt* contains the names of the files with mapping results.

NOTE: that the output files with extension "*.nodupl*" are opened with C++ "*app*" option (opens a file and appends output to the file's content). This means that once you have run remove-dupl, you will have ".nodupl" files, and if you want for some reason to run remove-dupl on the same files (and possibly some additional files), you need to remove ".nodupl" files for the corresponding files first and only then re-run remove-dupl. For example, as in the example above, you run remove-dupl and obtain ".nodupl" files:
output_pairs_lane1.txt.nodupl
output_pairs_lane2.txt.nodupl
output_singles_mates1_lane1.txt.nodupl
output_singles_mates1_lane2.txt.nodupl
output_singles_mates2_lane1.txt.nodupl
output_singles_mates2_lane2.txt.nodupl

Then you wish to add *output_pairs_lane3.txt* to *pairs_results.txt:*
output_pairs_lane1.txt
output_pairs_lane2.txt
output_pairs_lane3.txt
and to re-run remove-dupl on all files.

Make sure your delete existent files with extension ".nodupl":
rm output_pairs_lane1.txt.nodupl
rm output_pairs_lane2.txt.nodupl
rm output_singles_mates1_lane1.txt.nodupl
rm output_singles_mates1_lane2.txt.nodupl
rm output_singles_mates2_lane1.txt.nodupl
rm output_singles_mates2_lane2.txt.nodupl
and only then run remove-dupl.

If you don't remove these files, you will have previous output plus new output (for example: if *output_pairs_lane1.txt.nodupl* had 100 lines, then re-running remove-dupl without removing this file will result in 200 lines).

A command to run **convert-to-sam**

This program converts BRAT format to SAM format.

```
./convert-to-sam -P prefix -p pairs_results.txt -1 single_results_mates1.txt
-2 single_results_mates2.txt -s single_results.txt
```

NOTE: the file *pairs_results.txt* does not contain the actual results, it contains the names of the files with the results for paired-end mapping, and similarly, *single_results*.txt* files contain the names of the files with the actual results for single-end mapping (see examples in "A command to run remove-dupl").

NOTE: Use options -1 and -2 only when mapping paired-end reads as singles (for 5' mates and 3' mates respectively), and option s if the original sequenced reads were single reads.

The output will be in two files with names *prefix_forw.sam* and *prefix_rev.sam* if option **P** is provided, or *mapped_to_forw.sam* and *mapped_to_rev.sam* if not. The reason behind separating the mapped reads into two separate files is the following. To distinguish between methylated and unmethylated cytosines in aligned reads, one has to look at C mapped to C and T mapped to C respectively if reads come from forward strand, and one has to look at G mapped to G and A mapped to G respectively if reads come from reverse strand. If one has a mixture of reads from forward and reverse strands, it would be impossible to track methylated and unmethylated cytosines when visualizing aligned reads.

Paired-end mapped mates mapped as pairs are both either in forward output file (*prefix_forw.sam*) if 5' mate (the first mate) is mapped to positive strand, or both in reverse output file (*prefix_rev.sam*). Reads from files with option **1** are in forward output file if they are mapped to positive strand, and in reverse output file if they are mapped to negative strand. Reads from files with option **2** are in forward output file if they are mapped to negative strand, and in reverse output file if they are mapped to positive strand. Single reads from files with option **s** are in forward output file if they are mapped to positive strand, and in reverse output file if to negative strand.

NOTE: that the output files with extension "*.sam*" are opened with C++ "*app*" option. Please see details how to be careful with re-running convert-to-sam in "A command to run remove-dupl" (since it is similar situation as with remove-dupl).

A command to run **acgt-count**

To count mapped As, Cs, Gs and Ts at each base of forward and reverse strands of the references, use acgt-count.

```
./acgt-count -r references_names.txt -P prefix -p pairs_results.txt -s single_results.txt
```

the file *references_names.txt* contains the names of FASTA files with the references, which are needed to calculate the sizes of the references. The output will be in two files per a reference: *prefix_forw_aReference_name* and *prefix_rev_aReference_name*. The option **-p** is to specify the results of paired-end mapping (if any), and **-s** is to specify the results of single-end mapping (if any). **NOTE: the file *pairs_results.txt* does not contain the actual results, it contains the names of the files with the results for paired-end mapping, and similarly, *single_results.txt* file contains the names of the files with the actual results for single-end mapping.** At least one of these options must be provided. The files whose names are listed in the files *pairs_results.txt* and *single_results.txt* must be in BRAT's output format.

To make this point clear, assume, a user ran brat on paired-end reads and had the output file with the results in *output_pairs_results.txt*; to run acgt-count, the user must store the name of this file in *pairs_results.txt* file and run acgt_count using *pairs_results.txt* (*i.e. pairs_results.txt* will have in this case one line, namely, *output_pairs_results.txt*). The command for this example is:

```
./acgt-count -r references_names.txt -P prefix -p pairs_results.txt
```

Please note that if a user has paired-end reads (files with mates 1 and mates 2) and wishes to map the mates as single-end reads, the resulting files with mapped output don't need to be separated as with BRAT, i.e. BRAT-bw handles this during mapping, so just put names of the resulting files into *files_with_singles.txt* and use option -s:

```
./acgt-count -r references_names.txt -P prefix -s files_with_singles.txt
```

To produce a more concise output (outputs only cytocines), use option -B (recommended for large genomes):

```
./acgt-count -r references_names.txt -P prefix -p pairs_results.txt -s single_results.txt -B
```

This program takes care of overlapping mates: if two mates of a pair overlap, then ACGT-count is done only for one mate in the overlapped region. This program also takes care of producing un-biased ACGT-count from mates 2 (3` mates). Please see Details on ACGT-count section for details.

3 OUTPUT FORMAT

Output format for **brat**_bw for single-end mapping

- Read id < integer >: a consecutive number of a read in the reads input file that starts with 0;
- Read 1 < string >: the read given as in the input file (*prefix_reads1.txt*);
- Reference name < string >: a name of a reference to which the read is mapped (the first word following ">" in a FASTA file);
- Strand "+" if the read is mapped to forward strand, and "-" if the read is mapped to reverse strand;
- Position < integer >: position within the reference starting with 0, where the read is mapped (the leftmost position on forward strand).
- The number of non-BS-mismatches <int>
- Original position <integer>: position within the reference starting with 0, where the *original* read is mapped (the leftmost position on forward strand), where original read is the read before its ends have been trimmed. For example, if the number of trimmed bases at the beginning of a read is 2, and the read is mapped to positive strand at the position 10, then original position is 10 – 2 = 8. If the number of trimmed bases at the end of a read was 3 and the reverse-complement of the read is mapped to the position 10 on positive strand, then original position = position – 3 = 10 – 3 = 7. The original positions are used to identify copy-duplicates.

Output format for **brat** and **brat-large** for paired-end mapping

- Read id < integer >: a consecutive number of a read in the reads input file that starts with 0;
- Read 1 < string >: the first mate of a pair given as in the input file (*prefix_reads1.txt*);
- Read 2 < string >: the second mate of a pair given as in the input file (*prefix_reads2.txt*);
- Reference name < string >: a name of the reference to which the pair is mapped (the first word following ">" in a FASTA file);
- Strand "+" if 5` mate (from *prefix_reads1.txt*) is mapped to forward strand (consecutively, 3` mate, from *prefix_reads2.txt*, is mapped to reverse strand), and "-" if the 5` mate is mapped to reverse strand (and 3` mate to forward strand);
- Position 1 < integer >: position within the reference starting with 0, where 5` mate is mapped (the leftmost position on forward strand);
- Position 2 < integer >: position within the reference starting with 0, where 3` mate is mapped (the leftmost position on forward strand).
- The number of non-BS-mismatches < integer >: the number of mismatches in the alignment for 5` mate
- The number of non-BS-mismatches < integer >: the number of mismatches in the alignment for 3` mate
- Original position 1 <integer>: original position for 5` mate (see definition of original position above)
- Original position 2 <integer>: original position for 3` mate (see definition of original position above)

Output format for **acgt-count**

Starting with version *brat-1.1.17*, there are two choices for output format.

The first choice: The number of output files will be double the number of input references: two for each reference listed in *references_names.txt* file (one file for forward strand and the other for reverse strand). In each file, there are *M* lines, where *M* is the size of a corresponding reference in base pairs. Each line corresponds to a base of a strand and contains counts for As, Cs, Gs and Ts at that base for all mapped reads (*i.e.* there are four integers per line: from left to right for As, Cs, Gs and Ts).

For the reverse strand, the counts of As, Cs, Gs and Ts are given for the reads that are mapped to the reverse strand, but the counts are obtained by aligning the reverse-complements of these reads with the forward strand.

Following is an example to illustrate this point.

Let a read ACCGTT be mapped to a reverse strand at position *i*, then the corresponding forward strand starting at position *i* is AACGGT, and the counts for the reverse strand at positions *i …i+5* from this read are incremented for the following nucleotides: *i(A), i+1(A), i+2(C), i+3(G), i+4(G)* and *i+5(T).*

The second choice: If a user provides option *-B*:

```
./acgt-count -r references_names.txt -P prefix -p pairs_results.txt -s single_results.txt -B
```

then the output is in two files: one file for positive strand and another for negative strand (output files will contain words "forw" and "rev" to distinguish between strands). Each line in the output corresponds to a base in the genome that is either a cytocine on positive strand or cytocine on negative strand (given in separate files). Output format:

chrom, start, end, total, methylation_level, strand

where *chrom* is the reference name, *start* and *end* are positions in the genome (Note: base count in a reference starts with 0), *total* takes one of the values: CHH:X, CHG:X or CpG:X, where X is the sum of counts of Cs and Ts mapped to this base, methylation level is calculated as the number of Cs over the *total* (methylation level = count_C/(count_C + count_T)). CHH, CHG and CpG describe the sequence content following C: if two consecutive bases that follow C are not G, then *total* = CHH:X; if the first consecutive base following C is non-G and the second consecutive is G, then *total* = CHG:X; and finally, if G follows C (we have CG di-nucleotide), then *total* = CpG:X.

Output format for trim.

The tool trim accepts FASTQ files with reads/pairs as input, trims the ends of the reads whose base quality scores are lower than the user specified threshold or whose ends are Ns. The output for single reads is a single file with reads whose lengths might be different and whose lengths are greater than or equal to 24 bases. The output for pairs is four files: two for paired-end mapping, and two for single-end mapping. Trimming of paired-end reads produces two files with single reads: if one mate is shorter than the minimum length allowed, and the other's length is correct, then the mate with the correct length will be output into a corresponding file with single reads. Two files for single reads are necessary because BS-mapping for 5` and 3` mates is different. Each file contains a single line (raw reads format) with the following fields:

- Read < string >: a read after using trim;
- Start <int>: the number of bases trimmed at the beginning of the original read;
- End <int>: the number of bases trimmed at the end of the original read.

4 DETAILS ON *MULTI-SEED* MAPPING

BRAT-bw uses Burrows-Wheeler index to map reads. This index uses a concept of data structure that is a lexicographically sorted list of all possible suffixes of a given text (in our case given genome). With this approach, a read of length N is mapped base by base, and two values are calculated (we'll call them *sp* and *ep*) for each base. These values for a read base at *i*-th position denote start and end locations of the suffix [i…N] of the read within the lexicographically sorted list of all suffixes of the genome. Thus if sp < ep, then there exist more than one genomic suffixes starting with read's suffix [i…N] and hence genomic locations, where this suffix occurs, and if sp = ep, then this suffix is uniquely mapped within genome, but with sp > ep, there is no location within the genome, where the suffix [i…N] could be mapped.

BRAT-bw aligns longer reads using multi-seed approach. It is a well-known property that in a read of length N with k mismatches, there is at least one consecutive stretch of length N/(k+1) of the read without any mismatches. BRAT-bw makes K attempts (option -K) to align read from different locations within the read starting from the end of the read intending to find the region within the read that aligns perfectly. In the first attempt, BRAT-bw starts from the end of the read, aligns the read base by base until there is a unique match (suffix of the read or entire read aligns uniquely to the genome), or until sp > ep for some base at i-th position, i.e. there is no match for the suffix of the read starting at *i* (say, suffix READ[i…N]). Thus, READ[i…N] is the smallest suffix that does not align perfectly in a genome. If a suffix of the read aligns to one or more genomic locations, BRAT-bw makes additional full-length check-alignment and disregards alignments with the number of mismatches greater than the threshold provided with option -m. If a suffix READ[i…N] of the read could not being mapped exactly, but is longer than 31bp, then BRAT-bw makes additional full-length check-mapping on the last valid genomic location, to which a shorter suffix was mapped exactly, i.e. if READ[i…N] did not map exactly then BRAT-bw performs full-length checks for genomic locations to which READ[i+1…N] mapped exactly. The number of attempts is controlled by option -K. Next attempt is performed starting from D bases to the left of the previous attempt (see Figure below), and the process repeated until all attempts have been completed. Then BRAT-bw attempts perfectly align the first ***f*** (dynamically controlled by program dependent on the read length) bases in order to find a match or to identify ambiguous reads. If option ***F*** is set to 1, then BRAT-bw substitutes each base of the first ***f*** bases with the other two possible bases, and for each substitutions, aligns the first ***f*** bases (containing a substitution), if there is a match, it performs additional check-alignment on the entire read length.

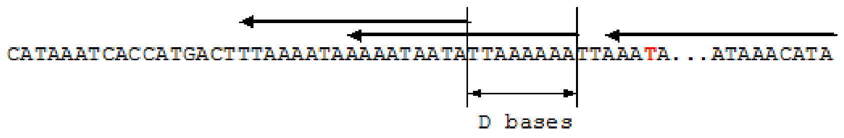

CATAAATCACCATGACTTTAAAATAAAAATAATATTAAAAAATTAAATA...ATAAACATA
D bases

Figure above shows attempts of read alignment by arrows above the read. We showed a sequenced error (or a mismatch) in red. All attempts that cover this mismatch will either fail or result in wrong alignments, but other attempts that have stretches of the read without mismatches will identify correct alignment(s). D determines the interval in bases between consecutive alignments (8 bases on Figure above). D is set by BRAT-bw dynamically dependent on read length, and for very long reads (longer than 200bp), users can set option D.

5 DETAILS ON *ACGT*-COUNT

BRAT-bw supports mapping of reads from two distinct bisulfite libraries: type-1 sequenced reads are bisulfite conversions of two original genomic strands, and type-2 with reads containing four strands, by-product of PCR.

BRAT-bw builds two indexes on original forward (or positive) genomic strand: in the first index Cs are converted to Ts and in the second Gs are converted to As. Mapping is done in two (or four) steps for type-1 (and type-2) library.

(1) An original read with Cs converted to Ts is mapped to the first index (which is the original forward genomic strand with Cs converted to Ts). This corresponds to the alignment of reads from PCR1+ strand on Figure 1.
(2) We take a reverse-complement of an original read, then change Gs to As in the reverse-complement, and then map the resulting read to the second index (the original forward genomic strand with Gs converted to As). This corresponds to the alignment of reads from PCR2- strand on Figure 1.

For the library by Cokus et al., we also align reads only to FORWARD strand of the original genome (using the same two indexes with Cs converted to Ts and another one with Gs converted to As).

For Cokus's library, in addition to two alignments described above, we perform two additional alignments:

(3) We convert Gs to As in an original read and align it to the second index (which is original genomic forward strand with Gs converted to As). This corresponds to the alignment of reads from PCR2+ strand on Figure 1.
(4) Next, we convert Cs to Ts in the reverse-complement of an original read and align it to the first index. This corresponds to the alignment of reads from PCR1- strand on Figure 1.

For any library, when there is a match for at least two out of two/four (for typeI/typeII BS-libraries) possible alignments for a read, then this read is considered to be ambiguous and is not reported in the output.

Methylation level is estimated for two strands separately (Figure 2): for cytocines on forward strand we count the number of Cs and Ts in reads mapped to Cs in the genome using alignments to the first index; and for cytocines on reverse strand we count the number of Gs and As in reads mapped to Gs in the genome using alignments with the second index.

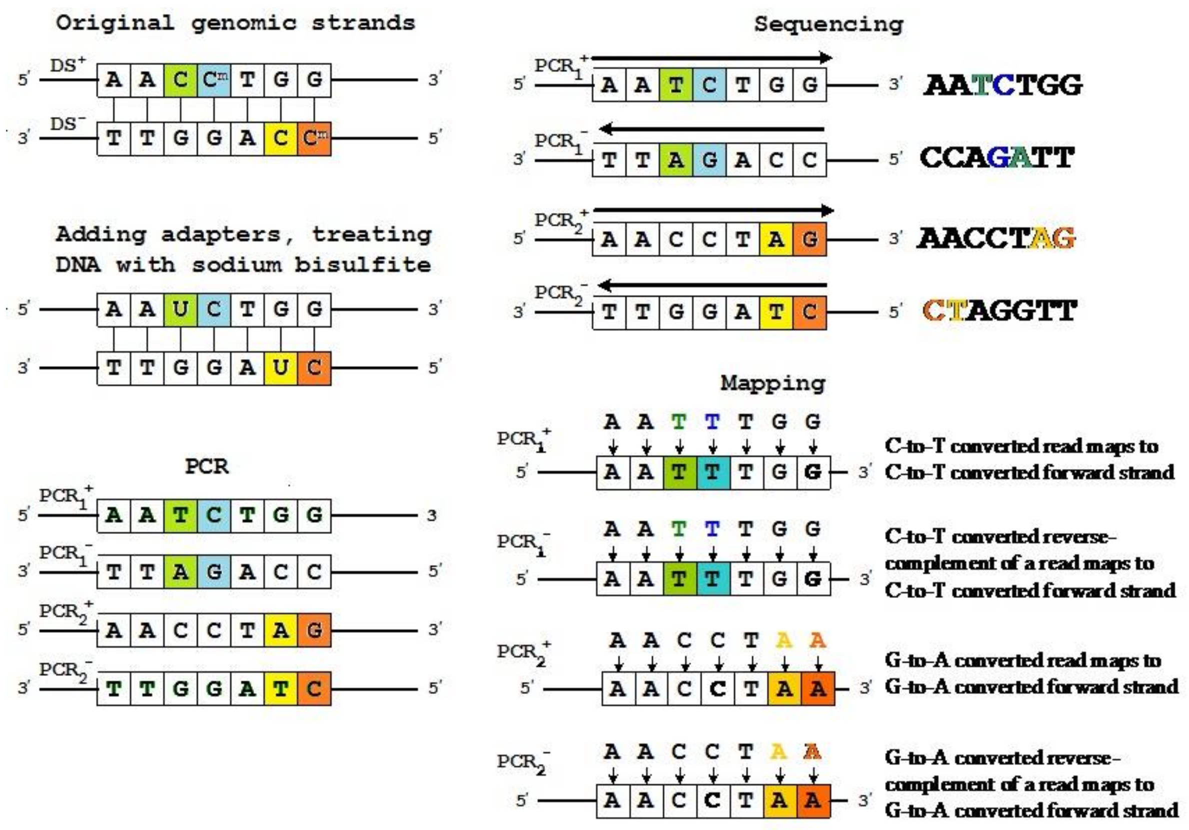

Original genomic strands
DS+
DS-
Adding adapters, treating DNA with sodium bisulfite
PCR
Sequencing
AATCTGG
CCAGATT
AACCTAG
CTAGGTT
Mapping
C-to-T converted read maps to C-to-T converted forward strand
C-to-T converted reverse-complement of a read maps to C-to-T converted forward strand
G-to-A converted read maps to G-to-A converted forward strand
G-to-A converted reverse-complement of a read maps to G-to-A converted forward strand

Figure 1. PCR_1^+ and PCR_2^- strands correspond to the original positive and negative genomic strands respectively (BS library by Lister et al. has only these two strands), and PCR_1^- and PCR_2^+ strands are reverse complements of PCR_1^+ and PCR_2^- respectively. Methylated and unmethylated cytosines on positive (forward) strand are shown in blue and green respectively. Methylated and unmethylated cytosines on negative strand are shown in orange and yellow respectively. In similar colors are shown converted versions (converted with sodium busilfite, further with PCR, and during alignment).

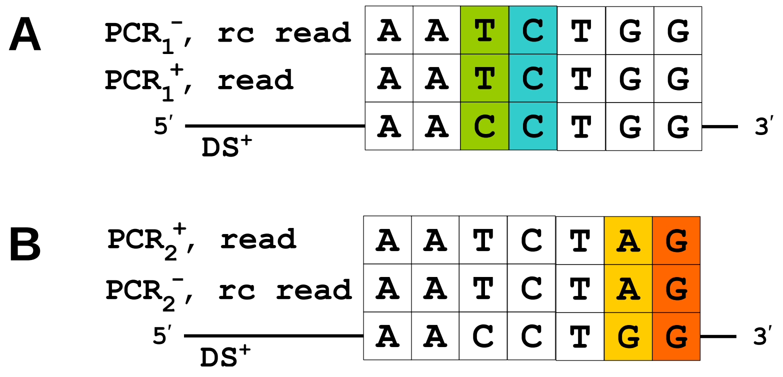


Figure 2. Methylation level is estimated for two strands separately. (A) For cytosines on forward strand we count the number of Cs and Ts in reads (or their reverse-complements) mapped to Cs in the genome using alignments to the first index. Here methylated cytosines are shown in blue and unmethylated in green. (B) For cytosines on reverse strand we count the number of Gs and As in reads (or reverse-complement of reads) mapped to Gs in the genome using alignments with the second index. Methylated cytosine on the reverse strand is shown in orange and unmethylated in yellow.

References

Lister, R. *et al.* (2009) Human DNA methylomes at base resolution show widespread epigenomic differences. *Nature*, 462, 315–322.

Cokus, S.J. *et al.* (2008) Shotgun bisulphite sequencing of the Arabidopsis genome reveals DNA methylation patterning. *Nature* ***452***, 215-219.

Ferragina, P. and Manzini, G. (2000) Opportunistic data structures with applications. Proceedings of IEEE Foundation of Computer Science.

Chen,P.Y. et al. (2010) BS Seeker: precise mapping for bisulfite sequencing. *BMC Bioinformatics*, 11, 203.

Krueger, F. and Andrews, S. (2011) Bismark: a flexible aligner and methylation caller for Bisulfite-Seq applications. *Bioinformatics,* 27, 1571-1572.

Burrows, M. and Wheeler, D. (1994) A Block Sorting Lossless Data Compression Algorithm. Technical Report #124, Digital Equipment Corporation.