USING BRAT-nova

BRAT-nova is a tool for BS-seq reads mapping, i.e. mapping of bisulfite-treated sequenced reads. BRAT-nova is a part of a suit for comprehensive analysis of BS-seq data. BRAT-nova currently supports bisulfite library of reads that are bisulfite-converted versions of *two* original genomic strands (Lister *et al.*, 2009). BRAT-nova supports single and paired-end reads, has limitation on the minimum read length (50bp), 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-nova supports one insertion or deletion in the middle of a read and trims the ends of the reads containing insertions/deletions/multiple mismatches. Feature comparison of BRAT-nova and other current tools that use Farragina-Manzini (FM) index based on Burrows-Wheeler transform for short reads mapping is provided in Table 1.

Table 1. Feature comparison of BRAT-nova, BRAT-nova, Bismark and BS-Seeker2 (as of on March, 2012)

Feature	**BRAT-nova**	**BRAT-bw**	**Bismark**	**BS-Seeker2**
Number of FM-instances	2	2	2/4	2/4
Paired-end (PE) support	Yes	Yes	Yes	Yes
Variable read length	Yes	Yes	Yes	Yes
Adjustable insert size (PE)	Yes	Yes	Yes	Yes
Supports typeI/typeII bisulfite libraries	Yes/No	Yes/Yes	Yes/Yes	Yes/Yes
Number of mismatches per read	Unlimited	Unlimited	Unlimited	Unlimited
Supports indels	No	Yes	Yes	Yes

BRAT-nova takes 6.0 GB to map reads to a human genome (human genome GRCh38 was tested, 24 chromosomes). BRAT-nova 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-nova, one has to build an index with the following command:

```
./build_bw -P path_to_index  -r file_with_references.txt
```

Option ***P*** provides absolute path to the directory with the resulting index and option ***r*** provides a file containing the names of FASTA files, each of which is with a single reference.

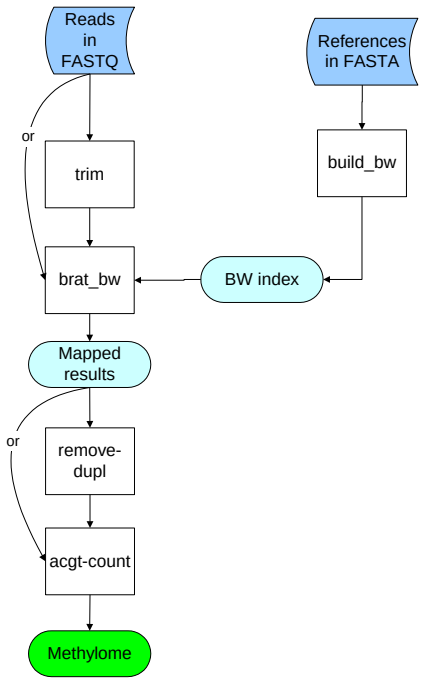
An additional option ***S*** with ***build_bw*** sets the total number of explicitly stored genomic positions (***S***= 2, 4, 8, 16 or 32, with default value of 16: each 16-th genomic position is stored). The smaller S, the larger RAM is used. It takes about 8-10 hours to build a genome index on human genome. Maximum space required for building an index is 3.8 GB.

Once the index is built, a user can re-use the index to map reads (as with Bowtie).

BRAT-nova currently supports only type-1 bisulfite library out of two distinct bisulfite libraries: type-1, containing bisulfite-converted original genomic strands; and type-2, containing four strands by-product of PCR.

1 ANALYSIS PIPELINE

Currently BRAT-nova includes the following tools: **build_bw**, **brat_bw**, **acgt-count**, **trim**, and **remove-dupl**. 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 FASTQ format that are accepted by **brat_bw**. One BW-index is built using **build_bw**. After index has been built and after using **trim**, **brat_bw** is used to map the reads to the reference genome. The output of brat_bw are mapped reads in **SAM** format. 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.

The results of mapping paired-end reads are given in the same output file.

2 COMMANDS AND INPUT

To uncompress run:

```
tar zxvf brat_nova.tar.gz
```

To build:

```
cd brat_nova
make
```

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

Input format of the reads for BRAT-nova is FASTQ.

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 50 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 table below). 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). For single reads, the output reads will be in *prefix.fastq* file.

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 in FASTQ format: *prefix_pair1.fastq*, *prefix_pair2.fastq*, *prefix_mates1.fastq* and *prefix_mates2.fastq*. To further map paired-end reads, use *prefix_pair1.fastq* and *prefix_pair2.fastq* as input files for paired-end mapping with brat_bw. The file *prefix_mates1.fastq* contains reads from the file *reads1.fastq* whose mates have shorter length than 50 bases after trimming. Similarly, the file *prefix_mates2.fastq* contains reads from the file *reads2.fastq* whose mates are shorter than 50 bases. The user can further map these files, *prefix_mates1.fastq* and *prefix_mates2.fastq*, as single-end reads: for BS-mapping of the reads in *prefix_mates2.fastq*, 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.fastq* as single reads).

-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

Commands to run brat_bw

BRAT-nova maps reads in FASTQ format using pre-built index (with **build_bw**). BRAT-nova 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.fastq -o output_results.sam   [Options]

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

Option **P** accepts absolute path to the pre-built with **build_bw** index. The file *output_results.sam* contains the results of the mapping: only uniquely mapped reads are in this file. BRAT-nova 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.fastq -2 prefix_reads2.fastq -pe -o output_results.sam  [Options]
```

The option -pe specifies paired-end mapping. The results of the mapping will be in *output_results.sam* in SAM format. BRAT-nova 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 (also, if one mate is unmapped, but the other is mapped uniquely, then the mapped mates are in the output files).

Command Options with brat_bw:

-A specifies 3`mates (in our examples above, either of *prefix_reads2.fastq* or *prefix_mates2.fastq* files must be used with this option when mapped as single reads). If the user does not specify this option, and provides either of the files, *prefix_mates2.fastq* or *prefix_reads2.fastq*, as input reads for single-end mapping, the mapping will NOT be correct;
-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.fastq*). 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.fastq*); 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
-**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 <int> alignment length (in percentage of a read length; please see below); default is 30;
-q <int> alignment score (in percentage of a read length; please see below); default is 90;
-L is local alignment allowed (default is *true*);
-G are indels allowed (default is *true*);
-E enforce only entire-length alignment (default is *false*);
-m <int> mismatch penalty (default is -3);
-go <int> gap opening penalty (default is -5);
-ge <int> gap extension penalty (default is -3);
-t <int> threshold on the total number of genomic positions, to which a seed from a read is exactly mapped: if the number of such positions exceeds ***t***, entire-length read mapping for such seed is not done (assumes that read is ambiguous); default is 1000;

To control the quality of an alingment, BRAT-nova uses two user-defined thresholds: one for an alignment length (l) and another for an alignment quality (q). The threshold for an alignment score is calculated as a function of the read length, r, using the formula *Score_Threshold* = $\left\lfloor \left(\frac{l}{100}\right) \cdot \left(\frac{q}{100}\right) \cdot r \right\rfloor$. For example, for a read of length 100bp, alignment length l = 0.9, and alignment quality q = 0.95, the threshold on the alignment score is 85. If a read is mapped with alignment quality less than 85, it is considered unmapped and the result is not reported. Moreover, the alignment quality q re-enforces the alignment to have q of bases matched perfectly (as a function of the aligned portion of the read). For our example, the read may be aligned with at most 5 mismatches using full-length alignment, or the contiguous 85bp of the read may be aligned with 0 mismatches using local alignment. To clarify, BRAT-nova uses the score threshold together with the quality alignment threshold to control high quality of the alignment of the aligned stretch of the read, where the stretch is contiguous. The alignment score is calculated using the following formula

$$Alignment\_Score = matches - (mismatches \cdot mismatch\_penalty) \\ -(gap\_opening + indel\_length \cdot gap\_extension)$$

where *mismatch_penalty*, *gap_opening* and *gap_extension* are user-defined.

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 -s mapped_reads.txt
```

NOTE: the file *mapped_reads.txt* does not contain the actual results, it contains the names of the files with the results for paired-end mapping or single-end mapping (or both).

For example, the content of *mapped_reads.txt* is:
output_pairs_lane1.sam
output_pairs_lane2.sam
output_singles_mates1_lane1.sam
output_singles_mates1_lane2.sam
output_singles_mates2_lane1.sam
output_singles_mates2_lane2.sam

The output of remove-dupl are the files with the same names as before with additional extension "*.nodupl*":

output_pairs_lane1.sam.nodupl
output_pairs_lane2.sam.nodupl
output_singles_mates1_lane1.sam.nodupl
output_singles_mates1_lane2.sam.nodupl
output_singles_mates2_lane1.sam.nodupl
output_singles_mates2_lane2.sam.nodupl

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.sam.nodupl
output_pairs_lane2.sam.nodupl
output_singles_mates1_lane1.sam.nodupl
output_singles_mates1_lane2.sam.nodupl
output_singles_mates2_lane1.sam.nodupl
output_singles_mates2_lane2.sam.nodupl

Then you wish to add *output_pairs_lane3.sam* to *mapped_reads.txt:*

output_pairs_lane1.sam
output_pairs_lane2.sam
output_pairs_lane3.sam
output_singles_mates1_lane1.sam
output_singles_mates1_lane2.sam
output_singles_mates2_lane1.sam
output_singles_mates2_lane2.sam

and to re-run remove-dupl on all files.

Make sure your delete existent files with extension ".nodupl":
rm output_pairs_lane1.sam.nodupl
rm output_pairs_lane2.sam.nodupl
rm output_singles_mates1_lane1.sam.nodupl
rm output_singles_mates1_lane2.sam.nodupl
rm output_singles_mates2_lane1.sam.nodupl
rm output_singles_mates2_lane2.sam.nodupl

and only then re-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 acgt-count

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

```
./acgt-count -r references_names.txt -P methylome.txt -s mapped_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 is in the file *methylome.txt* The option **-s** is to specify the results of mapping. **NOTE: the file *mapped_results.txt* does not contain the actual mapped reads, it contains the names of the files with the results for paired-end or single-end mapping.** The files whose names are listed in the file *mapped_results.txt* are in SAM format, the results of running ***brat_bw***.

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

```
./acgt-count -r references_names.txt -P methylome.txt -s mapped_results.txt
```

Please note that mapped_results.txt may contain the results for paired-end and single-end mapping:

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 is in SAM format.

https://samtools.github.io/hts-specs/SAMv1.pdf

Output format for acgt-count

The output is in a single file. 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 50 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. The output is in FASTQ format. If a read was trimmed, then the string with base quality scores was also trimmed accordingly. BRAT-nova adds additional information, the number of trimmed bases at 5'-end and the number of trimmed bases at 3'-end, for each read to the read id line: these numbers follow question marks. For example if a read id was ***@1948:1235:1800,*** and 10 bases were trimmed at 5'-end and 20 at 3'-end, then the new read id would be ***@1948:1235:1800?10?20***

4 DETAILS ON *MULTI-SEED* MAPPING

BRAT-nova 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-nova 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-nova makes K attempts (option -K) to align read from different locations within the read starting from the 5'end of the read intending to find the region within the read that aligns perfectly. In the first attempt, BRAT-nova starts from the 5'-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-nova makes additional full-length check-alignment and disregards alignments with low quality score. If a suffix READ[i…N] of the read could not being mapped exactly, but is longer than 31bp, then BRAT-nova 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-nova 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.

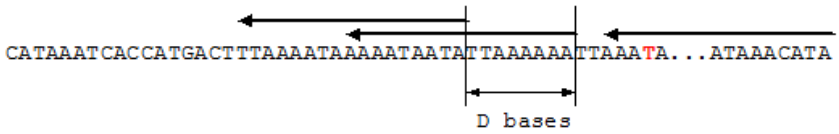


Figure above shows attempts of read alignment by arrows above the read. We showed a sequenced error (or a mismatch) in red. All attempts that cover this mismatch will either fail or result in wrong alignments, but other attempts that have stretches of the read without mismatches will identify correct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

5 DETAILS ON *ACGT*-COUNT

To build a single Burrows-Wheeler transform, BWT, and FM-index on a given reference genome G, first we replace all Cs with Ts and took the reverse of the resulting string, call the result S^R; for example, if G is CCATG, then S^R is GTATT. Then we took the reverse complement of G, replaced all Cs with Ts, and took the reverse of the resulting string, call the result S^C; for example, for G = CCATG, S^C = GGTAT. The BWT and FM-index are built on the concatenation of the strings S^C and S^R, i.e. on the string S^CS^R; for example, for G = CCATG, S^CS^R = GGTATGTATT. This allows aligning reads from 5'-ends that have less sequencing errors than 3'-ends of the reads.

An original read with Cs converted to Ts is mapped to the index. A single pass maps read simultaneously to the original reference and its reverse-complement.

Methylation level is estimated for two strands separately (Figure 2): we count the number of Cs and Ts in reads mapped to Cs in the genome.

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.