USING BRAT-1.1.16

1 SYSTEM AND SPACE REQUIREMENTS

In version 1.1.16, we added additional capability to BRAT: now it will map reads sequenced from four PCR-product strands. Please read details in Section 4, *Details on ACGT-count*.

In version 1.1.15, we fixed a bug in acgt-count and included two new tools: rev-compl and check-strands. Recently, we received *e*-mails from the users whose questions made us re-think the way we count ACGT-distribution of mapped reads for positive and negative strands. Please read a full account of these changes in section 4 *"Details on ACGT-count"* of the User Manual. Some users believe that the sequenced reads are produced by sequencing all four strands (two bisulfite-treated original genomic strands and their reverse-complements resulted from PCR for library amplification). In section 4, we describe how to use BRAT to check whether the reads are sequenced from two original strands as templates or from four PCR-product strands as templates for sequencing. Finally, we provide instructions on using BRAT in the case when the reads are sequenced from four strands as templates.

In version 1.1.14, we fixed a bug in tool acgt-count. If minimum insert size (option *i*) is set to be smaller than the minimum read length (in the set of input reads), then for paired-end mapping two mates could overlap each other. In previous versions, (A,C,G,T) count in the overlapping regions included nucleotide coverage from both mates. In this new version, nucleotide coverage in the overlapping region is counted only from one mate of a pair.

In version 1.1.13, we fixed bugs in acgt-count and brat-large-build (correspondingly, brat-large has been changed too). Please refer to Commands Options about how to use the option ***-P*** <directory name> with brat-large and brat-large-build.

In version 1.1.12, we added a new option, *L*, to use with trim to allow different base quality score types such as Phred or Solexa/Illumina.

We fixed a bug in trim in version 1.1.11, added a new option for trim, namely option *m*, and added additional output files for trim's output to track original reads' names and base quality scores.

In versions 1.1.8 and 1.1.9 we corrected some conditions that caused the programs to stop trying mapping reads. Now, more reads can be mapped when using BS-mapping allowing for non-BS-mismatches.

In version 1.1.7, we offered users two new options to use with BRAT-large: *P* and *S*. Option *S* gives the users control over the (space usage)-(mapping time) balance, and option *P* gives the users a choice of separating hashing of the genome from mapping reads to the genome. If users map reads to the same genome and wish to re-use the same hashing index, they now have a choice to pre-build the hashing index, save it on a hard disk and then re-use the index for mapping different reads. Mapping with *P* is done in two steps: first run the program brat-large-build and then run brat-large with *P* option.

BRAT stands for Bisulfite-treated Reads Analysis Tool. There are two versions of BRAT: BRAT and BRAT-large. Both programs run on 64-bit architecture under Linux/Unix operating system. Both versions work with reads of lengths 24 bases and above.

BRAT accepts up to 4.2G references with total size of references up to 4.2G base pairs. BRAT-large can work with up to 4.2G references with the size of the largest reference up to 4.2G base pairs.

BRAT works best with relatively small genomes because it uses significantly more space than BRAT-large. BRAT is faster than BRAT-large that maps reads to the references sequentially: it maps all reads to the first reference, then all reads to the second reference and so on. The space requirement of BRAT-large depends on the size of the largest reference in the set of input references, whereas BRAT's space requirement depends on the total size of all input references (measured in base pairs).

Let *N* be the total size of a genome in base pairs, *T* be the size of the largest reference in base pairs and *R* be the total number of reads (each read is counted, *i.e.* a pair has 2 reads). Then the space required for both programs is bound as follows:

$Space_{BRAT} = 269 \cdot 10^6 + 2 \cdot 4N + 24R + (3/8)N$ Bytes
$Space_{BRAT\text{-}large} = 269 \cdot 10^6 + 4T + 24R + (3/8)T$ Bytes
$Space_{BRAT\text{-}large} = 269 \cdot 10^6 + 2 \cdot 4T + 24R + (3/8)T$ Bytes (when option *S* is provided)

Whenever there is a space limitation, the user can choose to use BRAT-large. Here is an example of space usage with paired-end BS-mapping with non-BS-mismatches (mismatches other than T-to-C and A-to-G): BRAT uses 2.5 GB on 1 million pairs and human chromosome 1, and BRAT-large on 10 million pairs and an entire human genome uses 1.7 GB (or 2.7GB with option *S*).

In this version, BRAT allows for non-BS-mismatches with the restriction on the number of non-BS-mismatches in the first *X* bases (for BRAT) and *X* bases (for BRAT-large), where *X* is a threshold specified with the option *f* (*X* cannot be smaller than 24 or larger than 64). BRAT guarantees to map all reads that could be mapped to the genome with up to one non-BS-mismatch in the first *X* bases of reads. A user can specify any number of non-BS-mismatches with the option *m*, and as long as there is only one non-BS-mismatch in the first *X* bases of a read, BRAT will map this read. BRAT-large does not allow for non-BS-mismatches in the first *X* bases of reads. If a user specifies any number of non-BS-mismatches with the option *m*, BRAT-large will map all reads that could be mapped perfectly or with BS-mismatches within the first *X* bases, and any number of non-BS-mismatches in the other bases of reads.

The package includes three additional tools: trim, brat-large-build and acgt-count. The latter aligns mapped reads to the genome and counts mapped nucleotides at each base for forward and reverse strands separately. 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 and trims Ns at the ends of the reads. The tool brat-large-build is used with option P with BRAT-large.

2 COMMANDS AND INPUT

To uncompress run:

```
tar zxvf brat-1.1.*.tar.gz
```

To build:

```
cd brat-1.1.*
make
```

This will create five executable programs: brat, brat-large, brat-large-build, acgt-count and trim.

Input format of the reads for BRAT and BRAT-large is raw reads, *i.e.* reads given one per line. 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

NEW in this version: we added option *m* to run with trim so that to allow internal *N*s as long as the number of internal *N*s is less than or equal to the value provided with 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
```

This will trim bases whose base quality scores are lower than 20 from the ends of reads: bases are trimmed one at a time from both ends of a read until a base with quality score greater or equal than 20 is encountered. 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/. Please note that if all reads have at least one N in each of them, the output will be empty files. If this is the case, please use the following command:

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

This will allow each read having not more than 2 *N*s (not at the end of reads, i.e. internal *N*s).

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
```

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 and brat-large

BRAT and BRAT-large map raw reads to references that have to be in FASTA format: one reference per FASTA file. BRAT accepts a single file that contains names of FASTA files with the references. To map bisulfite single-end reads, run either of the commands:

```
./brat -r  references_names.txt  -s  prefix_reads1.txt -bs  -o output_results.txt

./brat-large -r  references_names.txt  -s  prefix_reads1.txt -bs  -o output_results.txt

./brat -r  references_names.txt  -s  prefix_reads2.txt -bs  -o output_results.txt  -A

./brat-large -r  references_names.txt  -s  prefix_reads2.txt -bs  -o output_results.txt  -A
```

The file *references_names.txt* contains the names of the FASTA files with the references. The file *output_results.txt* contains the results of the mapping: only uniquely mapped reads are in this file. The option -bs specifies that mapping is for bisulfite sequenced reads. To map normal (not bisulfite-treated) reads, run similar commands but without -bs option:

```
./brat -r  references_names.txt  -s  prefix_reads1.txt  -o output_results.txt

./brat-large -r  references_names.txt  -s  prefix_reads1.txt  -o output_results.txt
```

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

```
./brat -r  references_names.txt  -1  prefix_reads1.txt -2 prefix_reads2.txt -pe -bs  -o output_results.txt
./brat-large -r  references_names.txt  -1  prefix_reads1.txt -2 prefix_reads2.txt -pe -bs -o output_results.txt
```

The option -pe specifies paired-end mapping and as with single-end reads, -bs, specifies bisulfite mapping. The results of the mapping will be in *output_results.txt*.

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;
-u is used when a user wishes to output unmapped reads. The output will be in *unm_prefix_reads1.txt* file (assuming *prefix_reads1.txt* was provided in the command line with option **-s**) for single reads and with paired-end reads in *unm_prefix_reads1.txt* and *unm_prefix_reads2.txt* files (if *prefix_reads1.txt* and *prefix_reads2.txt* were used with the options **-1** and **-2** in the corresponding command line). Unmapped reads in the output files are in raw reads format: one per line, there are no ambiguous reads/pairs in the files with unmapped reads;
-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*);
-2 <paired-end reads file>: to specify the file with 3` mates for paired-end reads mapping (in our example above, *prefix_reads2.txt*);
-pe to specify paired-end reads mapping (default is false, *i.e.* single-end mapping);
-bs to specify bisulfite sequenced reads mapping. Without this option, mapping will be done as for normal, not bisulfite-treated, sequenced reads;
-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 is used when a user wishes to output ambiguous reads. The output will be in *amb_prefix_reads1.txt* for single reads and in *amb_prefix_reads1.txt* and *amb_prefix_reads2.txt* files with paired-end reads: raw reads one per line. This option does not output the mapping locations for ambiguous reads, but just the reads themselves;
-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, only one non-BS-mismatch is allowed in the first <integer> bases, and for BRAT-large NO non-BS-mismatches are allowed in the first <integer> bases (for BRAT, default is 36, and for BRAT-large, default is 24).
-S to specify speed mode for BRAT-large. Space usage with this option doubles, but running time is about three times faster.
-P <directory name> To use this option that allows to separate hashing of the genome from mapping of reads, one has to create a directory, *some_directory* (the name of a directory in the example below), and to run the following commands (1) to build a hashing index and 2) to map reads.

```
./brat-large-build -r  references_names.txt   -P some_directory [options: S, bs, f ]

./brat-large -P  some_directory   -1  prefix_reads1.txt -2 prefix_reads2.txt -pe -o output_results.txt [options]
```

If the directory *some_directory* is the directory inside the directory in which you run the commands above, then providing the name, *some_directory*, is sufficient. However, if you run these commands in the directory that does not have *some_directory* in it, then please provide a full path to *some_directory*.
The tool brat-large-build will output hashing index of the references into *some_directory*. For human genome, space on hard disk for hashing index is 15GB (and 26GB when option *S* is used). The command for tool brat-large with option *P* differs only by not specifying option *r*. If option *bs* was passed to brat-large-build, then this option must be passed to brat-large. In other words, for mapping normal and bisulfite-treated reads, two different hashing indices must be built. The same true when option *S* is used: if a user wishes to use this option and *P* option, then *S* must be used with brat-large-build. When *P* is used, during mapping option *f* will have the same value that was set with brat-large-build. If a user does not specify this option, the default value, 24, will be used. If a user wishes to speed up mapping time, and to use option *P*, then the user could set option *f* to a higher value with brat-large-build.
General rule for using option P: if a user wishes to change either of the parameters: *f*, *S* or *bs*; the user must apply the same parameters with brat-large-build. Default options fr brat-large-build are f = 24, *S* is not set (i.e. preference is given to slower mapping, but also smaller space usage), and normal mapping (i.e. *bs* is not set).

-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	Largest Value in

			ASCII representation	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

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 has 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
```

3 OUTPUT FORMAT

Output format for brat and brat-large 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>

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

Output format for acgt-count

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).*

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 read per line (raw reads format).

4 DETAILS ON *ACGT*-COUNT

Let us denote a mapping of a base in a read to a base in the reference as C→C, T→C, A→G, G→G. Initially, we thought that if a read maps to a positive strand, then the mappings C→C and T→C contribute to the count of methylation level of cytocines of the positive strand. Similarly, if the reverse-complement of a read maps to the positive strand (equivalently, a read maps to the negative strand), then the mappings A→G and G→G contribute to the count of methylation level of cytocines of the negative strand. Let us illustrate this idea in Figures 1-3 below.

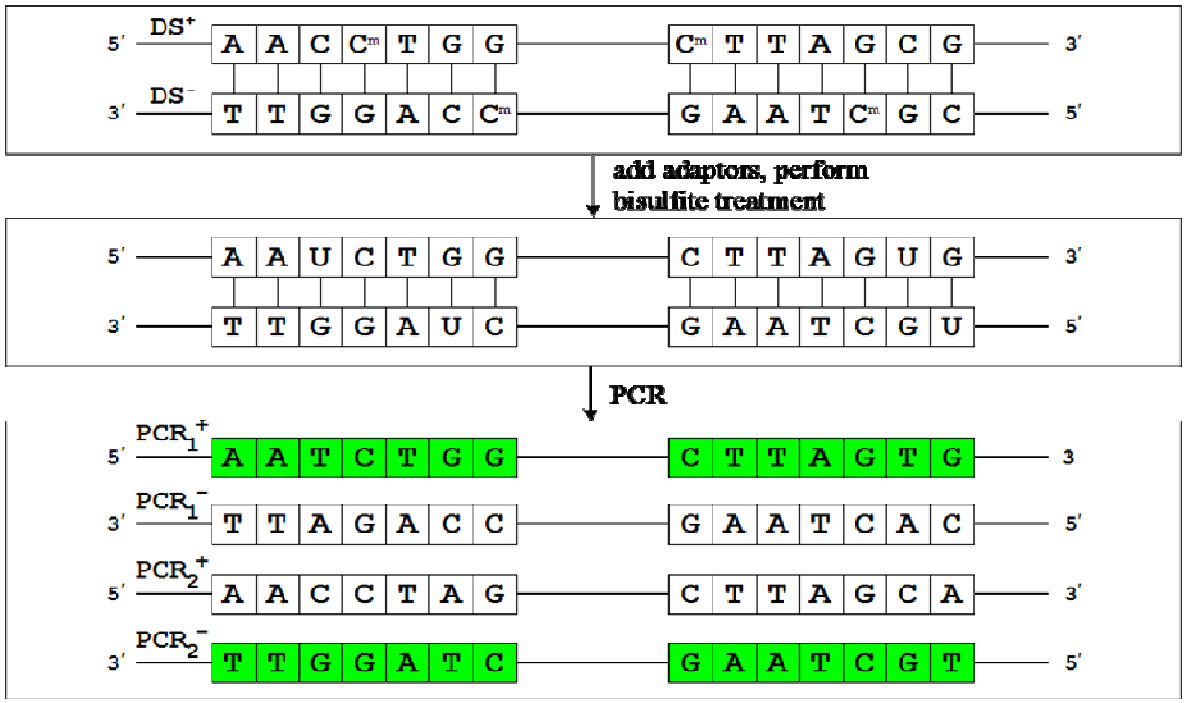

Figure 1. After the special adapters with methylated cytokines are ligated to DNA fragments, sodium bisulfite treatment (BS-treatment) is applied to DNA fragments, after which unmethylated cytocines are converted to uracils and later to thymines during PCR for library amplification. Note, that after BS-treatment, there are four distinct PCR-product strands: PCR1+ and PCR2- (correspond to original genomic strands) and PCR1- and PCR2+ (the reverse-complements of PCR1+ and PCR2- respectively). Note, that PCR1+ and PCR2- are T-rich and C-depleted (since unmethylated cytocines converted to thymines), and PCR1- and PCR2+ are A-rich and G-depleted (as the reverse-complements of PCR1+ and PCR2- respectively).

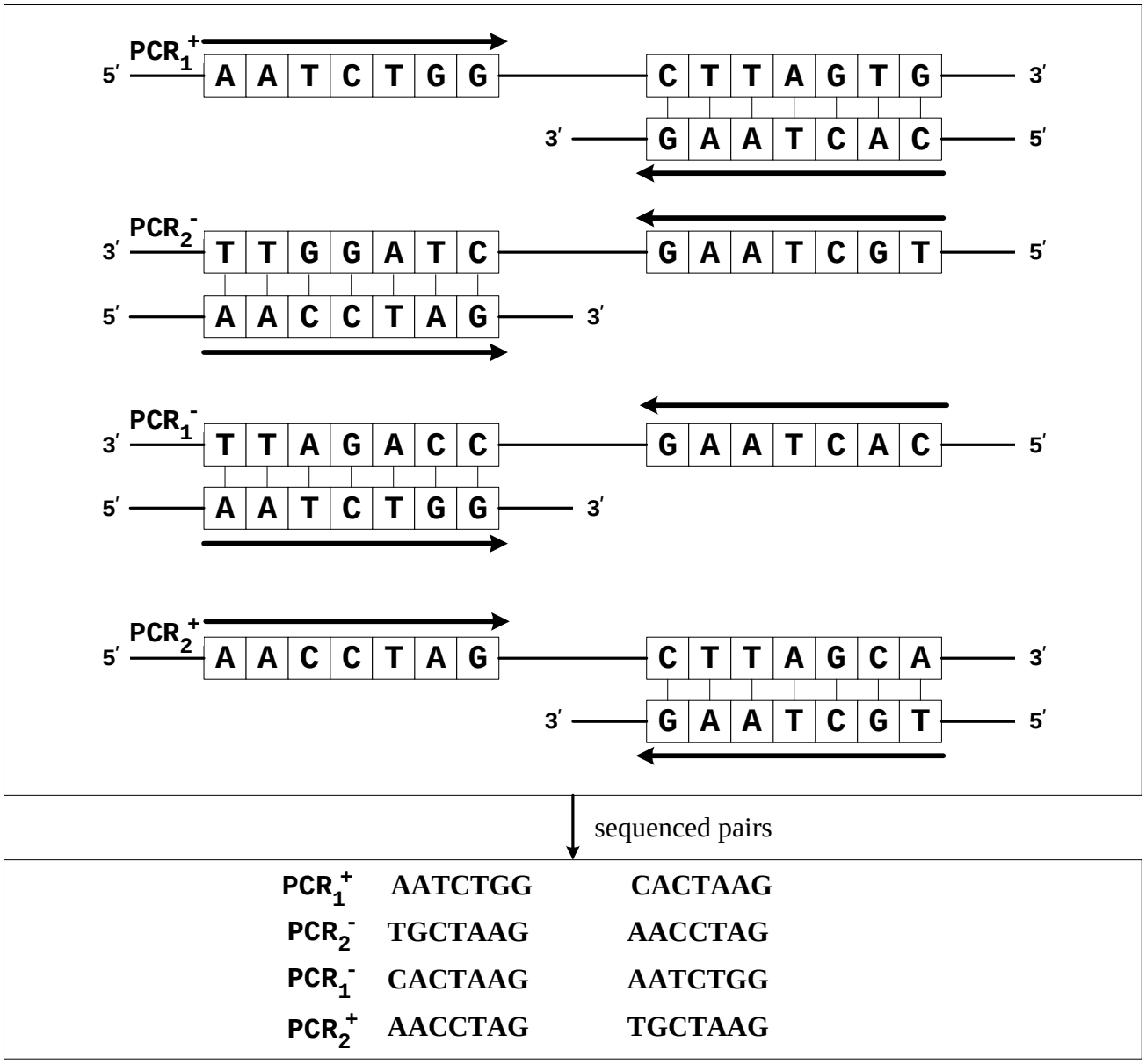

Figure 2. Here we show the sequenced pairs resulted from sequencing all four PCR-product strands, each of which serves as a template during sequencing. If PCR1+ is attached to the flow cell, then in single-end sequencing, 5'-end of this strand is sequenced; in paired-end sequencing, 5'-end of PCR1+ is sequenced, then PCR1+ is copied into its reverse-complement PCR1- and 5'-end of PCR1- is sequenced as the second mate of a pair. Similarly, sequencing produces reads for the rest of strands. If procedure has some technical details that ensure that only original genomic strands serve as templates for sequencing, then even though we have four PCR-product strands, we'll have reads sequenced only from PCR1+ and PCR2- with mates ordering shown above (in paired-end sequencing, the left column of reads are mates 1 will have reads IDs end with "1" and will be in one file, the right column of reads are mates 2 will have reads IDs end with "2" and will be in the other file of Illumina's sequenced reads).

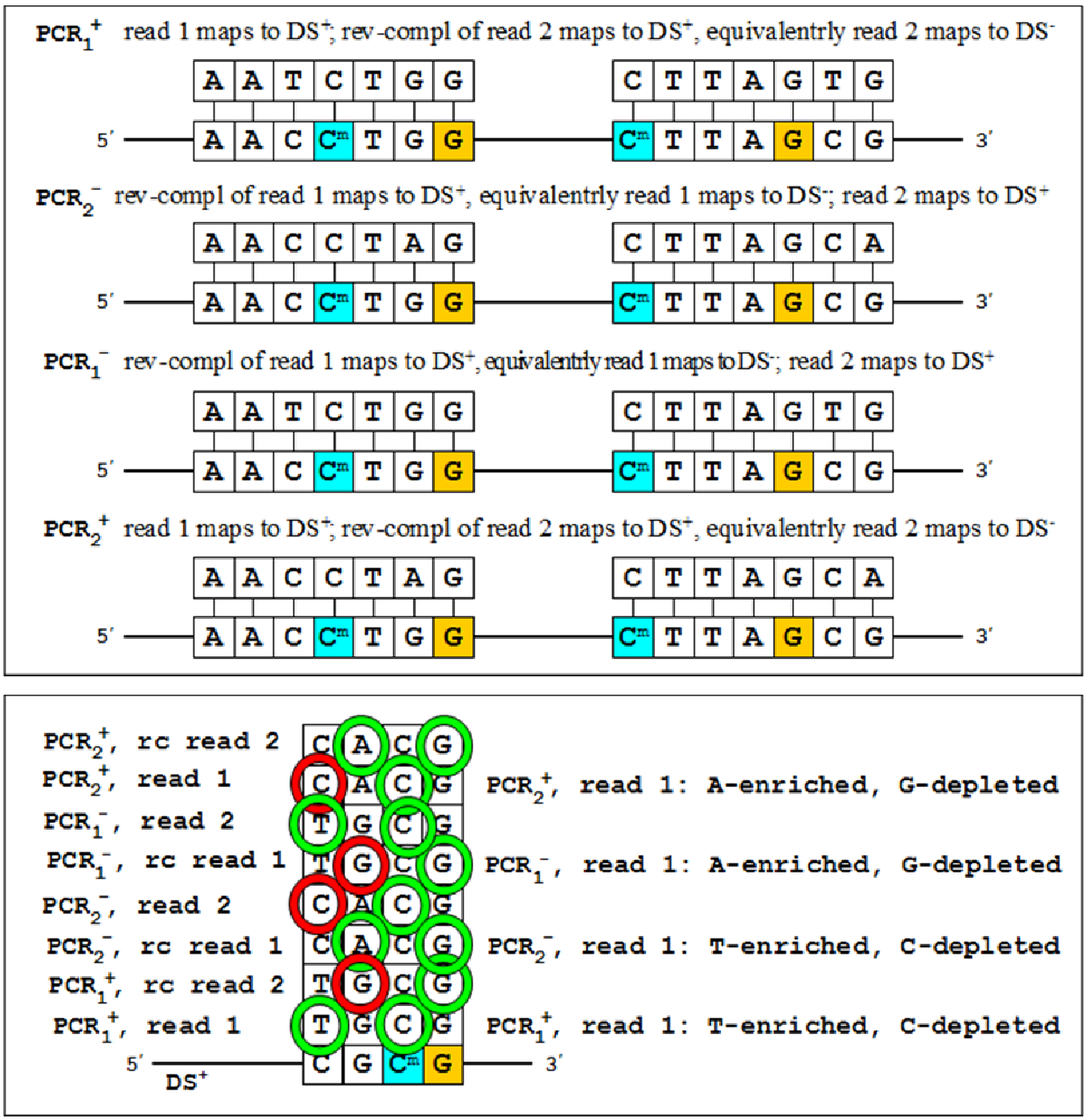

Figure 3. Here, we show mapping of the pairs sequenced from all four strands. The bottom strand shows positive strand of the reference with methylated cytocine shown in blue and methylated cytocine on negative strand shown as G in orange. This figure demonstrates that if we use simplified approach of counting methylated level of cytocines from positive and negative strands, then we introduce a bias in counting methylation level of unmethylated cytocines (either cytocines that are partially methylated across cells of a sample or completely unmethylated). For example, read 2 (mate 2) from PCR1+ maps to negative strand (its reverse-complement maps to positive strand), and therefore, initially we contributed ACGT-counts from this read toward negative strand. This was incorrect: in this Figure, bias is introduced by counting G from this read (in red circle) toward unmethylated G (i.e. unmethylated cytocine on negative strand). The count of another G from this read (in green circle) toward methylated G did not introduce a bias (since methylated G has only Gs mapped to it), but the coverage of methylated Gs is also not what it would have been were we using the correct counting. Similar bias is introduced when counting contribution of ACGT from mates 2 from PCR2+.

From Figures 1-3, we can observe that mate 2 of PCR1+ strand is reverse-complement 3'-end of PCR1+; thus, mate 2 must reflect methylation on positive strand rather than on negative strand. For example, methylated C on PCR1+ will be G on PCR1-, and unmethylated C (which is T) on PCR1+ will be A on PCR1-. Thus, T→C and C→C, where T and C respectively belong to PCR1-, must contribute to the count of methylation level of cytocines on positive strand.

Similarly, A→G and G→G (with A and G belonging to PCR2+) must contribute to the count of methylation level of cytocines on negative strand.

These changes have been made to acgt-count and now ACGT-counting from second mates of pairs sequenced from PCR1+ and PCR2- is done without bias, which is demonstrated in Figure 4.

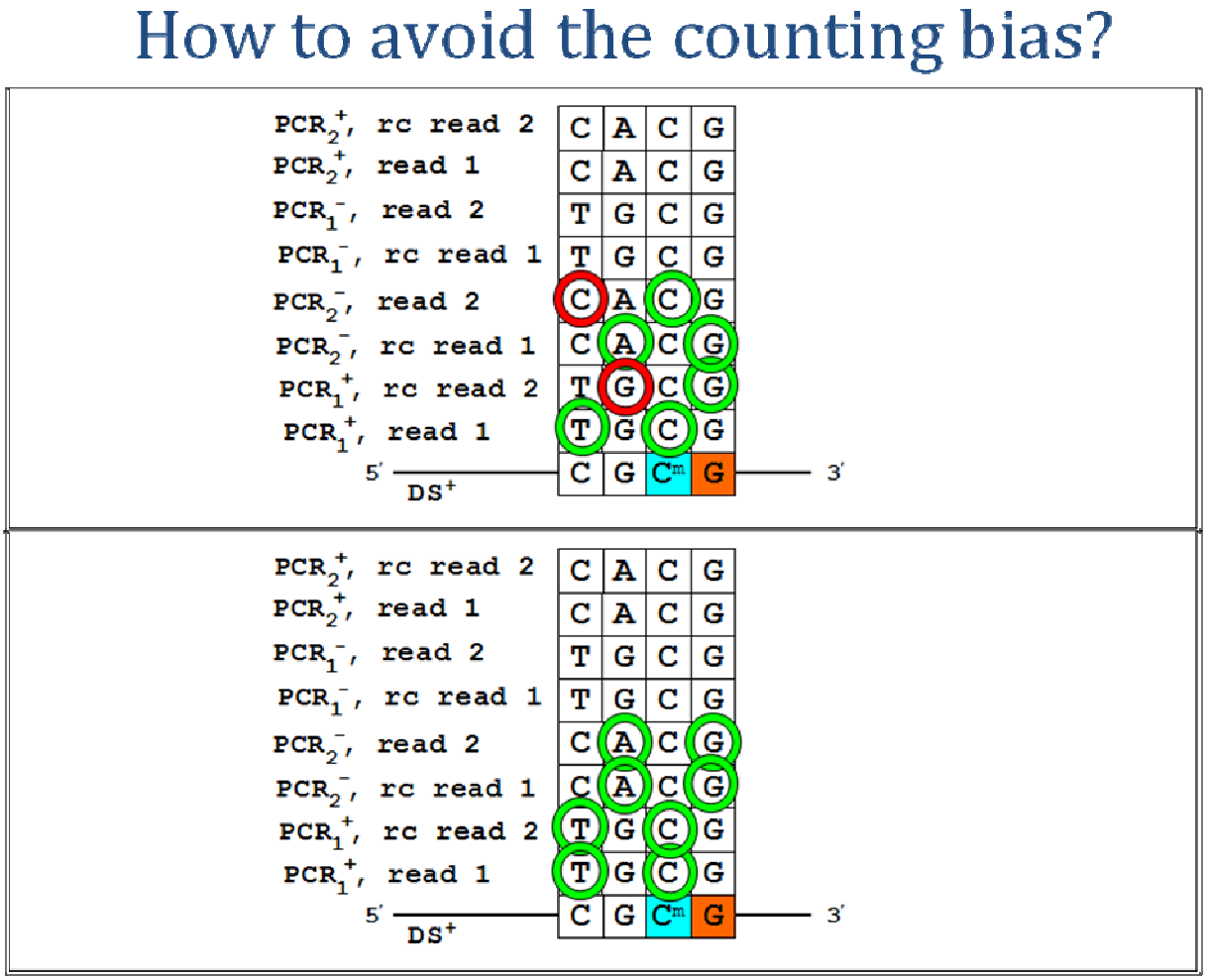

Figure 4. If mate 1 of a pair maps to positive strand (case of a pair sequenced from PCR1+), then we increment ACGT-count from both mates for positive strand, and methylation level of a cytocine on the positive strand can be measured as total Cs mapped to positive strand to the sum of total Cs and total Ts mapped to the positive strand. If mate 1 of a pair maps to negative strand (case of a pair sequenced from PCR2-), then we increment ACGT-count from both mates for negative strand. Methylation level can be determined for each G on positive strand of the reference (G on positive strand corresponds to C on negative strand) by looking into acgt-count output for negative strand and taking the ratio of total Gs mapped to this position and the sum of total Gs and As mapped to this position.

We received a couple of e-mails concerned that the sequenced reads must be products of sequencing of all four strands. The best way to be sure how many strands have been sequenced is talking to Illumina tech support, but there is also a quick and dirty method to answer this question. This approach will work best with genomes whose ACGT-distribution is close to uniform, or with genomes that CG-rich. We cannot be positive that the way we will explain shortly can be successfully applied to AT-rich genomes. We added a new tool, check-strands, that calculates ACGT-distribution in the genome, and in mates 1 and mates 2 of paired-end reads. For the case, when pairs are sequenced only from two original genomic strands, you should notice significant increase of Ts in mates 1 (and decrease of Cs) compared to Ts and Cs in the genome, and significant increase of As in mates 2 (and consequently decrease of Gs) compared to As and Gs in the genome. If four strands are sequenced, then the file with mates 1 contains a mixture of reads sequenced from original genomic strands as well as their reverse-complements. Thus, the increase of both As and Ts (and decrease of counts of Cs and Gs) should be observed in both files: with mates 1 and mates2. This conclusion is derived from the fact that

after BS-treatment original genomic strands are T-rich and C-depleted, and consequently their reverse-complements PCR1- and PCR2+ are A-rich and G-depleted.

Below is a command to run check-strands:

Single-end reads:

```
./check-strands -r  references_names.txt  -s  reads1.fastq -o output_results.txt
```

Paired-end reads:

```
./check-strands -r references_names.txt  -1  reads1.fastq -2 reads2.fastq -o output_results.txt
```

Here, *reference_names.txt* as before: it is the file with names of files in FASTA with references; *reads1.fastq* and *reads2.fastq* are FASTQ files from Illumina with sequenced reads (for paired-end reads, with mates 1 and mates 2). The output is in *output_results.txt*; it is self-explanatory with counts of A, C, G and T (in that order) for references, for reads from *reads1.fastq* and for reads from *reads2.fastq*.

How to use BRAT with reads sequenced from four strands.

For single-end reads, if reads are sequenced from four PCR-product strands, namely PCR1+, PCR2-, PCR1- and PCR2+, then run BRAT as explained before with option *-u* that will output unmapped reads into a separate file, say *unmapped.txt*. Then use our new tool rev-compl with *unmapped.txt* as follows:

```
./rev-compl -s unmapped.txt -o rev_compl_of_unmapped.txt
```

The output file *rev_compl_of_unmapped.txt* will contain reverse-complement of reads from *unmapped.txt* file.

Then map reads from *rev_compl_of_unmapped.txt* as usual (see how to map single-end reads with BRAT).

For paired-end reads, map reads as usual, but use -u option to output unmapped reads (use appropriate command of these two):

```
./brat -r  references_names.txt  -1  prefix_reads1.txt -2 prefix_reads2.txt -pe -bs  -o output_results.txt  -u
./brat-large -r  references_names.txt  -1  prefix_reads1.txt -2 prefix_reads2.txt -pe -bs -o output_results.txt -u
```

The unmapped reads from *prefix_reads1.txt* will be in *unm_prefix_reads1.txt* and unmapped reads from *prefix_reads2.txt* will be in *unm_prefix_reads2.txt*.

Then take reverse-complements of unmapped reads:

./rev-compl -s *unm_prefix_reads1.txt* -o rc_*unm_prefix_reads1.txt*
./rev-compl -s *unm_prefix_reads2.txt* -o rc_*unm_prefix_reads2.txt*

Then run BRAT again on reverse-complements of unmapped reads, but this time use option -1 with mates 2, and use option -2 with mates 1 and **use option *A*** (use either of the commands below):

./brat -r references_names.txt -2 rc_*unm_prefix_reads1.txt* -1 rc_*unm_prefix_reads2.txt* -o output_results2.txt -A
./brat-large -r references_names.txt -2 rc_*unm_prefix_reads1.txt* -1 rc_*unm_prefix_reads2.txt* -o output_results2.txt -A

The rest of the tools are used as before.