Using the RMAP software package

Andrew D Smith Wen-Yu Chung

August 19, 2009

RMAP is aimed to accurately map reads from next-generation sequencing technology. RMAP was developed for mapping Illumina reads, but has been used to map Roche/454 and ABI SOLiD reads. RMAP can map reads with or without error probability information (quality scores) and this current version supports paired-end reads or bisulfite-treated reads mapping. There is no limitaions on read widths, number of mismatches or the number of non-unique mappings.

1 Basic use of RMAP mapping programs

Quick start. Assuming the files are in the appropriate format, you can try the following command to start mapping right away:

```
$ ./rmap -o mapped_locations.bed -c chromosomes_dir reads.fa
```

The command will map all the reads in the file `reads.fa` to the chromosomes in FASTA format files in `chromosomes_dir`, which are assumed to have the suffix ".fa". The output will be written in the file `mapped_locations.bed` and will be in 6 column BED format (see below). Specifying an output file is not strictly necessary, but if not specified, the mapping results (which may be very large) will be printed to the terminal screen. If you want to see the progress of the program as it runs, then add the "-v" flag to the command line. The **rmapbs** and **rmappe** programs can be executed similarly:

```
$ ./rmapbs -o mapped_bs_locations.bed -c chromosomes_dir bs_reads.fa
$ ./rmappe -o mapped_pe_locations.bed -c chromosomes_dir pe_reads.fa
```

Above it is assumed that `pe_reads.fa` contains paired-end reads and `bs_reads.fa` contains reads sequenced following bisulfite treatment.

Specifying input. There are 2 kinds of input: reads files and chromosome files (the reads are mapped to the chromosomes). The chromosome files, which need not actually be chromosomes, must be specified in FASTA format files. The files can have multiple sequences in each. There are three options for specifying the chromosomes:

1. "-c chromosomes_dir" indicates a directory containing chromosome files, and the chromosome files will be assumed to be all files in "chromosomes_dir" having the suffix ".fa", unless the "-s" flag is used to specify some other suffix. Other files may be present in the chromosomes directory, but exactly those with the appropriate suffix will be used.

2. "-F chromosomes_file_set.txt" specifies a file that contains the path names (absolute or relative) of the chromosome files to which one wants to map. Use this if there are certain chromosomes in your chromosome directory that you want to exclude from the mapping. There must be exactly one chromosome file specified on each line, and there may be no blank lines.

3. "-c chromosomes_file.fa" specifies a single FASTA format chromosome file, which may contain one or multiple chromosomes.

The other kind of input is the reads, and specifying them in the correct format is critical. The RMAP programs will take reads in 3 formats:

1. FASTA format is pretty simple, and descriptions are all over the web. The important thing here is that individual reads must not span multiple lines. For the mapping, all read sequences are translated into {A,C,G,T,N}, with anything other than {a,A,c,C,g,G,t,T} automatically converted into "N". FASTA reads files are specified on the command line with no flags. Hopefully soon we will be able to use all the IUPAC degenerate nucleotide symbols in a meaningful way.

2. FASTQ format is more complicated. To understand this format, read the documentation on the web associated with MAQ/BWA. Note that the RMAP programs will automatically deduce whether a reads file is FASTA or FASTQ, and will also attempt to determine if the FASTQ is Solexa or Phred (again, see the MAQ documentation). FASTQ format files specify the called read sequence and a quality score for each base.

3. FASTA+PRB formats together give the greatest amount of information. Unfortunately it is increasingly difficult to get the PRB format files. The PRB files specify one line for each read, and have tab deliminted numerical values for the quality scores for the 4 bases at each position. For reads of width k there are $4k$ numerical values per line. These values must be actual quality scores, so it is best to just take the files as they are from the output of the Illumina pipeline. The FASTA format file must be provided with the PRB file because no read names are encoded in the PRB file. The PRB files generated by the Illumina pipeline are usually named like `*_prb.txt`.

Regarding the FASTA+PRB files, there must be exactly 2 times as many lines in the FASTA file, since each FASTA entry has a name line and a read line. Converting the reads from the Illumina format to FASTA format is trivial and can be done on the command line with some simple commands, or by using the following Python script which names each read according to tile (or however the Illumina files are named) and a unique number:

```
#!/usr/bin/env python
import sys, os
name = os.path.basename(sys.argv[1])[:-4]
c = 0
out = open(sys.argv[2], 'w')
for i in open(sys.argv[1]):
    out.write('>%s_%d\n%s\n' % (name, c, i.strip().split()[-1]))
    c += 1
out.close()
## Done!
```

An example using quality scores specified in a PRB file might look like this:

```
$ ./rmap -p reads.prb -o mapped_locations.bed \
                            -c chromosomes_dir reads.fa
```

When I work with Illumina reads, I generally name each read according to (1) the machine that produced them, (2) the lane in the machine, (3) the tile in the lane, and (4) the cluster number in that tile. This naming will allow me to do extra post-mapping diagnostics if I want to. Note that the quality score information can be used flexibly by the RMAP programs (see Section 2.1).

Specifying output and output formats. The output file is specified with the "-o" flag as indicated above. In addition to the locations of mapped reads, names of reads that map ambiguously can be output in a file specified with the "-a" flag. The definition of "ambiguous" is given below. Note that only the names of those ambiguously mapping reads are written in this file.

The output format is 6 column BED (Browser Extensible Data?), after the format used by the UCSC Genome Browser. The 6 columns give the following information:

```
chrom     start     end     name     score     strand
```

The chrom is taken from the name/header line for the sequences in the FASTA format chromosome files. If there are spaces in the names of the sequences (which again, need not be chromosomes), then only the characters up until the first space or tab are used. The "start" and "end" are positions in the chromosome/sequence, and specify the interval covered by the read. This means for a read of 36 bases, the "end - start" will be exactly 36 and therefore the "end" is redundant. Note that these are "zero-based half-open intervals". The "name" is the name of the read, the "score" indicates how well that read mapped to this position (see Section 2.1 for scoring information). The "strand" is either "+" or "-".

Ambiguous reads and multiple mappings. The RMAP mapping programs by default will only report mapping results for uniquely mapping reads. Reads map uniquely if they score better than the user specified scoring criteria (see Section 2.1) and there is no other location with a score at least as good. So for example if at most 2 mismatches are allowed, and a read matches one location with 2 mismatches, but a second location with 1 mismatch, then it will be reported as mapping to the second location, and as a unique map. The "-M" argument can be used on the command line to set the number of mappings to report. For example, the following command:

```
$ ./rmap -M 10 -o mapped_locations.bed -c chromosomes_dir reads.fa
```

will report up to 10 mapping locations for each read in the `reads.fa` file. The reported matches for a given read will all have the exact same score. If for a given read there are 5 locations where the read has 2 mismatches, and another 5 having 1 mismatch, only those 5 with 1 mismatch will be reported. If for some read there are 11 locations having the fewest number of mismatches (or best score according to some other scoring criteria), then none will be reported and the read will be considered to map ambiguously.

2 General options for RMAP mapping programs

There are a set of common options that control the mapping procedure in **rmap**, **rmapbs** anr **rmappe**. These options deal with how the matches are scored and specifying the searching strategy (*i.e.* the "seeds" used in the search).

2.1 Scoring matches and use of quality scores

Simple mismatches. In the most basic mode, the RMAP programs measure how well a read matches a segment of the reference genome by counting the mismatches. The number of allowed mismatches is specified using the "-m" flag on the command line. The following example:

```
$ ./rmap -o mapped_locations.bed -c chromosomes_dir -m 3 reads.fa
```

will map the reads allowing at most 3 mismatches anywhere in the read. This might be too few for reads that are now often over 100nt (and for each end, in the case of paired-end). By default the RMAP programs allow up to 10 mismatches anywhere in the read, and are fully sensitive to 2 mismatches in the first 32 bases. "Full sensitivity" is controlled by the "seed" parameters discussed below. It is probably Ok to allow large numbers of mismatches when mapping, and then use a program like the **mapsifter** described below to filter out those reads with fewer than some cutoff, as desired. The only issue is that the number of mismatches should be set so that at least 20-30 positions are required to match.

Wildcard matching. The original release of RMAP used quality scores to identify those positions in reads where mismatches were ignored when scoring the overall match between read and reference genomic segment. A similar but enhanced scoring mode is provided in the current release. The flag "-W" specified on the command line indicates to use "wildcard matching" and for this the reads must be accompanied by quality scores. The wildcard matching works as follows.

For each position in each read, the quality score is converted into an error probability, indicating the probability that each base at that position is *not* the correct base. If a position in a read is aligned with base b in the genome, and the probability of base b not being the appropriate base is greater than the cutoff, then a penality of 1 mismatch will be counted. Note that at a given position in the read, the sum over each base of one minus the error probability for that base, must equal exactly 1. So, for example, if one base has error probability of 0.4, and another has error probability of 0.61, then the final two would have error probability at least

$$\frac{1-((1-0.4)+(1-0.61))}{2} = 0.99.$$

Assuming a cutoff of 0.99 (which is the default), those two bases with error probability 0.99 would force a mismatch in the alignment. The value of 0.99 can be changed by giving a different value with the "-P" flag.

Position-weight matrix (PWM) matching. The full quality score information produced during the base-calling procedures by sequencing instruments generally produce a probability for each of the 4 bases at each position in the read. This information can be thought of as a matrix of probabilities. Such matrices have been used for many years in biological sequence analysis, and are often called position-weight matrices (PWMs) – though several other names have been used. The PRB files produced by the Illumina pipeline contain quality scores for every base at every position, and the PWM matching mode can use all of this information. The benefit is that some non-consensus bases (e.g. the second most likely base at a given position) may not be penalized as much as others when the quality score information indicates that they should not be. In theory this method should be the most accurate, but that depends on the data and whether the instrumentation actually provides meaningful values.

To use the PWM scoring method, the input must be specified in either FASTA+PRB format or FASTQ format. If it is only specified in FASTQ format, then there will be essentially a weight associated with mismatching the consensus at each position, and much of the benefit of using the full PWM is lost. In addition, the "-Q" flag must be present to use PWM matching if the input is in FASTQ format, or RMAP programs will ignore the quality score information.

In this mode the scoring for mappings is in terms of "equivalent mismatches" which correspond to the difference in overall score that would result if there is a mismatch at a position that had received a perfect base-call quality score. For example, in the original Illumina pipeline, with scores in the range -40 to +40, a single "equivalent mismatch" would correspond to a score of 80. Another benefit here is the precision of this scoring. Note that it is not full precision, as some information is lost by the actual implementation of PWM matching. More specifically, the scores in the range, e.g., -40 to +40 are projected onto the values $\{0,\ldots,15\}$ during the mapping procedure, and the projected back once the mapping is finished.

Quality scores for reads can be specified either in FASTQ format or in the full PRB format (*i.e.* `*_prb.txt` files) . For details on FASTQ format, the manual for the MAQ program has an excellent description. The PRB format has more information, as it can indicate when a non-consensus base at a position was nearly as likely as the consensus base.

	Program	*Score cutoff*	*Coverage (10-fold)*	*Enrichment*	*Mapped in target*
Initial release	RMAP (orig)	1	0.699	0.966	2490598
		2	0.750	0.958	3071044
	RMAP (orig; Q)	0	0.589	0.978	1829939
		1	0.742	0.975	2871745
Current release	RMAP (WC)	0	0.740	0.974	3375148
		1	0.766	0.965	3791773
	RMAP (Q)	1.75	0.611	0.980	2064228
		2	0.652	0.976	2273794
	BWA (default)	2	0.651	0.981	2702570

Which scoring parameters to select? The table above is copied from (Smith et al., 2009), and includes an extra row to illustrate the performance of BWA ((Li & Durbin, 2009)), which is a great mapping tool, with both speed and accuracy. The data is from the resequenced BAC, used in the original RMAP paper ((Smith et al., 2008)). Coverage is the proportion of bases in the target region covered by at least 10 reads (using the first base of each read). Enrichment is the proportion of mappable reads mapped inside the target region. The dataset contained 6721851 reads of 36 bases each, and the target region size was 162829 bases. The programs are: RMAP using both wildcard matching (WC) and weight-matrix matching (Q) with the original verion (orig) using mismatch counts and the original version of using quality scores (Q). The score column indicates number of allowed mismatches (fractional values arise from use of quality scores to weigh mismatches). The default values were used for BWA scoring. Only unique mappings were considered.

When selecting parameters to use for the scoring, the main criteria to consider are:

- **Sensitivity.** How important is it to have the greatest number of reads mapping? In projects where the relative read density is important, but the exact bases in the mapped reads are not really used (e.g. ChIP-Seq), then it might be acceptable to have more reads mapping even if it introduces more false positive mappings. Or when you know your sample is very clean, and have high-confidence in the sequencing quality, then the main criteria might be to map the greatest number of reads. To increase sensitivity, (1) allow more mismatches by increasing the value of the "-m" parameter (regardless of the scoring mode), (2) allow non-unique mappings by increasing the value of the "-M" parameter, or (3) use wildcard mapping.

- **Specificity.** Alternatively, is it critical to have correct mappings (e.g. in SNP calling applications or in DNA methylation profiling)? How about uniquely mapping reads? Will the reads be usable if they map to multiple locations? If false positives will confound the subsequent analysis, then it is probably wise to set parameters for high specificity. This can be done by using wildcard mapping with a low cutoff and allowing a low number of mismatches, or by using PWM matching with a low number of allowed equivalent mismatches.

The choice between mapping with wildcards and using the PWM matching can often be based on whether full precision is desired in the results, and whether full quality score information is available. Both of these methods can use base-call qualities, but the wildcard mode reports results in terms of mismatches. High-precision scoring can be very useful if one has a means of determining (after the mapping has been done) some mapping score cutoff to use. For example, if one has a particular target region (exons/junctions in the case of RNA-Seq; or target selected using hybrid capture), then one can tweak the cutoff without remapping using the **mapsifter** tool supplied with the RMAP software. By calculating the proportion of reads mapping into the target regions, for each score cutoff, one can select a cutoff that should be appropriate. In theory, using the PWM matching should allow more accuracy than using the wildcard mapping, but in practice this will depend on properties of individual datasets.

2.2 Parameters related to seed selection

It has long been known that the best techniques for exact string matching can be used to speed-up approximate matching through various techniques commonly referred to as "exclusion" methods. (Gusfield, 1997) gives an in-depth discussion of such techniques. The basic idea is as follows: whenever two sequences match approximately (*i.e.* with fewer than a specified number of mismatches) then some "part" of them must match exactly. Various results have been derived about conditions on the parts that must match exactly, and this remains an active area of research. A frequent procedure is to scan the larger text with part of the pattern. Locations in the text where that part matches exactly are called "hits" and the area surrounding each hit is examined more closely to determine if it matches well with the entire pattern. RMAP uses two kinds of seeds to implement the exclusion stage: layered seeds which are probably exclusive to RMAP, and a modified version of the seeds described by (Chen et al., 2009) for full sensitivity to 2 mismatches.

The idea of "layered seeds", which is similar to multiple filtration ((Pevzner & Waterman, 1995)). Seed structures indicate sets of positions in the reads that are required to match the genome exactly at any location where the read can map. Two distinct sets of seed structures are obtained such that if there is an approximate match, then each set of seed structures will contain at least one structure indicating positions that match exactly between the read and the genome. The two distinct sets of seed structures are combined creating a new set of seed structures corresponding to each pair of structures from the two initial sets. The combined (or *layered*) seed structures are more numerous, leading to an increased number of scans of the genome (one scan for each seed). However, these layered seeds are more specific, and therefore each scan excludes more full comparisons and is more efficient.

The following diagram illustrates layering of seed structures from two sets, producing a third set of seed structures:

$$\left\{\begin{array}{ll} a_1: & 111100000000 \\ a_2: & 000011110000 \\ a_3: & 000000001111 \end{array}\right\} \times \left\{\begin{array}{ll} b_1: & 100100100100 \\ b_2: & 010010010010 \\ b_3: & 001001001001 \end{array}\right\} = \left\{\begin{array}{ll} a_1b_1: & 111100100100 \\ a_1b_2: & 111110010010 \\ a_1b_3: & 111101001001 \\ a_2b_1: & 100111110100 \\ a_2b_2: & 010011110010 \\ a_2b_3: & 001011111001 \\ a_3b_1: & 100100101111 \\ a_3b_2: & 010010011111 \\ a_3b_3: & 001001001111 \end{array}\right\}$$

Three sets are presented, the third being obtained by layering the first two sets. The 1s in each seed structure indicate positions the structure requires to match between two sequences for a full comparison of the sequences to be triggered. Each of these seed structure sets can be used to identify matching between 12bp sequences having up to 2 mismatches. The set resulting from layering has a larger number of structures but each structure specifies a greater number of positions that must match between the two sequences, and therefore results in fewer random "hits" when scanning the genome. Note that the layered seeds can be layered arbitrarily, producing many seed sets of very high weight, but also that contain very many seeds.

For the seeds designed by Yang-Ho Chen and Tade Souaiaia ((Chen et al., 2009)) for full sensitivity to 2 mismatches in the first 28 bases of the read are:

$$\begin{array}{ll} s_1: & 1110100111010011101001110100XXXX \\ s_2: & 0111010011101001110100111010XXXX \\ s_3: & 0011101001110100111010011101XXXX \\ s_4: & 1001110100111010011101001110XXXX \\ s_5: & 0100111010011101001110100111XXXX \\ s_6: & 1010011101001110100111010011XXXX \\ s_7: & 1101001110100111010011101001XXXX \end{array}$$

The "X" positions pad the seeds up to 32, and for the implementation of RMAP, the seeds can be extended

to 32 bases. We do this as follows:

s_1 : 11101001110100111010011101000110
s_2 : 01110100111010011101001110100011
s_3 : 00111010011101001110100111011111
s_4 : 10011101001110100111010011100101
s_5 : 01001110100111010011101001111100
s_6 : 10100111010011101001110100111010
s_7 : 11010011101001110100111010011001

Note that when unique mappings are requested, RMAP removes ambiguous reads as soon as possible, thus making subsequent searches more efficient. Because of this, the seed s_3 is used as the first seed, since it is the heavies (has the greatest number of "1" positions), and is most efficient at the early stage when more reads are being considered.

In the RMAP mapping programs, the default is to use the layered seeds, and the associated options are "-S", which specifies the number of seeds that are layered, and "-h" which specifies the weight of the layered seeds. The default is "-S 3 -h 11", which is fully sensitive to 2 mismatches in the first 32 bases of a read, and which makes 9 passes over the genome (9 total seeds after layering). However, this set of default seeds is very sensitive to higher numbers of mismatches in the first 32 bases. If one wants full sensitivity to 3 mismatches, then the following settings for "-S" and "-h" will work:

```
$ ./rmap -S 4 -h 8 -o mapped_locations.bed -c chromosomes_dir reads.fa
```

The above settings will generate 16 layered seeds, each of weight 14. A much faster method, though much less sensitive, can be used by specifying "-f" on the command-line:

```
$ ./rmap -f -o mapped_locations.bed -c chromosomes_dir reads.fa
```

The above uses the periodic seeds designed by Yang-Ho Chen and Tade Souaiaia at USC for the PerM program. The periodic seeds are fully sensitive to 2 mismatches, and require only 7 passes over the genome, but are much less sensitive generally than the default parameter settings.

3 Mapping paired-end reads

One can use **rmappe** to map paired-end reads. The reads must be specified in a file like those mentioned in Section 1, except that the reads must be paired-end, with one sequenced end concatenated to the other, and both ends appearing 5' to 3'. There are parameters to control the distance between the two ends, and the ends are required to appear on opposite strands. The "-min-sep" parameter specifies the minimum distance between the two ends, and the "-max-sep" specifies the maximum allowed distance between the ends. These values should correspond roughly to the fragment size range expected in the sequenced sample. The following example will map paired-end reads requiring at least 200 but not more than 600 bases between the locations of the ends.

```
$ ./rmappe -min-sep 200 -max-sep 600 -o mapped_pe_locations.bed  \
           -c chromosomes_dir pe_reads.fa
```

The default values for the min and max separations between the reads are 100 and 400 bases. A slower, but very accurate running mode can be used by giving the "-A" option on the command line. This will cause **rmappe** to map both ends simultaneously. It is generally very slow, but in ongoing projects on plant and fly genomes, some users find it much more accurate than the more popular and faster mapping methods. In the mapping result, each end will be given either "L" or "R" to indicate the appropriate orientation.

If your paired-end reads are specified in two files (one for each end), the you will need to concatenate the two ends before mapping them with **rmappe**. A quick way to do this is to use `paste` in the shell to merge two files side by side, and applying following Python script:

```
#! /usr/bin/env python
import os, sys
out = open(sys.argv[2], 'w')
for i in open(sys.argv[1]):
    if i.startswith('>'): out.write('%s\n' % (i.split('/')[0]))
    else: out.write('%s\n' %(''.join(i.split())))
out.close()
## Done!
```

In the future we may add an option to **rmappe** to do this automatically.

4 Mapping bisulfite-treated reads

Bisulfite sequencing refers to sequencing DNA following bisulfite treatment. The purpose is to profile DNA methylation, and bisulfite treatment has the effect of converting all (or almost all) unmethylated Cs in the genome to Ts. Mapping bisulfite-treated reads is no different than the regular mapping (no need to convert Cs to Ts *in silico* in either reads or reference genomes). The **rmapbs** program can map the reads exploiting any Cs in the reads that were unconverted in the bisulfite treatment. This helps with specificity in mapping and eliminates any reads mapping at positions in bisulfite-specific deadzones (see below). There are 2 options specific to **rmapbs**. The "-B" flag controls whether the mapping will allow unconverted cytosines at CpG positions to assist in mapping:

```
$ ./rmapbs -B -o mapped_bs_locations.bed -c chromosomes_dir bs_reads.fa
```

This is potentially dangerous, because in subsequent analysis it will appear at some loci as though more methylation is present that actually is, since more highly methylated reads may be more mappable. To our knowledge, RMAP is the only existing method that handles this situation properly.

Most bisulfite sequencing projects use sequencing adaptors that ensure only the T-rich strands of the bisulfite-treated fragments are actually sequenced. The complements of those strands following PCR amplification will be A-rich, and if the protocol is able to sequence the A-rich strands, then **rmapbs** has an option to allow those strands to be mapped:

```
$ ./rmapbs -A -o mapped_bs_locations.bed -c chromosomes_dir bs_reads.fa
```

Note that by specifying "-A" the T-rich strands will not be mapped, so the mapping will need to be done twice if both A-rich and T-rich reads are present.

To give some indication of the accuracy of mapping using **rmapbs** compared with another popular method (Bowtie; (Langmead et al., 2009)), we tested the programs using simulated data. One million reads of width 36 bases were randomly simulated from and mapped back to human chromosome 1; assuming 100% bisulfite conversion and 100% CpG methylation. Both programs use default parameters. The numbers indicate how many reads were correctly mapped back to the original genomic region.

	simulated errors		
Program	0	3	5
Bowtie	447450	433011	294474
rmapbs	849830	838011	783592

Average running time using Bowtie is 2 minutes and **rmapbs** 30 minutes on a Quad Core Intel CPU, 64-bit machine, 8GB memory. In general **rmapbs** requires about twice the running time of the regular **rmap** for equivalent sets of scoring and seed options. This is due to the lower sequence complexity of bisulfite-treated reads.

5 Deadzones

Deadzones indicate locations in the genome that no uniquely mapped reads can start, *i.e.* the first base of any read cannot be mapped inside a deadzone. Deadzones correspond to exact repeats of length at least the length of the reads (and therefore are specific to the read length). Note that the deadzones can be as short as 1 bp, but the repeat associated with a 1bp deadzone would have length exactly equal to the read length. Software has been included to identify the locations of deadzones genome-wide, and also bisulfite-specific deadzones (which are strand-specific). If you have deadzones for read width k, you can use them for analysis based on reads of length $\geq k$. Too much of a difference in the read length assumed for the deadzones and that of the reads will result in conservative analysis (*i.e.* throwing out more reads than necessary).

This example shows to obtain bisulfite-specific deadzones (option -B) of read width 70. Increasing the number of -p option lowers the memory required, but requires more time.

```
$ ./deadzones -B -k 70 -p 4 -o deadzones_length70.bed genome.fa
```

The output follows the BED-format, but the fourth and fifth columns (names and scores, respectively) are pseudo-values. The bisulfite-specific deadzones are different because they can be strand-specific. That means a read can map unambiguously on one strand, but not on the opposite strand. Note also that these bisulfite deadzones assume full bisulfite conversion, so the **rmapbs** program might still be able to map reads into these locations. The unions and intersections of these regions can be obtained by using the **sortbed** program with the "collapse" option (`-c`), and the **bedoverlap** program, respectively.

6 Simulating data

Several programs for simulating test data have been included. One for each type of reads. These will generate simulated reads from a specified reference genome. The user can specify a number of mismatches, number of read and width of reads. There are 3 separate programs for simulating with and without quality scores, for regular reads, paired-end reads and bisulfite-treated reads. An example of how to use the program for simulating 100 regular reads from a reference genome is:

```
$ ./simreads -o test_reads.fa -n 100 chromosomes_dir/*.fa
```

To simulate paired-end reads of end width 36 bases, at most three mismatches and insertion size between 200bp and 600bp:

```
$ ./simreadspe -min-sep 200 -max-sep 600 -e 3 -w 36 \
               -o test_pe_reads.fa -n 100 chromosomes_dir/*.fa
```

To simulate 100 T-rich reads (for DNA methylation projects using bisulfite-treated technique) assuming 95% CpG methylation and 100% bisulfite convertion:

```
$ ./simreadsbs -m 95 -b 100 -o test_bs_reads.fa \
                            -n 100 chromosomes_dir/*.fa
```

To simulate reads and quality scores, add "-q" on the command line. This will produce FASTQ output. If the full quality score information is desired, add "-p" and specify an output file, like:

```
$ ./simreadsbs -m 95 -b 100 -o test_bs_reads.fa \
                     -p test_bs_reads.prb -n 100 chromosomes_dir/*.fa
```

References

Chen Y, Souaiaia T, Chen T (2009) Perm: Efficient mapping of short sequencing reads with periodic full sensitive spaced seeds. *Bioinformatics* pp. btp486+.

Gusfield D (1997) *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology* Cambridge University Press.

Langmead B, Trapnell C, Pop M, Salzberg S (2009) Ultrafast and memory-efficient alignment of short dna sequences to the human genome. *Genome Biology* 10:R25+.

Li H, Durbin R (2009) Fast and accurate short read alignment with burrows-wheeler transform. *Bioinformatics* 25:1754–60.

Pevzner PA, Waterman MS (1995) Multiple filtration and approximate pattern matching. *Algorithmica* 13:135–154.

Smith AD, Chung W, Hodges E, Kendall J, Hannon G, Hicks J, Xuan Z, , Zhang MQ (2009) Updates to the RMAP short-read mapping software (Submitted).

Smith AD, Xuan Z, Zhang MQ (2008) Using quality scores and longer reads improves accuracy of solexa read mapping. *BMC Bioinformatics* 9:128.