Bioinformatics, 31(19), 2015, 3130–3138
doi: 10.1093/bioinformatics/btv357
Advance Access Publication Date: 10 June 2015
Original Paper

OXFORD

Sequence analysis

A parallel and sensitive software tool for methylation analysis on multicore platforms

Joaquín Tárraga[1], Mariano Pérez[2], Juan M. Orduña[2,*], José Duato[3], Ignacio Medina[1] and Joaquín Dopazo[1,*]

[1]Department of Computational Genomics, Centro de Investigación Príncipe Felipe, [2]Departamento de Informática, Universidad de Valencia and [3]DISCA, Universidad Politécnica de Valencia, Valencia, Spain

*To whom correspondence should be addressed.

Associate Editor: Alfonso Valencia

Received on September 8, 2014; revised on May 8, 2015; accepted on June 5, 2015

Abstract

Motivation: DNA methylation analysis suffers from very long processing time, as the advent of Next-Generation Sequencers has shifted the bottleneck of genomic studies from the sequencers that obtain the DNA samples to the software that performs the analysis of these samples. The existing software for methylation analysis does not seem to scale efficiently neither with the size of the dataset nor with the length of the reads to be analyzed. As it is expected that the sequencers will provide longer and longer reads in the near future, efficient and scalable methylation software should be developed.

Results: We present a new software tool, called HPG-Methyl, which efficiently maps bisulphite sequencing reads on DNA, analyzing DNA methylation. The strategy used by this software consists of leveraging the speed of the Burrows–Wheeler Transform to map a large number of DNA fragments (reads) rapidly, as well as the accuracy of the Smith–Waterman algorithm, which is exclusively employed to deal with the most ambiguous and shortest reads. Experimental results on platforms with Intel multicore processors show that HPG-Methyl significantly outperforms in both execution time and sensitivity state-of-the-art software such as Bismark, BS-Seeker or BSMAP, particularly for long bisulphite reads.

Availability and implementation: Software in the form of C libraries and functions, together with instructions to compile and execute this software. Available by sftp to anonymous@clariano.uv.es (password 'anonymous').

Contact: juan.orduna@uv.es or jdopazo@cipf.es

1 Introduction

DNA methylation is an important mechanism of epigenetic regulation in development and disease. It is a heritable modifiable chemical process that affects gene transcription, and it is associated with other molecular markers (e.g. gene expression) and phenotypes (e.g. cancer or other diseases) (Jones, 2013). Although many methods for DNA methylation profiling have been developed, only bisulphite sequencing gives rise to comprehensive DNA methylation maps at single-base pair resolution (Laird, 2010). Bisulphite treatment converts unmethylated cytosines (Cs) into thymines, which gives rise to C-to-T polymorphisms after subsequent polymerase chain reaction (PCR) amplification, while leaving methylated cytosines unchanged. By aligning and comparing bisulphite sequencing reads to the genomic DNA sequence, it is possible to infer DNA methylation patterns at base pair-resolution.

Moreover, the introduction of new DNA sequencing technology, known as Next-Generation Sequencing (NGS), now makes it possible to sequence the genomic DNA in a few days, as well as at a very low cost. Current NGS sequencers can sequence short DNA or RNA fragments of lengths usually between 50 and 400 nt, though new sequencers with longer fragment sizes are being developed. Primary data produced by NGS sequencers consist of hundreds of

millions or even billions of short DNA fragments which are called reads. This big data trend has shifted the pressure from the sequencers to the software analysis tools (Fonseca *et al.*, 2012), which should be scalable enough to process increasing volumes of methylation data with acceptable sensitivity and reasonable execution times. Several software tools for methylation analysis have been proposed, amongst which Bismark (Krueger and Andrews, 2011), BS Seeker (Chen *et al.*, 2010), BSMAP (Xi and Li, 2009) and RRBSMAP (Xi *et al.*, 2012) can be cited as the state-of-the-art, due to their sensitivity and/or execution times. However, as sequencers are expected to provide longer and longer reads, the performance of these software tools decreases either in sensitivity and/or in execution times. Thus, there is an urgent need for a software tool that properly scales not only the reads lengths but also the amount of reads to be processed.

In this article, we present HPG-Methyl, a new software tool for mapping and determining the methylation state of bisulphite reads. Like other existing tools (Krueger and Andrews, 2011), HPG-Methyl is based on coding both the reference genome and the reads to be mapped with only three ASCII characters, in order to avoid the mapping problems generated by the methylated cytosines. However, it includes an innovative strategy that combines an efficient algorithm for those reads with a low rate of mutation errors, insertions or deletions (EIDs), and an algorithm with notably improved sensitivity that correctly aligns reads with a high rate of EIDs. This software tool (HPG-Methyl) shows excellent sensitivity and remarkable parallel performance for both short and long bisulphite reads, presenting runtimes that linearly depend on the number and the length of reads. HPG-Methyl uses a parallel pipeline similar to HPG-Aligner (Martínez *et al.*, 2013; Tárraga *et al.*, 2014). This parallel pipeline aligns the reads first by using the Burrows–Wheeler Transform (BWT) (Li and Durbin, 2009), next reads that are still unmapped are aligned with the Smith–Waterman algorithm (SWA) (Smith and Waterman, 1981). As BWT is faster but less precise than SWA, we employ the former in the early stages of the process, to obtain a rapid mapping of a large number of reads (those which contain few EIDs). SWA is applied in the final stages, to reliably map ambiguous reads. However, HPG-Methyl adds some important improvements with respect to the parallel pipeline of the HPG-Aligner: firstly, in this case the input data are bisulphite reads, rather than RNA reads. Therefore, different transformations are carried out on each read in order for them to be correctly processed. Secondly, under certain conditions the reads which are unmapped after the BWT stage are processed in a different way, rather than using the SWA. Hence, the cost of aligning these reads is greatly reduced, making this software scalable with the length of the reads.

The rest of the article is organized as follows: first, Section 2 describes the approach used for mapping the bisulphite reads. Next, Section 3 describes the baseline version of the parallel pipeline implemented by HPG-Methyl, as well as different improvements added to the baseline version of the parallel pipeline, in order to develop software that is scalable with the length of the reads, while keeping the same sensitivity. Section 4 presents detailed experimentation, evaluating the performance of the proposed software and comparing it against the most commonly used software tools for the analysis of methylation data. Moreover, comparisons are performed with simulated datasets as well as real datasets. Finally, Section 5 presents some concluding remarks and future work to be carried out.

2 Approach

Bisulphite treatment converts unmethylated cytosines (Cs) into thymines, which gives rise to C-to-T polymorphisms after subsequent

Fig. 1. DNA sequence and bisulphite-treated reads produced by NGS

PCR amplification, while leaving methylated cytosines unchanged. Nevertheless, PCR amplification can produce methylated sequences from the negative thread, and there are several possibilities. For example, let us consider the methylated DNA sequence shown in the upper left-hand part of Figure 1. We have coloured the methylated cytosines of that sequence in red, the non-methylated cytosines in yellow and the complementary guanines (in the opposite strand) of the non-methylated cytosines in green.

The upper right-hand part of Figure 1 shows the bisulphite-treated reads that a sequencer would obtain from the original sequence. The copy would have all the non-methylated cytosines turned into uracils (thymines). Finally, the lower part of this figure shows the four possible reads after PCR amplification. These four possibilities include the bisulphite-treated reads and their complementary reads (depending on the strand on which the PCR is applied). The first two sequences correspond to the reads derived from the forward strand (the direct and complementary sequence, respectively), and the other two to the reads derived from the reverse strand.

The problem that arises when trying to align any of these four possible bisulphite reads to the reference genome sequences shown in the upper left-hand part of Figure 1 (which do not include information about methylated cytosines, nor are they bisulphite treated) is that none of them completely matches the correct positions in the reference genome sequence. This is due to the fact that the original non-methylated Cs in the reference genome were converted to Ts in the bisulphite reads (also coloured in yellow in order to show the corresponding nucleotide in the reference genome), or their complementary Gs in the opposite strand in the reference genome are now As in the reads (also coloured in green). Thus, for example, the last read in the lower part of Figure 1 matches the forward strand of the reference genome sequence, except for the last nucleotide (A), which in the forward strand of the reference genome is a G.

In order to avoid this alignment problem, we propose an alignment based on an alphabet composed of three letters (ACT or AGT) instead of a four-letter alphabet (ACGT). This three-letter alphabet should be applied to both the reference genome and the bisulphite reads. The first step is to first convert all the Cs in the reference genome into Ts, providing what we will denote as *Genome_CT*, and then convert all the Gs in the original reference genome into As. We will denote this latter conversion as *Genome_GA*. As the reference genome contains only the forward strand, the sequence example of the reference genome shown in the upper left-hand part of Figure 1 would be converted into the sequences *ATTGTATG* (Genome_CT) and *ACTACATA* (Genome_GA).

The second step is to obtain the four possible versions of each read [the two possible conversions (C-T or G-A) and their reverse complement sequences]. Thus, if we consider the first of the four bisulphite, PCR-amplified reads shown in the lower part of Figure 1 (sequence *ACTGTATG*), the second step will generate the reads

ATTGTATG (C-T conversion, denoted as *read_CT*), *TAACATAC* (reverse complement of C-T conversion, denoted as *read_comp_CT*), *ACTATATA* (G-A conversion, denoted as *read_GA*) and *TGATATAT* (reverse complement of G-A conversion, denoted as *read_comp_GA*). The reverse complement sequences of the converted reads are needed because the reference genome only contains the forward strand; therefore, we need the reverse complement version of the read in order to align it to the reference genome assuming that it matches the reverse strand. Worth mentioning is that if we obtain the 16 possible conversions for the four bisulphite, PCR-amplified reads, only 8 different reads will be generated.

Once all the conversions are performed, a key aspect for the efficient alignment of the reads is the alignment strategy. As the reference genome only contains the forward strand, it may seem that the best way to take advantage of current multicore architectures is to simultaneously perform the alignment of the read and its complementary sequence on the reference genome. However, this strategy cannot be applied to bisulphite-treated reads, as they will not perfectly match the reference genome, as shown in the example in Figure 1. Instead, we propose the alignment of the reads in the genome version that has the same alphabet. That is, the reads denoted as read_CT and read_comp_GA, that contains the alphabet {AGT}, will be aligned on the Genome_CT version of the reference genome, and the other two reads will be aligned on the Genome_GA version. The alignments found for read_CT and read_GA will be assumed to occur on the forward strand, whereas the alignments found for the read_comp_CT and read_comp_GA will be assumed to occur on the reverse strand, as they are complementary to the converted original reads. This strategy will result in a set of four possible alignments for each read.

3 Methods

3.1 Baseline parallel framework

HPG-Methyl implements the alignment strategy described in Section 2. However, in order to take advantage of the parallel architectures available in current computers, it follows a very similar parallel pipeline to the one used by HPG-Aligner (Martínez *et al.*, 2013; Tárraga *et al.*, 2014), but adapted to the specific features of methylation analysis. The implementation described in this section represents the baseline version of HPG-Methyl that will be used later as a reference. In this section, we describe the parallel pipeline and the specific features that this pipeline contains for methylation analysis. One of the main features is the fact that the transformation of the original bisulphite read that is being processed is always kept in all the stages of this pipeline. It is necessary to properly detect on which strand the methylation is located, and which weights matrix should be applied when the SWA is applied (SWA stage).

The HPG-Methyl baseline pipeline maps bisulphite reads into the reference genome, with the mapping process being divided into stages A–F, illustrated in Figure 2. The interaction between two consecutive stages follows the well-known producer–consumer model, and it is synchronized through a shared data structure, where the producer inserts work that is to be processed by the consumer.

3.1.1 Stage A: load and generate input files

Initially, the bisulphite reads are stored in a disk file following the standard FASTQ format (Cock *et al.*, 2010). Due to the large number of reads involved per experiment (tens of millions) and the information provided per read, typical file sizes can reach up to 200 GB, greatly exceeding the memory capacity of most current

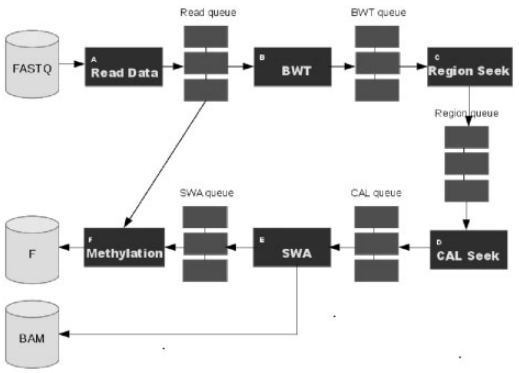

Fig. 2. Pipeline scheme of the Methylation mapper

workstations. Thus, the main task in this stage is to retrieve data from the disk in blocks, hereafter referred to as batches of reads, which can be processed independently from each other. In this way, the pipeline can take advantage of the degree of parallelism available in the underlying computer architecture. The batches are then stored in the Read queue for later use in the subsequent stage. The batches of reads are kept in main memory in an array list, which accommodates fast serial and indexed access. Among other information, this array records the header, sequence, size, and quality of each read. It should be noted that bisulphite reads apparently do not differ very much from other kinds of reads. The only difference is that bisulphite reads contain a much lower percentage of cytosines (only methylated cytosines), as all the nonmethylated cytosines are transformed into thymines.

Besides the original bisulphite reads, we need other input files: a compressed version of the reference genome, a compressed version of the original reference genome with all the Cs converted to Ts (*Genome_CT*) and another compressed version with all the Gs converted to As (*Genome_GA*). It should be noted that all three versions only contain the forward strand of the genome, and therefore the information about the reverse strand will have to be generated by converting the positive one. In order to be properly processed, the files containing the three versions of the reference genome have to first be pre-processed by applying the Burrows–Wheeler algorithm, so obtaining the indexes of all the genome positions through a suffix tree.

Finally, the context information about the Cytosines in the reference genome has to be obtained. Thus, both the Genome_CT and the Genome_GA files are processed as input files, and two binary files denoted as *Context_CT* and *Context_GA* are generated. These binary files code the context information with two bits for each nucleotide in the input file as follows: the value '00' means that the nucleotide under consideration in the input file is not a C in the case of input file Genome_CT, or it codes the absence of a G in the case of input file Genome_GA. The value '01' means that the context is CG (Context_CT) of GC (Context_GA). The value '10' means that the context is CHG (or GHC), and finally the value '11' means that the context is CHH (or GHH), where 'H' means a nucleotide different from G (or different from C in Context_GA). That is, the context information looks for Gs within the two next nucleotides to the right of each C (or it looks for Cs within the two next nucleotides to the left of a G) in the input files.

3.1.2 Stage B: BWT

In this stage, the four sets of possible alignments described in Section 2 for each original read are computed by using the BWT.

As described in Martínez *et al.* (2013), the BWT stage performs a fast mapping of reads to the genome, using our own implementation of the BWT, which allows a single EID within the whole read. The procedure extracts a batch from the read queue, and it applies the four possible conversions to each original bisulphite read [the two possible conversions (C-T or G-A) and their complementary sequences], as described in Section 2. The reads denoted as read_CT and read_comp_GA, which contain the alphabet {AGT}, are mapped onto the Genome_CT version of the reference genome. The other two conversions of each read, which contain the alphabet {ACT}, are aligned on the Genome_GA version. That is, four possible mappings should be searched for each bisulphite read. The mapping is performed by using the BWT-based algorithm, allowing up to 1 EID per read. If the read is successfully mapped, then this stage creates an alignment record for each mapping which identifies the chromosome, among other information, with the initial and final positions of the read within the chromosome, and the strand where the read has been mapped (those mappings found in conversions *read_comp_GA* or *read_comp_CT* are marked to be mapped on the negative strand, whereas those found in conversions *read_CT* or *read_GA* are marked to be mapped on the positive strand. Next, these mappings are transferred directly to the methylation phase). Otherwise, the information that identifies the unmapped read is stored in a target array. The alignment records and the target array make up a single data structure with the batch of reads, which is passed to the next stage once all the reads in the batch have been processed through the BWT queue (which contains both mapped and unmapped reads). Those reads which are unmapped in this stage will be processed in the seeding stage (Stage C), whereas those reads that have been mapped (in any of their conversions) will be processed in the POST PAIR stage. It should be noted that all the possible transformations of the unmapped reads are passed to the next stages.

3.1.3 Stages C and D: seeding and CAL seek

These stages are identical to the analogue stages in the HPG-Aligner software (Martínez *et al.*, 2013). Stage C consists of, given a batch in the BWT queue, processing the unmapped reads, leaving untouched those reads that were already mapped in the previous stage. Each unmapped read from the batch is split in this stage into a number of adjacent 'fragments', hereafter referred to as *seeds*. The real datasets used are the ones in the European Nucleotide Archive (http://www.ebi.ac.uk/ena/data/view/SRR309230 and SRR837425). Next, the BWT-based algorithm is again employed to map each one of these seeds into the reference genome, though this time no EIDs are permitted. Once the full batch of reads has been processed, the results are stored as part of the same data structure and passed to the next stage through the Region queue. Next, Stage D processes the batch, skipping the reads that were previously mapped in stage B. For each unmapped read in the batch, this stage uses the seeds produced by stage C to obtain a list of candidate alignment locations (CALs) or regions, which define potential mappings of that read. We have considered seed lengths, number of seeds and CAL sizes in a proportional way to the average read length in each dataset, in order to properly scale the parallel pipeline to the length of the reads.

3.1.4 Stage E: SWA

The purpose of this stage in HPG-Methyl consists, exclusively, of mapping the read under consideration, starting from the CALs. In order to achieve this goal, we use the SWA to align the whole read against each CAL region of the genome. The SWA will return a numerical value indicating the degree of similarity between the strings under consideration (the read and the CAL region of the genome). This value will have to be higher than a threshold value (determined by the length of the read), in order to consider that the read has been aligned. Nevertheless, the SWA should be properly modified with respect to the implementation used in Martínez *et al.* (2013) in order to keep its well-known sensitivity in the case of methylation analysis. Specifically, the matrix of relative weights assigned to the comparisons of characters of the two strings under consideration should be modified, to take into account on which transformation the CAL was found. If the CAL region was found on read_CT or read_comp_GA transformations, which contain the alphabet {AGT}, then the original bisulphite read should be aligned on the Genome_CT. Table 1 should be used to compute the numerical value for the alignment. The first row on this table shows the four possible nucleotides in the reference genome, and the left-most column shows the possible nucleotides in the string to be aligned. The values in the matrix correspond to the relative weights assigned to all the possible occurrences of comparisons between characters, taking into account the probability of appearance of that symbol on both the read and the genome.

Table 1. Relative weight of matches and mismatches used in the SWA

		Genome				
		A	C	G	T	N
Read	A	5	−4	−4	−4	−4
	C	−400	500	−400	−400	−4
	G	−4	−4	5	−4	−4
	T	−2	2.5	−2	2.5	−4
	N	−4	−4	−4	−4	−4

As we are looking at methylated reads, the occurrence of a C in both strings denotes that a methylated cytosine has been aligned, and therefore the alignment of that string should be 'rewarded'. Additionally, the mismatches of Cs with Ts and Ts with Cs should be 'punished' to a lower degree, as they represent the finding of non-methylated cytosines. An analogous table should be used when the transformations on which the CAL have been found are read_GA or read_comp_CT. In this case, the original bisulphite read should be aligned on the Genome_GA version of the genome.

3.1.5 Stage F: methylation status

The output of the previous stage consists of two lists of alignments for each read, one coming from the *read_CT* and *read_comp_CT* conversions, denoted as *CT_List*, and the other from the *read_GA* and *read_comp_GA* conversions, denoted as *GA_List*. The first step in this stage is to filter all the duplicate alignments that are present in both lists, in order to remove redundant information. Next, for each alignment in these lists, the second step consists of annotating the conversion and the strand on which the alignment was found. Depending on these two criteria, we use the methylation context information contained in the *Context_CT* and *Context_GA* files to determine the methylation present in that read. If the alignment is in the *CT_List* and is in the forward strand, or the alignment is in the *GA_List* and it is in the reverse strand, then the methylation context file *Context_CT* will be used, meaning the methylation is in the same strand as the alignment. On the other hand, if the alignment is in the *GA_List* and is in the forward strand, or the alignment is in

the *CT_List* and is in the reverse strand, then the methylation context file *Context_GA* will be used, meaning the methylation is in the opposite strand to the alignment. The methylation information is then written on an output text file, denoted as F in Figure 2.

3.2 Improved parallel framework

We have added different improvements to the baseline version shown above. In this section, we describe these improvements. The stages that are not modified in the improved version (with regard to the baseline version) are not mentioned, but they are present in the improved version of the pipeline.

3.2.1 Stage B: BWT

We have improved this stage by exploiting the information present in the original bisulphite read, thus avoiding the unnecessary mapping processes of those transformations that are very unlikely to happen. In order to achieve this goal, HPG-Methyl computes the number of Cs and Gs in the original read prior to the alignment of the four possible transformations of the original read. The idea is to avoid the alignment of certain transformations of the original read, depending on the relative number of appearances of Cs and Gs in the read. We have defined N_c as the number of Cs in the original bisulphite-treated read, divided by the sum of Cs and Gs in the original read. In the same way, N_g is computed as the number of Gs divided by the sum of Cs and Gs in the original read. We have experimentally analyzed the reads in real datasets (SRR309230), obtaining in all cases histograms very similar to the ones shown in Figures 3 and 4. Specifically, the *X*-axis on these figures shows the percentage values that each variable (N_c and N_g) can reach, with a precision of 0.01 (that is, from 0.00 to 1.00). On the *Y*-axis, these figures show the number of reads in the real dataset that show each of the percentage values and have been mapped.

Figures 3 and 4 show that the vast majority of reads have a high percentage of either Gs or Cs, over the sum of both. Thus, we have developed a 'filter' to process only the most likely transformations. In this stage, if N_g is lower than a threshold value *Th* (experimentally set to 0.8, with regard to the results shown in the figures), then the *read_CT* and *read_comp_CT* transformations are aligned against *Genome_CT* and *Genome_GA*, respectively. Similarly, if N_c is lower than the threshold value *Th*, then the *read_GA* and *read_comp_GA* transformations are aligned against *Genome_GA* and *Genome_CT*, respectively (it should be noted that the histograms indicate when the threshold value is exceeded, the opposite transformations must be avoided because most of the reads only need two of the transformations). As $N_c = 1 - N_g$, the alignment of the four possible transformations of the original read will be performed when both N_c and N_g are lower than *Th*. Let us assume, for example, that $Th = 0.8$, $N_c = 0.9$, and $N_g = 0.1$. In this case, N_g is lower than *Th* but N_c is higher than *Th*. Therefore, only the *read_CT* and *read_comp_CT* transformations will be aligned, as the histograms suggest. We have denoted this version of the BWT stage as the 'filter'. It should be noted that the idea behind this filtering can be applied not only to the BWT but also to any software or algorithms that process bisulphite-treated reads.

3.2.2 Stage E: SWA/direct search

We have added another improvement in the SWA stage, which takes advantage of the information provided by stages C and D. One of the main problems with using the SWA is that its computational cost is proportional to the length of the reads being analyzed. This feature prevents any software based on the SWA from achieving scalability

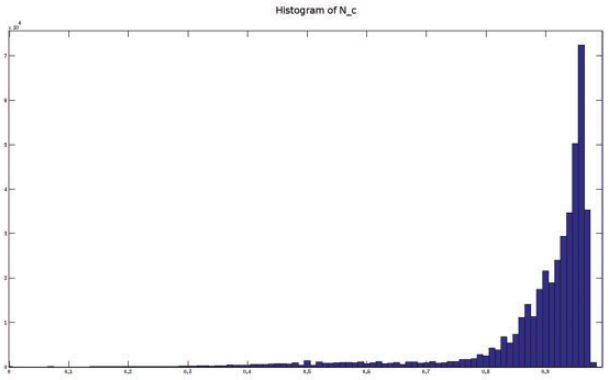

Fig. 3. Histogram for N_c

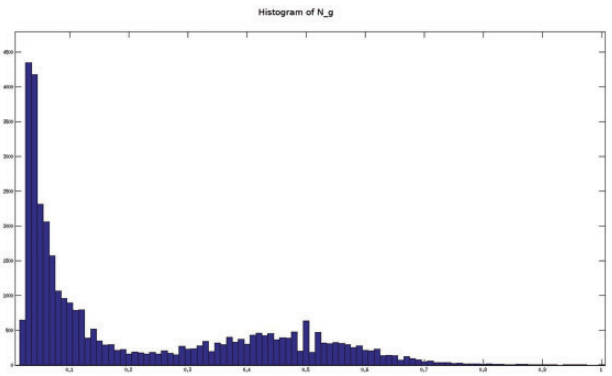

Fig. 4. Histogram for N_g

with the length of the read. In order to minimize this problem, we have modified the SWA stage. The CAL stage provides a CAL region formed by some seeds (that have been already aligned with BWT algorithm) and some regions located between seeds. In the improved version of the SWA stage, which we have denoted as 'SW' stage, the SWA algorithm is not executed for the entire read. Instead, the length of each region (in the genome) between two consecutive seeds (already aligned by means of BWT) is analyzed. If the length of that region in the genome is identical to the length of the region in the read to be aligned, then this probably means that neither insertions nor deletions are found, and only mismatches are found. Therefore, a direct comparison between the characters of the genome and the characters in the read is performed for that region. If the length of the region between the seeds is different, then the SWA is applied to align that region of the read on that region of the genome. This process is repeated until all the regions between two consecutive seeds are processed. If there are still some unmapped regions at the beginning or end of the read, the same process is applied, until the whole read is aligned. As the SW stage only uses the SWA for regions between seeds, the length of the strings analyzed by the SWA no longer depends on the length of the read, but on the length of the unmapped regions in the CAL. Additionally, there are some regions where SWA is not used. As a result, the pipeline becomes more scalable with the length of the reads, as shown in Section 4.

4 Performance evaluation

This section presents a comparative performance evaluation among the different versions of HPG-Methyl, as well as a comparative performance evaluation between HPG-Methyl and other well-known

Methylation software. We have measured the performance of the software in terms of both sensitivity and execution time. For performance evaluation purposes, we have used synthetic datasets, artificially extracted from the reference genome. In this way, we know a priori the correct alignment for each of the reads. Also, we have used real datasets obtained from the European Nucleotide Archive (SRR309230 and SRR837425). In order to make the proposed software publicly available, the source code of the two versions of the HPG-Methyl software (baseline and improved), as well as all the datasets (both real and synthetic) used to obtain the results shown in this section, can be downloaded by an anonymous sftp (password 'anonymous') from the host clariano.uv.es. The two versions of the source code are packed in each.tar.gz compressed file in the /src directory. There is also a README.txt file explaining the installation instructions, the compilation and execution options of the source code, the parameter values used in the HPG-Methyl software to obtain the results shown in this section, as well as the commands used for generating the synthetic datasets from the human genome.

Unlike other existing software tools (for example, BSMAP can only use eight cores), HPG-Methyl can take advantage of all the existing cores in current multicore processors, providing shorter execution times particularly for datasets made up of very long reads. Therefore, instead of running all the tools using the same number of processor cores (therefore limiting the number of cores to the one that the worst tool can exploit), we have used as many processor cores as each piece of software allows, for maximum speedup.

4.1 Version comparison

Figure 5 shows the comparative performance evaluation, in terms of mapping efficiency (sensitivity) achieved by the different versions of HPG-Methyl when executed with a synthetic dataset of different read lengths. All the datasets in this case have mutation rate of 0.1% with respect to the reference genome, and they are composed of 2 million reads. The *X*-axis shows the read lengths under consideration in the different synthetic datasets, and the *Y*-axis shows the percentage of correct mappings achieved by HPG-Methyl. We have looked at three different versions of HPG-Methyl for evaluation purposes: the baseline version, corresponding to the plot labelled 'original' in the figure, the SW version, and the version including SW and the filter in the BWT stage. Each of the plots in the figure shows one of these versions. Figure 5 shows that there are no significant differences among the three versions in terms of sensitivity when the mutation rate is 0.1%.

Figure 6 shows the execution times (in minutes) required by each version of HPG-Methyl to process the datasets. This figure shows the execution time in minutes on the *Y*-axis. The figure shows that for read lengths shorter than 250 nt the SW and filter additions do not actually improve on the execution times achieved by the baseline version. However, for large read lengths (400 nt) the performance of the baseline version largely decreases, whereas the improved versions clearly outperform the baseline version. Nevertheless, these results show short execution times, and therefore any variation does not represent a significant amount of time in absolute values.

Figure 7 shows the results, in terms of mapping efficiency (sensitivity), achieved by the different versions of HPG-Methyl for datasets with a mutation rate of 1% with respect to the reference genome. This figure shows that for a high rate of mutations, the sensitivity of both the SW and the SW + filter versions is identical, and lower than the baseline version. That is, the application of the SWA to the entire read provides better sensitivity. Also, as could be expected, the filter does not affect sensitivity. Nevertheless, it is worth

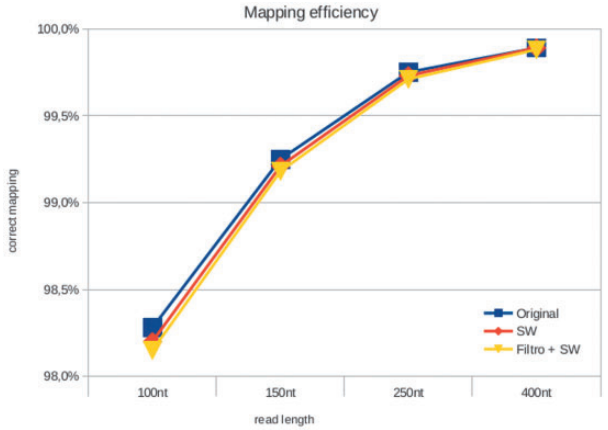

Fig. 5. Software sensitivity for a synthetic dataset with 0.1% of mutations

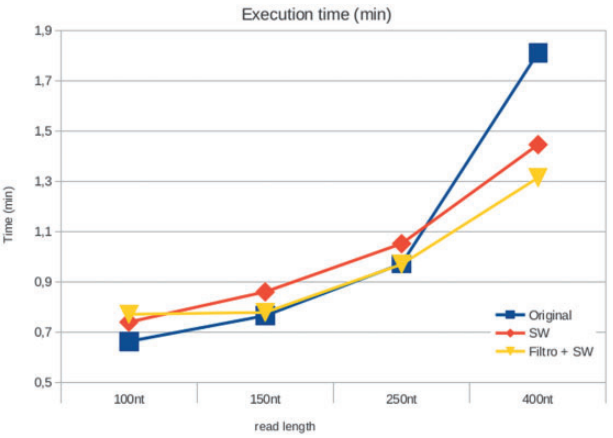

Fig. 6. Execution times for a synthetic dataset with 0.1% of mutations

mentioning that the sensitivity lost by the improved version does not exceed 3% for short reads, and this difference decreases with the length of the reads.

Finally, Figure 8 shows the execution times (in minutes) required by each version of HPG-Methyl to process the datasets. In this case, the plots are similar to those shown in Figure 6, except for the fact that the plot for the version including the SW and the filter stages provides the shortest execution times for all the lengths under consideration. Moreover, for lengths of 250 and 400 nt, the differences between this version and the other ones clearly increase, indicating that this version is the most scalable with the length of the reads. These results show that the most efficient version of HPG-Methyl is the one including the SW and the filter stages, as it slightly reduces software sensitivity, but greatly reduces the required execution time. Additionally, the loss of sensitivity tends to decrease as the length of the reads increases. Thus, in the rest of the article the version including the SW and the filter stages will be denoted as HPG-Methyl.

4.2 Comparative study

In this subsection, we present a comparative study of HPG-Methyl with other existing software tools for methylation analysis. We have analyzed synthetic datasets with read lengths ranging from 100 to 800 nt. All the tests whose results are shown in this subsection have been executed on a desktop computer comprising of an Intel i7-3930 K processor (http://ark.intel.com/products/63697), with

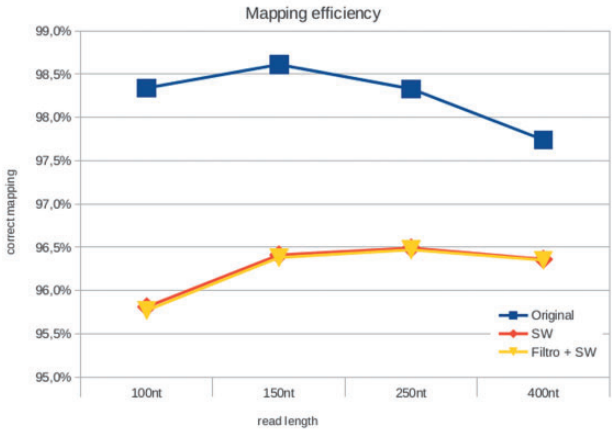

Fig. 7. Software sensitivity for a synthetic dataset with 1% of mutations

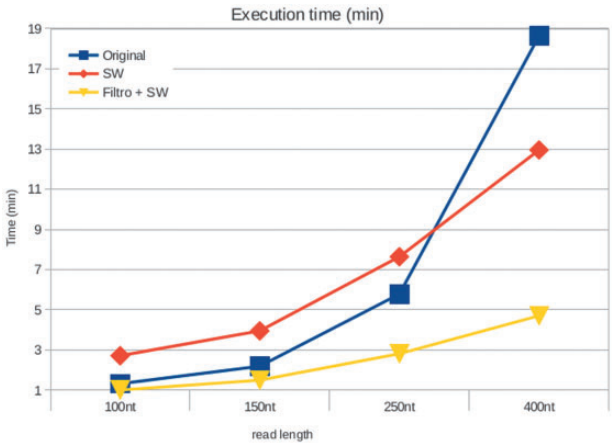

Fig. 8. Execution times for a synthetic dataset with 1% of mutations

Table 2. Comparative study of sensitivity on a synthetic dataset with a mutation rate of 0.1%

Length (nt)	HPG-Methyl		Bismark	
	R	W	R	W
75	96.31	0.06	89.57	0.01
150	99.20	0.03	95.15	0.00
400	99.90	0.02	97.66	0.00
800	99.96	0.01	98.51	0.00
Length (nt)	BSMAP		BS-Seeker	
	R	W	R	W
75	93.9	5.88	92.07	0.11
150	96.92	2.39	96.40	0.03
400	48.49	50.84	98.07	0.01
800	48.78	51.21	—	—

6 cores and 12 threads, 48 GB of RAM, two hard disks each one being 2 TB. Table 2 shows the sensitivity (in terms of the percentage of reads aligned) that HPG-Methyl as well as Bismark, BSMAP and BS-Seeker software have provided for synthetic datasets composed of 4 million reads with 0.1% of mutations. For each of the software tools, there are two columns. The one labelled 'R' shows the percentage of reads that have been correctly aligned, and the column labelled

Table 3. Comparative study of execution times (min) required for processing a synthetic dataset with a mutation rate of 0.1%

Length (nt)	HPG-Methyl	Bismark	BSMAP	BS-Seeker
75	1.283	63.436	3.464	118.229
150	1.600	106.338	3.230	139.766
400	2.916	244.733	3.464	315.765
800	9.266	1220.34	3.530	—

'W' shows the percentage of reads that have been wrongly aligned. The sum of both columns is the total percentage of reads aligned. HPG-Methyl and Bismark software have been executed using 12 threads (all the existing ones in the machine), whereas BSMAP and BS-Seeker are only able to use 8 of the 12 threads. The '—' value means that the executions with datasets of that length did not finish within three days, and were aborted because it makes no sense to compare software tools with such long execution times. Table 2 shows that HPG-Methyl provides the greatest sensitivity; the percentage of correctly aligned reads being greater than 96% in the worst case. Also worth mentioning is that HPG-Methyl yields incorrect alignments for less than 0.06% of the reads, regardless of the read length. Although Bismark yields a lower percentage of incorrectly aligned reads, it yields a slightly lower percentage of correctly aligned reads. Additionally, it can be seen that HPG-Methyl yields a lower percentage of incorrectly aligned reads than BSMAP, particularly for long reads (400 and 800 nt). Even for a short read length of 75 nt, HPG-Methyl yields incorrect alignments for 0.01% of the reads, whereas BSMAP yields incorrect alignments for 5.88% of the reads (a factor of more than 58). This table also shows that HPG-Methyl, Bismark and BS-Seeker yield similar sensitivity for all the read lengths under consideration. However, BSMAP yields half the sensitivity for read lengths of 400 and 800 nt. The reason for this behaviour is that the maximum read length that BSMAP can process is 144 nt. For longer reads, this software exclusively uses the first 144 nt. Therefore, it cannot yield good sensitivity for long reads.

Table 3 shows the execution times (in minutes) required to process the synthetic dataset. Again, the '—' value means that the execution for that read length was aborted after 3 days. These results show that HPG-Methyl is the fastest software for read lengths of 75 and 150 nt, followed by BSMAP, which requires longer execution times. The other two software tools provide execution times of 1 or even 2 orders of magnitude. For read lengths of 400 and 800 nt, BSMAP requires very similar execution times to those required for the shortest lengths, as this tool exclusively processes the first 144 nt of each read, and therefore longer reads do not affect the required execution time. Among the rest of the tools, HPG-Methyl requires execution times 2 orders of magnitude lower than Bismark or BS-Seeker. If we take into account both the sensitivity and the execution times, these results show that HPG-Methyl yields the best results for this dataset, as BSMAP only can process datasets with read lengths lower than 145 nt. We have performed the same evaluation with other synthetic datasets, and the results obtained are similar; however, the results are not shown here for the sake of brevity.

We have also tested the tools with a similar synthetic dataset including a higher rate of mutations (1%), in order to test the robustness of the results. Table 4 shows the sensitivity achieved by the software for this mutation rate. This table shows that HPG-Methyl yields higher percentages of correctly aligned reads than the rest of the tools for short read lengths (75 and 150 nt). For long reads, HPG-Methyl yields similar percentages of correctly aligned reads to Bismark, and higher percentages than BSMAP or BS-Seeker. Also, a

Table 4. Comparative study of sensitivity on a synthetic dataset with a mutation rate of 1%

Length (nt)	HPG-Methyl		Bismark	
	R	W	R	W
75	93.37	0.62	88.30	0.1
150	96.87	0.80	94.59	0.08
400	97.55	0.48	97.55	0.1
800	97.58	0.43	98.45	0.08
Length (nt)	BSMAP		BS-Seeker	
	R	W	R	W
75	91.36	6.5	89.11	1.71
150	90.68	2.67	92.68	0.37
400	45.29	48.05	76.22	0.08
800	48.87	51.13	—	—

comparison of this table with Table 2 shows that the tools under consideration do not yield very different sensitivities when the rate of mutations increases.

Next, Table 5 shows the required execution times (in minutes) when the rate of mutations is 1%. This table shows that HPG-Methyl requires the shortest execution times for read lengths of 75 and 150 nt. BSMAP requires constant execution times, and HPG-Methyl requires execution times 1 order of magnitude lower than those required by Bismark or BS-Seeker. These results are similar to those shown in Table 3, demonstrating that a significant mutation rate does not affect the performance of HPG-Methyl. Again, if we take into account both sensitivity and execution times, these results also show that HPG-Methyl yields the best results.

We have tested the behaviour of the software tools when the number of reads in the dataset increases, not shown here due to space limitations, and we have seen that the execution time required by all the tools increases in a quadratic way with the number of reads in the dataset, as could be expected.

Finally, we have tested the programmes with current real datasets (SRR309230 and SRR837425). These two datasets contain 16.6 million *Homo sapiens* bisulphite reads of 75 and 100 nt, respectively. Table 6 shows the percentage of reads in the datasets that have been mapped, provided by each of the programmes for these real datasets. This table shows that HPG-Methyl and BSMAP yield similar sensitivities for both datasets, and these sensitivities are higher than those yielded by Bismark and BS-Seeker. In the case of 75 nt long reads, BSMAP seems to be slightly superior to HPG-Methyl. However, reads used currently are longer than 100 nt, a length for which HPG-Methyl provides a better percentage than BSMAP. It must be taken into account that the percentage of incorrectly aligned reads yielded by HPG-Methyl is, at most, half of that provided by BSMAP for synthetic datasets (Tables 2 and 4). Therefore, if we assume similar behaviour for the 100 nt long reads, then it is very likely that HPG-Methyl will yield a greater percentage of correctly aligned reads compared to BSMAP. Table 7 shows the execution times, (measured in minutes) required to provide the alignments whose sensitivity is shown in Table 6. This table shows that BSMAP requires the shortest execution times, whereas HPG-Methyl requires slightly longer execution times and the other two programmes require much longer ones. These results, together with the sensitivity results, show that HPG-Methyl yields good sensitivity while requiring short execution times for dataset with short reads, similar to the ones provided by BSMAP.

Table 5. Comparative study of execution times (min) for a synthetic dataset with a mutation rate of 1%

Length (nt)	HPG-Methyl	Bismark	BSMAP	BS-Seeker
75	1.366	62.579	4.601	116.044
150	1.950	106.173	6.220	141.442
400	10.850	248.207	4.601	324.616
800	50.600	1246.89	4.159	—

Table 6. Comparative study of percentage of reads mapped on real datasets

Dataset	HPG-Methyl	Bismark	BSMAP	BS-Seeker
SRR309230_1	87.71	71.81	89.21	84.18
SRR837425_1	82.75	68.42	82.84	77.35

Table 7. Comparative study of execution times (min) for real datasets

Dataset	HPG-Methyl	Bismark	BSMAP	BS-Seeker
SRR309230_1	12.053	82.120	9.975	250.686
SRR837425_1	19.047	95.194	15.286	271.048

5 Conclusions

HPG-Methyl is a new software tool for mapping and determining the methylation state of bisulphite reads. HPG-Methyl shows excellent sensitivity and remarkable parallel performance for both short and long bisulphite reads, presenting runtimes that linearly depend on the number and the length of reads. HPG-Methyl adds some important contributions for processing bisulphite reads: first, different transformations are carried out on each read in order to correctly process them. Second, all the reads are first processed with the BWT, and then under certain conditions the reads which are unmapped after the BWT stage are directly processed. The use of SWA is reduced to some strings of some reads, in order to make this software scalable with the length of the reads. The performance evaluation results show that HPG-Methyl yields good performance for current, real datasets as well as for synthetic datasets containing reads of relatively short lengths when compared to some other well-known programmes for methylation analysis. Nevertheless, the results for synthetic datasets containing long reads show that HPG-Methyl yields the best sensitivity, while requiring one of the shortest execution times. These results validate HPG-Methyl as a sensitive and efficient software tool able to cope with real datasets with the longer reads which are expected in the near future.

Acknowledgements

This work was supported by grants BIO2011-27069 from the Spanish MINECO and PROMETEOII/2014/025 from the Valencia Regional Government. We also thank the support of the Spanish INB.

Funding

This work was supported by grants BIO2011-27069 from the Spanish MINECO and PROMETEOII/2014/025 from the Valencia Regional Government and by Spanish INB.

Conflict of Interest: none declared.

References

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

Cock,P.J.A. *et al*. (2010) The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic Acids Res.*, **38**, 1767–1771.

Fonseca,N.A. *et al*. (2012) Tools for mapping high-throughput sequencing data. *Bioinformatics*, **28**, 3169–3177.

Jones,P.A. (2013) Functions of DNA methylation: islands, start sites, gene bodies and beyond. *Nat. Rev. Genet.*, **13**, 484–492.

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

Laird,P.W. (2010) Principles and challenges of genome-wide DNA methylation analysis. *Nat. Rev. Genet.*, **11**, 191–203.

Li,H. and Durbin,R. (2009) Fast and accurate short read alignment with burrows-wheeler transform. *Bioinformatics*, **25**, 1754–1760.

Martínez,H. *et al*. (2013) Concurrent and accurate RNA sequencing on multicore platforms. *Technical report icc 2013-03-01*. Universitat Jaume I, Castelln, Spain.

Smith,T.F. and Waterman,M.S. (1981) Identification of common molecular subsequences. *J. Mol. Biol.*, **147**, 195–197.

Tárraga,J. *et al*. (2014) Acceleration of short and long DNA read mapping without loss of accuracy using suffix array. *Bioinformatics*, **30**, 3396–3398.

Xi,Y. and Li,W. (2009) BSMAP: whole genome bisulfite sequence MAPping program. *BMC Bioinformatics*, **10**, 232+.

Xi,Y. *et al*. (2012) Rrbsmap: a fast, accurate and user-friendly alignment tool for reduced representation bisulfite sequencing. *Bioinformatics*, **28**, 430–432.