BIOINFORMATICS ORIGINAL PAPER

Vol. 30 no. 13 2014, pages 1837–1843
doi:10.1093/bioinformatics/btu146

Sequence analysis

Advance Access publication March 13, 2014

Lacking alignments? The next-generation sequencing mapper segemehl revisited

Christian Otto[1,2], Peter F. Stadler[2,3,4,5,6,7,8] and Steve Hoffmann[1,2,*]

[1]Transcriptome Bioinformatics Junior Research Group, LIFE—Leipzig Research Center for Civilization Diseases, [2]Interdisciplinary Center for Bioinformatics, [3]Bioinformatics Group, Department of Computer Science, University Leipzig, [4]RNomics Group, Fraunhofer Institute for Cell Therapy and Immunology, Leipzig, Germany, [5]Santa Fe Institute, Santa Fe, New Mexico, USA, [6]Department of Theoretical Chemistry, University of Vienna, Austria, [7]Max-Planck-Institute for Mathematics in Sciences, Leipzig, Germany and [8]Center for non-coding RNA in Technology and Health, University of Copenhagen, Denmark

Associate Editor: Inanc Birol

ABSTRACT

Motivation: Next-generation sequencing has become an important tool in molecular biology. Various protocols to investigate genomic, transcriptomic and epigenomic features across virtually all species and tissues have been devised. For most of these experiments, one of the first crucial steps of bioinformatic analysis is the mapping of reads to reference genomes.

Results: Here, we present thorough benchmarks of our read aligner `segemehl` in comparison with other state-of-the-art methods. Furthermore, we introduce the tool `lack` to rescue unmapped RNA-seq reads which works in conjunction with `segemehl` and many other frequently used split-read aligners.

Availability: `lack` is distributed together with `segemehl` and freely available at www.bioinf.uni-leipzig.de/Software/segemehl/.

Contact: steve@bioinf.uni-leipzig.de

Supplementary information: Supplementary data are available at *Bioinformatics* online.

Received on December 11, 2013; revised on February 18, 2014; accepted on March 6, 2014

1 INTRODUCTION

The problem of aligning (short) sequencing reads to (larger) reference genomes has received considerable attention in recent years, and many different alignment tools based on a variety of distinct algorithmic approaches have been published so far. A survey at the EBI currently counts more than 80 different mappers (Fonseca *et al.*, 2012). This competitive field has seen quite a bit of evolution and progress. Early mappers were restricted to aligning short reads with no or few mismatches and reads with insertions and deletions were excluded more often than not. `Bowtie` (Langmead and Salzberg, 2012; Langmead *et al.*, 2009) and `segemehl` (Hoffmann *et al.*, 2009) were among the first next-generation sequencing (NGS) aligners that explicitly implemented strategies for aligning reads with indels. The rise of RNA-seq protocols has added yet another layer of complexity to the problem: splicing. When mapping cDNA reads that join two or more exons, the aligner is required to 'split' the read and align its parts to the appropriate exons in the reference genome. Alternatively, the mapper needs to be provided with junction or paired-end information to predict or construct mRNA references. Today, most of the tools allow only a single split, while reads that span multiple exon–exon junctions may not be properly aligned. As read lengths are constantly increasing, algorithms that allow multiple splits are clearly favorable. `segemehl` facilitates multi-split alignments using a local transition algorithm that was shown to perform very well in simulated and read data benchmarks (Hoffmann *et al.*, 2014). The alignment of reads of bisulfite-treated DNA, i.e. methylcytosine sequencing, also requires specialized algorithms and methods (Chen *et al.*, 2010; Krueger and Andrews, 2011; Smith *et al.*, 2009). The split-read and bisulfite features of `segemehl` have recently been published along with extensive benchmarks (Hoffmann *et al.*, 2014; Otto *et al.*, 2012). The diversity of tools and the rapid development of algorithms and software requires frequent, transparent and reproducible benchmarks. Here, we present the results of performance tests for DNA-seq and RNA-seq read alignments and provide detailed information on them. We have therefore assembled an extensive electronic supplement (http://www.bioinf.uni-leipzig.de/publications/supplements/13-008) comprising all data, custom scripts and detailed descriptions on how to re-run the analyses. In reference to recent heated debates on the comparison of mappers, we would like to stress that benchmarks only measure specific aspects and may not be used to claim any universal superiority or inferiority of any tool. We would like to encourage all readers to reproduce this data and to come up with alternative benchmarks.

In addition to the benchmarking, we propose a novel tool, `lack`, for remapping previously unmapped RNA-seq reads using supplied splice junctions. The workflow of common, split-read and bisulfite mapping using `segemehl` and remapping by `lack` is depicted in Figure 1.

2 RESULTS

2.1 Comparison of read aligners

Because several aligners have limitations in finding multiple hits, we evaluated the performance of alignment programs in two different manners as proposed by Holtgrewe *et al.* (2011).

*To whom correspondence should be addressed.

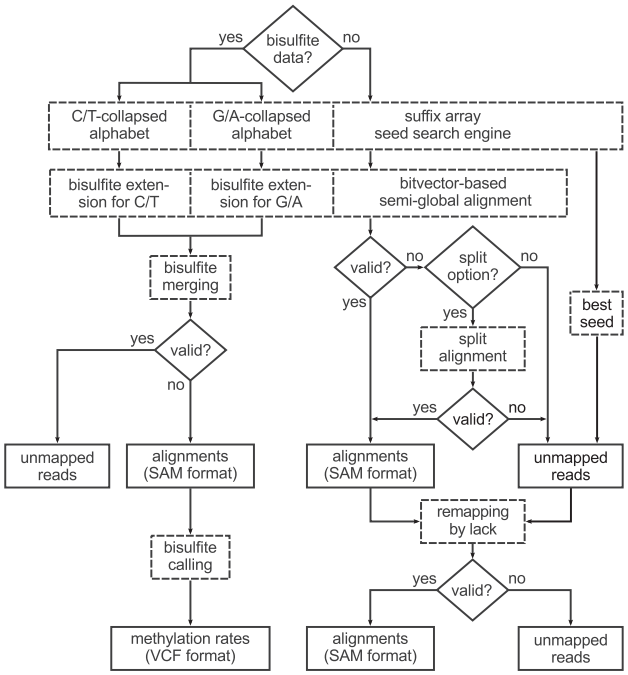

Fig. 1. Workflow of `segemehl` and `lack`. `segemehl` is able to align paired and unpaired DNA-seq, RNA-seq and bisulfite-treated DNA-seq data. Previously unmappped reads can be rescued with `lack`

First, we measured the sensitivity and the number of false positive (FP) alignments for each program in finding 'at least one' optimal hit (any-best) with respect to the unit edit distance. The second benchmark measured the performance in finding 'all' optimal hits (all-best). The tests were carried out on different datasets comprising simulated and real DNA- and RNA-seq datasets from Illumina and 454 sequencing technologies. Simulated data was generated using `Mason` (Holtgrewe, 2010). For the case of Illumina, we evaluated short (~22–30 nt) as well as long (~100 nt) sequencing reads. The median length of 454 reads was 407 nt for simulated and 524 nt for read 454 data. To obtain the complete set of optimal alignments, we used `RazerS 3` (Weese *et al*., 2012). Using a classical pigeonhole principle, the algorithm of `RazerS 3` reportedly guarantees to find all optimal alignments (up to a given maximum number) with an edit distance (mismatches + insertions + deletions) of less than or equal to k. Because of the relatively long computation times for the full sensitivity alignment, it was necessary to sample a smaller set of reads, $\sim 10^5$, for each dataset and estimate sensitivity and number of false positive alignments from this sample (see Section 4 and Supplementary Material).

All aligners were benchmarked both with their default parameters and with parameter settings optimized for sensitivity and number of false positive alignments, respectively.

The evaluation with default parameters shows that their performance varied quite strongly with length and type of the input. Nevertheless, `segemehl` performed better with respect to sensitivity and number of false positive alignments than most of the other tools (Fig. 2A). In the case of Illumina reads, `segemehl` achieved the highest sensitivity for simulated reads as well as real DNA- and mRNA-seq reads. At the same time, it reported the lowest number of sub-optimal alignments with mRNA-seq data (Fig. 2A). This is also the case for paired-end mRNA-seq data. In all-best benchmarks, it outperformed all other tools tested in terms of sensitivity while maintaining low false positive rates (Supplementary Fig. S1 and Table S3). In any-best benchmarks, a better sensitivity was only achieved by `BWA-MEM` on Illumina paired-end DNA-seq data and `BWA` on Illumina shortRNA-seq data (Supplementary Fig. S2 and Table S3) with 0.1 and 5% increase, respectively. In the latter case, `BWA` reported 80% more false positive alignments as compared to `segemehl`. While `segemehl` performed similarly well in the all-best and any-best scenarios, relatively large differences can be observed for `Bowtie 2`, `BWA` and `BWA-MEM` since their default parameters are presumably tailored to find one instead of all optimal alignments with significant effects on the run time.

A larger difference among the read aligners can be observed in Illumina short-read and 454 data (Supplementary Figs S1 and S2). In the first case, `segemehl`'s closest competitor was `BWA`, which, however, achieved low sensitivities in most of the Illumina single-end benchmarks. In all 454 scenarios, `segemehl` and `BWA-MEM` turned out to be the best aligners among the tested tools.

To explore the trade-off between sensitivity and the number of false positive alignments, benchmarks with different parameter settings were carried out. For each tool, we selected those parameter sets with highest sensitivity and lowest number of false positive alignments. Regardless of evaluation type (all-best or any-best) and parameter setting (default, best-sensitivity, best-FP), the sensitivities of `segemehl` exceeded 99% in all datasets except for shortRNA-seq (Supplementary Figs S1–S2 and Tables S3–S5) where `segemehl` still achieved the best or second-best results (>91%). In the comparison of best-FP parameter settings, `segemehl` performed best or second-best in terms of number of false positives in seven out of 10 datasets. The closest competitor of `segemehl` with best-sensitivity settings was `GEM` despite some performance issues with paired-end data. Apart from `segemehl`, several aligners (`BWA-MEM`, `BWA`, `GEM`) showed good performances with best-FP parameter settings, depending on the dataset used. In terms of number of mapped reads, `segemehl` performed comparable or better than the other tools tested (Supplementary Tables S3–S5).

The performance of `segemehl` in terms of sensitivity and false positive rate came at the cost of higher running times and memory consumption. With default parameters, `segemehl` was on average slower than the competitors. `STAR` was the fastest tool in this benchmark (Fig. 2B). Using best-sensitivity settings, however, the running times of several aligners including `Bowtie 2`, `BWA`, `GEM`, became significantly longer (Supplementary Fig. S3 and Table S4). The peak virtual memory footprint of `segemehl` (70 GB) was higher than that of `STAR` (28 GB) and the other aligners (3–6 GB). Unexpectedly, the memory consumption of `GEM` depended on its parameter values, strongly varying in the benchmarks with best-sensitivity settings (4–70 GB). Note that we compared the virtual memory consumption. The required physical memory is considerably smaller. For large mammalian genomes, `segemehl` may not be feasibly applied on computers with <50 GB of memory. The memory consumption of `segemehl` is considerably smaller for smaller genomes:

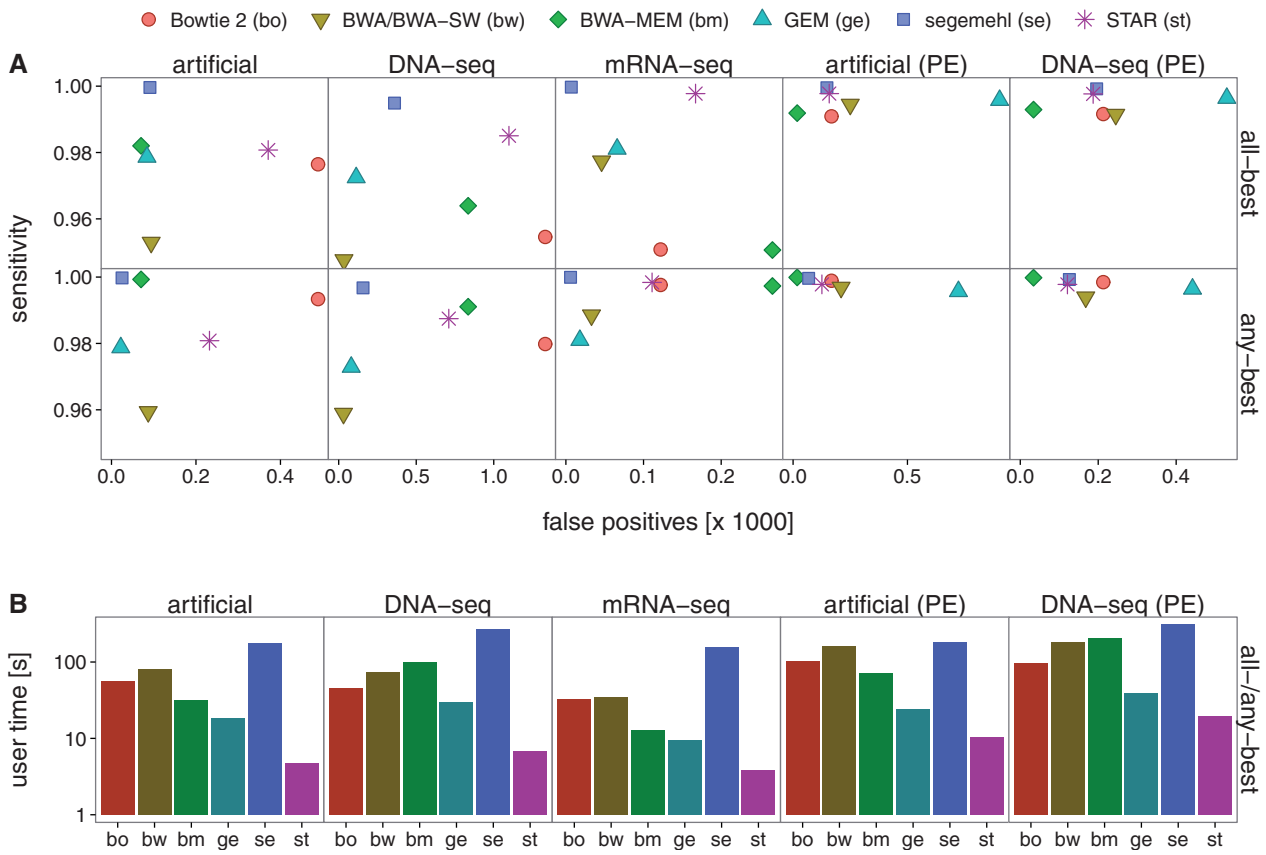

Fig. 2. Comparison of different read aligner with default parameter. The performance is assessed in terms of (A) sensitivity and false positives as well as (B) user time. `segemehl` performed better with respect to sensitivity and number of false positive alignments than most of the other tools with default parameters at the cost of higher running times. The number of reads used for evaluation is given in Supplementary Table S1

Escherichia coli 0.7 GB, *Caenorhabditis elegans* 1.5 GB, *Drosophila melanogaster* 2.6 GB or *Arabidopsis thaliana* 1.8 GB.

Interestingly, throughout all test scenarios we observed a difference between simulated and real data. Most of the aligners achieved higher sensitivities and lower number of false positive alignments with simulated compared to real data. The opposite effect was only present in 454 data which, however, may be caused by differences in the read lengths (407 versus 524 nt for simulated and real data, respectively).

2.2 Rescuing reads with `lack`

The objective of `lack` is to rescue previously unmapped RNA-seq reads that may have emerged from splicing events. It utilizes *de novo* splice junction information from alignments reported by state-of-the-art split-read aligners. In contrast to other methods, `lack` is able to map reads across multiple splice junctions. The benefit of this multi-junction remapping is illustrated in Figure 3B where previously unlinked splice junctions were connected by `lack`-remapping. We tested `lack` on simulated and real Illumina and 454 RNA-seq datasets as well as artificial Ion Torrent data, all of which were initially mapped by `Blat` (Kent, 2002), `segemehl` (Hoffmann *et al.*, 2014) `TopHat 2` (Kim *et al.*, 2013) and `STAR` (Dobin *et al.*, 2013). Details about the evaluation and datasets are given in the Supplementary Material.

Overall, the evaluation shows that `lack` was able to rescue a substantial portion of the previously unmapped reads (Fig. 3A). Using the alignments and unmapped reads of the split-read aligners as input, `lack` was able to rescue 51% of the unmapped reads on average with every split-read aligner and for every Illumina and 454 dataset (Supplementary Table S6). When considering only those unmapped reads that were *de facto* aligning across exon–exon junctions, the benefit of `lack` was more apparent: on average it rescued 70% of them (Supplementary Table S7). The accuracy of the alignments reported by `lack` was high (Supplementary Table S8). The number of splice sites with at least 20 reads increased by 34% on average with Illumina and 454 data (Supplementary Fig. S4A). `lack` performed particularly well with 454 data. In case of real 454 data, there was a considerable number of potential splice sites (394 for `Blat`, 396 for `STAR`, 790 for `TopHat 2`) with >20 additional reads per junction (Supplementary Fig. S4B). Moderate remapping rates in the real Illumina dataset resulted from a number of low-quality reads that were not mapped by any of the four split-read aligners tested. For simulated Ion Torrent data, `lack` was able to achieve an average remapping rate of 45% and an average split-read remapping rate of 69%. However, the rates of `lack`

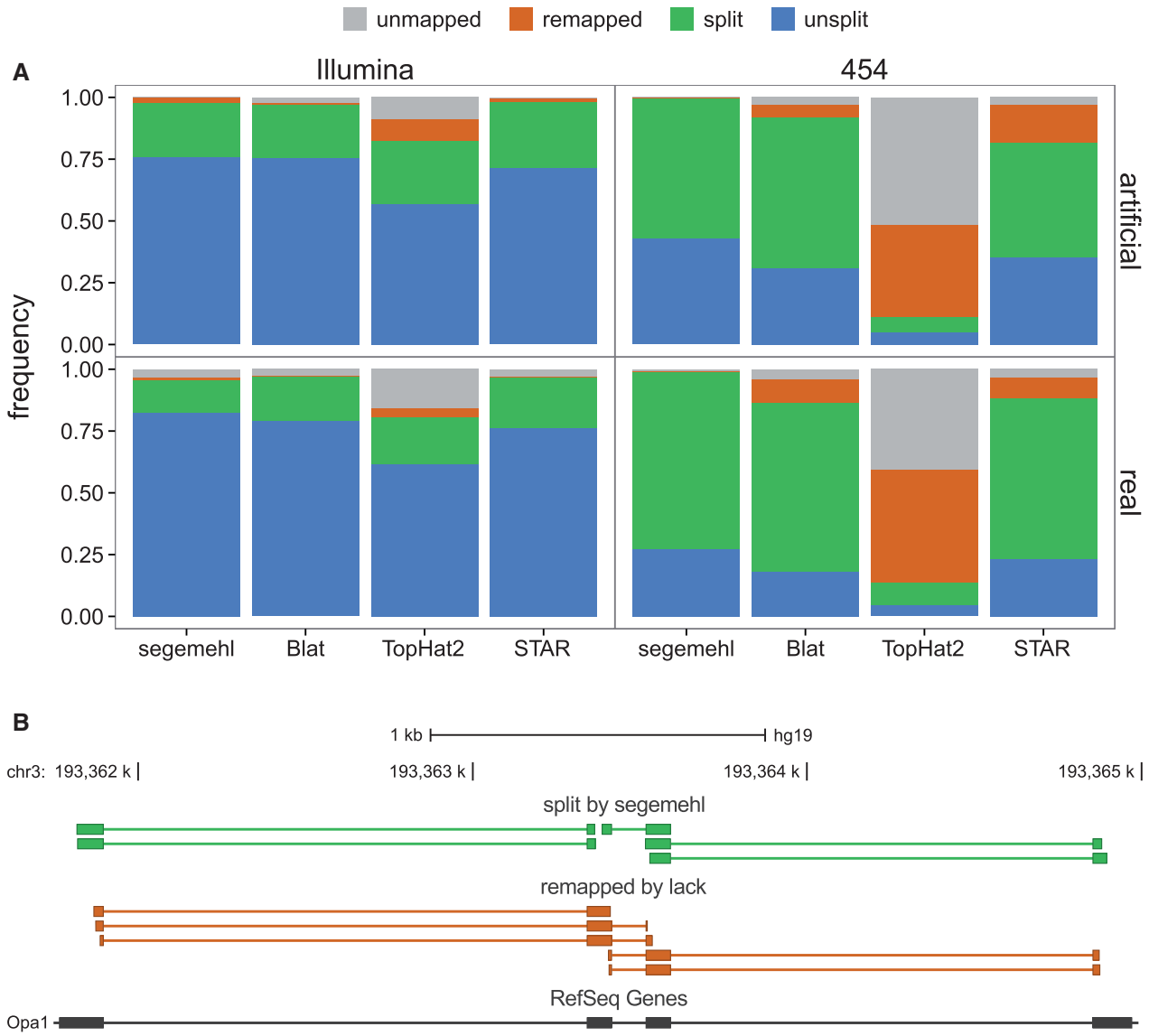

Fig. 3. Performance of `lack`. (A) Frequency of unsplit-mapped and split-mapped reads of different split-read aligners as well as initially unmapped reads recovered by `lack`. Reads that were not mapped by the aligner and `lack` are termed unmapped. All split-read aligners as well as `lack` were executed under default parameters. Overall, `lack` was able to rescue a substantial proportion of unmapped reads missed by the aligner (51% on average). (B) Example on real Illumina data that illustrates the benefits of using `lack`. With three *de novo* splice junctions, extracted from five single-split mapped reads mapped by `segemehl`, `lack` retrieves split-alignments for another five previously unmapped reads, each of which spans over two of three junctions. For the purpose of isoform reconstruction, previously independent splice junctions have become correlated

differed widely with very high to moderate (split-read) remapping rates for `segemehl`, `STAR` and `Blat`. The rather poor remapping rates for `TopHat 2` can be explained by the program's difficulties to split-map reads with high insertion and deletion rates.

`STAR` and `TopHat 2` provide similar tools with less extensive functionality. Most importantly, these tools cannot align unmapped reads to chimeric junctions. Moreover, remapping with `STAR` via its 'second pass' method requires a new index for every set of input splice junctions. This is demanding with respect to computation time as well as disk space. Both approaches were compared to `lack` on simulated and real data from Illumina and 454 sequencing technology. `lack` outperformed the remapping tools of `STAR` and `TopHat 2` in terms of number of remapped reads and total running time (Supplementary Material and Tables S8 and S9). `TopHat 2` recovered only a few unmapped reads (<0.2%). Also in comparison with `STAR`'s second pass method, `lack` achieved a significantly higher number of remapped reads, in particular for 454 data with an increase of >60 and 25% for artificial and real data, respectively. The difference may be explained by the fact that simply aligning reads over a set of given splice junctions differs fundamentally from the greedy extension algorithm of `lack` in which arbitrary paths over multiple splice

junctions are allowed. This particularly took effect in case of long 454 reads.

In terms of running time, lack is on average between 4- and 32-fold times faster than segemehl, TopHat 2 or Blat but 14-fold slower than STAR (Supplementary Fig. S5A). The memory consumption of lack is ~6.3 GB for each dataset (Supplementary Fig. S5B). It is lower than the consumption of segemehl and STAR and in the ballpark of the memory footprint of other tools.

3 DISCUSSION

In this article we show that segemehl is a versatile and accurate read aligner that performs equally well for reads from DNA- and RNA-seq experiments and is largely independent of read length and technology. Since the initial publication, segemehl has been continuously updated and extended. Here, we have focused updating segemehl's performance in aligning DNA-seq or unspliced RNA-seq reads. While the core algorithms remained unchanged, we have adjusted several parameterizations to optimize the tool. Our results indicate that segemehl is not only more sensitive in finding the optimal alignment with respect to the unit edit distance, but also very specific compared to the most commonly used alternative read mappers. These advantages are observable for both real and simulated reads.

In addition, we have presented lack to rescue previously unmapped RNA-seq reads that have emerged from splicing events. It shows excellent performance for every dataset with all split-read aligner tested and hence represents a valuable extension to RNA-seq analysis pipelines.

4 METHODS

4.1 Comparison of read aligners

In total, 10 datasets were used for benchmarking: three datasets with long single-end Illumina reads (artificial, DNA-seq, mRNA-seq), three with long paired-end Illumina reads (artificial, DNA-seq, mRNA-seq), two with short single-end Illumina reads (artificial, shortRNA-seq) and two with 454 reads (artificial, DNA-seq). An overview of the benchmarking datasets, their sequencing platforms, library types and average read lengths is given in Supplementary Table S1. Details about simulations and read data preprocessing are given in the Supplementary Material.

For benchmarking, we compared segemehl v.0.1.7 to five read aligners: Bowtie 2 v.2.1.0, BWA/ BWA-SW v.0.7.4 (Li and Durbin, 2009, 2010), BWA-MEM v.0.7.4 (Li, 2013), GEM pre-release 3 (Marco-Sola *et al*., 2012) and STAR v.2.3.0e (Dobin *et al*., 2013). The aligners were run on all datasets while keeping track of the user time and peak virtual memory consumption. Note that user time measurements did not include the preprocessing time for building the index structures of the reference, required by each aligner. In case of BWA where separate commands for alignment (aln) and post-processing the intermediate alignments (samse/sampe) were executed, time and memory were measured to include both commands. For 454 datasets, as recommended by the authors, BWA was exchanged by BWA-SW and Bowtie 2 was run in local mode. If necessary, the output of the aligners was converted into SAM format (Li *et al*., 2009).

Since our benchmark only considered optimal alignments with respect to the unit edit distance, a best-only filter was applied to the output of all tools. In case of paired-end alignments, the optimal alignment was defined as a properly paired alignment with the minimum sum of the edit distances in the first and second mate. In some cases, aligners report local instead of semi-global read alignments, marked by soft-clipped bases. To apply the best-only filter in these cases, local alignments were treated as semi-global ones by considering soft-clipped bases as errors. In addition to the default parameters, we evaluated a number of different parameter settings for each aligner (analogously to Langmead and Salzberg, 2012) to explore the tradeoff between sensitivity and number of false positive alignments (Supplementary Table S2). In such a way, best-sensitivity and best-false positive parameter settings were selected for each read aligner and dataset.

To obtain the set of optimal read alignments, RazerS 3 v.3.1 was applied to each dataset in its full-sensitivity mode. More specifically, RazerS 3 was run with the parameters -r 100, -i 90, -dr 0, -m 10, -pa, -ds and -of sam. Given a maximum edit distance and maximum number of optimal alignments per input, it guarantees to find all optimal alignments satisfying these constraints. For the paired-end data, RazerS 3 was not executed in paired-end but in single-end mode on both ends separately. Reads with >10 alignments or alignments with an error rate > 10% were discarded and not considered in all subsequent statistics. Subsequently, concordant optimal single-end alignments with insert sizes between 250 and 500 nt were paired and added to the test set. This strategy ensured that both alignments of a pair were optimal itself and the insert size constraint was always fulfilled. In this way, no aligner is put at a disadvantage because it favors paired-end alignments with lower edit distance but out-of-range insert sizes.

An alignment was considered optimal if a similar alignment was reported by RazerS 3 with the minimum edit distance on the same chromosome, strand and almost identical position. We permitted a deviation of twice the alignment edit distance from the position of the alignment reported by RazerS 3. Otherwise, the alignment was marked sub-optimal.

For the all-best benchmarks, the sensitivity was calculated as the normalized number of optimal read alignments. The normalization corrected for reads with multiple equivalent alignments, i.e. each optimal read alignment counted as $1/n$ with n being the total number of optimal read alignments of this read. The number of false positives was given by the number of sub-optimal read alignments. To compare read aligners that report multiple alignments per read to those that report only a single alignment in the any-best scenario, we randomly selected one single-/paired-end alignment per read. For the any-best benchmarks, the sensitivity and the number of false positives was given by the number of optimal and sub-optimal read alignments, respectively. In case of paired-end data, the alignments of both mates were evaluated separately. In addition to sensitivity and number of false positive alignments, we assessed the number of mapped reads of each aligner. To assure compatibility of local and semi-global alignments, only reads with at least one alignment with ≤10% mismatches, indels and clipped bases were considered.

4.2 Algorithm of lack

lack tries to rescue unmapped RNA-seq reads. In brief, it starts from a seed alignment and iteratively extends it across potential splice junctions. Subsequently, the read is aligned to the loci given by the extension path using a transition alignment (Hoffmann *et al*., 2014). In the following, the algorithm of lack is described in detail.

Initially, the splice junction data base is built up from split-read alignments provided by the user. Regardless of splice site consensus motifs and strandedness of the alignments, the genomic locations of read splits are clustered and categorized into types L and R (Fig. 4A). For reads that are mapped to the plus strand, type L sites denote 'donor' sites while type R sites denote 'acceptor' sites. For reads that are mapped to the minus strand, it is the other way round. Subsequently, clusters are linked according to the split read information, i.e. two clusters A and B are linked if a split-read alignment from locus A to locus B (or B to A) exists. Thus, for each cluster we obtain one or more cluster junctions. Note that for

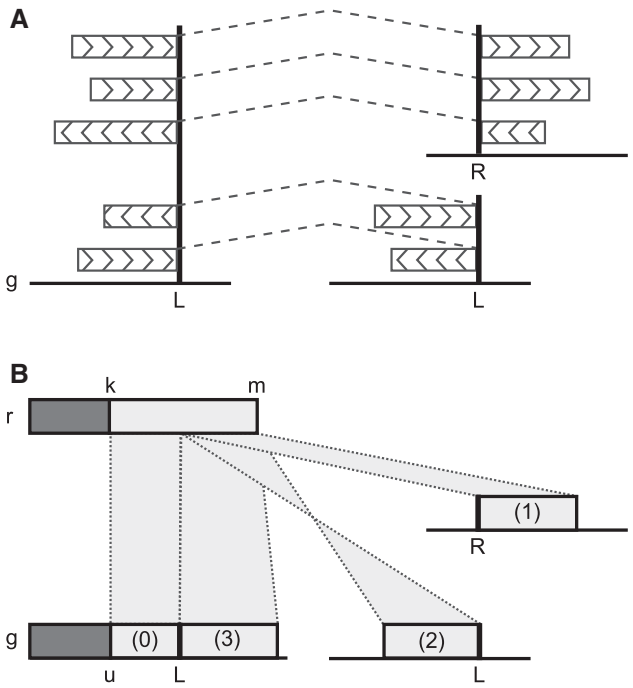

Fig. 4. Concepts of `lack`. (A) Splice site clusters of type L and R including split-read alignments connect three different loci on the reference *g*. Splice sites are illustrated as vertical black bars. The shading of the split-read alignments denotes the alignment strand whereby plus strand alignments are pictured by right-pointing arrows. (B) Example of forward-extension step on plus strand. The previous alignment (dark gray shaded) between read *r* and reference *g* was extended until *k* and *u*, respectively. At the current step, there are two spliced extensions candidates (1) and (2) and the unspliced alternative (3). The transition alignments are calculated between the remainder of *r* (light gray) and the two reference loci, i.e. common region (0) and specific region (1, 2 or 3). Dotted lines indicate the correspondence of alignment boundaries between *r* and *g*. The candidate with best alignment score determines at which reference loci the extension continues if the best score is obtained by the alignment of one of the spliced candidates. Otherwise, the forward extension is stopped. The backward extension works similarly

regular splice events, only clusters of different types are linked whereas strand-switch events produce links between clusters of same type.

Let *r* be the read sequence of length *m* and *g* be the reference sequence of length *n*. Furthermore, we assume that there is at least one seed alignment available for each read. The objective is to find the best split-read alignment between *r* and *g* using cluster junctions from the data base. The best seed alignment serves as anchor between *r* and *g* and is extended greedily in forward and backward direction. Let *k* and *u* be the current alignment boundaries on *r* and *g*, respectively. Starting with the forward extension, *k* and *u* are initially set to the best seed alignment end on *r* and the downstream alignment boundary on *g*, respectively. An example of one extension step is shown in Figure 4B.

During the extension, splice site clusters in the vicinity of *u* are looked up in the data base. Only clusters within the margin *M* are considered. *M* is calculated as the sum of maximum permitted edit distance *e* and the length of the remainder of the query sequence, i.e. $m-k-1+e$ during forward and $k+e$ during backward extension. Once a cluster *A* is found in the vicinity of the current extension front, the read is aligned across all cluster junctions associated with this cluster. More precisely, for each cluster junction from *A* to *B*, a local transition alignment is computed between the remainder of *r* and two reference loci on *g* with a total length of *M* (Fig. 4B). In case of junctions between clusters of the same type, a strand-switch event is represented and the alignment strand on *g* is switched. To control the false positive rate, spliced extenstions are valid only if they fulfill the following quality criteria (analogously to `segemehl`): the minimum alignment score (option `-Z`) and minimum alignment length (option `-U`) must be met for each alignment block. The best spliced extension is the one with valid split alignment and maximal score. In case of ties, the cluster junction with the highest split-read support is selected. In addition, we compute an optimal semi-global alignment between the remainder of the query and a reference subsequence of length *M* starting at g_u (Fig. 4B). To avoid unnecessary splits, we require the split alignment to have a higher score compared to the semi-global alignment. If these criteria are met, the split of *r* from *A* to *B* is accepted and the extension is iteratively continued at the locus of *B*. Otherwise, the current extension path is finished. Subsequently, the backward extension is carried out analogously. For the backward extension, *k* and *u* are initialized to the best seed alignment start on *r* and upstream alignment boundary on *g*. After completion of the extension procedure, the optimal local transition alignment is computed between the query sequence and all reference loci, confined by the extension steps. Only alignments with a minimum accuracy (option `-A`) and minimum coverage (option `-W`) are reported.

To limit the computational effort per step, the set of spliced extensions for each extension step is limited to *m* (option `-M`) and only *m* candidates with highest splice junction support are evaluated. Note that if the number of spliced extensions of each step is less than or equal to *m*, the computational effort as well as outcome of the algorithm will not change.

Overall, the time requirement of `lack` mainly depends on the read length and the number of nearby splice-site clusters. The number of clusters is influenced by the splice-junction data base and the choice of the parameter *m*. In contrast, the memory requirement of `lack` depends on the length of the reference sequence, while the number of unmapped reads and the splice junctions plays a minor role.

Funding: This publication was supported by LIFE and BMBF through ICGC MMML-Seq (01KU1002J). LIFE—Leipzig Research Center for Civilization Diseases, Universität Leipzig, is funded by means of the European Union, by the European Regional Development Fund (ERDF) and by means of the Free State of Saxony within the framework of the excellence initiative.

Conflict of interest: none declared.

REFERENCES

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

Dobin,A. *et al.* (2013) STAR: ultrafast universal RNA-seq aligner. *Bioinformatics*, **29**, 15–21.

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

Hoffmann,S. *et al.* (2014) A multi-split mapping algorithm for circular RNA, splicing, trans-splicing, and fusion detection. *Genome Biol.*, **15**, R34.

Hoffmann,S. *et al.* (2009) Fast mapping of short sequences with mismatches, insertions and deletions using index structures. *PLoS Comput. Biol.*, **5**, e1000502.

Holtgrewe,M. (2010) Mason – a read simulator for second generation sequencing data. *Technical Report TR-B-10-06*. Department of Mathematics and Computer Science, Free University of Berlin.

Holtgrewe,M. *et al.* (2011) A novel and well-defined benchmarking method for second generation read mapping. *BMC Bioinform.*, **12**, 210.

Kent,W. (2002) BLAT–the BLAST-like alignment tool. *Genome Res.*, **12**, 656–664.

Kim,D. *et al.* (2013) TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. *Genome Biol.*, **14**, R36.

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

Langmead,B. and Salzberg,S. (2012) Fast gapped-read alignment with Bowtie 2. *Nat. Methods*, **9**, 357–359.

Langmead,B. *et al.* (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol.*, **10**, R25.
Li,H. (2013) Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *arXiv preprint arXiv:1303.3997*.
Li,H. and Durbin,R. (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, **25**, 1754–1760.
Li,H. and Durbin,R. (2010) Fast and accurate long-read alignment with Burrows-Wheeler transform. *Bioinformatics*, **26**, 589–595.
Li,H. *et al.* (2009) The sequence Alignment/Map format and SAMtools. *Bioinformatics*, **25**, 2078–2079.
Marco-Sola,S. *et al.* (2012) The GEM mapper: fast, accurate and versatile alignment by filtration. *Nat. Methods*, **9**, 1185–1188.
Otto,C. *et al.* (2012) Fast and sensitive mapping of bisulfite-treated sequencing data. *Bioinformatics*, **28**, 1698–1704.
Smith,A. *et al.* (2009) Updates to the RMAP short-read mapping software. *Bioinformatics*, **25**, 2841–2842.
Weese,D. *et al.* (2012) RazerS 3: faster, fully sensitive read mapping. *Bioinformatics*, **28**, 2592–2599.