Send Orders of Reprints at bspsaif@emirates.net.ae

The Open Bioinformatics Journal, 2013 *7*, 1-8

Open Access

Comparison of Sequencing Utility Programs

Erik Aronesty*

Expression Analysis, A Quintiles Company, Durham, NC, USA

Abstract: High throughput sequencing (HTS) has resulted in extreme growth rates of sequencing data. At our lab, we generate terabytes of data every day. It is usually seen as required for data output to be "cleaned" and processed in various ways prior to use for common tasks such as variant calling, expression quantification and assembly.

Two common tasks associated with HTS are *adapter trimming* and *paired-end joining*. I have developed two tools at Expression Analysis, Inc. to address these common tasks. The names of these programs are *fastq-mcf* and *fastq-join*. I compared the performance of these tools to similar open-source utilities, both in terms of resource efficiency, and effectiveness.

Keywords: Sequencing, utilities, Illumina, clipping, ngs and next-generation.

1. INTRODUCTION

Before proceeding to analyze data from a high throughput sequencing instrument, the sequences are typically preprocessed using various computational techniques. Two of these techniques, adapter clipping and paired-end joining are discussed.

1.1. Adapter Clipping

Most high throughput sequencing instruments are not capable of sequencing very long molecules. Typical sequencing runs produce 50bp to 200bp of sequence per read. During sample preparation, molecules are first fragmented to a certain desired "fragment size". Typical fragment sizes range from 150bp to 500bp.

After fragmenting, sequencing technologies use alien "adapter" sequences which are ligated to the ends of DNA fragments (Fig. **1**) so that they can be bound to the flowcell, smartwell, chip, etc. before sequencing [1].

As long as the fragment size is longer than the sequenced length, the output will contain data only from the molecules of interest. In practice, however, fragmentation is not completely precise, and there may be a portion of fragments that are shorter than the sequenced length. The resulting sequence may therefore be contaminated with the sequence of the adapter itself. Without thorough removal, these adapters can potentially be miscalled as variants, or can interfere with assembly [2].

In addition, there are some protocols, such as smallRNA sequencing, where significant adapter presence is expected on *every* read, because the molecules sequenced (21bp on average) have sizes that are nearly always less than the sequenced length.

*Address correspondence to this author at the Expression Analysis, a Quintiles Company, Durham, NC, USA; Tel: 919-287-4011;
E-mail: erik.aronesty@quintiles.com

There are several published tools for performing adapter clipping, including cutadapt [3], fastx_clipper [4], Seqprep [5], and TagDust [6].

1.2. Paired-End Joining

Several high-throughput sequencing technologies perform "paired end" sequencing, often used for improving alignment specificity [1]. In this technique, the fragments are sequenced from *both ends* (Fig. **2**). With long insert sizes, these can be used for improved assembly (known as scaffolding), transcriptome determination and other applications.

However, when the insert size is shorter than the total number of bases read, the sequencer will read the same region twice, once in one direction and then again in the other.

These reads can be "joined" using several publically available tools, including SeqPrep, fastq-join, mergePairs.py (code.google.com/p/standardized-velvet-assembly-report), and Audy's "stitch" program (github.com/audy/stitch).

These overlapping regions will then be overrepresented (sequenced twice for the same molecule), and this may result in bias, especially for exome capture or other smaller regions. For this reason, and for testing library preparation, "joining" may be done.

1.3. Terms

In this article, the term "Smith Waterman [7] matching" is used to describe *any* matching algorithm that allows for inserts & deletions when matching two sequences. The term "clipping" is used to refer to the task of removing adapter sequences from the ends of reads.

2. IMPLEMENTATION AND ALGORITHM

The two tools covered in this article, fastq-mcf and fastq-join, were implemented in C++, using standard libraries.

1875-0362/13

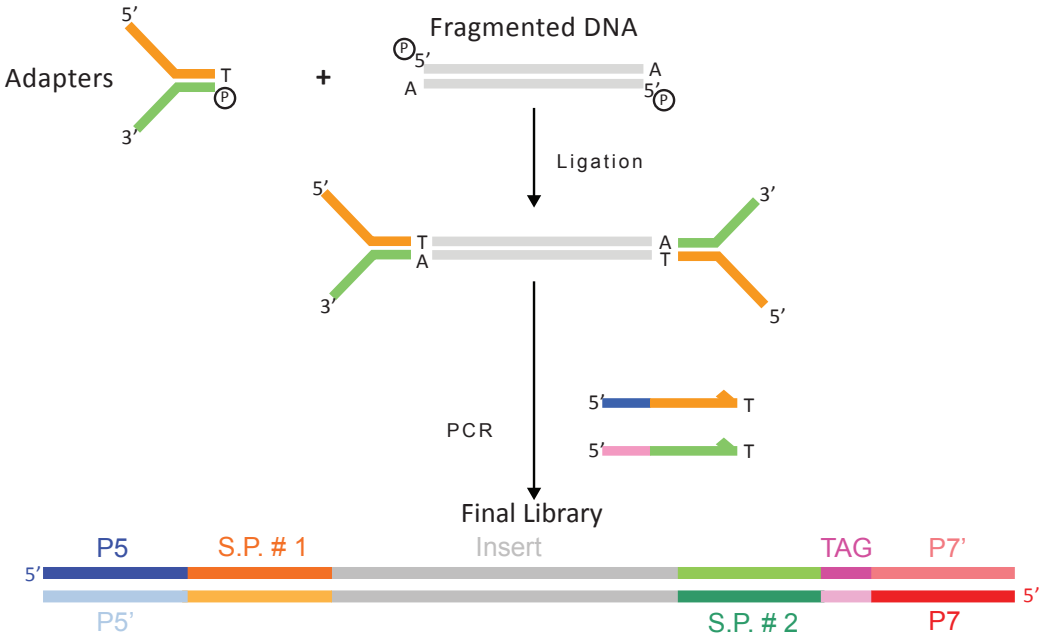

Fig. (1). Typical adapter layout.

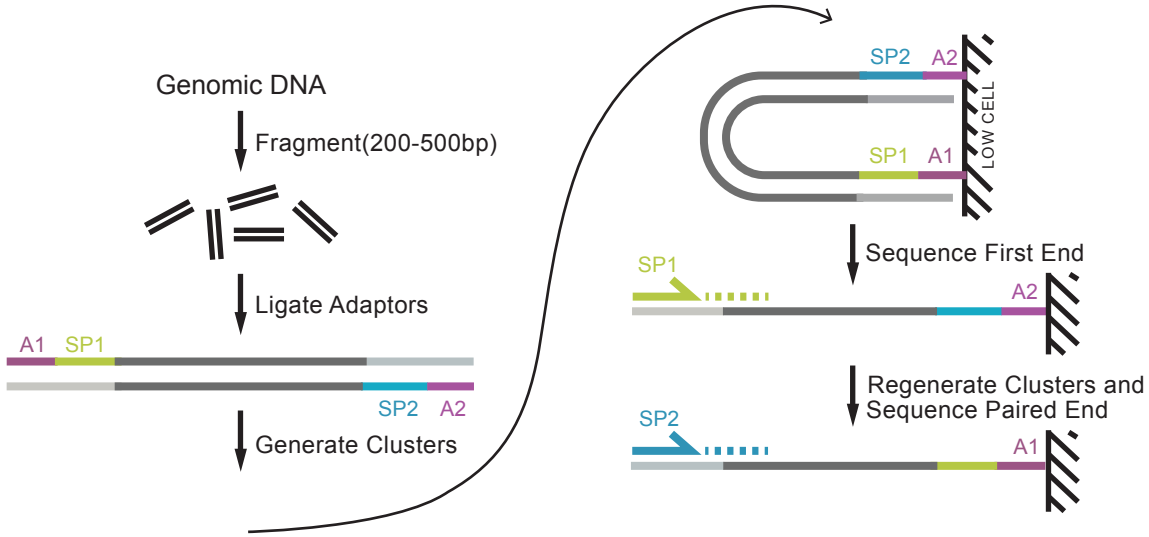

Fig. (2). Paired-end sequencing chemistry, and adapter layout.

2.1. Fastq-Mcf and Fastq-Join Use a Novel end-Joining Algorithm

Unlike many common utilities fastq-mcf and fastq-join do not perform Smith-Waterman alignment. Instead, they employ a strategy appropriate only to sequencing technologies that do not suffer from high insertion or deletion errors and was specifically optimized for the Illumina platform. This is what enables both the high speed & sensitivity of these programs:

Each of the programs selects the minimum scoring alignment by scoring the potential overlapping pairs of sequences and choosing a maximum score: allowable _overlaps

{L=overlap_length,

d=hamming_distance(adapter, sequence) [8] if

(d < minimum _allowable_distance) {score=(d*d+1)/L}}

By using the squared distance, *better* matches are weighted, and on dividing by the length, *longer* matches are also weighted. This scoring allows fastq-mcf and fastq-join to be highly selective while eschewing spurious short, but exact, matches - which are frequent when performing these sorts of alignments.

3. ANALYSIS OF ADAPTER CLIPPING

There have been many open access algorithms published, but few have been subjected to an analysis which compares them and reports the "best parameters" for typical sequencing. Each program was run with a broad variety of settings, to capture a range of performance characteristics.

3.1. Testing Method

The tool wgsim (from SAMTools [9]) was used for read simulation against the human genome build "hg19" downloaded from UCSC. 3 sets of 100,000 reads were

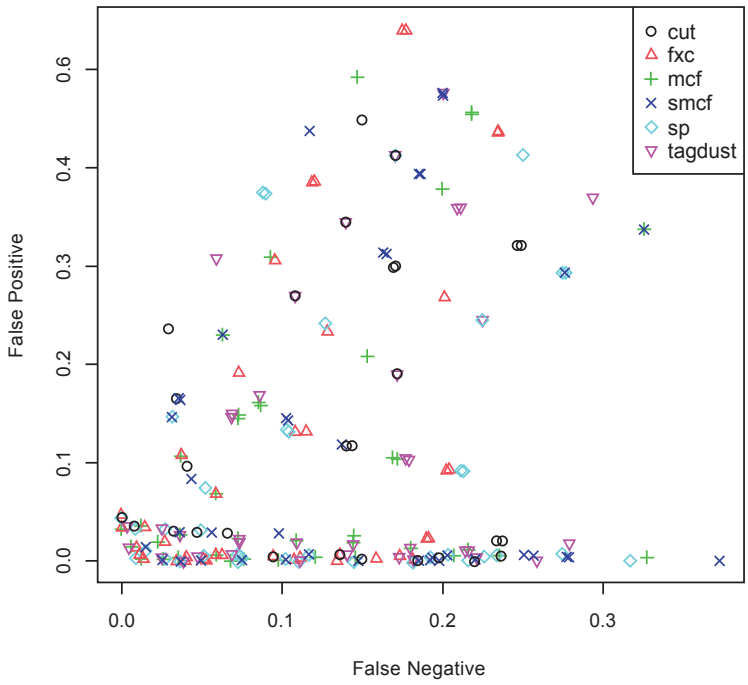

Fig. (3). FP vs FN, sweep settings, no filter.

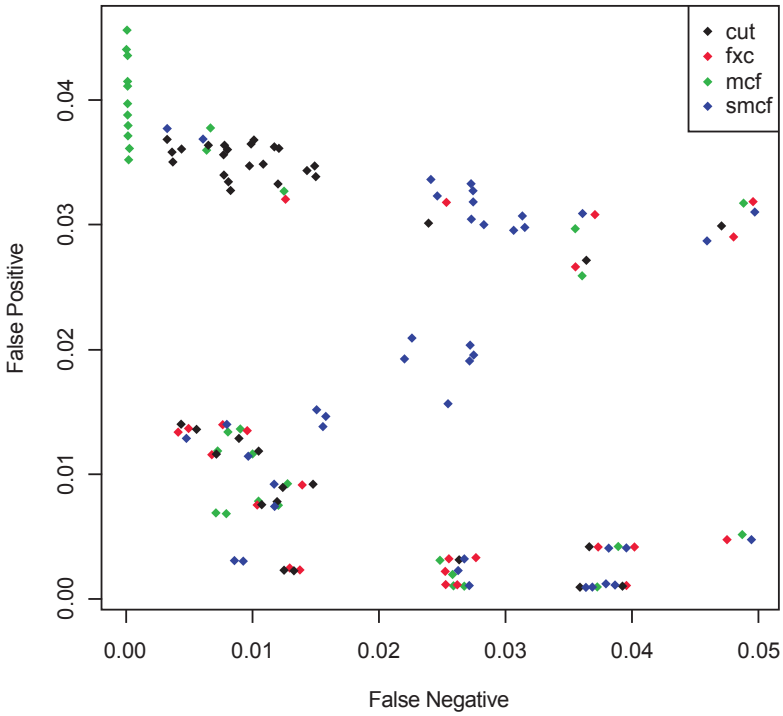

Fig. (4). Filtered <5% FP vs FN.

generated. Illumina adapters were "in silico" ligated, using a perl script, to the end of reads with random lengths at different percentages of the overall file (ranges from 10% to 60% contamination were used). Random number generation, instead of an attempt to model actual adapter ligation and illumina error rate, was used. The error rate was fixed at .2% (phred 26).

The clipping tools tested were: fastx_clipper [4], fastq-mcf (code.google.com/p/ea-utils), SeqPrep [5], tagdust [6], and cutadapt [3].

Parameters, for each clipping tool, were varied by overlap size and match percentage.

Two different reports were generated. False positives were recorded when the clipping of bases was performed when it should not have been. False negatives were recorded if clipping was incomplete. The first report allows for no clipping beyond the adapter, i.e. all *excess* removal seen as false positive (no lenience, Fig. **3**, **5**). In the second report (Fig. **4**), correctness was defined as a match of the original sequence without adapter and *up to 4 extra bases removed*. It

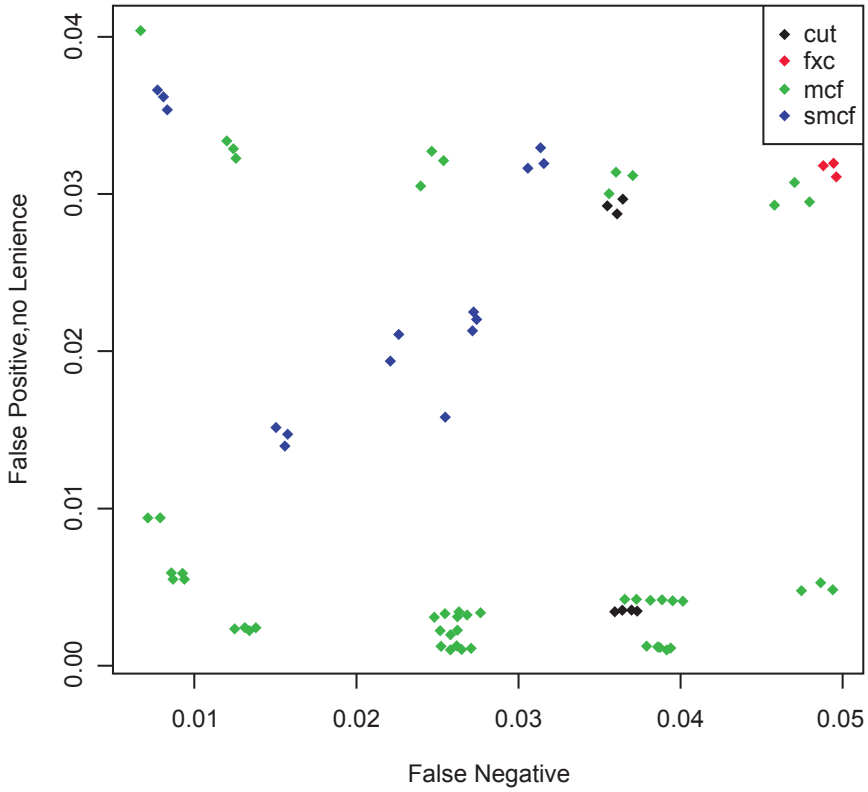

Fig. (5). Filtered < 5%, No lenience.

is my view that extra clipping is generally preferable to under-clipping, as the removal of a few extra bases should not adversely affect downstream programs as much as the inclusion of extra adapter sequence.

3.2. Results

Overall, most adapter trimming programs were capable of operating with both low false positive and false negative rates (over 95%) with the correct parameter settings. The settings that generated this "ideal clipping" (shown in the lower left corners of Figs. **3**, **4**) are reported in the Appendix for each program.

Programs that use Smith Waterman alignment strategies without accounting for ligation "end-ness", such as SeqPrep and TagDust have poor behavior if lenient settings, such as small overlap levels or FDR (false discovery rate) settings are used. These results are represented by the points in Fig. (**3**) radiating out from the lower left.

In the "no lenience" report (Fig. **3**), fastq-mcf's performance can result in as much as 50% over-clipping (with a mean over-clip of 1.83 bases). This results if the "overlap level" is set to 1 base, or if the "scale" is set to less than 0.6. With the minimum clip level set to 13, the false positive rate for most programs reduces to < 0.1% however the false negative rate can be unacceptably high, depending on the application.

Only fastq-mcf, fastx toolkit, and cutadapt were capable of sustaining both very low false positive (>95%) and low false negative rates (>95%) with any measure of success tested (with lenience for over-discarding, no lenience for over-clipping). Note: fastq-mcf in "autoscale" mode with the scale set to .6 *seemed* to perform best at preventing false negatives, however, it was an artifact of "over-discarding" some reads in that mode.

fastq-mcf with a minimum match of 1 and a percentage set to 10 had a false negative rate of < .005, while still maintaining a false positive rate of < .04. Even with zero lenience, it could operate with a false neg rate of <.01 *and* a false pos rate of 25% (what would be expected if every 4th read has 1 base falsely removed). This is the best performing program if false-negatives are of chief concern, such as in miRNA discovery.

Since all runs were done in triplicate it's possible to report the stability of these metrics, however the differences were minor, visible as clusters of 3 points in some of the graphs below.

Optimal results for fastq-mcf, cutadapt and fastx_clipper were obtained with either the "scale" set between 1.1 and 2.1, and with the "percent match" set to .05, and "overlap length" set to 9 if false positive rate (over-clipping) was of chief concern, and with the minimum *turned off* (set to 1) if under-clipping was more concerning (such as in small RNA applications). Other programs, such as TagDust and SeqPrep, were capable of good performance, but only over a very narrow range of settings, and did not meet more stringent criteria.

Clip Timing results: Programs were compiled on Ubuntu 11, with default compilation options and run on a 2.1MHz Intel Xeon Processor with 16GB ram free. C++ programs (tagdust, SeqPrep and fastq-mcf) had the –O3 option added for speed. RAM utilization was negligible (<5mb). Tagdust and fastq-mcf were I/O bound, rather than CPU bound, capable of clipping over 100K reads per second (Fig. **6**). The tool "cutadapt" also had adequate performance.

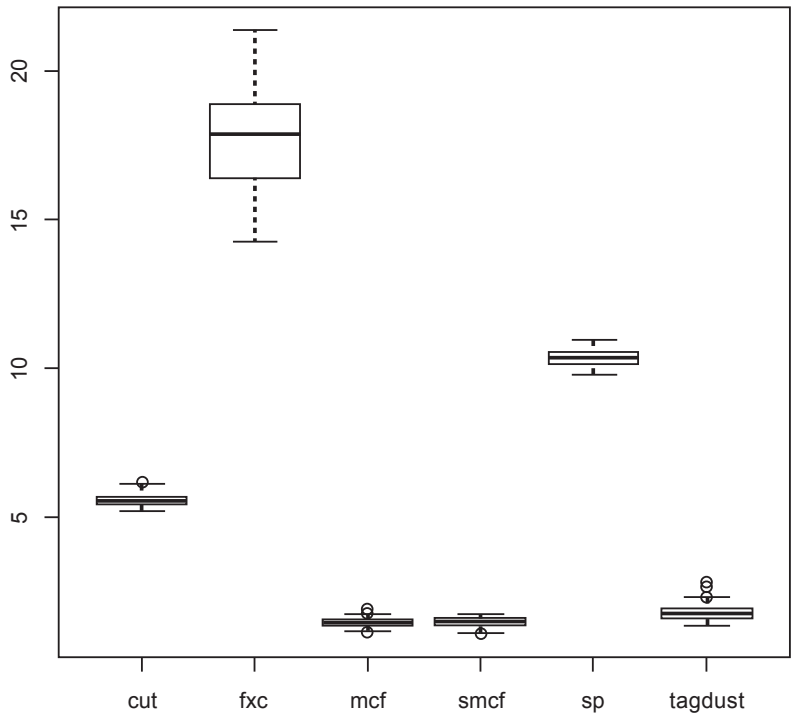

Fig. (6). Timing Output (Seconds).

Cutadapt would not operate at low overlap levels / very high sensitivity, the minimum false negative rate as therefore 4.3%. Otherwise, since the algorithm was similar, and performed similarly with the difference in weighting accounting for a 1% improvement in performance at the same false negative rates. I suspect if I were able to control the parameters correctly, it would operate the same as fastx_clipper.

4. ANALYSIS OF PAIRED-END JOINING

4.1. Testing Method

The tool "wgsim" was used for read simulation against the human genome build "hg19" downloaded from UCSC. Insert lengths were varied from 100bp to 120bp with read-length fixed at 70bp. Reads with all "N's" were discarded.

Joining parameters were varied by overlap size and match percentage. SeqPreps "-n" and "-m" values were also independently varied in ranges from .05 to .20 and .95 to .80 respectively. Segmentation faults and other errors were discarded from the data sets and ignored.

The definition of "correctness" was defined as an exact match of original sequence or a match with exactly equal lengths and a hamming distance of less than 5. The reason for this lenience is that joining programs often use the "better quality base" in the overlap region. Failure to join at all was a "false negative", joining reads that should not be are "false positive" (a more serious error in this case). "Bad joining" was measured separately and used for an alternative evaluation.

4.2. Results

If the only purpose in joining is to output statistics for quality control of the paired-end protocol, all algorthims operate with high enough specificity to be used for end-join statistics generation (>95%). However, in the downstream analysis, high numbers of false positives could potentially result in biased variation.

(In Figs. **7**, **8**), the size of the symbol is proportional to the the log of the number of exact matches).

Only fastq-join and SeqPrep were capable of operating at that high level of specificity while still maintaining a false negative rate of less than 15%. The algorithms seemed to match each other's performance (see Fig. **8**). For some applications, false positives can lead to variants, so minimizing this is desirable. Clearly the choice of settings is critical.

4.3. Join Timing Results

Regardless of performance, speed is essential in evaluating HTS tools. There are many situations, such as in quality control pipelines, where speed and stability are more important than absolute accuracy. Only fastq-mcf and SeqPrep performed with stable speeds appropriate to HTS tools, capable of operating on 100k reads in under 1 second.

5. PROGRAM USAGE AND AVAILABILITY

Both fastq-mcf and fastq-join come as a part of the open source "ea-utils" toolkit, which also includes programs for calculating sequencing and alignment statistics, demulti-plexing, and variant calling.

5.1. Fastq-Mcf

This is a general purpose filtering, validation, quality trimming and filtering tool – clipping is only one of the functions. The simplest usage mode automatically detects adapter presence, and optimizes the match lengths based on

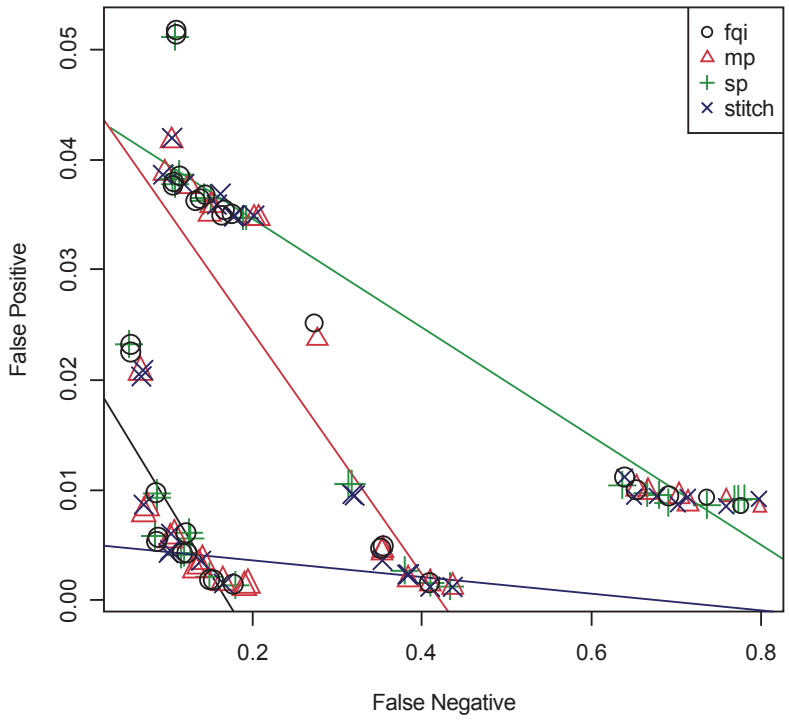

Fig. (7). Joining FP vs FN, no filter.

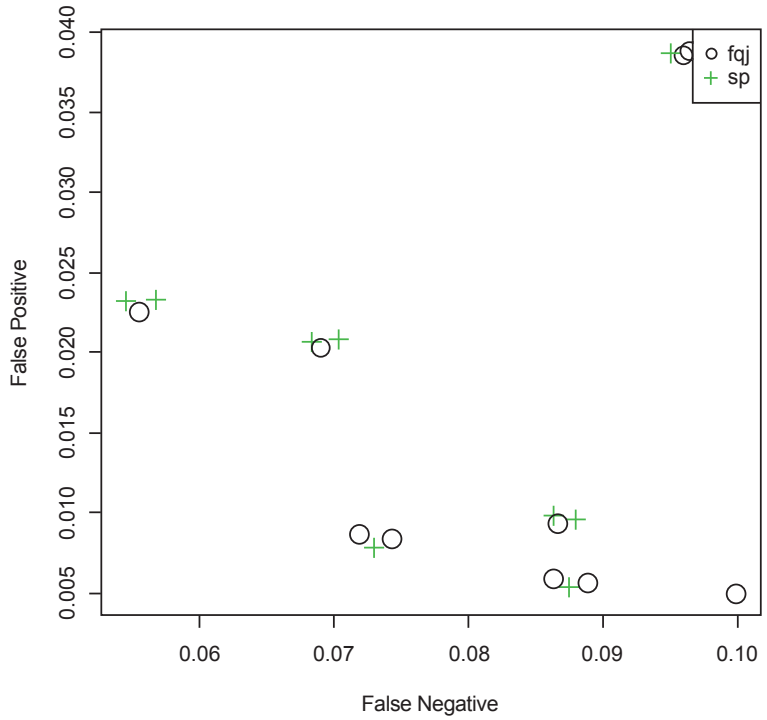

Fig. (8). (FP < .01, FN < .15).

their prevalence (the user can override this). The adapters to be removed are placed in a FASTA file, and the percentage mismatch and minimum remaining sequence length can be specified.

Example:

fastq-mcf -p 10 -l 25 clip_seq.fa read1.fq.gz read2.fq.gz -o read1.clp.fq.gz -o read2.clp.fq.gz

Additional features: By default, fastq-mcf will verify that the paired-end read sets are "proper" pairs by matching the identifiers of each read. It will also search for end-of-read cycles that contain "skew", an error with older Illumina sequencing where certain cycles will contain "all A's". Reads shorter than the minimum length (25 specified above, default is 19) are discarded. Reads containing low quality bases at the ends (phred 10 by default) are trimmed. All of

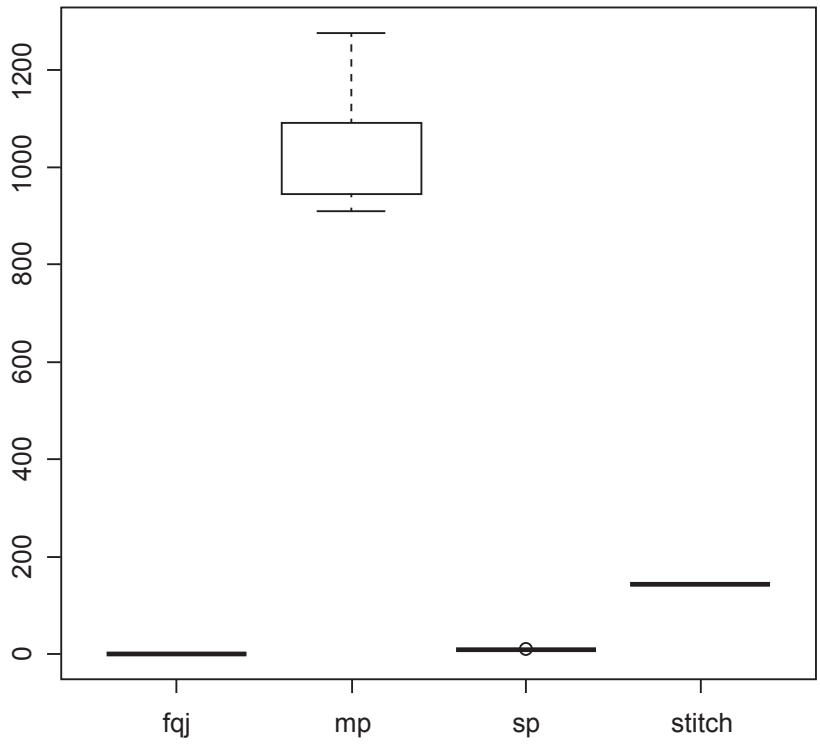

Fig. (9). Join timing in seconds.

these features were enabled during performance and accuracy testing. There are a thorough set of options for filtering and preprocessing reads so that pipelines can be shortened to a single pass on the data.

5.2. Fastq-Join

This is also a command-line tool. Users specify two reads to join, and optionally a "mate" read that gets copied into the output files (typically a barcode read). Then, an output "template" is specified for the results.

Example

fastq-join read1.fq read2.fq -o foo.%.fq

The strings "un, un2, and join" are inserted where the "%" sign is in the output template, representing un-joined reads, and joined results respectively.

Additional features: If a mate read is specified, then "un3" and "join3" files are created as well (it is important to preserve mates, and this feature is often missing from preprocessing tools). fastq-join can produce a statistics file containing the overlap lengths of all of the joined reads. These results then can be easily analyzed for quality purposes.

6. CONCLUSION

It's clear that choosing parameters carefully is very important for sequencing preprocessing tools. Performances of all tools were greatly impacted by the choice of parameters. All tools were capable of performing poorly when inappropriate values were chosen. A table of parameters used and performance is available as supplementary data.

Overall, fastq-mcf and fastq-join perform at least as well or better than other available methods, and were far more efficient. The novel scoring algorithm allows fastq-mcf to perform slightly better than other methods in situations where false-negatives are a primary concern. Only SeqPrep and fastq-join performed acceptably when joining overlapping reads.

As a more general conclusion for sequencing technologies that have a low error rate and low indel rates, Smith Waterman-style alignment is most likely not appropriate for "end-overlap" clipping tasks, such as that of adapter removal and paired-end joining algorithms. Methods that utilizing hamming-distance scoring schemes consistently perform better, both in terms of quality and efficiency.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflicts of interest.

ACKNOWLEDGEMENT

Declared none.

REFERENCES

[1] G. Döring and W. Chen, "An overview of the analysis of next generation sequencing data", *Methods Mol. Biol.,* vol. 802, pp. 249-257, 2012.

[2] R. Schmieder, Y. W. Lim, F. Rohwer and R. Edwards, "TagCleaner: identification and removal of tag sequences from genomic and metagenomic datasets", *BMC Bioinformatics*, vol.11, p. 341, 2010.

[3] M. Martin, "Cutadapt removes adapter sequences from high-throughput sequencing reads", *EMBnet. J.*, vol. 17, no. 1, pp. 10-12, 2011.

[4] A. Gordon, FASTX Toolkit: Available: http://hannonlab.cshl-.edu/fastx_toolkit/, 2009.

[5] J. St. John, SeqPrep. Available: https://github.com/jstjohn/SeqPrep, 2011.

[6] T. Lassmann,Y. Hayashizaki and C. O. Daub, "TagDust--a program to eliminate artifacts from next generation sequencing data", *Bioinformatics,* vol. 25, no. 21, pp. 2839-2840, 2009.

[7] M.S.Waterman, T. F. Smith and W.A. Beyer, "Some biological sequence metrics", *Adv. Math*., vol. 20, pp. 367-387, 1976.

[8] Hamming and W. Richard, "Error detecting and error correcting codes", *Bell Syst. Tech. J.,* vol. 29, no. 2, pp. 147-160, 1950.

[9] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan and N. Homer, "The Sequence Alignment/Map format and SAMtools", *Bioinformatics,* vol. 25, no. 16, pp. 2078-2079, 2009.

Received: November 01, 2012 Revised: November 15, 2012 Accepted: November 17, 2012