Atropos: specific, sensitive, and speedy trimming of sequencing reads

John P Didion[1] **and Francis S Collins**[1]

[1]**National Human Genome Research Institute, National Institutes of Health, Bethesda, MD**

Corresponding author:
John P Didion[1]

Email address: john.didion@nih.gov

ABSTRACT

Summary. A key step in the transformation of raw sequencing reads into biological insights is the trimming of adapter sequences and low-quality bases. Read trimming has been shown to increase the quality and reliability while decreasing the computational requirements of downstream analyses. Many read trimming software tools are available; however, no tool simultaneously provides the accuracy, computational efficiency, and feature set required to handle the types and volumes of data generated in modern sequencing-based experiments. Here we introduce Atropos and show that it trims reads with high sensitivity and specificity while maintaining leading-edge speed. Compared to other state-of-the-art read trimming tools, Atropos achieves a XX% increase in trimming accuracy and a decrease in execution time of at least XX% (using 16 parallel execution threads). Furthermore, Atropos maintains high accuracy even when trimming simulated data with a high rate of error. The accuracy, high performance, and broad feature set offered by Atropos makes it an appropriate choice for the pre-processing of most current-generation sequencing data sets. **Availability.** Atropos is open source and free software written in Python and available at `https://github.com/jdidion/atropos`.

1 INTRODUCTION

Sequencing on all current-generation sequencing technologies, including Illumina, SOLiD, PacBio, and Nanopore, requires a library construction step that involves ligation of short adapter sequences to the ends of the template DNA fragments. Depending on the sequencing platform and the fragment size distribution of the sequencing library, an often substantial fraction of reads will consist of both template and adapter sequences (Figure 1A). Additionally, the error rates of these sequencing technologies vary from ˜0.1% on Illumina to 5% or more on long-read sequencing platforms. Error rates tend to be enriched at the ends of reads (where adapters are located), thus exacerbating the effects of adapter contamination. Adapter contamination and sequencing errors can lead to increased rates of misaligned and unaligned reads, which results in errors in downstream analysis including spurious variant calls (Del Fabbro et al., 2013; Sturm et al., 2016). Certain sequencing protocols may introduce other artifacts in sequencing reads. For example, some methylation sequencing (Methyl-Seq) protocols result in artificially methylated bases at the 3' ends of reads that can result in inflated estimates of methylation levels (Bock, 2012).

Read trimming is an important step in the analysis pipeline to mitigate the effects of adapter contamination, sequencing errors, and other artifacts. The development of tools for read trimming is an active area of bioinformatics research, thus there are currently many options. In terms of adapter trimming, these tools fall into two general categories (Figure 1B): 1) those that rely solely on matching the adapter sequence (*adapter-match trimming*) using semi-global alignment (which is the only option available for single-end reads); and 2) those that leverage the overlap between paired-end reads to identify adapter starting positions (*insert-match trimming*) (Sturm et al., 2016). Cutadapt (Martin, 2011) is a mature and feature-rich example of a tool that provides adapter-match trimming, while SeqPurge (Sturm et al., 2016) is a recent example of a highly accurate insert-match trimmer designed specifically for paired-end data. Additionally, hybrid tools are available that optimize their choice of read trimming method based on the type of data. Skewer (Jiang et al., 2014) is a fast and accurate hybrid trimmer that works with both

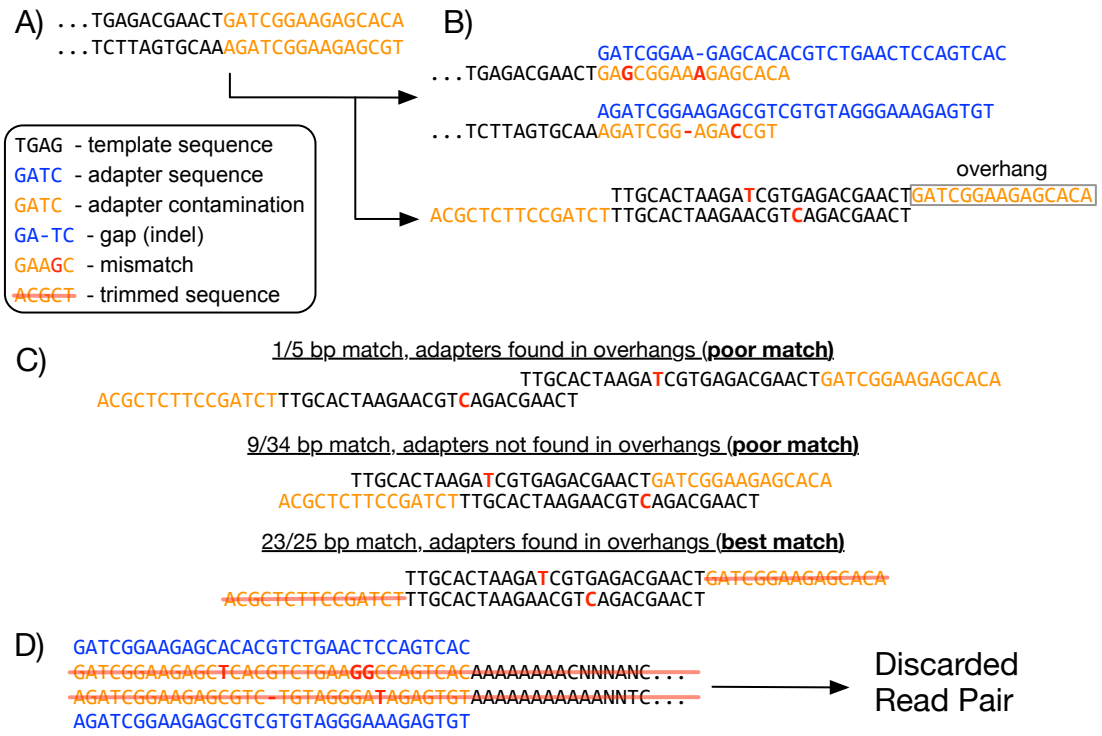

Figure 1. Adapter detection and trimming. A) When a fragment (insert) is shorter than the read length, the read sequence will contain partial to full-length adapter sequences (orange). B) Adapters contamination may be detected by adapter-match (top) or insert-match (bottom). C) Atropos uses semi-global alignment to find the best insert match (if any) for which the overhangs contain adapter contamination. D) If insert-match fails – for example when the adapter appears at or near the beginning of the read – adapter-match is optionally performed. Reads that are too short after trimming are discarded.

single-end and paired-end data. However, choosing a read-trimming tool currently requires a trade-off between feature set, efficiency, and accuracy. Furthermore, even state-of-the-art tools still have a relatively high rate of over-trimming (removing usable template bases from reads) and/or under-trimming (leaving low-quality and adapter-derived bases in the read sequence) (Sturm et al., 2016).

We sought to develop a read trimming tool that would combine the best aspects of currently available software to provide high speed and accuracy while also offering a rich feature set. To accomplish this aim, we used Cutadapt as a starting point, as it provides the broadest feature set of currently available tools and is published under the MIT license, which allows modification and improvement of the code. We focused on making three specific improvements to Cutadapt: 1) improve the accuracy of paired-end read trimming by implementing an insert-match algorithm; 2) improve the performance by adding multiprocessing support (as it is currently only able to use a single processor); and 3) add important additional features such as automated trimming of Methyl-Seq reads and automated detection of adapter sequences in reads where the experimental protocols are not known to the analyst. Because these modifications required substantial changes to the Cutadapt code base, and because there are software tools that depend on the current implementation of Cutadapt, we chose to ”fork” the Cutdapt code base and develop our software, Atropos, as a separate tool. Here, we show that we have accomplished our three aims. In addition to extending the already rich set of features provided by the original Cutadapt tool, Atropos demonstrates paired-end read trimming accuracy that is superior to other state-of-the-art tools, and it is among the fastest read trimming tools when a moderate number of parallel threads are used (4 or 8). Furthermore, Atropos achieves a performance increase that is roughly linear with the number of threads used, making it the fastest tool when 16 or more threads are available.

2 IMPLEMENTATION

Atropos is developed in Python and is available to install from GitHub or via the pip package manager (see Data Availability).

2.1 Insert Match Algorithm

For each read pair, the insert-match algorithm finds all possible alignments between the first read and the reverse complement of the second read that satisfy user-specified minimum overlap and maximum mismatch thresholds (Figure 1C). The candidate alignments are tested in order of decreasing length until one is found in which the overhanging sequences on either end match the user-specified adapter sequences.

To determine whether an adapter sequence is present in the overhang, our algorithm uses the same probabilistic approach as in the SeqPurge algorithm (Sturm et al., 2016). Briefly, starting at the end of the insert overlap, a pairwise comparison is made between the adapter and the read at each possible offset. The probability of a random match at k bases out of the n bases being compared is computed using the binomial distribution:

$$P = \sum_{i=k}^{n} \frac{n!}{i!(n-i)!} p^i (1-p)^{n-i} \tag{1}$$

The offset having the lowest random-match probability is taken to be the location of the adapter sequence, and all bases from that position to the 3' end of the read are removed. If an adapter is only found in one of the two reads, then the same offset is used to trim both reads, under the assumption that the location of the adapter sequence must be symmetric across the read pair. If no adapter is found, then an adapter-match approach is optionally attempted in each read separately (Figure 1D).

2.2 Parallel processing

The performance improvements in Atropos relative to Cutadapt and other read trimming tools are based in two observations: 1) each read (or read pair) is trimmed separately, and thus trimming can be parallelized across multiple processor cores, and 2) a significant fraction of the execution time is spent decompressing input files and re-compressing results. Compression of sequencing data is increasingly becoming necessary due to the large volumes of data generated in sequencing experiments.

To address the first bottleneck, we implemented a parallel processing pipeline based on the Python `multiprocessing` module . Briefly, a single thread is dedicated to a "reader" process that loads reads (or read pairs) from input file(s), with support for a variety of data formats and automatic decompression of compressed data. Reads are loaded in batches, and each batch is added to an in-memory queue. A user-specified number of "worker" threads (which is constrained by the number of processing cores available on the user's system) extract batches from the queue and perform trimming and filtering operations on the reads in the same manner as Cutadapt. Atropos addresses the second bottleneck by offering a choice of three modes for writing the results to disk. The first two modes involve adding the results to a second queue, from which a dedicated "writer" process extracts batches and performs the serialized write operation. These modes differ in how the trimmed reads are compressed – in "worker compression" mode, each worker is responsible for compressing the results using the Python `gzip` module prior to placing the results on the queue, whereas in "writer compression" mode, the writer process performs compression using the much faster system-level gzip program. The choice between these two modes is selected automatically based on the number of worker threads used, with worker compression becoming faster than writer compression when at least 8 threads are available. The third output mode, called "parallel writing," does not use a dedicated writer process (and thus an additional worker process can be used in its place). Instead, each worker process writes its results to a separate file. This can dramatically reduce the execution time of the program (˜100% reduction in our experiments; see Results) and is generally compatible with downstream analysis since most modern mapping and assembly tools accept multiple input files. An additional speed-up is gained by recognizing that the reader process often finishes loading data well before the worker processes finish processing it, thus an additional worker thread is started as soon as the reader process completes.

2.3 Adapter detection

Often, details of sequencing library construction are not fully communicated from the individual or facility that generated the library to the individual(s) performing data analysis. Manual determination of sequencing adapters and other potential library contaminants can be a tedious and error-prone task. Thus, we implemented in Atropos a new 'detect' command that automatically identifies adapters/contaminants from a sample of read sequences. First, a profile is built of k-mers (where k is a fixed number of consecutive nucleotides, defaulting to $k = 12$) within N read sequences (where N defaults to 10,000). When at least 8 consecutive A bases are detected, those bases along with all subsequent bases (in the 3' direction) are first trimmed, as that pattern is a strong indicator that the sequencer scanned past the end of the read (i.e. the length of the fragment + adapter is less than the read length; Figure 1D). Additionally, low-complexity reads are excluded, where complexity $X(S)$ is defined as follows. Let $C(i,S)$ be the number of elements of a nucleotide sequence $S = s_1, ..., s_n$, that are nucleotide $i \in A, C, G, T$.

$$X(S) = -\sum \frac{C(i,S) * log(C(i,S))}{log(2)} \tag{2}$$

Sequences with $X(S) < 1.0$ are defined as low-complexity. All remaining k-mers are counted, and each k-mer is linked to all of the sequences from which it originated. This process continues iteratively for increasing values of k, with only those read sequences linked to high-abundance k-mers in the previous iteration being used to build the k-mer profile in the next iteration. k-mer K is considered high-abundance when:

$$|K| > \frac{N * (l - k + 1) * O}{4^k} \tag{3}$$

where l is the read length and $O = 100$ by default. Finally, high-abundance k-mers of all lengths are merged to eliminate shorter sequences that are fully contained in longer sequences. Optionally, the high-abundance k-mers are matched to a list of known adapters/contaminants. Atropos reports to the user an ordered list of up to 20 of the most likely contaminants.

3 RESULTS

3.1 Benchmarks

Data Set	Error Rate	Read Length	Total Read Pairs	Reads w/ Adapters	Adapter Bases
Simulated 1	0.20%	125	781,923	325,982	12,447,262
Simulated 2	0.60%	125	780,899	325,105	12,416,861
Simulated 3	1.20%	125	782,237	325,860	12,464,235
XXX Amplicon	Unknown	???	1,000,000	???*	???*
GM12878 WGBS	Unknown	125	1,000,000	16,999*	1,020,017*

Table 1. Description of data sets. *For the real data sets, total adapter content is unknown; we provide the number of full-length adapters and adapter bases present.

We evaluated both the speed and the accuracy of Atropos relative to other state-of-the-art read trimming tools using both simulated and real-world data (Table 1). As trimming of single-end reads is unchanged from the original Cutadapt method and is also decreasing in relevance as most current experiments used paired-end data, we focused our benchmarking on trimming of paired-end reads. Sturm et al. demonstrate that Skewer (Jiang et al., 2014) and SeqPurge (Sturm et al., 2016) stand out as having superior performance in paired-end read trimming; thus, we chose to benchmark Atropos against these tools. We also evaluated the accuracy of our new insert-match algorithm against the original Cutadapt adapter-match algorithm.

To simulate paired-end read data, we use the ART simulator (Huang et al., 2012) that was modified by Jiang et al. to add adapter sequences to the ends of simulated fragments. ART simulates reads based on empirically derived quality profiles specific to each sequencing platform. A quality profile consists

of distributions of quality scores for each nucleotide at each read position, expressed as read counts aggregated from multiple sequencing experiments, where quality scores are in Phred scale ($-10log_{10}(e)$, where e is the probability that the base call is erroneous). We developed an *R* script to artificially inflate the error rates in an ART profile to a user-defined level. For each row in the profile with quality score bins $e_1..e_n$ and corresponding read counts $r_1..rn$, the overall error rate can be computed as:

$$E = \frac{\sum_{i=1}^{n} e_i r_i}{\sum_{i=1}^{n} r_i} \tag{4}$$

We use the R function *optim* with the variable metric ("BFGS") algorithm to optimize a function that adds an equal number of counts C to each Phred-score bin in the distribution and then compares the overall error rate to the user-desired error rate E':

$$f(C, E') = \frac{\sum_{i=1}^{n} e_i (r_i + C)}{\sum_{i=1}^{n} (r_i + C)} - E' \tag{5}$$

We simulated ~800k 125 bp paired-end reads using the Illumina 2500 profile at error rates that were low/typical (~0.2%, the unmodified profile), intermediate (~0.6%), and high (~1.2%).

We evaluated the accuracy of the tools on the simulated data by comparing each trimmed read pair to the known template sequence. We counted the frequency of following outcomes: the fragment does not contain adapters but is trimmed anyway ("wrongly trimmed"), the fragment is under-trimmed, or the fragment is over-trimmed. We also counted the total number of under- and over-trimmed bases.

We also benchmarked the tools on a two real-world data sets of 1M read pairs each: an amplicon sequencing data set generated in our lab (REF: Maria's paper) and 1M randomly sampled read pairs from a whole-genome bisulfite sequencing (WGBS) library generated from the GM12878 cell line as part of the ENCODE project (ENCODE Project Consortium, 2012). Since the genomic origins of the templates are not known *a priori*, we instead compared the read trimming tools based on improvement in the results of mapping the untrimmed versus trimmed reads. We used BWA MEM (Li, 2013) to map the amplicon reads to human genome build 37 (GRCh37), and we used bwa-meth (Pedersen et al., 2014) to map the WGBS reads to the bisulfite-converted GRCh37. We also compared the results of only adapter trimming to the results of adapter trimming plus additional quality trimming, using a minimum quality threshold of 20. The supplement describes in detail how bisulfite reads are mapped and how we used the mapping information to determine the effects of read trimming.

Although sequence analysis is sometimes performed using a desktop computer, analysis of the volumes of data currently being generated increasingly requires the use of high-throughput computing facilities ("clusters"). The hardware architecture of a cluster is often different from that of a desktop computer. Most importantly, storage in a cluster is typically centralized and accessed by the compute nodes via high-speed networking. Such an architecture inevitably adds latency to file reading and writing operations ("I/O"). Cluster nodes also typically have more processing cores and memory available than desktop computers. This means that the performance of software with intensive I/O usage (such as read trimming) is likely to be quite different on a desktop versus a cluster. To examine the impact of these architectural differences, we ran all benchmarks on both a desktop computer (a Mac Pro) having a 3.7 GHz quad-core Intel Xeon E5 processor and 32 GB RAM, and on a cluster node having 4 X 3.2 GHz eight-core Intel Xeon E5 processors and 128 Gb memory, and with all data being read from and written to network-accessible storage.

Finally, we provide scripts in our GitHub repository that can be used to re-run our analysis, and that also enable other tool developers to benchmark their software against Atropos.

3.2 Simulated Data

3.2.1 Performance

On a desktop computer with 4 processing cores, we found that Skewer had the fastest overall execution time and SeqPurge the slowest (2). Atropos running in parallel-write mode and using the adapter-match algorithm was nearly as fast as Skewer.

As expected, execution times on a cluster node using 4 threads were approximately twice those observed on a desktop computer (3). Part of this difference may be explained by a slight difference in

Program	Execution Time (sec.) Min	Max
Atropos (adapter + writer compr.)	41.07	55.51
Atropos (adapter + parallel write)	23.18	30.75
Atropos (insert + writer compr.)	41.75	56.22
Atropos (insert + parallel write)	32.98	41.33
SeqPurge	96.67	114.24
Skewer	**20.75**	29.40

Table 2. Execution time for programs running on desktop with 4 threads. Each program was executed multiple times, and Atropos was run with all combinations of alignment algorithm (insert-match or adapter-match) and output mode (writer-compression or parallel-write). The minimum and maximum execution times for each program are shown, with the lowest overall execution time in bold.

	Execution Time (Min|Max sec.)					
Program	4 Threads		8 Threads		16 Threads	
Atropos (adapter + worker compr.)	99.39	493.85	56.64	99.05	**30.38**	64.05
Atropos (adapter + writer compr.)	75.87	346.80	77.63	109.05	72.37	102.00
Atropos (adapter + parallel write)	46.35	121.18	**27.63**	34.39	**21.27**	25.40
Atropos (insert + worker compr.)	115.28	292.61	65.36	100.84	**34.72**	53.82
Atropos (insert + writer compr.)	77.27	119.45	77.53	108.97	72.01	109.27
Atropos (insert + parallel write)	65.30	137.21	36.65	47.14	**21.57**	28.11
SeqPurge	372.54	517.83	192.85	226.01	94.88	122.39
Skewer	**40.12**	122.07	40.22	58.78	40.80	100.71
	Memory Usage (Min|Max Gb)					
Atropos	1.2	2.0	0.011	0.012	0.008	0.008
SeqPurge	0.47	2.0	0.011	0.012	0.009	0.011
Skewer	0.51	1.8	0.011	0.012	0.012	0.014

Table 3. Top: Execution time for programs running on a cluster with 4, 8, or 16 threads. Each program was executed multiple times, and Atropos was run with all combinations of alignment algorithm (insert-match or adapter-match) and output mode (worker-compression, writer-compression or parallel-write). The minimum and maximum execution times for each program are shown, and notable results are highlighted in bold. Bottom: The minimum and maximum memory usage for each program across all executions.

processor speeds (3.7 GHz on the desktop versus 3.2 GHz on the cluster node); however, we expect that most of the difference is due to the increased latency involved in network-based I/O on the cluster.

Interestingly, when increasing the number of threads from 4 to 8 and 16, the execution time of Skewer remains unchanged, while Atropos and SeqPurge achieve roughly linear decreases in execution time. With 8 and 16 threads, Atropos in parallel-write mode is the fastest of the tools, and with 16 threads Atropos in worker-compression mode is also faster than SeqPurge and Skewer.

On the other hand, Atropos uses significantly more memory than SeqPurge or Skewer (3). We expect this is partially due to overhead of automatic memory management in Python compared to C++ (in which SeqPurge and Skewer are implemented), but in larger part results from Atropos' use of of in-memory queues to communicate between parallel processes. We note that Atropos provides parameters to limit memory usage (although typically at the expense of performance).

3.2.2 Accuracy

We found that the four adapter matching algorithms had different biases toward under- and over-trimming (Table 4). Across the three sequencing error rates, Skewer had the lowest frequency of wrongly trimming reads while SeqPurge had the highest. The Atropos adapter-match algorithm and Skewer had similarly low frequencies of over-trimming, while the Atropos insert-match algorithm and SeqPurge had similarly low frequencies of over-trimming. Overall, the Atropos insert-match algorithm and SeqPurge demonstrated the lowest error rates (0.01%).

Program	Reads				Bases		
	Wrongly Trimmed	Over-trimmed	Under-trimmed	Total Error	Over-trimmed	Under-trimmed	Total Error
Error rate 0.2%							
Atropos							
adapter-match	51 (0.01%)	**1** (0.00%)	28,991 (9.72%)	3.71%	490	102,133	0.054%
insert-match	134 (0.03%)	26 (0.01%)	**27** (0.01%)	**0.01%**	1,036	66	**0.001%**
SeqPurge	2,639 (0.55%)	37 (0.01%)	29 (0.01%)	**0.01%**	7,708	**29**	0.004%
Skewer	**10** (0.00%)	7 (0.00%)	283 (0.09%)	0.04%	**21**	22,029	0.012%
Error rate 0.6%							
Atropos							
adapter-match	72 (0.01%)	6 (0.00%)	28,843 (9.68%)	3.69%	733	101,839	0.054%
insert-match	168 (0.03%)	14 (0.00%)	**31** (0.01%)	**0.01%**	1,434	62	**0.001%**
SeqPurge	2,595 (0.54%)	26 (0.01%)	38 (0.01%)	**0.01%**	6,701	**41**	0.004%
Skewer	**6** (0.00%)	**3** (0.00%)	413 (0.14%)	0.05%	**9**	36,685	0.019%
Error rate 1.2%							
Atropos							
adapter-match	76 (0.02%)	**5** (0.00%)	30,152 (10.12%)	3.86%	721	117,027	0.062%
insert-match	175 (0.04%)	12 (0.00%)	46 (0.02%)	**0.01%**	1,440	110	**0.001%**
SeqPurge	2,652 (0.55%)	24 (0.01%)	**34** (0.01%)	**0.01%**	7,628	**36**	0.004%
Skewer	**7** (0.00%)	**5** (0.00%)	644 (0.22%)	0.08%	**12**	67,066	0.035%

Table 4. Trimming accuracy on simulated data with three different base-call error rates.

In terms of numbers of over- and under-trimmed bases, the Atropos insert-match algorithm and SeqPurge clearly had the best performance (Table 4) at all sequencing error rates. The two algorithms had similarly low numbers of under-trimmed bases, but the Atropos insert-match algorithm had a lower number of over-trimmed bases, giving it the lowest overall error rate (0.001%). On the other hand, Skewer and the Atropos adapter-match algorithm left substantial numbers of under-trimmed bases, resulting in about 10-fold higher overall error rates.

Additionally, we found that all tools discarded very similar numbers of reads (˜1.8%) that were below the minimum length threshold of 25 bp after trimming. These were reads with short insert sizes, which have a high rate of spurious mapping, and thus it is common practice to discard them.

3.3 Real Data

We first tested Atropos' adapter detection module on the real data sets. Using the first 10,000 reads in each pair of FASTQ files, Atropos correctly detected that sequences of the adapters used in constructing each library. In the case of the WGBS data set, the adapters were found in the list of known contaminants (read 1: "TruSeq Adapter, Index 7" and read 2: "TruSeq Universal Adapter"). On the other hand, a custom index was used in the read 1 adapter of the amplicon library. Atropos correctly identifies the custom index (AACGTGAT), and also correctly identifies the TruSeq Universal Adapter as the read 2 adapter.

3.3.1 Performance

We performed adapter trimming on the real data set in a cluster environments using 16 parallel cores. Again, we found Atropos (in worker-compression mode) to have the best performance both with and without additional quality trimming (Table 5). We also performed read mapping on the cluster with 16 cores and observed that the trimmed reads produced by each combination of tool and quality threshold resulted in substantially different execution times (Table 5). The mapping of reads trimmed by Atropos using the adapter-match algorithm took twice as long as those trimmed by any other tool, while the reads trimmed by Atropos using the insert-match algorithm and no additional quality trimming took the least amount of time to map. At a quality trimming threshold of 20, the trimmed reads generated by SeqPurge mapped slightly faster than those generated by Atropos.

Program	Trimming Time (sec.) Q0	Q20	Mapping Time (min.) Q0	Q20
WGBS Data				
Untrimmed reads			45:45	
Atropos	**56.46**	**55.58**	37:32	**30:37**
SeqPurge	70.37	80.33	39:10	31:55
Skewer	69.26	72.21	**36:06**	32:15

Table 5. Execution and read mapping times for programs trimming real data on a cluster with 16 threads. Each program was executed with no additional quality trimming (Q0) and with quality trimming at a minimum base quality of 20 (Q20). Atropos was run with the insert insert-match algorithm in worker-compression mode. SeqPurge and Atropos were run with error correction enabled (Skewer performs error correction by default).

3.3.2 Accuracy

We chose to assess read trimming accuracy in practical terms – as the number of trimmed reads mapped at at given quality (MAPQ) cutoff, relative to the number of untrimmed reads mapped at that cutoff. We found that trimming by Atropos resulted in the greatest increase in number of mapped reads at all quality cutoffs (Figure 2). We found that trimming with SeqPurge resulted in similar, but smaller, gains in mapping quality, while trimming with Skewer resulted in the smallest gains in quality.

Additionally, we found that additional quality trimming in addition to adapter trimming has a substantial negative effect on read mapping, at least for bisulfite reads mapped using bwa-meth. However, quality trimming by Skewer had the least negative effect on mapping quality of the three programs, and quality trimming by SeqPurge had the greatest negative effect on mapping quality.

4 CONCLUSIONS

Our results demonstrate that adapter trimming tools are approaching optimal accuracy, at least for the (currently) most common type of data – paired-end short reads with 3' adapters. On synthetic data with varying error rates, Atropos (using our new insert-match algorithm) and SeqPurge both demonstrated nearly negligible overall error rates of 0.01% at the read level, and Atropos has the lowest base-level error rate of 0.001%.

On real data, we also find that Atropos has superior performance, leading to the greatest increase in read mapping quality. We also find unequivocally that stringent quality trimming has a negative effect on WGBS data. For reads trimmed with a quality threshold of 20, all mapping statistics are worse than those for untrimmed reads.

In terms of performance, Atropos is faster than SeqPurge, and is the overall fastest tool when at least 16 processing cores are available.

Thus, overall, Atropos offers the best combination of accuracy and performance of the tools that we evaluated. Furthermore, Atropos has the richest feature set of the three tools, including Methyl-Seq-specific trimming options, automated adapter detection, and support for data generated by many sequencing methods (ABI SOLiD, Illumina NextSeq, mate-pair libraries, and single-end sequencing). Finally, Atropos is easily installable on any system with Python 3.3+ (using the command 'pip install atropos').

5 DATA AVAILABILITY

- Atropos can be installed from the Python Package Index (pypi) using the *pip* tool: 'pip install atropos'.
- The Atropos source code, including all scripts needed to execute the analyses in this paper, are available at `https://github.com/jdidion/atropos`. The portions of Atropos developed by JPD are a work of the US government, and thus all copyright is waived under a CC0 1.0 Universal Public Domain Dedication (`https://creativecommons.org/publicdomain/zero/1.0/`).

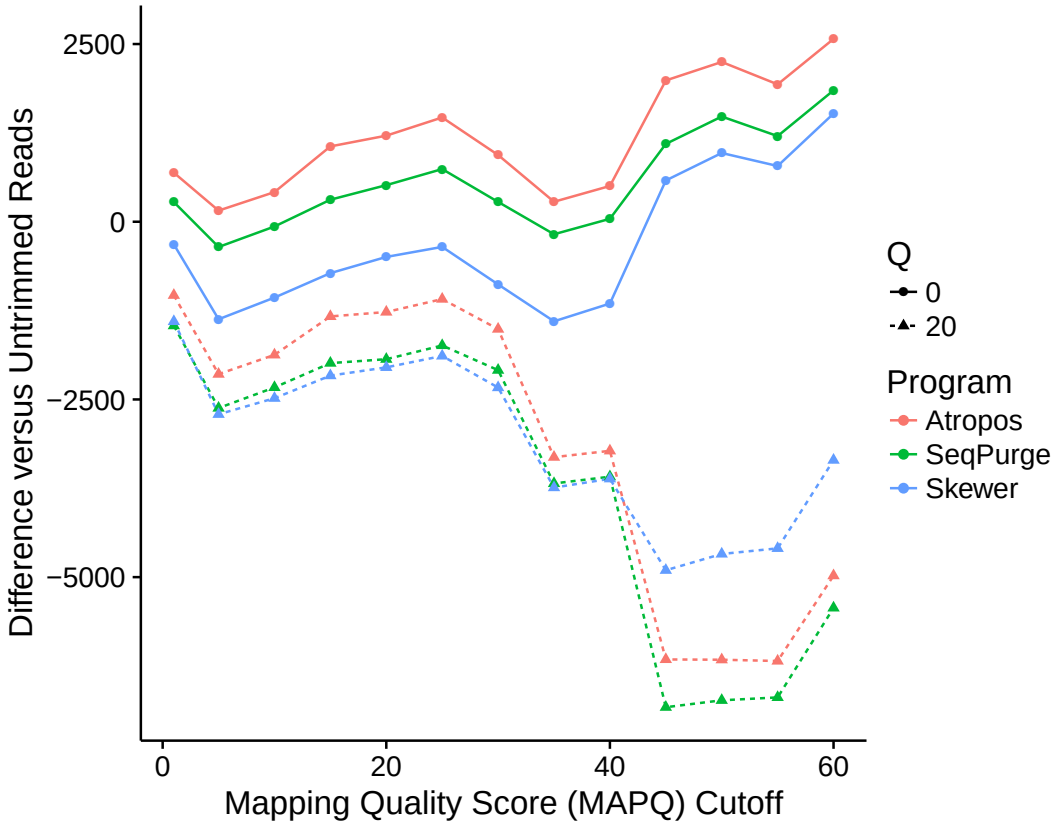

Figure 2. Atropos trimming best improves mapping of real sequencing reads. Difference between the number of trimmed and untrimmed reads mapped with quality (MAPQ) greater than or equal to each cutoff (1..60 at intervals of 5). We mapped both untrimmed and trimmed reads to the genome. Reads were trimmed with all three programs, both without additional quality trimming (Q=0) and with quality trimming at a threshold of Q=20.

- The simulated data sets are also available in the Atropos GitHub repository.
- The real amplicon data set is available at XXX.
- The real GM12878 WGBS data (accession ENCLB794YYH) is available from the ENCODE project website: `https://www.encodeproject.org/experiments/ENCSR890UQO/`.

6 ACKNOWLEDGEMENTS

We would like to thank Marcel Martin and colleagues for publishing Cutadapt under an open source license that is compatible with modifications and improvements. JPD and FSC are funded by the NIH intramural program.

REFERENCES

Bock, C. (2012). Analysing and interpreting DNA methylation data. *Nature Reviews Genetics*, 13(10):705–719.

Del Fabbro, C., Scalabrin, S., Morgante, M., and Giorgi, F. M. (2013). An extensive evaluation of read trimming effects on Illumina NGS data analysis. *PLoS One*, 8(12):e85024–None.

ENCODE Project Consortium (2012). An integrated encyclopedia of DNA elements in the human genome. *Nature*, 489(7414):57–74.

Huang, W., Li, L., Myers, J. R., and Marth, G. T. (2012). ART: a next-generation sequencing read simulator. *Bioinformatics*, 28(4):593–594.

Jiang, H., Lei, R., Ding, S.-W., and Zhu, S. (2014). Skewer: a fast and accurate adapter trimmer for next-generation sequencing paired-end reads. *BMC Bioinformatics*, 15:182–None.

Li, H. (2013). Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *arXiv:1303.3997 [q-bio]*. arXiv: 1303.3997.

Martin, M. (2011). Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet.journal*, 17(1):pp. 10–12.

Pedersen, B. S., Eyring, K., De, S., Yang, I. V., and Schwartz, D. A. (2014). Fast and accurate alignment of long bisulfite-seq reads. *arXiv:1401.1129 [q-bio]*. arXiv: 1401.1129.

Sturm, M., Schroeder, C., and Bauer, P. (2016). SeqPurge: highly-sensitive adapter trimming for paired-end NGS data. *BMC Bioinformatics*, 17:208.