Atria: An Ultra-fast and Accurate Trimmer for Adapter and Quality Trimming

Jiacheng Chuan [1,2], Aiguo Zhou[1,3], Lawrence Richard Hale [2], Miao He[4], Xiang Li [1,*]

[1] *Canadian Food Inspection Agency, Charlottetown, PE C1A5T1, Canada*

[2] *Department of Biology, University of Prince Edward Island, Charlottetown, PE C1A4P3, Canada*

[3] *Guangdong Laboratory for Lingnan Modern Agriculture, South China Agricultural University, Guangzhou 510642, China*

[4] *School of Life Sciences, Sun Yat-sen University, Guangzhou, Guangdong, 510275, China*

E-mail:

jiacheng.chuan@inspection.gc.ca (Chuan J)

aiguozhou@scau.edu.cn (Zhou A)

lhale@upei.ca (Hale L)

lsshem@mail.sysu.edu.cn (He M)

sean.li@inspection.gc.ca (Li X)

* Corresponding author: Xiang Li, sean.li@inspection.gc.ca

Abstract

Background: As Next Generation Sequencing takes a dominant role in terms of output capacity and sequence length, adapters attached to the reads and low-quality bases hinder the performance of downstream analysis directly and implicitly, such as producing false-positive single nucleotide polymorphisms (SNP), and generating fragmented assemblies. A fast trimming algorithm is in demand to remove adapters precisely, especially in read tails with relatively low quality.

Findings: We present a trimming program named Atria. Atria matches the adapters in paired reads and finds possible overlapped regions with a super-fast and carefully designed byte-based matching algorithm (*O(n)* time with *O(1)* space). Atria also implements multi-threading in both sequence processing and file compression and supports single-end reads.

Conclusions: Atria performs favorably in various trimming and runtime benchmarks of both simulated and real data with other cutting-edge trimmers. We also provide an ultra-fast and lightweight byte-based matching algorithm. The algorithm can be used in a broad range of short-sequence matching applications, such as primer search and seed scanning before alignment.

Availability & Implementation: The Atria executables, source code, and benchmark scripts are available at https://github.com/cihga39871/Atria under the MIT license.

Research Area: Software and Workflows

Classifications: Bioinformatics, Software Engineering

Statement of Need

Background

Next generation sequencing (NGS) is a revolutionary new technology that produces massive, high-resolution genome sequence data to facilitate a broad range of biological applications. Illumina paired-end sequencing can read a DNA fragment from both ends and generate accurate reads for downstream bioinformatics analysis, such as assembly, resequencing, transcriptome profiling, variant calling, epigenome profiling, chromatin interaction, and chromosomal rearrangements [1, 2].

In paired-end library preparation, adapter sequences are the technical sequences ligated to both sides of inserts, which are the DNA fragments of interest. Then, DNA molecules with adapters are sequenced from both ends of the inserts so paired-end reads are generated. If insert sizes of paired-end reads are less than the read lengths, inserts are reversely complementary, and adapters are sequenced after reading through the inserts (**Fig. 1**). Thus, adapter contamination in the 3' end needs to be removed before downstream analysis.

Cleaning adapters can therefore be achieved by searching adapter sequences and/or aligning paired reads (**Fig. 1**). To date, some trimmers, such as AdapterRemoval [3], Trim Galore [4], and Trimmomatic [5], use both types of information to clean adapters. However, when the quality of sequencing reads decreases, the trimming process employing both types of information is likely to give different trimming suggestions.

Trimmers thus face a bottleneck when working on trimming adapters at accurate positions. Also, extremely short adapters at the low-quality 3' end are sometimes difficult to detect. Thus, trade-offs between trimming truncated adapters, and retaining inserts intact, become necessary.

These two issues hinder trimmers from cleaning adapter sequences and leaving DNA inserts intact. To combat this, we launch Atria, an integrated trimming program for NGS data. Atria uses a super-fast byte-based matching algorithm to detect adapters and reverse complementary regions of paired reads, and integrates carefully designed decision rules to infer true adapter positions. Thus, Atria can trim extremely short adapter sequences at accurate positions and not over-trim reads without adapters (**Fig. 1**).

In addition to adapter trimming, Atria integrated a set of trimming and filtering methods, such as consensus calling for overlapped regions, quality trimming, homopolymer trimming, N trimming, hard clipping from both ends, and read complexity filtration.

Implementation

The adapter finding algorithms used in Atria can be categorized in the following portions: DNA encoding, matching algorithm, matching and scoring, decision rules, consensus calling, quality trimming, and IO optimization (**Fig. 2**).

DNA encoding

The DNA encoding algorithm is developed based on BioSequences, a Julia package from BioJulia [6]. The original BioSequences package encodes DNA bases A, C, G, T as four-bit codes 0001, 0010, 0100, 1000, respectively. Extended codes are also supported, such as N (1111), S (0110), and gap (0000). DNA sequences are encoded and stored in a contiguous block of Random-Access Memory (RAM) as a dense array of unsigned 64-bit integers (UInt64) (**Fig. 2A**).

Atria makes use of the property of dense arrays to extract sequences as unsigned integers from available memory locations. When accessing the last several indices of a sequence, the extraction is illegal because operating systems do not allow the loading of data outside of sequence boundary. To solve the issue, Atria constructs a bit-safe sequence array, which elongates the sequence boundary by appending a UInt64 to the end of the original array, and setting all bits after the end of encoded DNA to 0 (**Fig. 2A**).

It is noticeable that the smallest addressable unit of memory is one byte (8 bits) while each DNA is encoded in four bits, so only the even indices of sequence can be directly extracted (defining indices start from 0) (**Fig. 2A**). The extraction from odd indices requires extra operations, which is avoidable in many scenarios of a well-designed algorithm.

We denote a UInt64 extracted from the memory position n of sequence a by a_n. a_n is a 16-mer and represents the subsequence of a indexed from $2n$ to $2n$+15, which is denoted by $a[2n{:}2n+15]$ (**Fig. 2A**).

Matching algorithm

Given two sequences a and b, we plan to match the 16-base-long head of a to each index of b. However, only the even indices of b can be extracted from memory without bitwise operations, so we prepared two UInt64 of a: a_0 and a_-. a_0 is the 16-mer UInt64 loaded from the position 0 of a, and a_- can be computed from the following bitwise operations: ($a_0 >> 4 \mid a_1 << 4$). In this way, a_0 represents the subsequence of a indexed from 0 to 15 ($a[0{:}15]$), and a_- represents $a[1{:}16]$ (**Fig. 2B**).

In this way, the problem of matching the 16-base-long head of a to each index of b is converted to the problem of matching two 16-mers, a_0, and a_-, to each addressable memory position of b. The latter requires less bitwise operations.

The number of mismatches K is computed in the formula:

$$K_{an,bn} = 16 - count\_ones(an \,\&\, bn)$$

where *count_ones* counts the number of ones in the binary representation of the UInt64.

Let k denote the user-defined number of mismatches allowed in the 16-mer comparison of UInt64 a_n and b_n (k = 2 by default). After matching a_0 and a_- to each addressable memory position of b, if the minimum number of mismatches is not greater than k, the smallest index of b of the minimum mismatches is reported.

Therefore, the complexity of the matching algorithm is *O(n)* time with *O(1)* space, so its speed is extremely fast. One limitation is that when computing the number of mismatches of a_n and b_n, and if they have ambiguous bases in the same indices, the number of mismatches is underestimated. Another limitation is that the algorithm does not handle indels. Those limitations are compensated in the design of adapter matching, scoring, and decision rules.

Matching and scoring

We implement four pairs of matching to utilize properties of paired-end reads thoroughly: (1) matching adapter 1 head to read 1, (2) matching adapter 2 head to read 2, (3) matching read 1 head to reverse complement of read 2 and (4) matching read 2 head to reverse complement of read 1 (**Fig. 2C**). If the maximum number of bases matched of (1) and (2) is less than a user-defined cut-off (default is 9), (3) and (4) will be performed with a loosed k (= $k_{original}$ + 1). If the largest number of matched bases of the four matches is greater than the cut-off, and some matches do not meet the requirement, we will re-run those matches with a loosed k (= $k_{original}$ + 3) at the insert size indicated from the best match. If the new number of matched bases is greater than the cut-off, the old match will be discarded.

The scoring system measures the matching reliability of the whole 16-mer rather than each base. The Phred quality score Q of each base is converted to the probability P of that the corresponding base being correct using the formula:

$$P = 1 - 10^{\left(-\frac{Q}{10}\right)}$$

Then, the average base quality $\bar{P}$ of 16-mer sub-sequence *a* at the memory position *n* is computed:

$$\overline{P_{a_n}} = \frac{1}{16}\sum_{i=2n}^{2n+15} P_{a[i]}$$

Notably, if the read quality is too low, it would imply an invalid match. However, in reality, invalid matches are filtered out by the kmer-based algorithm. To solve the discordance, we limit the lower bound of $\bar{P}$ to 0.75 manually.

The matching score *S* between a_n and b_m is defined as:

$$S_{a_n,b_m} = count\_ones(a_n \mathbin{\&} b_m) \cdot \overline{P_{a_n}} \cdot \overline{P_{b_m}}$$

where *count_ones* counts the number of ones in the binary representation of the UInt64. When sequence *a* is a user-defined adapter, $\bar{P}_a = 1$ is used. Generally, the matching score *S* is ranged from 0 to 16.

Pseudocode 1: Matching and scoring

```
r1_pos_adpt, r1_nmatch_adpt = match(adapter1, r1, k)
r2_pos_adpt, r2_nmatch_adpt = match(adapter1, r1, k)
k_extra = max(r1_nmatch_adpt, r2_nmatch_adpt) < 9 ? 1 : 0
r1_pos_pe, r1_nmatch_pe = match(reverse_complement(r2), r1, k + k_extra)
r2_pos_pe, r2_nmatch_pe = match(reverse_complement(r1), r2, k + k_extra)

max_nmatch = max(r1_nmatch_adpt, r2_nmatch_adpt, r1_nmatch_pe, r2_nmatch_pe)
max_pos = corresponding position of max_nmatch
if max_nmatch > 9
```

```
      for matches with any nmatch < 9
          redo match with loosed k = k + 3 at max_pos
          replace old results if nmatch > 9

r1_prob_adpt = average_16mer_quality(r1, r1_pos_adpt)
r2_prob_adpt = average_16mer_quality(r2, r2_pos_adpt)
r1_prob_head = average_16mer_quality(r1, 1)
r2_prob_head = average_16mer_quality(r2, 1)
r1_prob_pe = average_16mer_quality(r1, r1_pos_pe)
r2_prob_pe = average_16mer_quality(r2, r2_pos_pe)
r*_prob_* = 0.75 if any r*_prob_* < 0.75

r1_score_adpt = r1_nmatch_adpt * r1_prob_adpt
r2_score_adpt = r2_nmatch_adpt * r2_prob_adpt
r1_score_pe = r1_nmatch_pe * r1_prob_pe * r2_prob_head
r2_score_pe = r2_nmatch_pe * r2_prob_pe * r1_prob_head
```

Decision rules

This module infers correct adapter positions from the four pairs of matching described in the previous section. It is illustrated and self-explanatory in **Fig. 2D**. First, in each read, the adapter and paired-end matches are compared. The one with the higher matching score is chosen. If both matches support the same adapter position, the matching score of the read is the sum of adapter and paired-end matching scores. Then, the matches of the two paired-end reads are compared using the same strategy. If one read finds an ideal adapter (matching score > 10 by default) while the other read is too short to check or the average base accuracy of its 16-mer is less than 0.6 (Phred Q < 5), both reads will be trimmed. If the matching score of a given read pair is less than 10 (by default), the read pair will not be trimmed.

Other read pairs will be taken a further examination to reduce false positives, which are usually adapter matches at read tails. A read tail is defined as the last several bases (default is 12 bp) of each read. Reads are not trimmed if both statements are true: (1) In any paired read, the adapter is found at the tail, but the paired-end match is not; (2) In both paired reads, adapter and pair-end matches suggest different trimming positions.

Before the final trimming, one additional step is required for the accurate positioning of adapter sequences. The previous steps usually assume the read 1 and 2 have the same length of insert sizes, but indel in reads usually lead to over or under trim one base. To prevent this circumstance, Atria re-positions the adapter by matching one adjacent base with the first four bp of adapter sequences. The position of the highest number of bases matched is chosen to trim. This step is ignored when the inferred insert size is greater than the read length minus three because, in this situation, the adapter sequence is too short to check.

Pseudocode 2: Decision rules

```
function correct_insert_size(pos1, score1, pos2, score2)
    if pos1 == pos2
        return pos1, score1 + score2
    else
        score = max(score1, score2)
```

```
          pos = corresponding pos of max score
          return pos, score

r1_pos, r1_score = correct_insert_size(r1_pos_adpt, r1_score_adpt, r1_pos_pe, r1_score_pe)
r2_pos, r2_score = correct_insert_size(r2_pos_adpt, r2_score_adpt, r2_pos_pe, r2_score_pe)
r12_pos, r12_score = correct_insert_size(r1_pos, r1_score, r2_pos, r2_score)

if r1_pos != r2_pos
     if r1_score > 10
       r2_prob = average_16mer_quality(r2, r1_pos)
       @goto “trim” if r2_prob < 0.6
     elseif r2_score > 10
       r1_prob = average_16mer_quality(r1, r2_pos)
       @goto “trim” if r1_prob < 0.6

function check_read_tail(read)
     E_adpt = whether adapter found at read tail
     E_pe = whether pair-end match found at read tail
     E = E_adpt & E_pe  # both matches in read tail
     R = rx_pos_adpt == rx_pos_pe  # adapter and pair-end match at same position
     return E, R

E1, R1 = check_read_tail(r1)
E2, R2 = check_read_tail(r2)
E = E1 | E2  # at least one read matching in read tail
R = R1 | R2  # at least one read matching at the same position
```

```
is_false_positive = E & !R

if r12_score > trim_score & !is_false_positive
    @label "trim"
    r1_pos_adjusted = adjacent_one_bp_check(r1, adapter1, r12_pos)
    r2_pos_adjusted = adjacent_one_bp_check(r2, adapter2, r12_pos)
    trim(r1, r1_pos_adjusted)
    trim(r2, r2_pos_adjusted)
```

Consensus calling

In this module, the overlapped base pairs of read 1 and 2 are corrected to the corresponding bases with higher quality scores. It has three steps, prediction, assessment, and correction.

In the prediction step, Atria makes a preliminary estimate of whether a read pair contains an overlapped region. If adapters are trimmed and the remaining lengths of read 1 and 2 are the same, the prediction passes. If no adapter can be trimmed, two additional matching and scoring are required. The head of the reverse complement of read 2 is matched to read 1, and the head of the reverse complement of read 1 is matched to read 2. If the two matches reach a consensus, the prediction passes. Otherwise, the prediction fails and consensus calling is skipped.

In the assessment step, Atria compares the whole overlapped region using a similar matching algorithm, except that ambiguous bases (N, 1111) are converted to gaps (0000) before matching. If the ratio of mismatch is greater than a user-defined value (28% by default), the assessment fails, and consensus calling is skipped.

In the correction step, each base pair in the overlapped region is corrected to the corresponding base with the highest quality score.

Quality trimming

Atria implements a traditional sliding window algorithm to remove the low-quality tail. The sliding window scans from the front of the read and computes the average Phred quality score of the sliding window. If the average quality is less than a given threshold, the read tail is removed.

IO optimization

Atria spends more time on reading and writing than matching and trimming, so the key to reducing runtime is to optimize IO usage. Considering that a large amount of RAM is easily accessible nowadays, Atria trades increased RAM usage with decreased time. A large block of memory is allocated for reading input files, which is then wrapped and encoded to FASTQ objects parallelly using multi-threading. On the contrary, in the writing process, Atria unboxes and decodes FASTQ objects to string vectors in parallel and writes sequentially to files. In addition, pigz (parallel gzip) and pbzip2 (parallel bzip2) are called for compression and decompression when needed [7, 8]. Atria also support running with a single thread.

Comparison to related work

The performance of adapter trimming on a simulated dataset

We simulated 8.9 G bases with 100 bp paired-end reads from the *Arabidopsis thaliana* reference genome using the Skewer modified ART, a public NGS read simulator to allow adapters in the reads [9, 10]. The simulation profile was trained from a 101 bp paired-end public dataset SRR330569, and the 33 bp adapter pair used in read simulation is AGATCGGAAGAGCACACGTCTGAACTCCAGTCA and AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT [11].

Atria v3.0.0 was benchmarked with cutting-edge and popular trimmers, including AdapterRemoval v2.3.1 [3], Skewer v0.2.2 [10], Fastp v0.21.0 [12], Ktrim v1.2.1 [13], Atropos v1.1.29 [14], SeqPurge v2012_12 [15], Trim Galore v0.6.5 [4] and Trimmomatic v0.39 [5]. Only adapter trimming was used, and other trimming and filtration were disabled. Detailed command line arguments are listed in **Table S1**. Each trimming software was running on an idle Ubuntu 19.10 server with a 32-thread Intel i9-9960X Central Processing Unit (CPU) @ 3.10 GHz, 128 gigabyte (GB) DDR4-3200 RAM, and a 2 terabyte (TB) Samsung 970 EVO Solid State Drive (SSD) (sequential reads and writes up to 3.5 and 2.5 TB/s).

The trimming performance was evaluated based on the following metrics: positive predictive value (PPV), as the fraction of the number of correctly trimmed reads to all trimmed reads; sensitivity, as the fraction of the number of correctly trimmed reads to the reads with adapters; specificity, as the fraction of the number of untrimmed reads without adapters to all reads without adapters; and Matthew's correlation coefficient (MCC) measuring overall quality of pattern recognition, as

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FN)(TN + FP)}}$$

where TP is the number of reads trimmed correctly, TN is the number of untrimmed reads without adapters, FP is the number of over-trimmed reads, and FN is the number of under-trimmed reads [3, 10].

Table 1 Adapter trimming performance on the 8.9 G bases with 100 bp paired-end simulated data

Trimmer	PPV (%)	Sensitivity (%)	Specificity (%)	MCC (%)
Atria	99.35	99.81	99.82	99.51
AdapterRemoval	99.42	99.94	99.83	99.61
Atropos	99.57	97.34	99.88	98.00
Fastp	98.73	99.58	99.61	98.92
Ktrim	91.51	85.85	98.84	87.84
SeqPurge	57.92	99.80	76.84	66.62
Skewer	99.58	99.53	99.88	99.44
Trim Galore	40.05	82.98	62.39	38.96
Trimmomatic	99.29	57.86	99.88	71.05

The adapter trimming performance is shown in **Table 1**. AdapterRemoval, Atria and Skewer were the top-class adapter trimmers in terms of MCC (99.61%, 99.51%, 99.44%, respectively) (**Table 1**). Fastp (98.92%) and Atropos (98.00%) were in the second tier (**Table 1**). Ktrim obtained a good specificity (98.84%) but sacrificed its sensitivity (85.85%), and Trimmomatic achieved an exceptional specificity (99.88%) by trading off its sensitivity (57.86%) (**Table 1**).

To compare speed and efficiency, elapsed time (wall time) and average CPU consumption of each trimmer were recorded in different threading (1-32 threads) for uncompressed and gzip compressed data formats (**Fig. 3, Table S1**). Efficiency was defined as the fraction of processing speed to the percent of CPU utilized, so it was a better measurement, especially in CPU-intensive scenarios, such as running on a server with a job scheduling system or trimming multiple samples at the same time. Ktrim and Atria were two of the fastest trimmers in terms of speed and efficiency, from one to 16 threads (**Fig. 3, Table S1**). For uncompressed data, Trimmomatic was faster than Atria using 8-32 threads, but its real CPU usage was much greater than Atria (**Fig. 3, Table S1**). The speed and efficiency of AdapterRemoval and Skewer were generally 2-4 times less than Atria, and Atropos was the slowest one (**Fig. 3, Table S1**). SeqPurge did not support the output of uncompressed data, so it was only tested in the compressed benchmark.

When trimming compressed data, the speed of AdapterRemoval, Skewer, Fastp, Atropos and Trimmomatic kept constant when the number of threads increased from 4 to 32, because they failed to utilize more than four CPU in the IO process, while Atria and Trim Galore did not have the limitation (**Fig. 3, Table S1**). Atria was faster than Trim Galore, and the efficiency of Atria was constantly two to three times greater than Trim Galore (**Fig. 3, Table S1**). SeqPurge showed strange speed curves; when assigning a single thread to SeqPurge, the average CPU usage was 300%, and the speed and average CPU usage dropped when assigning 8 to 32 threads (**Fig. 3, Table S1**). In addition, Ktrim did not support output compressed files, so we ignored it. In general, Atria was the fastest trimmer when trimming compressed files.

The detailed statistics of adapter trimming accuracy on a simulated dataset

The previous portion benchmarks on a whole dataset. This section evaluates trimming accuracy regarding different read properties, including adapter presence or absence, base error, and adapter length. To achieve the goal, Atria integrates a benchmarking toolkit for read simulation and trimming analysis.

The read simulation method was inspired by how sequencers read DNA. First, an original DNA fragment (insert) with a given original insert size is simulated base by base. Adenine, thymine, cytosine, and guanine are randomly chosen repetitively. Then, the insert and adapter sequences are copied base by base with an error profile, which simulates the procedure of sequencing by synthesis. The error profile defines substitution rate, insertion rate, and deletion rate.

Twenty-one million read pairs were simulated with a uniform read length (100 bp), different error profiles, adapter length, and original insert sizes. The baseline error profile comprises a 0.1% substitution rate, 0.001% insertion rate, and 0.001% deletion rate, inspired by an Illumina error profile analysis [16]. 1x, 2x, 3x, 4x, and 5x baseline error profile, 16, 20, 24, 28, and 33 adapter lengths, and 66 to 120 even insert sizes are chosen. In this way, the reads with the least insert size have full lengths of adapters. The reads with 66-98 original insert sizes contain adapters, and the reads with 100-120 original insert sizes are free from adapter contamination, except for few reads with a 100 bp insert size containing indels. Therefore, in each condition combination, 30 thousand read pairs were simulated to avoid random errors. The reads were trimmed with the same method described in the last section.

The average trimming performance among different conditions is shown in **Fig. 4 A**. When adapters are present, Atria trims 99.9% adapters accurately, and SeqPurge, Fastp, and Atropos follow closely with an accuracy of 99.7% (**Fig. 4 A1**). When adapters are absent, AdapterRemoval, Skewer, Trimmomatic, Atropos, and Atria successfully leave 100.0% reads intact, and Fastp falls behind with 99.8% accuracy (**Fig. 4 A2**).

Fig. 4 B illustrates the trimming accuracy on different read error profiles. When adapters are present, the accuracy of all trimmers drops as error rates increase (**Fig. 4 B1**). Atria keeps the highest accuracy from 100.0% to 99.9%, and is almost not affected by different error rates (**Fig. 4 B1**). The accuracy of SeqPurge, Fastp, and Atropos decrease from 99.9% to 99.6%, 99.5%, and 99.4%, respectively (**Fig. 4 B1**). With no adapter present in reads, the accuracy is hardly influenced by error profiles (**Fig. 4 B2**), so the performance is similar to **Fig. 4 A2**. In addition, adapter lengths ranging from 16 to 33 bp are not relevant to most trimmers' accuracy, including Atria (**Fig. 4 C**).

The performance of adapter trimming on real sequencing dataset

RNA-Seq paired-end dataset (SRR330569)

SRR330569 is a real RNA-Seq dataset sequenced from *Drosophila simulans* with 5.46 G bases and 2 x 101 bp read length. It contains 38 bp adapter sequences AGATCGGAAGAGCGGTTCAGCAGGAATGCCGAGACCG and AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGTAGAT in read 1 and read 2, respectively. Adapter trimming was performed by different trimmers without other trimming or filtering methods. Then, a sliding-window based quality trimming was performed to remove low-quality tails (sliding window size = 5 and average Q score ≥ 15). The adapter-trimmed reads and adapter-and-quality-trimmed reads were mapped to the *Drosophila simulans* genome version 2.02 from FlyBase using Hisat2 v2.2.1, respectively [17, 18]. Mapping statistics were collected using SAMTools Stat v1.10 [19]. Skewer did consensus calling after adapter trimming, and no option was provided to disable it. To achieve benchmark parity, Skewer was compared to Atria with consensus calling enabled, and other trimmers were compared to Atria without consensus calling. Time trimming was recorded in accordance with a common scenario: inputs were gzip-compressed and trimmed with eight threads, and outputs were also gzip-compressed to reduce massive disk use. All tested trimmers worked in the scenario except that Ktrim could not output gzip files **(Table 2)**.

Atria was the fastest program to process and output compressed data in terms of wall time (**Table 2**). It also achieved the highest number of reads mapped and paired, and the percent of properly paired reads with or without quality trimming. Generally, higher base mapped is accompanied with higher error rate in the mapping process, so it is important to interpret the two metrics together. Atria had the lowest mapping error rate of 8.1833‰ and the forth highest base mapped (**Table 2**). The trimmers (AdapterRemoval, Fastp, and Atropos) with the highest three error rates has the highest base mapped (**Table 2**). Our program generally improved more than 5% compared to other trimmers for the data without quality trimming (**Table 2**). The mapping statistics of data without quality trimming were generally worse than with quality trimming except for Atria. Specifically, the properly paired rates of other trimmers without quality trimming were 0.5 to 4% less than with quality trimming (**Table 2**). Quality trimming also increased the number of mapped and paired reads and reduced the number of unmapped reads (**Table 2**).

Table 2 Performance of trimmers on real data

	Trimming and consensus		Trimming only							
Metric	**Atria**	**Skewer**	**Atria**	**AR**	**Atropos**	**Fastp**	**Ktrim***	**SeqPurge**	**Trim Galore**	**Trimmomatic**
Low-quality dataset (SRR330569, RNA, Hisat2 mapping)										
Elapsed time (min:sec)*	2:38	9:19	**2:32**	11:29	10:08	9:17	1:34 + GZ	3:53	3:39	9:38
No quality trimming										
Reads mapped and paired	26,126,804	24,694,330	**25,781,268**	24,559,060	24,505,656	24,545,646	24,196,658	24,240,072	24,046,542	22,797,620
Reads unmapped	27,276,761	28,254,804	**27,379,299**	28,338,455	28,410,022	28,350,294	28,747,248	28,591,442	28,647,873	29,649,287
Properly paired reads (%)	48.3	45.6	**47.6**	45.3	45.2	45.3	44.5	44.7	44.2	38.3
Base mapped (cigar)	2,387,212,225	2,354,164,041	2,322,436,545	**2,346,791,204**	2,341,355,822	2,344,847,438	2,316,915,673	2,304,776,514	2,317,321,510	2,237,846,534
Error rate (‰)	7.3952	9.5897	**8.1833**	9.8902	9.8536	9.8683	9.7920	9.7904	9.6994	9.3106
With quality trimming										
Reads mapped and paired	25,942,092	25,787,464	**25,728,206**	25,721,788	25,714,956	25,725,480	25,473,670	25,364,392	25,654,498	24,744,754
Reads unmapped	27,245,720	27,364,827	**27,361,655**	27,369,773	27,373,854	27,360,527	27,556,820	27,736,292	27,400,932	28,064,739
Properly paired reads (%)	47.9	47.6	**47.5**	**47.5**	**47.5**	**47.5**	46.9	46.8	47.3	42.3
Base mapped (cigar)	2,317,238,536	2,316,981,456	2,302,740,463	**2,304,584,269**	2,304,325,743	2,304,437,244	2,292,034,762	2,263,465,110	2,297,815,439	2,246,076,618
Error rate (‰)	7.1114	7.7882	7.8902	7.9160	7.9141	7.9149	7.9059	7.8787	7.8921	**7.5649**
High-quality dataset (ERR4695159, cell-free DNA, Bowtie2 mapping)										
Elapsed time (min:sec)*	3:08	11:34	**3:03**	13:48	13:41	11:29	1:41 + GZ	4:05	4:34	11:44
No quality trimming										
Reads mapped and paired	54,367,548	54,287,616	54,324,964	54,319,438	54,299,922	54,322,088	53,087,420	**54,446,104**	54,218,344	54,128,760
Reads unmapped	1,094,145	1,016,244	1,119,103	989,335	1,002,745	**978,665**	2,317,968	999,099	1,041,005	1,094,752
Properly paired reads (%)	96.8	96.7	96.7	96.7	96.7	96.7	94.1	**97.0**	96.4	88.6
Base mapped (cigar)	7,703,820,585	7,700,134,673	7,700,298,217	7,701,482,302	7,700,749,164	7,699,298,008	7,607,799,306	7,512,839,360	7,677,087,845	**7,720,352,493**
Error rate (‰)	3.3082	3.8388	3.8724	3.8771	3.8834	3.8564	4.3239	**3.8007**	3.9173	6.1984
With quality trimming										
Reads mapped and paired	54,553,566	54,526,276	54,546,192	54,541,948	54,539,502	54,549,462	53,335,674	**54,608,308**	54,482,002	54,403,982
Reads unmapped	965,447	984,845	967,917	970,869	973,217	**826,424**	2,136,081	890,884	999,918	914,003
Properly paired reads (%)	97.0	97.0	97.0	97.0	97.0	97.0	94.4	**97.1**	96.8	89.0
Base mapped (cigar)	7,653,879,312	7,649,380,218	7,646,989,362	7,647,893,624	7,648,184,196	7,646,574,606	7,556,468,882	7,461,588,482	7,625,484,706	**7,668,777,971**
Error rate (‰)	2.9547	3.2535	3.2678	3.2698	3.2792	3.2634	3.7183	**3.2117**	3.3109	5.5798

Note: AR = AdapterRemoval. In the trimming-only benchmark, bold and underline formats indicate the first and second trimmers (including tie) in terms of each metric, respectively. *Elapsed time (wall time) is benchmarked based on trimming and output gzip files with 8 threads, except that Ktrim cannot output gzip files (marked with time + GZ).

Genome-wide human cell-free DNA dataset (ERR4695159)

Generally, plasma cell-free DNA is short in length [20], and trimming is extremely important in medical diagnosis. Here, we chose a human genome-wide cell-free DNA dataset ERR4695159. It has 8.4 G bases with 2 x 150 bp read length with 33 bp adapter sequences AGATCGGAAGAGCACACGTCTGAACTCCAGTCA in read 1 and AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT in read 2. The benchmark workflow was the same as the RNA-Seq analysis, except that the clean reads were mapped to the human reference genome hg38 (GRCh38.p13) using Bowtie2 v2.3.5.1 [21].

Atria was also the fastest trimmer in the scenario (3 min 3 s) (**Table 2**). SeqPurge and Trim Galore finished the task in more than 4 minutes, while others spent more than 11 minutes (**Table 2**).

In adapter-trimming-only statistics, SeqPurge had the highest mapped and paired reads (54,446,104) and the highest properly paired reads (97.0%) (**Table 2**). Atria followed with 54,324,964 mapped and paired reads. Atria, AdapterRemoval, Atropos, and Fastp all had 96.7% properly paired reads (**Table 2**).

With quality trimming, the overall performance increased, and properly paired reads were closer; SeqPurge had 97.1% properly paired reads, with Atria, AdapterRemoval, Atropos, and Fastp close behind at 97.0% (**Table 2**). Only 89.0% of reads were properly paired with Trimmomatic (**Table 2**).

Discussion

Atria performs favorably with other cutting-edge adapter trimmers in accuracy, robustness, speed, and efficiency. Its performance is ascribed to the byte-based matching algorithm. The design concept of the algorithm is to minimize any unnecessary CPU operations by taking advantage of the data structure of dense arrays.

Matrix-based algorithms, such as the Needleman-Wunsch algorithm and the Smith-Waterman algorithm, allocate and update a matrix and perform base-to-base comparison [22, 23]. They report every mismatch and gap between two sequences while Atria skips this step since it is focussed on the start positions of successful matches. Despite that, the matching algorithm used in Atria is able to identify mismatch loci when needed.

The byte-based matching algorithm is lightweight and designed for short sequence scanning. Each DNA is encoded in four bits and stores continuously in RAM. A sub-sequence can be extracted as an unsigned integer from a given memory position. For example, a 64-bit unsigned integer (UInt) represents a 16-mer, and a 128-bit UInt represents a 32-mer. The comparison between two sub-sequences is completed within the accumulator register, a CPU unit for arithmetic or logical operation. It does not require addressing or updating a scoring matrix from RAM. When comparing a short sequence, such as an adapter, to a long sequence, such as the read, the 16-mer of the short sequence is compared to every position of the long sequence. Hence, the byte-based matching algorithm has *O(n)* expected time complexity and *O(1)* space complexity in adapter matching, where n is the length of the long sequence.

The algorithm also has its limitations. It only reports the number of matched bases and does not report the positions of mismatches, so it cannot be used for sequence alignment solely. Besides, the algorithm does not handle insert and deletion. However, the average indel rate of Illumina library is 10^{-6} to 10^{-5} [16], and the low indel rate is almost negligible in real data analysis. In addition, Atria does four pairs of matches in different locations to compensate for the limitation. If one match is failed because of indel, other matches will suggest the real adapter positions.

In the runtime benchmark, we compared how trimmers performed using extremely high CPU cores. In general, efficiency marginally decreased as CPU usage increased due to the trimmers' parallel implementation and the inevitable cost of multi-threading, such as task scheduling and context switching. In addition, IO could be the main bottleneck for most hard disk drives and some solid-state drives. Thus, if the system IO reaches a bottleneck, an efficiency plateau would be expected sooner.

Conclusions

We introduce not only Atria, a cutting-edge trimming software for sequence data, but also the ultra-fast and lightweight byte-based matching algorithm. The algorithm can be used in a broad range of short-sequence matching applications, such as primer search and seed scanning before alignment. Atria is implemented in Julia, a programming language designed specifically for high performance. The source code, executables, and benchmark scripts are available on Atria's Github page [24].

Availability and requirements

Project name: Atria

Project home page: https://github.com/cihga39871/Atria

Operating system(s): Linux, OSX

Programming language: Julia

Other requirements: Julia v1.4, Pigz v2.4 or higher, Pbzip v1.1.13 or higher

License: MIT

Research Resource Identification Initiative ID: SCR_021313

Data Availability

The datasets SRR330569, and ERR4695159 analyzed during the current study are available in the Sequence Read Archive from the National Center for Biotechnology Information [11, 25].

The Atria source codes, releases, documents, and benchmark scripts can be downloaded from Atria's Github page [24].

Abbreviations

CPU: Central processing unit; DNA: Deoxyribonucleic acid; GB: Gigabyte; MCC: Matthew's correlation coefficient; NGS: Next-generation sequencing; PPV: Positive predictive value; RAM: Random-access memory; RNA: Ribonucleic acid; SNP: Single nucleotide polymorphism; SSD: Solid-state drive; TB: Terabyte; UInt: Unsigned integer; UInt64: Unsigned 64-bit integer; WGS: Whole-genome sequencing.

Competing interests

The authors declare that they have no competing interests.

Funding

This study was partially funded by the Interdepartmental funding of Genomics Research and Development Initiatives (GRDI), Canada to XL.

Authors' contributions

JC developed Atria software, performed benchmark experiments under the supervision of XL. Both XL and LH serve as co-supervisors and participates in the design of the study. MH contributes to the optimization of the algorithm. AZ participates in benchmark validation. JC, LH, and XL drafted the manuscript. All authors read and approved the final version of the manuscript.

Acknowledgments

The technical assistance of Jingbai Nie is greatly acknowledged. The financial support of CFIA to JC is greatly appreciated. The advice on algorithm optimization, encouragement, and support of Drs. Christian Lacroix and Stevan Springer to JC are greatly appreciated.

References

1. Schluth-Bolard C, Diguet F, Chatron N, Rollat-Farnier PA, Bardel C, Afenjar A, et al. Whole genome paired-end sequencing elucidates functional and phenotypic consequences of balanced chromosomal rearrangement in patients

with developmental disorders. J Med Genet. 2019;56 8:526-35. doi:10.1136/jmedgenet-2018-105778.

2. Tan G, Opitz L, Schlapbach R and Rehrauer H. Long fragments achieve lower base quality in Illumina paired-end sequencing. Sci Rep. 2019;9 1:2856. doi:10.1038/s41598-019-39076-7.
3. Schubert M, Lindgreen S and Orlando L. AdapterRemoval v2: rapid adapter trimming, identification, and read merging. BMC Res Notes. 2016;9 1:88. doi:10.1186/s13104-016-1900-2.
4. Krueger F. Trim galore. A wrapper tool around Cutadapt and FastQC to consistently apply quality and adapter trimming to FastQ files. 2015;516:517.
5. Bolger AM, Lohse M and Usadel B. Trimmomatic: a flexible trimmer for Illumina sequence data. Bioinformatics. 2014;30 15:2114-20. doi:10.1093/bioinformatics/btu170.
6. BioJulia/BioSequences.jl: Biological sequences for the Julia language. https://github.com/BioJulia/BioSequences.jl. Accessed 1 Dec 2020.
7. Pigz - Parallel gzip. https://zlib.net/pigz/. Accessed 1 Dec 2020.
8. Parallel BZIP2 (PBZIP2). http://compression.ca/pbzip2/. Accessed 1 Feb 2021.
9. Huang W, Li L, Myers JR and Marth GT. ART: a next-generation sequencing read simulator. Bioinformatics. 2012;28 4:593-4. doi:10.1093/bioinformatics/btr708.
10. Jiang H, Lei R, Ding SW and Zhu S. Skewer: a fast and accurate adapter trimmer for next-generation sequencing paired-end reads. BMC Bioinformatics. 2014;15 1:182. doi:10.1186/1471-2105-15-182.
11. Barrett T, Clark K, Gevorgyan R, Gorelenkov V, Gribov E, Karsch-Mizrachi I, et al. BioProject and BioSample databases at NCBI: facilitating capture and organization of metadata. Nucleic Acids Res. 2012;40 D1:D57-D63.
12. Chen S, Zhou Y, Chen Y and Gu J. fastp: an ultra-fast all-in-one FASTQ preprocessor. Bioinformatics. 2018;34 17:i884-i90. doi:10.1093/bioinformatics/bty560.
13. Sun K. Ktrim: an extra-fast and accurate adapter- and quality-trimmer for sequencing data. Bioinformatics. 2020;36 11:3561-2. doi:10.1093/bioinformatics/btaa171.
14. Didion JP, Martin M and Collins FS. Atropos: specific, sensitive, and speedy trimming of sequencing reads. PeerJ. 2017;5:e3720. doi:10.7717/peerj.3720.
15. Sturm M, Schroeder C and Bauer P. SeqPurge: highly-sensitive adapter trimming for paired-end NGS data. BMC Bioinformatics. 2016;17 1:208. doi:10.1186/s12859-016-1069-7.
16. Schirmer M, D'Amore R, Ijaz UZ, Hall N and Quince C. Illumina error profiles: resolving fine-scale variation in metagenomic sequencing data. BMC Bioinformatics. 2016;17 1:125. doi:10.1186/s12859-016-0976-y.

17. Thurmond J, Goodman JL, Strelets VB, Attrill H, Gramates LS, Marygold SJ, et al. FlyBase 2.0: the next generation. Nucleic Acids Res. 2019;47 D1:D759-D65. doi:10.1093/nar/gky1003.
18. Kim D, Paggi JM, Park C, Bennett C and Salzberg SL. Graph-based genome alignment and genotyping with HISAT2 and HISAT-genotype. Nat Biotechnol. 2019;37 8:907-15. doi:10.1038/s41587-019-0201-4.
19. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, et al. The Sequence Alignment/Map format and SAMtools. Bioinformatics. 2009;25 16:2078-9. doi:10.1093/bioinformatics/btp352.
20. Sun K, Jiang P, Wong AIC, Cheng YKY, Cheng SH, Zhang H, et al. Size-tagged preferred ends in maternal plasma DNA shed light on the production mechanism and show utility in noninvasive prenatal testing. Proceedings of the National Academy of Sciences. 2018;115 22:E5106-E14.
21. Langmead B, Wilks C, Antonescu V and Charles R. Scaling read aligners to hundreds of threads on general-purpose processors. Bioinformatics. 2019;35 3:421-32. doi:10.1093/bioinformatics/bty648.
22. Needleman SB and Wunsch CD. A general method applicable to the search for similarities in the amino acid sequence of two proteins. J Mol Biol. 1970;48 3:443-53. doi:10.1016/0022-2836(70)90057-4.
23. Smith TF and Waterman MS. Identification of common molecular subsequences. J Mol Biol. 1981;147 1:195-7. doi:10.1016/0022-2836(81)90087-5.
24. cihga39871/Atria: An ultra-fast and accurate NGS adapter and quality trimmer. https://github.com/cihga39871/Atria. Accessed 31 Mar 2021.
25. Sequence Read Archive from the National Center for Biotechnology Information. https://www.ncbi.nlm.nih.gov/sra/. Accessed 15 Jan 2021.

Figure legends

Figure 1 Overview of Atria workflow

Figure 2 Adapter trimming algorithms

Figure 3 Benchmark of adapter-trimming speed for uncompressed and compressed files on different threading options

The 8.9 G bases simulated paired-end data with a 100 bp read length was trimmed in both uncompressed and compressed format using up to 32 threads. Speed is the ratio of the number of bases to elapsed time (wall time). SeqPurge does not support uncompressed outputs, so it is not shown in the uncompressed benchmark. In the trimming for compressed data, the speed of AdapterRemoval, Skewer, Fastp, Atropos, and Trimmomatic kept constant when the number of threads increased from 4 to 32, so we only benchmark those trimmers using 1, 2, and 4 threads. Ktrim does not support output compressed files, so it is not shown in the compressed benchmark.

Figure 4 Adapter trimming accuracy on adapter presence and absence, different base errors, and adapter lengths

A1, B1, and C1 are statistics for reads with adapter contamination, while A2, B2, C2 for reads without adapters. A1 and A2 show the accumulated rates of accurate trim, one bp over trim, one bp under trim, multiple bp over trim, and multiple bp under trim. In A1, the accuracy of Trimmomatic is 41.0%. In A2, the accuracy of SeqPurge is 78.8%, the accuracy of Trim Galore is 68.3%. B1 and B2 show the trimming accuracy on different error profiles. In B1, the accuracy of Trimmomatic drops from 41.9% to 40.1%. In B2, the accuracy of SeqPurge is 78.8%, and the accuracy of Trim Galore is 68.2 - 68.3%. C1 and C2 show the trimming accuracy on different adapter lengths. In C1, the accuracy of Trimmomatic is 0.0% at 16 bp adapter length, 50.7% to 51.6% at adapter lengths from 20 to 33 bp. In C2, the accuracy of SeqPurge ranges from 78.7% at 16 bp to 78.9% at 33 bp, and the accuracy of Trim Galore ranges in 68.2 - 68.3% from 16 to 33 bp.

Supplementary material

Table S1 Trimming speed on the 8.9 G bases 100 bp paired-end simulated data

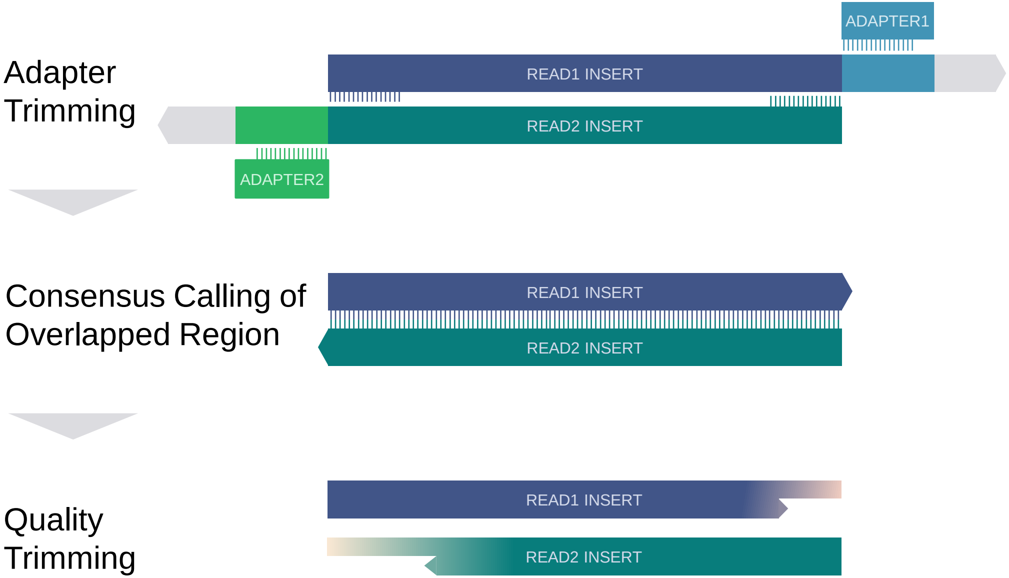
Atria (consensus) does both adapter trimming and paired-end consensus call (base correction of overlapped regions). In the trimming for uncompressed data, SeqPurge does not support uncompressed outputs, so it is not shown in the uncompressed benchmark. Fastp does not support 32 threads, so only 1-16 threads were tested. In the trimming for compressed data, the speed of AdapterRemoval, Skewer, Fastp, and Trimmomatic kept constant when the number of threads increased from 4 to 32, so we only benchmarked those trimmers using 1, 2, and 4 threads. Atropos was too slow to trim compressed data, and Ktrim did not support compressed outputs, so they are not shown in the compressed benchmark.

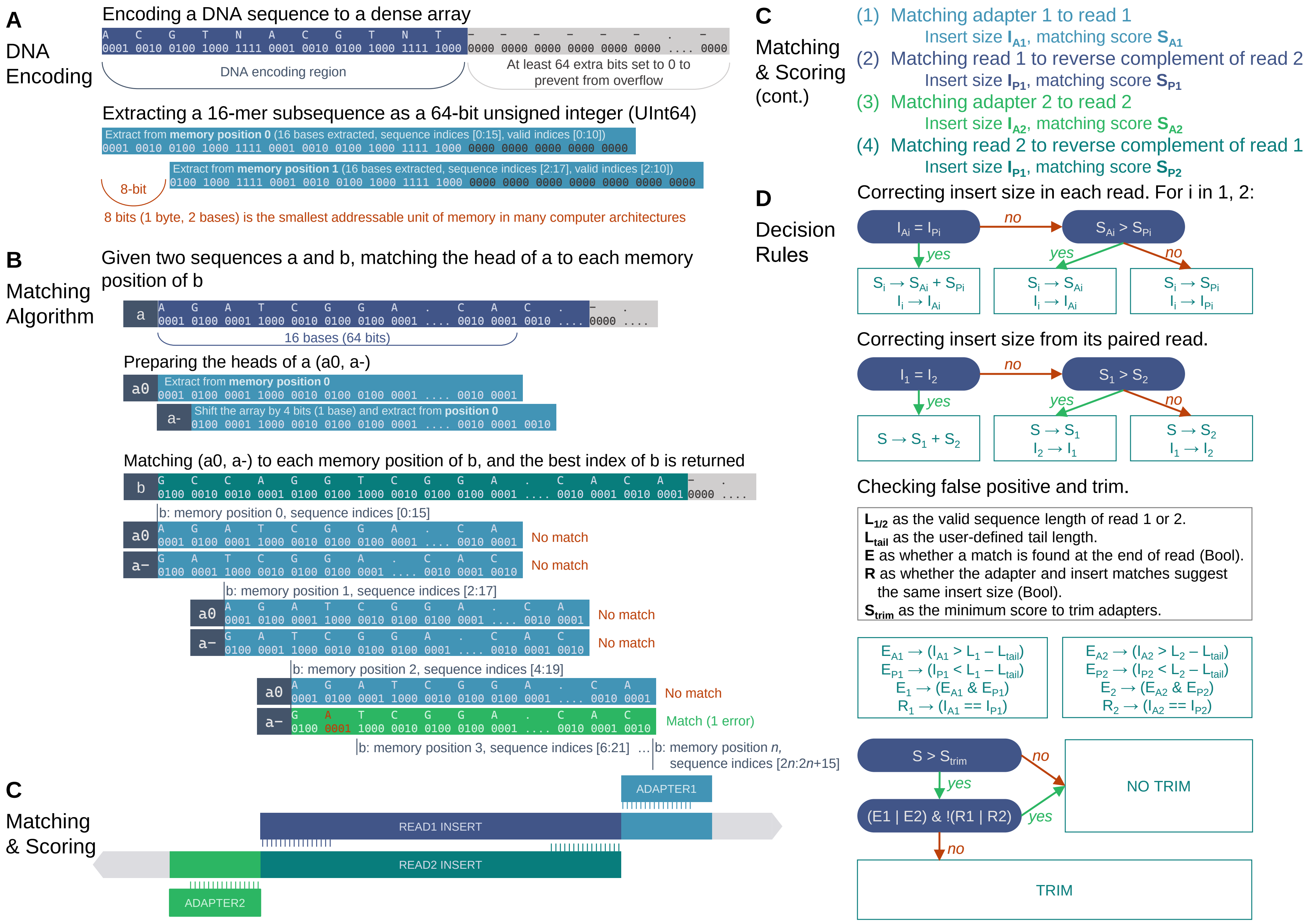
Table 2 Performance of trimmers on real data (larger than A4)

	Trimming and consensus		Trimming only							
Metric	**Atria**	**Skewer**	**Atria**	**AR**	**Atropos**	**Fastp**	**Ktrim***	**SeqPurge**	**TrimGalore**	**Trimmomatic**
Low-quality dataset (SRR330569, RNA, Hisat2 mapping)										
Elapsed time (min:sec)*	2:38	9:19	**2:32**	11:29	10:08	9:17	1:34 + GZ	3:53	3:39	9:38

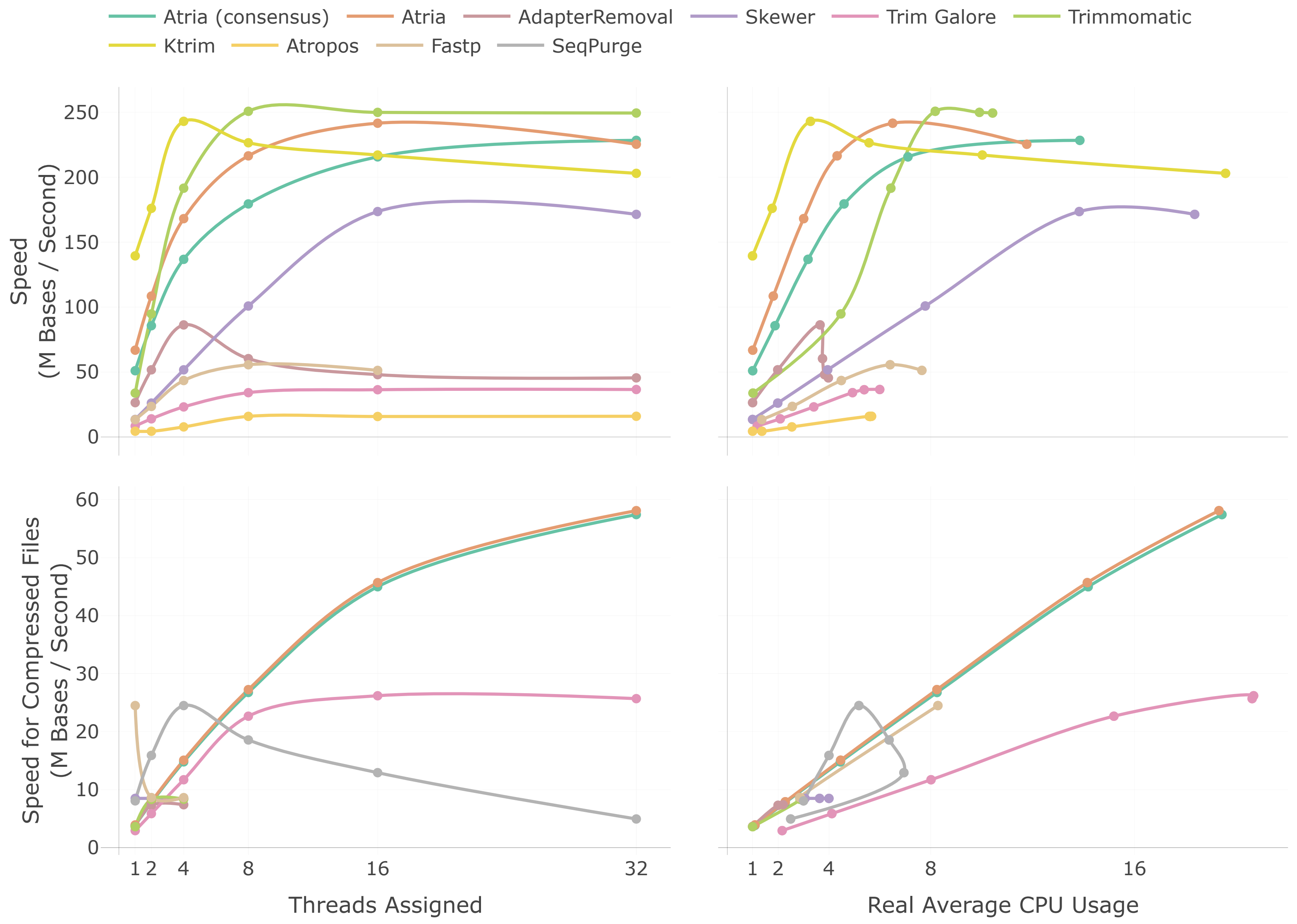
No quality trimming										
Reads mapped and paired	26,126,804	24,694,330	**25,781,268**	24,559,060	24,505,656	24,545,646	24,196,658	24,240,072	24,046,542	22,797,620
Reads unmapped	27,276,761	28,254,804	**27,379,299**	28,338,455	28,410,022	28,350,294	28,747,248	28,591,442	28,647,873	29,649,287
Properly paired reads (%)	48.3	45.6	**47.6**	45.3	45.2	45.3	44.5	44.7	44.2	38.3
Base mapped (cigar)	2,387,212,225	2,354,164,041	2,322,436,545	**2,346,791,204**	2,341,355,822	2,344,847,438	2,316,915,673	2,304,776,514	2,317,321,510	2,237,846,534
Error rate (‰)	7.3952	9.5897	**8.1833**	9.8902	9.8536	9.8683	9.7920	9.7904	9.6994	9.3106
With quality trimming										
Reads mapped and paired	25,942,092	25,787,464	**25,728,206**	25,721,788	25,714,956	25,725,480	25,473,670	25,364,392	25,654,498	24,744,754
Reads unmapped	27,245,720	27,364,827	**27,361,655**	27,369,773	27,373,854	27,360,527	27,556,820	27,736,292	27,400,932	28,064,739
Properly paired reads (%)	47.9	47.6	**47.5**	**47.5**	**47.5**	**47.5**	46.9	46.8	47.3	42.3
Base mapped (cigar)	2,317,238,536	2,316,981,456	2,302,740,463	**2,304,584,269**	2,304,325,743	2,304,437,244	2,292,034,762	2,263,465,110	2,297,815,439	2,246,076,618
Error rate (‰)	7.1114	7.7882	7.8902	7.9160	7.9141	7.9149	7.9059	7.8787	7.8921	**7.5649**
High-quality dataset (ERR4695159, cell-free DNA, Bowtie2 mapping)										
Elapsed time (min:sec)*	3:08	11:34	**3:03**	13:48	13:41	11:29	1:41 + GZ	4:05	4:34	11:44
No quality trimming										
Reads mapped and paired	54,367,548	54,287,616	54,324,964	54,319,438	54,299,922	54,322,088	53,087,420	**54,446,104**	54,218,344	54,128,760
Reads unmapped	1,094,145	1,016,244	1,119,103	989,335	1,002,745	**978,665**	2,317,968	999,099	1,041,005	1,094,752
Properly paired reads (%)	96.8	96.7	96.7	96.7	96.7	96.7	94.1	**97.0**	96.4	88.6

Base mapped (cigar)	7,703,820,585	7,700,134,673	7,700,298,217	7,701,482,302	7,700,749,164	7,699,298,008	7,607,799,306	7,512,839,360	7,677,087,845	**7,720,352,493**
Error rate (‰)	3.3082	3.8388	3.8724	3.8771	3.8834	3.8564	4.3239	**3.8007**	3.9173	6.1984
With quality trimming										
Reads mapped and paired	54,553,566	54,526,276	54,546,192	54,541,948	54,539,502	54,549,462	53,335,674	**54,608,308**	54,482,002	54,403,982
Reads unmapped	965,447	984,845	967,917	970,869	973,217	**826,424**	2,136,081	890,884	999,918	914,003
Properly paired reads (%)	97.0	97.0	97.0	97.0	97.0	97.0	94.4	**97.1**	96.8	89.0
Base mapped (cigar)	7,653,879,312	7,649,380,218	7,646,989,362	7,647,893,624	7,648,184,196	7,646,574,606	7,556,468,882	7,461,588,482	7,625,484,706	**7,668,777,971**
Error rate (‰)	2.9547	3.2535	3.2678	3.2698	3.2792	3.2634	3.7183	**3.2117**	3.3109	5.5798

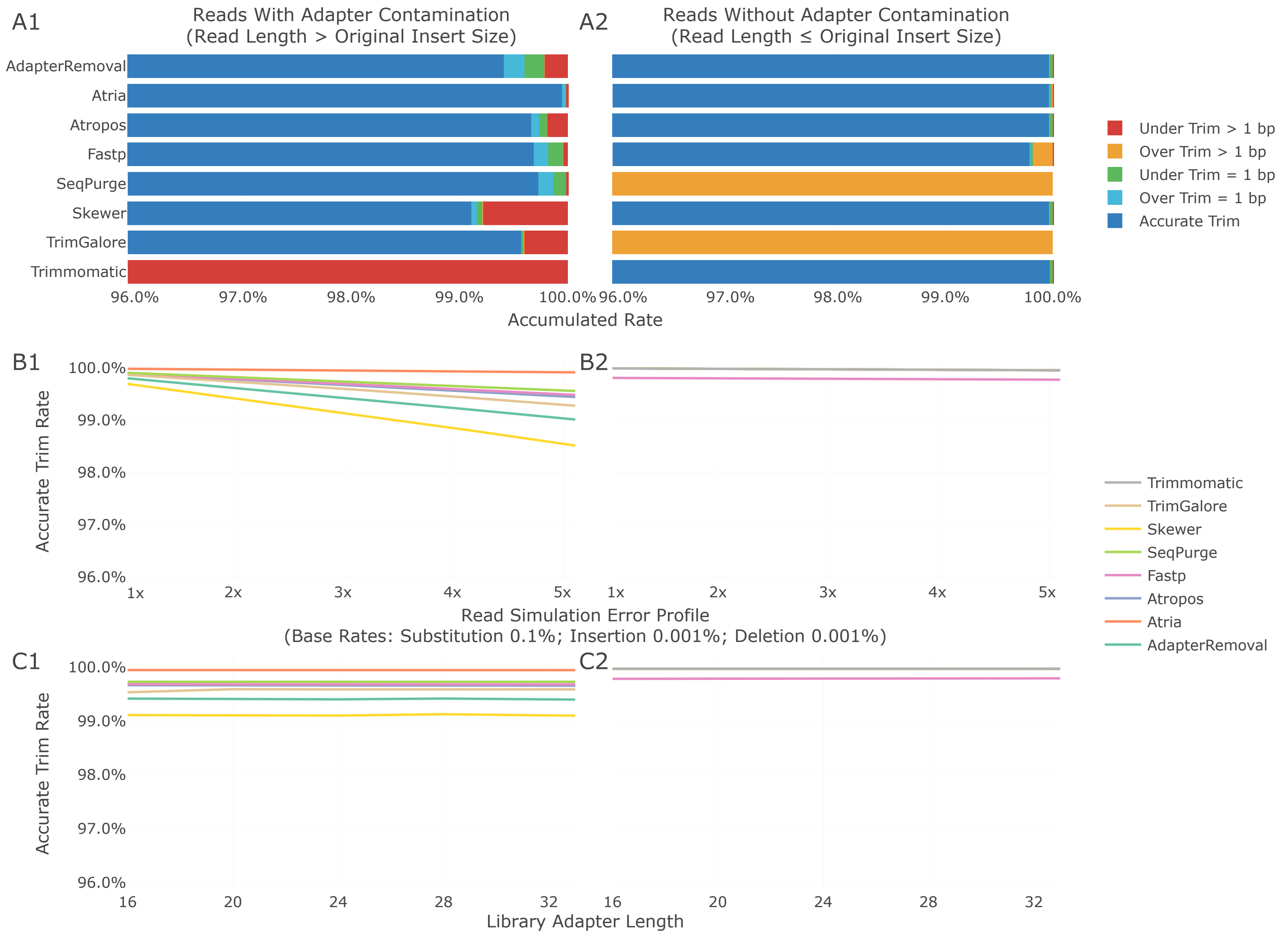

Adapter Trimming
ADAPTER1
READ1 INSERT
READ2 INSERT
ADAPTER2
Consensus Calling of Overlapped Region
READ1 INSERT
READ2 INSERT
Quality Trimming
READ1 INSERT
READ2 INSERT



A
DNA Encoding
Encoding a DNA sequence to a dense array
A C G T N A C G T N T
0001 0010 0100 1000 1111 0001 0010 0100 1000 1111 1000 0000 0000 0000 0000 0000 0000 0000
DNA encoding region
At least 64 extra bits set to 0 to prevent from overflow
Extracting a 16-mer subsequence as a 64-bit unsigned integer (UInt64)
Extract from memory position 0 (16 bases extracted, sequence indices [0:15], valid indices [0:10])
0001 0010 0100 1000 1111 0001 0010 0100 1000 1111 1000 0000 0000 0000 0000 0000
Extract from memory position 1 (16 bases extracted, sequence indices [2:17], valid indices [2:10])
0100 1000 1111 0001 0010 0100 1000 1111 1000 0000 0000 0000 0000 0000 0000 0000
8-bit
8 bits (1 byte, 2 bases) is the smallest addressable unit of memory in many computer architectures
B
Matching Algorithm
Given two sequences a and b, matching the head of a to each memory position of b
a
A G A T C G G A . C A C .
0001 0100 0001 1000 0010 0100 0100 0001 .... 0010 0001 0010 .... 0000 ....
16 bases (64 bits)
Preparing the heads of a (a0, a-)
a0
Extract from memory position 0
0001 0100 0001 1000 0010 0100 0100 0001 .... 0010 0001
a-
Shift the array by 4 bits (1 base) and extract from position 0
0100 0001 1000 0010 0100 0100 0001 .... 0010 0001 0010
Matching (a0, a-) to each memory position of b, and the best index of b is returned
b
G C C A G G T C G G A . C A C A
0100 0010 0010 0001 0100 0100 1000 0010 0100 0100 0001 .... 0010 0001 0010 0001 0000 ....
b: memory position 0, sequence indices [0:15]
No match
No match
b: memory position 1, sequence indices [2:17]
No match
No match
b: memory position 2, sequence indices [4:19]
No match
Match (1 error)
b: memory position 3, sequence indices [6:21]
… b: memory position n, sequence indices [2n:2n+15]
C
Matching & Scoring
ADAPTER1
READ1 INSERT
READ2 INSERT
ADAPTER2
C
Matching & Scoring (cont.)
(1) Matching adapter 1 to read 1
Insert size I_A1, matching score S_A1
(2) Matching read 1 to reverse complement of read 2
Insert size I_P1, matching score S_P1
(3) Matching adapter 2 to read 2
Insert size I_A2, matching score S_A2
(4) Matching read 2 to reverse complement of read 1
Insert size I_P1, matching score S_P2
D
Decision Rules
Correcting insert size in each read. For i in 1, 2:
I_Ai = I_Pi
S_Ai > S_Pi
no
yes
yes
no
S_i → S_Ai + S_Pi, I_i → I_Ai
S_i → S_Ai, I_i → I_Ai
S_i → S_Pi, I_i → I_Pi
Correcting insert size from its paired read.
I_1 = I_2
S_1 > S_2
S → S_1 + S_2
S → S_1, I_2 → I_1
S → S_2, I_1 → I_2
Checking false positive and trim.
L_1/2 as the valid sequence length of read 1 or 2.
L_tail as the user-defined tail length.
E as whether a match is found at the end of read (Bool).
R as whether the adapter and insert matches suggest the same insert size (Bool).
S_trim as the minimum score to trim adapters.
E_A1 → (I_A1 > L_1 − L_tail)
E_P1 → (I_P1 < L_1 − L_tail)
E_1 → (E_A1 & E_P1)
R_1 → (I_A1 == I_P1)
E_A2 → (I_A2 > L_2 − L_tail)
E_P2 → (I_P2 < L_2 − L_tail)
E_2 → (E_A2 & E_P2)
R_2 → (I_A2 == I_P2)
S > S_trim
(E1 | E2) & !(R1 | R2)
NO TRIM
TRIM



Atria (consensus)
Atria
AdapterRemoval
Skewer
Trim Galore
Trimmomatic
Ktrim
Atropos
Fastp
SeqPurge
Speed (M Bases / Second)
Speed for Compressed Files (M Bases / Second)
Threads Assigned
Real Average CPU Usage
0
50
100
150
200
250
0
10
20
30
40
50
60
1
2
4
8
16
32


A1
Reads With Adapter Contamination
(Read Length > Original Insert Size)
A2
Reads Without Adapter Contamination
(Read Length ≤ Original Insert Size)
AdapterRemoval
Atria
Atropos
Fastp
SeqPurge
Skewer
TrimGalore
Trimmomatic
96.0%
97.0%
98.0%
99.0%
100.0%
Accumulated Rate
Under Trim > 1 bp
Over Trim > 1 bp
Under Trim = 1 bp
Over Trim = 1 bp
Accurate Trim
B1
B2
Accurate Trim Rate
1x
2x
3x
4x
5x
Read Simulation Error Profile
(Base Rates: Substitution 0.1%; Insertion 0.001%; Deletion 0.001%)
Trimmomatic
TrimGalore
Skewer
SeqPurge
Fastp
Atropos
Atria
AdapterRemoval
C1
C2
16
20
24
28
32
Library Adapter Length