To the Editor and Reviewers,

Thank you for the constructive and timely reviews of our manuscript “Atropos: specific, sensitive, and speedy trimming of sequencing reads.” We apologize for the delay in resubmitting a revised manuscript. These delays were due to the following: 1) we chose to implement some new features in the Atropos software to make it more useful; and 2) we re-wrote our benchmarking workflow to be easily and completely reusable and reproducible. In addition, we feel that we have addressed most of your concerns.

New features

Since submitting the first version of our manuscript, we have added the following features:

- Ability to collect quality control (QC) metrics on reads before and after trimming. The scope of QC statistics is similar to what is offered by the FastQC tool (http://www.bioinformatics.babraham.ac.uk/projects/fastqc/). Statistics can either be collected as part of the trimming step, or separately using the new ‘qc’ subcommand.
- Major improvements to the summary reports. Specifically, the text-based report is now easier to read, and we also provide the ability to output summary statistics in JSON format. We also implemented an Atropos module for MultiQC (Ewels et al., Bioinformatics (2016) 32 (19): 3047-3048) that enables results from one to many data sets to be visualized. The module is currently in a separate repository (https://github.com/jdidion/atropos-multiqc) but is in the process of being integrated into MultiQC itself.

Benchmarking improvements

Previously, we offered a way to reproduce our benchmark workflow, but it was quite difficult to use and involved installing many software dependencies. In addition, the shell scripts were brittle and not easily modified to incorporate new tools or parameters.

To improve on this, we first created Docker containers for all of the software tools we use, and pushed these to a public Docker Hub repository (https://hub.docker.com/r/jdidion/). Now, rather than install many tools, it is only necessary to install a free Docker client (or free Singularity client, on some linux platforms that are not supported by Docker).

Second, we implemented our benchmark workflow in Nextflow (https://www.nextflow.io/). A key strength of Nextflow versus shell scripts or other workflow tools is that it can transparently execute containerized software using Docker or Singularity. Singularity support was critical for us, because our HPC environment runs RedHat linux, which is not supported by (the free version of) Docker. Nextflow workflows are also relatively easy to read, understand, and modify. Thus, we hope that others will adopt our workflow and add additional tools and datasets for more extensive benchmarking.

We also added AdapterRemoval2 as an additional tool in our benchmarks, as suggested by Reviewer #3.

Response to reviewer comments

Reviewer 1

Decide whether to use hyphens or not in terms like "writer(-)compression", "parallel(-)write" and "worker(-)compression" since these are written differently when looking at the tables and the text.

We chose to consistently use these terms as adjectives modifying the mode of operation (*e.g.* "writer-compression mode"), and have updated the text to reflect this.

In addition, I found some minor errors when reading the text:
l. 285 'of' used twice.
l. 322 at → a

Fixed

Lastly, I suggest to remove *-signs from formula 3-4, since these are not used in the other formulas or alternatively use multiplication sign (\cdot in LaTex).

'*' replaced with '\cdot'

Last sentence of description text for figure 2-3 is a bit confusing to read and understand. I suggest reformulating these and in addition to use >= signs instead of "greater than or equal to".

We changed these sentences to define the x-axis values using set notation, which we believe is clearer than the textual description.

MAPQ in Fig. 3 seems to be more a qualitative score than in fig 2. Does it make sense to make a curve plot for the qualitative score, since there are no values in between?

This is a good point. We changed Figure 3 (now Figure 5) to a bar plot.

Furthermore, what tool/method is used for mapping WGBS seq. reads to the genome?

As stated in the "Benchmarks" section, we use bwa-meth (Pedersen et al., 2014) for WGBS read mapping.

The caption for table 4 only contains one line and needs to be elaborated. I suggest: Explain if the Overtrimmed in bases is both bases for "over trimmed" and "wrongly trimmed", which it seems like when looking at the numbers, but is not obvious. Describe the percent in parenthesis (total number of reads I guess). Describe the different error rates and notable results highlighted in bold.

We have added a more detailed caption, incorporating the reviewer's suggestions. Over-trimmed and wrongly trimmed are mutually exclusive categories.

In the caption of table 2, the worker-compression is missing when describing the combinations of Atropos.

Fixed

Use capital letters for the different tools in the legend for figure 3.

Fixed

Table 3: Not crucial, but I think it would make more sense to shift the order of the two titles; "Execution time" and "4 Threads 8 Threads 16 Threads" so that "Execution time" comes after "4 Threads..", since "4 Threads.." is the "global" title and "Execution time" is the "local" title of the table.

This is a good suggestion, and we made this change.

One concern I have about the data in connection with the validity in the manuscript is the use of the older GRCh37 in stead of the newest GRCh38 reference genome. Since the manuscript tries to measure mapping differences between trimmed and untrimmed reads on real data, I think there is some motivation of always using the newest and most reliable genome reference consortium available. Therefore it could be interesting to see the same analysis done for GRCh38 data to confirm the results obtained in this manuscript.

This is an excellent point. We used GRCh37 out of convenience, since we already had pre-generated indexes available. Now we provide Docker containers that contain pre-generated GRCh37 and GRCh38 indexes for the BWA (DNA-Seq), bwa-meth (WGBS), and STAR (RNA-Seq) aligners. Thus, a user is free to modify the workflow script to use a different container and test the effect of using one genome version versus the other. We continue to use the simulated data from our initial submission (which was simulated from GRCh37), but we now use GRCh38 for mapping the real data.

We note that the Docker containers with STAR indexes have not been uploaded to Docker Hub due to space restrictions (they are ~28 GB, which is larger than the maximum size allowed by either Docker Hub or Quay.io), so users will need to build those themselves. In our GitHub repository, we provide shell scripts to automate the downloading of the reference sequences and the building of the containers.

Reviewer 2 (Stephen Piccolo)

In the Abstract, it says, "makes it an appropriate choice for the pre-processing of most current- generation sequencing data sets." It would be helpful to state more specifically what types of sequencing data for which this method is suitable.

We revised this sentence, as well as the first sentence of the introduction, to only mention short-read technologies, as we have not thoroughly benchmarked on long-read (*e.g.* PacBio, Nanopore) data. We added a sentence in the discussion to mention that a near-term future goal is to add support for long-read trimming.

It would be slightly more clear if you changed "(as it is currently only able to use a single processor)" on line 59 to "(as Cutadapt is currently only able to use a single processor)". When reading it the first time, I had to think twice about whether you were referring to Atropos or Cutadapt there.

Fixed

On line 81, change "as a intuitive parameter" to "as an intuitive parameter".

Fixed

"Let s be the adapter, t the read and m = |s|, n = |t|." It would be more intuitive to use a variable called "a" to represent the adapter and "r" the reader. This may be due to my limited math background, but I am not sure I understand how s is represented. Is it a numeric value? How is that number obtained from the adapter?

Apologies for the confusion. We have renamed s -> a and t -> r. We have clarified in the text that a and r are the nucleotide sequences of the adapter and sequencing read, and |a| and |r| are the cardinalities (lengths) of the sequences.

On line 244, it says, "the fragment is under-trimmed, or the fragment is over-trimmed." I assume over-trimmed is distinct from wrongly trimmed in that the fragment does contain an adapter but the tool trims more than the adapter sequence. It would be helpful to clarify this, just to make sure readers understand the distinction.

We have clarified that “wrongly trimmed” means that there is no adapter sequence in the read but trimming was performed, whereas under- and over-trimmed mean that there was adapter sequence in the read but the algorithm removed too few/too many bases.

On line 316, skewer should be capitalized.

Fixed

The authors use bisulfite sequencing and mRNA sequencing data. But there's no real-world DNA-Sequencing data. This type of data would be helpful to include because this type of data is so common. You could potentially use the Genome in a Bottle data to compare the trimming methods and see which leads to more accurate identification of known DNA variants.

We simulated DNA-Seq data, and the characteristics of this data are similar to read data, so we chose not to add another DNA-Seq benchmark dataset.

I love the adapter detection feature. I am not quite sure I understood how the user can avoid false positives when using this feature. It would be helpful to add some type of explanation about this.

We added a sentence explaining that adapter sequences have been designed not to match any known sequence in nature, thus a sequence (or pair of sequences) that occurs at high frequency and matches a known adapter sequence is likely to be the true sequence(s) used as adapters in the dataset. When the adapter does not match a known sequence, we advise the user to take caution as the sequence might simply be *e.g.* derived from a frequently repeated element in the genome.

The parallelization feature is very nice. It was great that the authors showed how the performance gains improve with an increasing number of threads. This is just personal preference, but it would be more intuitive if Table 3 were represented as a figure rather than a table.

We converted tables 3 and 5 into figures showing time versus number of threads used. We moved the tables to the supplementary material.

It is nice that you can install this tool via pip. However, there are several dependencies to install. Importantly, this tool only runs on Python 3.3+. For people (like me) who are still using Python 2, it is a barrier to install 3.3+ and have two versions. It would be convenient if this software and its dependencies were packaged in a Docker container so that it would be easier to install but also so that I would not be required to install Python 3.3+. It should be not much work to build the Docker image and store it on DockerHub.com. Below I have pasted the contents of the Dockerfile that I used to test

your software.

Thank you for providing a starting point for the Docker container. As you can see, we applied this approach to our entire benchmark workflow.

It's fantastic that the authors provide their analysis scripts and tests in their GitHub repository. I did have trouble accessing the simulated data in the GitHub repository. I cloned the repository. But it only had a short description of the FASTQ files. It seems there is a special way to download these larger files, but I wasn't sure how to do that. Providing a README would be helpful.

Apologies - these files were originally uploaded to GitHub using git lfs ("large file storage"); cloning these files requires installing the git lfs plugin on the client. We have moved those data to Docker containers (https://hub.docker.com/r/jdidion/).

Reviewer 3 (Anonymous)

The authors selected existing adapter trimming software for comparison based on the benchmarks presented in Sturm et al., 2016. However, I note that Sturm et al., 2016 made use of a version of AdapterRemoval (1.5.4) that was already greatly outdated when that paper was published. I would be curious as to how Atropos performs in comparison to a more recent version of AdapterRemoval (2.x), which greatly outperform the version tested in Sturm et al., 2016 and implements multi-threaded operation: http://bmcresnotes.biomedcentral.com/articles/10.1186/s13104-016-1900-2

We appreciate the reviewer's suggestion. We have added AdapterRemoval2 (v2.2.0) to our benchmarks. We also provide a Docker container definition file (https://github.com/jdidion/atropos/blob/master/paper/containers/tools/adapter-removal/Dockerfile), Docker container (https://hub.docker.com/r/jdidion/adapterremoval/), and CWL tool definition (https://github.com/jdidion/atropos/blob/master/paper/containers/tools/adapter-removal/adapter-removal.cwl) for AdapterRemoval2, along with the other tools we benchmark.

Secondly, the authors benchmark the mapping runtime following trimming, but do not specify what versions of the mapping software, and what parameters (if any), they used to carry out this mapping. Nor could I find any mention of the exact number of times that programs were executed to estimate the minimum and maximum runtimes.

We now provide the exact software we used for our benchmarks in Docker containers. The Nextflow script shows the exact parameters we used to run these tools.

I would suggest that the authors specify 'Python 3' on lines 72, rather than just 'Python'. This is a very minor issue, and is included simply because my first attempt at installing Atropos failed due to using the Python 2 version of pip.

We have updated all references to Python to specify that version 3.3+ is required. Additionally, our tool is available in a Docker container and can be run using either a Docker or Singularity client, without needing to have Python 3 installed on the host system.