9-25-2015

The khmer software package: enabling efficient nucleotide sequence analysis

Michael R. Crusoe
Michigan State University

Hussein F. Alameldin
Michigan State University

Sherine Awad
University of California, Davis

Elmar Boucher
Oregon Health and Science University

Adam Caldwell
San Jose State University

[list page for additional authors]

The khmer software package: enabling efficient nucleotide sequence analysis

Abstract
The khmer package is a freely available software library for working efficiently with fixed length DNA words, or k-mers. khmer provides implementations of a probabilistic k-mer counting data structure, a compressible De Bruijn graph representation, De Bruijn graph partitioning, and digital normalization. khmer is implemented in C++ and Python, and is freely available under the BSD license at https://github.com/dib-lab/khmer/.

Disciplines
Agriculture | Bioresource and Agricultural Engineering | Computational Biology | Numerical Analysis and Scientific Computing

Comments
This article is from *F1000Research* 2015, **4**:900 (doi:10.12688/f1000research.6924.1). Posted with permission.

Authors
Michael R. Crusoe, Hussein F. Alameldin, Sherine Awad, Elmar Boucher, Adam Caldwell, Reed Cartwright, Amanda Charbonneau, Bede Constantinides, Greg Edvenson, Scott Fay, Jacob Fenton, Thomas Fenzl, Jordan Fish, Leonor Garcia-Gutierrez, Phillip Garland, Jonathan Gluck, Iván González, Sarah Guermond, Jiarong Guo, Aditi Gupta, Joshua R. Herr, Adina C. Howe, Alex Hyer, Andreas Härpfer, Luiz Irber, Rhys Kidd, David Lin, Justin Lippi, Tamer Mansour, Pamela McA'Nulty, Erin McDonald, Jessica Mizzi, Kevin D. Murray, Joshua R. Nahum, Kaben Nanlohy, Alexander Johan Nederbragt, Humberto Ortiz-Zuazaga, Jeramia Ory, Jason Pell, Charles Pepe-Ranney, Zachary N. Russ, Erich Schwarz, Camille Scott, Josiah Seaman, Scott Sievert, Jared Simpson, Connor T. Skennerton, James Spencer, Ramakrishnan Srinivasan, Daniel S. Standage, James A. Stapleton, Susan R. Steinman, Joe Stein, Benjamin Taylor, Will Tremble, Heather L. Wiencko, Michael Wright, Brian Wyss, Qingpeng Zhang, en zyme, and C. Titus Brown

SOFTWARE TOOL ARTICLE

The khmer software package: enabling efficient nucleotide sequence analysis [version 1; referees: 2 approved, 1 approved with reservations]

Michael R. Crusoe[1], Hussien F. Alameldin[2], Sherine Awad[3], Elmar Boucher[4], Adam Caldwell[5], Reed Cartwright[6], Amanda Charbonneau[7], Bede Constantinides[8], Greg Edvenson[9], Scott Fay[10], Jacob Fenton[11], Thomas Fenzl[12], Jordan Fish[11], Leonor Garcia-Gutierrez[13], Phillip Garland[14], Jonathan Gluck[15], Iván González[16], Sarah Guermond[17], Jiarong Guo[18], Aditi Gupta[1], Joshua R. Herr[1], Adina Howe[19], Alex Hyer[20], Andreas Härpfer[21], Luiz Irber[11], Rhys Kidd[22], David Lin[23], Justin Lippi[24], Tamer Mansour[3,25], Pamela McA'Nulty[26], Eric McDonald[11], Jessica Mizzi[27], Kevin D. Murray[28], Joshua R. Nahum[29], Kaben Nanlohy[30], Alexander Johan Nederbragt[31], Humberto Ortiz-Zuazaga[32], Jeramia Ory[33], Jason Pell[11], Charles Pepe-Ranney[34], Zachary N. Russ[35], Erich Schwarz[36], Camille Scott[11], Josiah Seaman[37], Scott Sievert[38], Jared Simpson[39,40], Connor T. Skennerton[41], James Spencer[42], Ramakrishnan Srinivasan[43], Daniel Standage[44,45], James A. Stapleton[46], Susan R. Steinman[47], Joe Stein[48], Benjamin Taylor[11], Will Trimble[49], Heather L. Wiencko[50], Michael Wright[11], Brian Wyss[11], Qingpeng Zhang[11], en zyme[51], C. Titus Brown[1,3,11]

[1]Microbiology and Molecular Genetics, Michigan State University, East Lansing, MI, USA
[2]Department of Plant, Soil and Microbial Sciences, Michigan State University, East Lansing, MI, USA
[3]Population Health and Reproduction, University of California, Davis, Davis, CA, USA
[4]Department of Biomedical Engineering, Oregon Health and Science University, Portland, OR, USA
[5]Biology Department, San Jose State University, San Jose, CA, USA
[6]School of Life Sciences and The Biodesign Institute, Arizona State University, Tempe, AZ, USA
[7]Genetics, Michigan State University, East Lansing, MI, USA
[8]Computational and Evolutionary Biology, Faculty of Life Sciences, University of Manchester, Manchester, UK
[9]Micron Technology, Seattle, WA, USA
[10]Invitae, San Francisco, CA, USA
[11]Computer Science and Engineering, Michigan State University, East Lansing, MI, USA
[12]Independent Researcher, Munich, Germany
[13]Mathematics Institute, University of Warwick, Warwick, UK
[14]Eastlake Data, Seattle, WA, USA
[15]Graduate Program, University of Maryland, College Park, MD, USA
[16]Athinoula A. Martinos Center for Biomedical Imaging, Department of Radiology, Massachusetts General Hospital, Charlestown, MA, USA
[17]Independent Researcher, Seattle, WA, USA
[18]Center for Microbial Ecology, Michigan State University, East Lansing, MI, USA

[19]Department of Agricultural and Biosystems Engineering, Iowa State University, Ames, IA, USA
[20]Department of Biology, University of Utah, Salt Lake City, UT, USA
[21]ConSol* Software GmbH, Munchen, Germany
[22]Independent Researcher, Sydney, Australia
[23]Verdematics, Fremont, CA, USA
[24]Independent Researcher, San Francisco, CA, USA
[25]Clinical Pathology, Mansoura University, Mansoura, Egypt
[26]Addgene, Cambridge, MA, USA
[27]Biochemistry and Molecular Biology, Michigan State University, East Lansing, MI, USA
[28]ARC Centre of Excellence in Plant Energy Biology, The Australian National University, Canberra, ACT, Australia
[29]BEACON Center, Michigan State University, East Lansing, MI, USA
[30]Independent Researcher, New Orleans, LA, USA
[31]Centre for Ecological and Evolutionary Synthesis, Dept. of Biosciences, University of Oslo, Oslo, Norway
[32]Department of Computer Science, Rio Piedras Campus, University of Puerto Rico, San Juan, Puerto Rico
[33]Biochemistry, St. Louis College of Pharmacy, St. Louis, MO, USA
[34]Crop and Soil Sciences, Cornell University, Ithaca, NY, USA
[35]Department of Bioengineering, UC Berkeley, Berkeley, CA, USA
[36]Department of Molecular Biology and Genetics, Cornell University, Ithaca, NY, USA
[37]Data Visualization, Newline Technical Innovations, Windsor, CO, USA
[38]Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN, USA
[39]Ontario Institute for Cancer Research, Toronto, ON, Canada
[40]Computer Science, University of Toronto, Toronto, ON, Canada
[41]Division of Geological and Planetary Sciences, California Institute of Technology, Pasadena, CA, USA
[42]Dept of Physics and Dept of Materials, Imperial College London, London, UK
[43]Genetics and Genomic Sciences, Icahn School of Medicine at Mount Sinai, New York, NY, USA
[44]Department of Biology, Indiana University, Bloomington, IN, USA
[45]Bioinformatics and Computational Biology Graduate Program, Iowa State University, Ames, IA, USA
[46]Chemical Engineering & Materials Science, Michigan State University, East Lansing, MIS, USA
[47]The New York Eye and Ear Infirmary of Mount Sinai, New York, NY, USA
[48]Independent Researcher, Providence, RI, USA
[49]Mathematics and Computer Science Division, Argonne National Laboratory, Lemont, IL, USA
[50]Department of Genetics, Smurfit Institute, Trinity College Dublin, Dublin, Ireland
[51]Independent Researcher, Boston, MA, USA

v1 **First published:** 25 Sep 2015, **4**:900 (doi: 10.12688/f1000research.6924.1)
Latest published: 25 Sep 2015, **4**:900 (doi: 10.12688/f1000research.6924.1)

Abstract

The khmer package is a freely available software library for working efficiently with fixed length DNA words, or k-mers. khmer provides implementations of a probabilistic k-mer counting data structure, a compressible De Bruijn graph representation, De Bruijn graph partitioning, and digital normalization. khmer is implemented in C++ and Python, and is freely available under the BSD license at https://github.com/dib-lab/khmer/.

Open Peer Review

Referee Status: ? ✓ ✓

	Invited Referees 1	2	3
version 1 published 25 Sep 2015	? report	✓ report	✓ report

1 **Ewan Birney**, European Bioinformatics Institute UK

2 **Daniel Katz**, University of Chicago USA

3 **Rob Patro**, Stony Brook University USA

Discuss this article

Comments (3)

Corresponding author: C. Titus Brown (titus@idyll.org)

How to cite this article: Crusoe MR, Alameldin HF, Awad S *et al.* **The khmer software package: enabling efficient nucleotide sequence analysis [version 1; referees: 2 approved, 1 approved with reservations]** *F1000Research* 2015, **4**:900 (doi: 10.12688/f1000research.6924.1)

Grant information: khmer development has largely been supported by AFRI Competitive Grant no. 2010-65205-20361 from the USDA NIFA, and is now funded by the National Human Genome Research Institute of the National Institutes of Health under Award Number R01HG007513, as well as by the the Gordon and Betty Moore Foundation under Award number GBMF4551, all to CTB.
The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: No competing interests were disclosed.

First published: 25 Sep 2015, **4**:900 (doi: 10.12688/f1000research.6924.1)

Introduction

DNA words of a fixed-length k, or "k-mers", are a common abstraction in DNA sequence analysis that enable alignment-free sequence analysis and comparison. With the advent of second-generation sequencing and the widespread adoption of De Bruijn graph-based assemblers, k-mers have become even more widely used in recent years. However, the dramatically increased rate of sequence data generation from Illumina sequencers continues to challenge the basic data structures and algorithms for k-mer storage and manipulation. This has led to the development of a wide range of data structures and algorithms that explore possible improvements to k-mer-based approaches.

Here we present version 2.0 of the khmer software package, a high-performance library implementing memory- and time-efficient algorithms for the manipulation and analysis of short-read data sets. khmer contains reference implementations of several approaches, including a probabilistic k-mer counter based on the CountMin Sketch[1], a compressible De Bruijn graph representation built on top of Bloom filters[2], a streaming lossy compression approach for short-read data sets termed "digital normalization"[3], and a generalized semi-streaming approach for k-mer spectral analysis of variable-coverage shotgun sequencing data sets[4].

khmer is both research software and a software product for users: it has been used in the development of novel data structures and algorithms, and it is also immediately useful for certain kinds of data analysis (discussed below). We continue to develop research extensions while maintaining existing functionality.

The khmer software consists of a core library implemented in C++, a CPython library wrapper implemented in C, and a set of Python "driver" scripts that make use of the library to perform various sequence analysis tasks. The software is currently developed on GitHub under https://github.com/dib-lab/khmer, and it is released under the BSD License. There is greater than 87% statement coverage under automated tests, measured on both C++ and Python code but primarily executed at the Python level.

Methods

Implementation

The core data k-mer counting data structures and graph traversal code are implemented in C++, and then wrapped for Python in hand-written C code, for a total of 10.5k lines of C/C++ code. The command-line API and all of the tests are written in 13.7k lines of Python code. C++ FASTQ and FASTA parsers came from the SeqAn library[5].

Documentation is written in reStructuredText, compiled with Sphinx, and hosted on ReadTheDocs.org.

We develop khmer on github.com as a community open source project focused on sustainable software development[6], and encourage contributions of any kind. As an outcome of several community events, we have comprehensive documentation on contributing to khmer at https://khmer.readthedocs.org/en/latest/dev/[7]. Most development decisions are discussed and documented publicly as they happen.

Operation

khmer is primarily developed on Linux for Python 2.7 and 64-bit processors, and several core developers use Mac OS X. The project is tested regularly using the Jenkins continuous integration system running on Ubuntu 14.04 LTS and Mac OS X 10.10; the current development branch is also tested under Python 3.3, 3.4, and 3.5. Releases are tested against many Linux distributions, including RedHat Enterprise Linux, Debian, Fedora, and Ubuntu. khmer should work on most UNIX derivatives with little modification. Windows is explicitly not supported.

Memory requirements for using khmer vary with the complexity of data and are user configurable. Several core data structures can trade memory for false positives, and we have explored these details in several papers, most notably Pell *et al.* 2012[2] and Zhang *et al.* 2014[1]. For example, most single organism mRNAseq data sets can be processed in under 16 GB of RAM[3,8], while memory requirements for metagenome data sets may vary from dozens of gigabytes to terabytes of RAM.

The user interface for khmer is via the command line. The command line interface consists of approximately 25 Python scripts; they are documented at http://khmer.readthedocs.org/ under User Documentation. Changes to the interface are managed with semantic versioning[9] which guarantees command line compatibility between releases with the same major version.

khmer also has an unstable developer interface via its Python and C++ libraries, on which the command line scripts are built.

Use cases

khmer has several complementary feature sets, all centered on short-read manipulation and filtering. The most common use of khmer is for preprocessing short read Illumina data sets prior to *de novo* sequence assembly, with the goals of decreasing compute requirements for the assembler as well as potentially improving the assembly results.

Prefiltering sequence data for *de novo* assembly with digital normalization

We provide an implementation of a novel streaming "lossy compression" algorithm in khmer that performs abundance normalization of shotgun sequence data. This "digital normalization" algorithm eliminates redundant short reads while retaining sufficient information to generate a contig assembly[3]. The algorithm takes advantage of the online k-mer counting functionality in khmer to estimate per-read coverage as reads are examined; reads can then be accepted as novel or rejected as redundant. This is a form of error reduction, because the net effect is to decrease not only the total number of reads considered for assembly, but also the total number of errors considered by the assembler. Digital normalization results in a decrease of the amount of memory needed for *de novo* assembly of high-coverage data sets with little to no change in the assembled contigs.

Digital normalization is implemented in the script `normalize-by-median.py`. This script takes as input a list of FASTA or FASTQ files, which it then filters by abundance as described above;

see 3 for details. The output of the digital normalization script is a downsampled set of reads, with no modifications to the individual reads. The three key parameters for the script are the k-mer size, the desired coverage level, and the amount of memory to be used for k-mer counting. The interaction between these three parameters and the filtering process is complex and depends on the data set being processed, but higher coverage levels and longer k-mer sizes result in less data being removed. Lower memory allocation increases the rate at which reads are removed due to erroneous estimates of their abundance, but this process is very robust in practice[1].

The output of `normalize-by-median.py` can be assembled using a *de novo* assembler such as Velvet[10], IDBA[11], Trinity[12] or SPAdes[13].

K-mer counting and read trimming

Using a memory-efficient CountMin Sketch data structure, khmer provides an interface for online counting of k-mers in streams of reads. The basic functionality includes calculating the k-mer frequency spectrum in sequence data sets and trimming reads at low-abundance k-mers. This functionality is explored and benchmarked in 1.

Basic read trimming is performed by the script `filter-abund.py`, which takes as arguments a k-mer countgraph (created by khmer's `load-into-counting.py` script) and one or more sequence data files. The script examines each sequence to find k-mers below the given abundance cutoff, and truncates the sequence at the first such k-mer. This truncates reads at the location of substitution errors produced by the sequencing process. When processing sequences from variable coverage data sets, `filter-abund.py` can also be configured to ignore reads that have low estimated abundance.

K-mer abundance distributions can be calculated using the script `abundance-dist.py`, which takes as arguments a k-mer countgraph, a sequence data file, and an output filename. This script determines the abundance of each distinct k-mer in the data file according to the k-mer countgraph, and summarizes the abundances in a histogram output.

We recently extended digital normalization to provide a generalized semi-streaming approach for k-mer spectral analysis[4]. Here, we examine read coverage on a per-locus basis in the De Bruijn graph and, once a particular locus has sufficient coverage, call errors or trim bases for all following reads belonging to that graph locus. The approach is "semi-streaming"[4] because some reads must be examined twice. This semi-streaming approach enables few-pass analysis of high coverage data sets. More, the approach also makes it possible to apply k-mer spectral analysis to data sets with uneven coverage such as metagenomes, transcriptomes, and whole-genome amplified samples.

Because our core data structure sizes are preallocated based on estimates of the unique k-mer content of the data, we also provide fast and low-memory k-mer cardinality estimation via the script `unique-kmers.py`. This script uses the HyperLogLog algorithm to provide a probabilistic estimate of the number of unique k-mers in a data set with a guaranteed upper bound[14]. A manuscript on this implementation is in progress (Irber and Brown, unpublished).

Partitioning reads into disconnected assembly graphs

We have also built a De Bruijn graph representation on top of a Bloom filter, and implemented this in khmer. The primary use for this so far has been to enable memory efficient *graph partitioning*, in which reads contributing to disconnected subgraphs are placed into different files. This can lead to an approximately 20-fold decrease in the amount of memory needed for metagenome assembly[2], and may also separate reads into species-specific bins[15].

Reformatting collections of short reads

In support of the streaming nature of this project, our preferred paired-read format is with pairs interleaved in a single file. As an extension of this, we automatically support a "broken-paired" read format where orphaned reads and pairs coexist in a single file. This enables single input/output streaming connections between tools, while leaving our tools compatible with fully paired read files as well as files containing only orphaned reads.

For converting to and from this format, we supply the scripts `extract-paired-reads.py`, `interleave-reads.py`, and `split-paired-reads.py` to respectively extract fully paired reads from sequence files, interleave two files containing read pairs, and split an interleaved file into two files containing read pairs.

In addition, we supply several utility scripts that we use in our own work. These include `sample-reads-randomly.py` for performing reservoir sampling of reads and `readstats.py` for summarizing sequence files.

Summary

The khmer project is an increasingly mature open source scientific software project that provides several efficient data structures and algorithms for analyzing short-read nucleotide sequencing data. khmer emphasizes online analysis, low memory data structures and streaming algorithms. khmer continues to be useful for both advancing bioinformatics research and analyzing biological data.

Software availability

Software available from

https://khmer.readthedocs.org/en/v2.0/

Link to source code

https://github.com/dib-lab/khmer/releases/tag/v2.0

Link to archived source code as at time of publication

http://dx.doi.org/10.5281/zenodo.31258[16]

Software license

Author contributions

CTB is the primary investigator for the khmer software package. MRC is the lead software developer from July 2013 onwards. Many significant components of khmer have their own paper describing them (see "Use Cases", above). The remaining authors each have one or more Git commits in their name.

Competing interests

No competing interests were disclosed.

Grant information

khmer development has largely been supported by AFRI Competitive Grant no. 2010-65205-20361 from the USDA NIFA, and is now funded by the National Human Genome Research Institute of the National Institutes of Health under Award Number R01HG007513, as well as by the the Gordon and Betty Moore Foundation under Award number GBMF4551, all to CTB.

I confirm that the funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

References

1. Zhang Q, Pell J, Canino-Koning R, *et al.*: **These are not the k-mers you are looking for: Efficient online k-mer counting using a probabilistic data structure.** *PLoS One.* 2014; **9**(7): e101271.
PubMed Abstract | Publisher Full Text | Free Full Text
2. Pell J, Hintze A, Canino-Koning R, *et al.*: **Scaling metagenome sequence assembly with probabilistic de Bruijn graphs.** *Proc Natl Acad Sci U S A.* 2012; **109**(33): 13272–7.
PubMed Abstract | Publisher Full Text | Free Full Text
3. Brown CT, Howe A, Zhang Q, *et al.*: **A reference-free algorithm for computational normalization of shotgun sequencing data**. *arXiv preprint.* 2012.
Reference Source
4. Zhang Q, Awad S, Brown CT: **Crossing the streams: a framework for streaming analysis of short DNA sequencing reads**. *PeerJ PrePrints.* 2015; **3**: e1100
Publisher Full Text
5. Döring A, Weese D, Rausch T, *et al.*: **SeqAn an efficient, generic C++ library for sequence analysis.** *BMC Bioinformatics.* 2008; **9**(1): 11.
PubMed Abstract | Publisher Full Text | Free Full Text
6. Crusoe MR, Brown CT: **Walking the talk: adopting and adapting sustainable scientific software development processes in a small biology lab**. *figshare.* 2013.
Publisher Full Text
7. Brown CT, Crusoe MR: **Channeling community contributions to scientific software: a sprint experience**. *figshare.* 2014.
Publisher Full Text
8. Lowe EK, Swalla BJ, Brown CT: **Evaluating a lightweight transcriptome assembly pipeline on two closely related ascidian species**. *PeerJ Preprints.* 2014; **2**.
Publisher Full Text
9. Preston-Werner T: **Semantic versioning 2.0.0**. 2015. [Online; accessed 3-August-2015].
Reference Source
10. Zerbino DR, Birney E: **Velvet: algorithms for *de novo* short read assembly using de Bruijn graphs.** *Genome Res.* 2008; **18**(5): 821–9.
PubMed Abstract | Publisher Full Text | Free Full Text
11. Peng Y, Leung HCM, Yiu SM, *et al.*: **IDBA–a practical iterative de Bruijn graph *de novo* assembler.** In *Research in Computational Molecular Biology.* 2010; 426–440.
Publisher Full Text
12. Haas BJ, Papanicolaou A, Yassour M, *et al.*: ***De novo* transcript sequence reconstruction from RNA-seq using the Trinity platform for reference generation and analysis.** *Nat Protoc.* 2013; **8**(8): 1494–512.
PubMed Abstract | Publisher Full Text | Free Full Text
13. Bankevich A, Nurk S, Antipov D, *et al.*: **SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing.** *J Comput Biol.* 2012; **19**(5): 455–477.
PubMed Abstract | Publisher Full Text | Free Full Text
14. Flajolet P, Fusy E, Gandouet O, *et al.*: **HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm.** *DMTCS Proceedings.* 2008; (1).
Reference Source
15. Howe AC, Jansson JK, Malfatti SA, *et al.*: **Tackling soil diversity with the assembly of large, complex metagenomes.** *Proc Natl Acad Sci U S A.* 2014; **111**(13): 4904–9.
PubMed Abstract | Publisher Full Text | Free Full Text
16. Crusoe MR, Alameldin HF, Awad S, *et al.*: **The khmer project v2.0.** *Zenodo.* 2015.
Data Source

Open Peer Review

Current Referee Status:   

Version 1

Referee Report 08 October 2015

doi:10.5256/f1000research.7456.r10508

Rob Patro
Computer Science Department, Stony Brook University, Stony Brook, NY, USA

This paper describes version 2 of the khmer software suite. The software is developed to provide both a set of directly usable tools (e.g. normalize-by-median for digital normalization) as well as an experimental framework for developers looking to design new algorithms and methods. It has proven very useful on both of these fronts. The repository is highly watched and starred on GitHub, the developers are very responsive (see more below), and both the senior author's group and other researches seem to be leveraging this framework to build new tools and algorithms.

The paper itself does a good job of describing the software at a high level, including the overall design and goals. I would have appreciated slightly more detail about the motivation behind the design decisions, and the tradeoffs they entail (e.g. Why have a Python front-end? Why use hand-written binding code rather than a binding generator, like SWIG, that would allow interfaces to other languages as well?). I understand that a comprehensive description is not feasible in a manuscript of this length. It would be very interesting to know, however, the cost paid for using the high-level interface rather than the C++ library directly. When the underlying computation is trivial, simply having to iterate over an enormous number of things in Python could add non-trivial overhead. Despite these desiderata, I find that the paper is generally well written and does a good job of describing what a new user might want to know about khmer, and so I approve of this manuscript.

Like Daniel, I also downloaded and built the software using the instructions provided in the ReadTheDocs documentation. The process was simple, and worked well, with the exception of a minor glitch running the tests. After debugging the cause of the problem, I posted an issue to the GitHub repository, and received a response in less than a day. I bring this up because, while not an aspect of the paper itself, good developer support is crucial to the long-term survival and utility of a software package — khmer seems to have this.

This brings me to my final point, about the (currently) controversial authorship policy on this paper, which is ancillary to the quality of the paper (and software) itself. At this point, *I must reserve judgement* on whether I think the authorship policy adopted by this paper is "good" or "bad" (for science, the community, etc.). Incidentally, this is a dichotomy that does not capture the subtlety or importance of this issue well. In the manuscript, the authors state "We develop khmer on github.com as a community open source project focused on sustainable software development, and encourage contributions of any kind." Thus, contributions to khmer are of a potentially wide variety in character (and also, I believe, not simply related to improving or maintaining the code). Those who contribute to the design, improve the usability, work on documentation, support new and existing users, and develop and propagate best practices are all

contributing something valuable to the khmer software "ecosystem". It is unreasonable to expect a piece of software that is ~25k lines of code (and growing) to be actively developed, maintained, and supported by only a small contingent of people, many of whom may be graduate students soon to graduate and move on. Thus, if we are interested in the long-term viability and quality of such software, we must adopt a system of credit that values and recognizes a variety of different types of contribution. On the other hand, I do share the concern that, in the midst of the current authorship system, bestowing that recognition in the form of authorship may have the adverse effect of diminishing the public perception of the very credit one is trying to grant. Perhaps there is a solution *along* the lines adopted by this paper, or perhaps something drastically different needs to be considered.

I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Competing Interests: No competing interests were disclosed.

Referee Report 06 October 2015

doi:10.5256/f1000research.7456.r10513

Daniel Katz
Computation Institute, University of Chicago, Chicago, IL, USA

Regarding the paper, it is a fairly straightforward description of a software package, containing all the things that such a paper should have - a description of the goals, the implemented methods, the hardware and software dependencies (systems on which the software has been tested), some guidance on usage, pointers to the software and documentation, and references.

Regarding the software, I did download and build the software, which seemed to work, other than a fair number of warnings. I was not able to successfully test the software, however, due to issues in https://khmer.readthedocs.org/en/v2.0/user/install.html#run-the-tests Does this mean I should not approve the article? Or should I ask the authors for help in understanding the error and hold off on submitting this report?

I would have liked to have chosen "Approved with reservations" for the status of this review, but my reservations are with the F1000 system for this type of paper, not with this specific paper, so in fairness to the authors, given the lack of clarity of what I should be doing as a reviewer for a software paper, I approve this paper based on its quality as a good description of the software, and not on the quality of software (and related documentation) itself.

I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Competing Interests: No competing interests were disclosed.

Referee Response 06 Oct 2015

Daniel S. Katz, University of Chicago, USA

In addition to my report, regarding software papers in general under F1000, I believe that much more should be required from their reviewers, and what **is required** should be made clear. Software journals (e.g., Ubiquity Press's Journal of Open Research Software, Elsevier's Software X) have specific statements of what a reviewer should do, which say a lot about the quality of the review. For JORS, this is defined on a web page (http://openresearchsoftware.metajnl.com/about/editorialPolicies/). For Software X, the criteria are not on the web (as far as I know) but are embedded in the review form/process, and are roughly equivalent.

Competing Interests: none

Reader Comment 08 Oct 2015

F1000 Research, UK

Thanks for this helpful feedback on our guidelines.

Our current guidelines for reviewing software tools are focused around the content of the article itself, and what information should be included. However, reporting issues with the software itself is clearly also important, so we would always encourage referees to download and test the software and include any feedback within the referee reports, so the authors may rectify them. We'll revise our instructions to software tool referees in light of your comments to make this clearer.

Competing Interests: No competing interests were disclosed.

Referee Report 05 October 2015

doi:10.5256/f1000research.7456.r10514

Ewan Birney
European Molecular Biology Laboratory, European Bioinformatics Institute, Hinxton, UK

This is an update of a widely used tool, khmer, which is in broad use in the technical community around de Bruijn graphs and short reads, based on Bloom filters. It is a good update, provides link to the code, and is sensibly written with tests. I have no concern about the scientific aspect of this paper.

I do find the author inclusion list taking a concept and going to the extreme, and I don't think it is sensible to have an anonymised author (en zyme) on the list, with in effect no way to attribute to a person this. Science's openness in publication is also about attribution. Although I understand Titus' consistency of having all git committers as authors, I think it is sensible to make a distinction of substantial/scientific changes, of which the vast majority of the authors are. Acknowledgements are precisely there to handle these other cases.

I believe it is uncontroversial to appropriately trim the author list, to use the acknowledgements for anonymous improvement (happens regularly in science) and small details (again, a commonplace practice).

I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Competing Interests: No competing interests were disclosed.

Discuss this Article

Version 1

Reader Comment 29 Sep 2015

Russell Neches, UC Davis, USA

I wold like to make a comment regarding Lior Pachter's comment.

The use of pseudonyms has a long and important history in scientific discourse.

Despite the fact that we now know its author's identity, the *t*-statistic is known as "Student's *t*-test" because that was the name under which he published it. Pseudonyms can be ad hoc tools used to allow researchers to participate in science despite prejudice among their peers. For example, mathematician Sophie Germain studied, corresponded and published under the name Monsieur Antoine Auguste Le Blanc owing to the near-total exclusion of women from all domains of science in the eighteenth century. Pseudonyms have also been deployed to shield researchers from prejudice beyond the scientific community; mathematician Jacques Feldbau published under the less-Jewish-sounding name Jacques Laboureur shortly before he was deported to Auschwitz. Sometimes the motives behind the choice to publish pseudonymously are obscure or personal, such as Carl Ludwig Siegel's choice in 1926 to publish his reduction of a hyperelliptic equation to a unit equation under the name "X." Even Isaac Newton published his alchemical dabblings as "Jehovah Sanctus Unus."

There is a long-standing tradition of etiquette regarding pseudonyms in science. Simply put, one endeavors to respect the author's choice. Of course, there are limits to how far to carry this respect. Most people agree that the courtesy ought not be extended to protect people who use pseudonyms to obtain impunity when attacking others.

Lior writes that, "Authors who did contribute should be listed with full name with affiliation so that they can be contacted if the need arises." The author that Lior has singled out here has made him/herself available for anyone to contact under their pseudonym via email, Twitter, LinkedIn and in person at a variety of professional conferences. Even if one accepts the premise under which it was raised, the objection is unfounded. I respectfully suggest the editors expunge the identifying information Lior placed in his comment. I also feel that Lior's actions in this matter should remain part of the record.

Competing Interests: Russell Neches is a graduate student at the same university as some of the authors, though shares no departmental affiliations, program affiliations, publications or funding sources with them. They do, however, occasionally enjoy beer together.

Reader Comment 28 Sep 2015

F1000 Research, UK

Thank you for bringing this to our attention.

Because F1000Research does not have editors and the authors are in charge of their publication, one of the key requirements for publication is that the 'lead' authors, who have to engage in the public discussion with referees and readers, are active researchers and meet our authorship criteria. For an author-driven model to work, this is a key check done on submission.

The ICMJE "Uniform requirements", which specify what type of contribution justify full authorship, constitute best practice in STM publishing and are listed in our policy; the Author Contribution section is meant to ensure transparency for readers, outlining why authors were indeed included in the author list.

We appreciate that readers may not always agree that an individual author's contribution in a paper is 'substantial' enough to justify full authorship. However, consistent with the F1000Research publishing ethos generally applied to the content of a paper (where no editors judge whether the finding in a paper is 'significant' or substantial enough to justify publication), the in-house editorial team does not usually judge whether an individuals' contribution is sufficient to justify authorship – a call that can be subjective. As with many traditional journals, on submission, we ask the submitting authors confirm that all the co-authors have agreed to the submission of the article.

Competing Interests: No competing interests were disclosed.

Reader Comment 26 Sep 2015

Lior Pachter, University of California, Berkeley, USA

This article appears to violate the F1000 criteria for authorship mentioned here and defined in the "uniform requirements". Specifically, the contribution of "one or more Git commits" in the code which is the sole contribution listed for the majority of authors fails to satisfy the first requirement, namely

- Substantial contributions to the conception or design of the work; or the acquisition, analysis, or interpretation of data for the work.

Most of the authors' Git commits consist of fixing very minor typos (e.g. see here and here). Such "contributions" clearly do not rise to the level of authorship qualification as specified in the "uniform requirements" and the individuals who made such contributions should instead be mentioned in the acknowledgements section.

Authors who did contribute should be listed with full name with affiliation so that they can be contacted if the need arises. This may be necessary to confirm another "uniform requirement" for authorship:

- Agreement to be accountable for all aspects of the work in ensuring that questions related to the accuracy or integrity of any part of the work are appropriately investigated and resolved.

I noticed that "en zyme" is listed as an author with the affiliation of "independent Researcher in Boston, MA". This individual appears to be Nathan Kohn, a part time lecturer at Boston University Metropolitan College and should be listed as such (assuming his contribution merits authorship).

Competing Interests: I have no competing interests to disclose.