Bioinformatics, 34(22), 2018, 3924–3925
doi: 10.1093/bioinformatics/bty432
Advance Access Publication Date: 1 June 2018
Applications Note

Sequence analysis

Axe: rapid, competitive sequence read demultiplexing using a trie

Kevin D. Murray* **and Justin O. Borevitz**

ARC Centre of Excellence in Plant Energy Biology, Department of Plant Science, Research School of Biology, ANU, Canberra, Australia

*To whom correspondence should be addressed.

Associate Editor: Bonnie Berger

Received on March 5, 2018; revised on April 30, 2018; editorial decision on May 21, 2018; accepted on May 30, 2018

Abstract

Summary: We describe a rapid algorithm for demultiplexing DNA sequence reads with in-read indices. Axe selects the optimal index present in a sequence read, even in the presence of sequencing errors. The algorithm is able to handle combinatorial indexing, indices of differing length and several mismatches per index sequence.
Availability and implementation: Axe is implemented in C, and is used as a command-line program on Unix-like systems. Axe is available online at https://github.com/kdmurray91/axe, and is available in Debian/Ubuntu distributions of GNU/Linux as the package axe-demultiplexer.
Contact: axe@kdmurray.id.au
Supplementary information: Supplementary data are available at *Bioinformatics* online

1 Introduction

The incredible yield of modern DNA sequencing technologies has enabled the multiplexing of DNA samples into a single sequencing unit. Multiplexing is achieved by the addition of short sequences (indices) to each molecule to be sequenced. When sequenced, these index sequences uniquely identify the sample to which a sequence read belongs. Many commercial protocols use platform specific features to add these DNA indices such that sequencing platforms can automatically demultiplex these samples. However, many custom sequencing protocols, including GBS (Elshire *et al.*, 2011), add indices which end users must themselves demultiplex. Combinatorial indexing schemes add independent index sequences to both pairs of a paired-end sequencing protocol, and samples are identified by the combination of these two index sequences [e.g. (Peterson *et al.*, 2012)].

Many sequencing read demultiplexers have been published. For example, both Flexbar (Dodt *et al.*, 2012) and the Fastx-toolkit's fastx_barcode_splitter.pl (Available at http://hannonlab.cshl.edu/fastx_toolkit/) accept single- and paired-end reads, however they cannot demultiplex combinatorial indices. AdapterRemoval (Schubert *et al.*, 2016) can demultiplex combinatorial indices, but cannot demultiplex indexes which differ in length. The same is true of DeML (Renaud *et al.*, 2015), which also uses a trie data structure. We developed axe to address these shortcomings.

2 Materials and methods

2.1 Algorithm

Axe matches the prefix of a sequence read against a pre-computed trie of index sequences. To do so, axe first calculates all sequences within a given Hamming distance (Hamming, 1950) of each index sequence. Axe then associates each of these sequences with its respective sample identifier using a double-array trie. Reads are demultiplexed by finding the read's longest prefix in the trie of (possibly mutated) index sequences, and assigning that read to its associated sample. This algorithm extends easily to combinatorial indexing, where two independent indices prefix each read of a read pair. The constant-time nature of these lookups allows Axe to remain rapid even with many thousand possible samples. Although this algorithm is agnostic as to which end of a sequencing read contains a index, only 5' (prefix) index demultiplexing is currently implemented.

2.2 Operation

To demultiplex sequence reads, one uses the command axe-demux. This command takes input reads as FASTQ or FASTA files which may contain single- or paired-end reads. Paired-end reads may be interleaved, and output reads can be written in any of these formats.

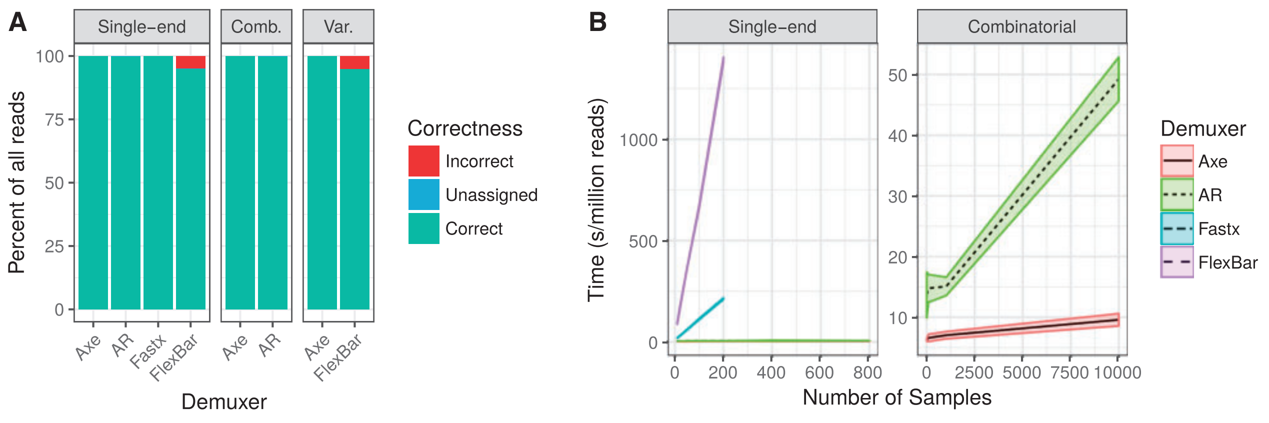

Fig. 1. (**A**) Accuracy of read assignment. Axe is able to perfectly demultiplex all reads, as is fastx. Only flexbar incorrectly assigns reads. Note: 'Comb.' refers to combinatorial index sets, and 'Var.' refers to index sets with variable length index sequences. (**B**) Computational performance of demultiplexers (seconds per million reads, mean ± SD). Axe is the fastest in all cases, closely followed by AdapterRemoval. fastx and flexbar are appreciably slower, especially when the number of indices is large

Axe is implemented in the C language, and is available at https://github.com/kdmurray91/axe. It may be built from source code on any modern POSIX operating system (including GNU/Linux and Mac OS X). The only dependencies not bundled with the source distribution are CMake and zlib. Axe is also available in the Debian and Ubuntu GNU/Linux distributions as the axe-demultiplexer software package.

3 Results

3.1 Demultiplexing accuracy and performance

We benchmark the speed and accuracy of axe, flexbar, AdapterRemoval and fastx_barcode_splitter.pl (hereafter 'fastx'). When demultiplexing read pairs with an index sequence on one read only (single-end), both axe and fastx are able to perfectly demultiplex all reads, with no error and with no reads left unassigned. AdapterRemoval fails to assign a minuscule proportion of reads, while flexbar mis-assigns several percent of reads (Fig. 1A). When demultiplexing combinatorially indexed read pairs, axe again demultiplexes all reads perfectly and AdapterRemoval fails to assign a small proportion. When demultiplexing reads with variable-length index sequences, axe performs perfectly, while flexbar mis-assigns several percent of reads. In all cases, axe is the fastest demultiplexer tested. AdapterRemoval performs several times slower than axe. fastx and flexbar perform hundreds of times slower than axe and AdapterRemoval (Fig. 1B). The methods underlying these experiments are available as online Supplementary Material.

3.2 Summary

Here, we implement a rapid and accurate algorithm for demultiplexing 5'-indexed reads. We show equal or improved accuracy and reduced computational cost compared to previous software developed to perform this task. In addition, more complex indexing schemes including combinatorial and/or variable length index sequences are supported. While in-read indexing is being phased out in some protocols, it persists in others such as GBS (Elshire *et al.*, 2011) and RNAseq using unique molecular identifiers (Kivioja *et al.*, 2012). Additionally, Axe's algorithm is applicable to demultiplexing out-of-read indexing schemes, though the implementation does not currently support this.

Funding

This work was supported by the Australian Research Council Centre of Excellence in Plant Energy Biology [CE140100008], and was undertaken with the assistance of resources from the National Computational Infrastructure (NCI), which is supported by the Australian Government. KDM is supported by an Australian Government Research Training Program (RTP) Scholarship.

Conflict of Interest: none declared.

References

Dodt,M. *et al.* (2012) FLEXBAR: flexible barcode and adapter processing for next-generation sequencing platforms. *Biology*, **1**, 895–905.

Elshire,R.J. *et al.* (2011) A robust, simple genotyping-by-sequencing (GBS) approach for high diversity species. *PLoS One*, **6**, e19379.

Hamming,R.W. (1950) Error detecting and error correcting codes. *Bell Syst. Tech. J.*, **29**, 147–160.

Kivioja,T. *et al.* (2012) Counting absolute numbers of molecules using unique molecular identifiers. *Nat. Methods*, **9**, 72–74.

Peterson,B.K. *et al.* (2012) Double digest RADseq: an inexpensive method for de novo SNP discovery and genotyping in model and non-model species. *PLoS One*, **7**, e37135.

Renaud,G. *et al.* (2015) deML: robust demultiplexing of Illumina sequences using a likelihood-based approach. *Bioinformatics*, **31**, 770–772.

Schubert,M. *et al.* (2016) AdapterRemoval v2: rapid adapter trimming, identification, and read merging. *BMC Res. Notes*, **9**, 88.