Bioinformatics, 36(10), 2020, 3254–3256
doi: 10.1093/bioinformatics/btaa112
Advance Access Publication Date: 24 February 2020
Applications Note

Sequence analysis

Nubeam-dedup: a fast and RAM-efficient tool to de-duplicate sequencing reads without mapping

Hang Dai* and Yongtao Guan*

Department of Biostatistics and Bioinformatics, Duke University School of Medicine, Durham, NC 27705, USA

*To whom correspondence should be addressed.

Associate Editor: Inanc Birol

Received on October 2, 2019; revised on February 6, 2020; editorial decision on February 12, 2020; accepted on February 14, 2020

Abstract

Summary: We present *Nubeam-dedup*, a fast and RAM-efficient tool to de-duplicate sequencing reads without reference genome. *Nubeam-dedup* represents nucleotides by matrices, transforms reads into products of matrices, and based on which assigns a unique number to a read. Thus, duplicate reads can be efficiently removed by using a collisionless hash function. Compared with other state-of-the-art reference-free tools, *Nubeam-dedup* uses 50–70% of CPU time and 10–15% of RAM.

Availability and implementation: Source code in C++ and manual are available at https://github.com/daihang16/nubeamdedup and https://haplotype.org.

Contact: hang.dai@duke.edu or yongtao.guan@duke.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Duplicate reads are a commonplace in sequencing data. The duplications often occur during the library preparation when the polymer chain reaction (PCR) was used to obtain more DNA molecules or to enrich DNA molecules that have adaptors attached to both ends (Head *et al.*, 2014). Because PCR is a stochastic process, it can create artifacts that may suggest one molecule is more abundant than the other. Removing duplicate reads is often one of the important QC steps in a pipeline that performs variant calling, differential gene expression analysis and more. One approach is to identify reads that mapped to the same location of a reference genome and consider them as duplicates of each other even these reads have one or a few base-pair differences. Popular software performing mapping-based de-duplications are Picard MarkDuplicates (http://broadinstitute.github.io/picard/) and SAMTools rmdup (Li *et al.*, 2009).

When reference genomes are unavailable, incomplete or in low quality, such as in the *de novo* assembly and the analysis of metagenomics whole-genome sequencing data, *de novo* (or reference-free) de-duplication is indispensable to remove PCR duplicates before downstream analysis (Audoux *et al.*, 2017; Lu *et al.*, 2017; Rahman *et al.*, 2018). Currently, there are three categories of methods for *de novo* de-duplication. The first category sorts reads and compares adjacent pairs (Xu *et al.*, 2012). The second category clusters reads of same prefix and then compares suffixes of reads within each cluster to identify duplicates (Burriesci *et al.*, 2012; Expósito *et al.*, 2017; González-Domínguez *et al.*, 2016; Manconi *et al.*, 2016). The third category is a hash-based method using cuckoo filter, a probabilistic data structure to test element inclusion, which allows false positives (Gaia *et al.*, 2019). The first two categories are slow and RAM-intensive. Though the prefix–suffix method was designed to parallelize the process, we found it not necessarily faster than sorting-based method without using Apache Spark or GPU; its advantage over sorting-based method is the ability to account for sequencing error. The last category uses less RAM, but remains slow and may remove unique reads; moreover, in our limited trials, we found it failed to remove all duplicates in very small datasets. All three methods have difficulty to handle duplicates on different strands.

Here, we present *Nubeam-dedup*, a fast and RAM-efficient tool to de-duplicate sequencing reads *de novo*. *Nubeam-dedup* is based on the *Nubeam* (Dai and Guan, 2019), which represents nucleotides by matrices, transforms reads into products of matrices and based on which assigns a unique number to a read. With different reads assigned different numbers, *Nubeam* provides a perfect hash function for DNA sequences, which enables fast and RAM-efficient de-duplication.

2 Materials and methods

We assume reads have the same length L and consist of four nucleotides A, T, C and G, with an unknown nucleotide denoted by N. We first construct four binary sequences from a read by using each of the four nucleotides as the reference, with the reference nucleotide being 1 and the others 0, then, we concatenate the four binary sequences into a single binary sequence B (of length $4L$). We use a matrix $\mathbf{M}_1 = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ to represent 1, and its transpose $\mathbf{M}_0 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$ to represent 0. For a binary sequence $B = b_1 b_2 \ldots b_{4L}$,

we obtain the product matrix $\mathbf{M}_B = \Pi_{j=1}^{4L} \mathbf{M}_{b_j}$. Let $\mathbf{W} = \begin{pmatrix} 1 & \sqrt{3} \\ \sqrt{2} & \sqrt{5} \end{pmatrix}$ be a weight matrix, we designate $\mathrm{tr}(\mathbf{W}\mathbf{M}_B)$ as the Nubeam number to the read (Fig. 1a).

If two reads are different, their corresponding binary sequences must be different. We proved previously that if two binary sequences differ, then their Nubeam numbers differ (Dai and Guan, 2019). Thus, if two reads are different, their assigned Nubeam numbers must be different (Fig. 1a). Technically, the mapping from a read to its Nubeam number is a collisionless hash function. We use unordered associative containers in C++ to determine the uniqueness of Nubeam numbers and thereby the uniqueness of reads (details in the sections below). Unique reads will be written to new FASTQ file on-the-fly.

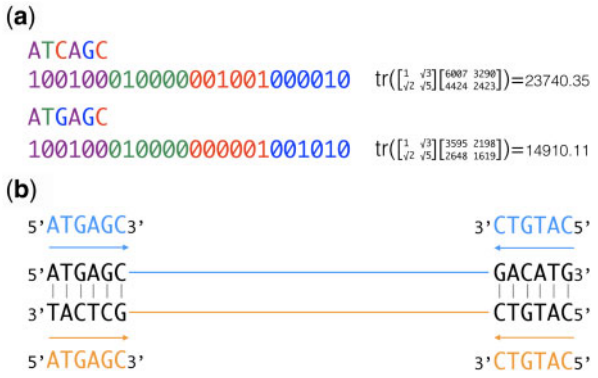

Fig. 1. (**a**) Cartoon of how to obtain Nubeam numbers for reads. Convert each read to a binary sequence, turn each binary sequence into a product matrix, and obtain a number from each product matrix. The two reads with the difference of one nucleotide (C versus G at position 3) are assigned by different Nubeam numbers. (**b**) How to consider strand flip? The DNA strands in blue (top one) and orange (bottom one) are reverse complement of each other. Interchanging the two PE reads for blue strand (top ones) results to the two PE reads for orange strand (bottom ones). (Color version of this figure is available at *Bioinformatics* online.)

2.1 Single-end reads

To process single-end reads, we use a container `unordered_set` to record as keys Nubeam numbers of already processed reads. For each new read, we compute its Nubeam number and query whether it shares the same value with a key already exists in the `unordered_set`. If yes, the read is considered a duplicate and we move to the next read. If no, the Nubeam number is inserted to the `unordered_set` as a key and the read will be recorded as a unique read. And we move to the next read. To account for the possibility that two reads may be duplicates but come from opposite strands, if a read is recorded as a unique read, we also insert the Nubeam number of its reverse complement into the `unordered_set`.

2.2 Paired-end reads

To process paired-end reads, we use two containers to record Nubeam numbers of already processed read-pairs. Let i denotes the line index of a read-pair in the input FASTQ file and i_1 and i_2 denote the Nubeam numbers of read 1 and read 2, respectively. We use an `unordered_multimap` to store the key-value pair $<i_1, i>$ and an `unordered_map` to store key-value pair $<i, i_2>$. In this way, the Nubeam numbers for the two reads in a read-pair are linked through the line index. If a new read-pair at j-th line in the FASTQ file with Nubeam numbers (j_1, j_2) is a duplicate of some read-pair that has already been processed, we would observe that (i) j_1 exists in the `unordered_multimap` as a key; and (ii) among a set of values in `unordered_map` that linked to j_1 there exists one that equals j_2. If either condition (i) or (ii) fails, we record the read-pair into containers by inserting $<j_1, j>$ and $<j, j_2>$ into the `unordered_multimap` and the `unordered_map`, respectively.

To account for the same read-pairs originating from complementary strand, for each read-pair, we first check if the pair itself is a duplicate and then check if the read-pair from the complementary strand is a duplicate. If none of them is duplicate, the original read-pair can be recorded into the containers. Note that, the chosen data structure requires no additional storage to check complementary

Table 1. CPU time, memory footprint and pairs remained after applying *FastUniq*, *ParDRe*, *Nubeam-dedup* and *Nubeam-dedup-s* on different datasets

Performance measures		HMP 69 025 235×2 100 bp	GTEx 91 077 485×2 76 bp	BGI 700 616 762×2 50 bp	GIAB 3 099 165 624×2 148 bp
CPU time (s)	*Nubeam-dedup*	478	507	4312	36 947
	Nubeam-dedup-s	523	544	5361	41 443
	FastUniq	732	766	8740	>53 300[a]
	ParDRe[d]	1434	1424	>10 008[b]	>83 700[c]
Memory footprint (GB)	*Nubeam-dedup*	6.2	6.3	56.8	280.2
	Nubeam-dedup-s	6.2	6.2	56.5	279.5
	FastUniq	52.8	61.1	386.3	>2756.0[a]
	ParDRe[d]	68.0	79.9	>596.9[b]	>3983.4[c]
Pairs remained	*Nubeam-dedup*	69 024 928	74 069 285	683 118 530	3 090 659 918
	Nubeam-dedup-s	68 729 758	70 047 566	679 045 669	3 078 856 656
	FastUniq	69 023 559	74 069 285	683 118 531	NA
	ParDRe	69 024 928	74 148 969	NA	NA

Note: Compared with *Nubeam-dedup*, *Nubeam-dedup-s* checks for strand flip. The CPU time is the average of five independent runs.

HMP: Illumina HiSeq2000 PE100, metagenomics sample SRS024075 from Human Microbiome Project (Human Microbiome Project Consortium *et al.*, 2012).

GTEx: Illumina HiSeq2000 PE76, RNA-seq sample SRR665369 from Genotype-Tissue Expression Project (GTEx Consortium *et al.*, 2015).

BGI: BGISEQ-500 PE50, WGS of HapMap sample NA12878 (Huang *et al.*, 2017).

GIAB: Illumina HiSeq2500 PE148, WGS of HapMap sample NA12878 from Genome in a Bottle (Zook *et al.*, 2019).

[a]Estimated by using 15% of data.

[b]Estimated by using 12.5% of data.

[c]Estimated by using 1% of data.

[d]All CPU time and memory footprint for *ParDRe* were measured using single-core mode. The multi-core modes, though took marginally less elapsed real time, consumed much larger memory than single-core mode (Supplementary Table S1).

strand, and the extra computational cost to check complementary strand is marginal (Fig. 1b).

3 Results

We compared *Nubeam-dedup* with two representative and state-of-the-art *de novo* de-duplication tools: *FastUniq* (Xu *et al.*, 2012), which implements sorting-based algorithm, and *ParDRe* (González-Domínguez *et al.*, 2016), which implements prefix–suffix algorithm. Our benchmarking computer has two 2.1 GHz Intel Xeon Platinum 8160 CPUs, 512 GB RAM and running Ubuntu 18.04.2 LTS. For *FastUniq*, we used the default setting. For *ParDRe*, to optimize its performance, we invoked multi-threading option and used 16 or all the 48 cores with two threads per core (Supplementary Material). The CPU time and memory footprint were recorded by Linux bash command *time* and *process_watcher* (White, 2018), respectively. We used four datasets to compare the performance, including two different sequencing platforms (Illumina and BGI) and three types of biological samples (DNA from microbiomes, RNA from human tissues and DNA from humans).

Our comparison demonstrated that *ParDRe* was slower than *FastUniq* and had a larger RAM footprint (Table 1). *ParDRe* was unable to finish running even one-fourth of BGI data under 48-core mode due to extraordinarily large RAM requirement. In contrast, *Nubeam-dedup* was about 1.5–2 times faster than *FastUniq*, with RAM footprint only a fraction (about 10–15%) of that of *FastUniq*. Both *FastUniq* and *ParDRe* were not able to de-duplicate the huge GIAB dataset (3.1 billon of PE148), whereas *Nubeam-dedup* finished running in about 11 h (Table 1). *Nubeam-dedup* removed all duplicate reads without a single false positive. Upon examining the differences between the results of *Nubeam-dedup* and *FastUniq*, we discovered that *FastUniq* considered two reads with different lengths identical if the shorter one matching 5′ end of the longer one, whereas *Nubeam-dedup* considered them different reads.

Finally, *Nubeam-dedup* has its limitation as it cannot conveniently account for sequencing error. However, since the read lengths are short and the sequencing error rate is 0.1% (or less) per base for current sequencing technology, such as Illumina, the probability that the duplicates were masked by sequencing error is negligible.

Funding

This work was supported by start-up fund from the Duke University School of Medicine.

Conflict of Interest: none declared.

References

Ardlie,K.G.,*et al.*; The GTEx Consortium. (2015) The Genotype-Tissue Expression (GTEx) pilot analysis: multitissue gene regulation in humans. *Science*, **348**, 648–660.

Audoux,J. *et al.* (2017) DE-kupl: exhaustive capture of biological variation in RNA-seq data through k-mer decomposition. *Genome Biol.*, **18**, 243.

Burriesci,M. *et al.* (2012) Fulcrum: condensing redundant reads from high-throughput sequencing studies. *Bioinformatics*, **28**, 1324–1327.

Dai,H., and Guan,Y. (2019) Human microbiome sequences in the light of the Nubeam. *bioRxiv*. doi : 10.1101/763631.

Expósito,R. *et al.* (2017) MarDRe: efficient MapReduce-based removal of duplicate DNA reads in the cloud. *Bioinformatics*, **33**, 2762–2764.

Gaia,A. *et al.* (2019) NGSReadsTreatment–a Cuckoo Filter-based tool for removing duplicate reads in NGS data. *Sci. Rep.*, **9**, 1–6.

González-Domínguez,J. *et al.* (2016) ParDRe: faster parallel duplicated reads removal tool for sequencing studies. *Bioinformatics*, **32**, 1562–1564.

Head,S. *et al.* (2014) Library construction for next-generation sequencing: overviews and challenges. *Biotechniques*, **56**, 61–77.

Huang,J. *et al.* (2017) A reference human genome dataset of the BGISEQ-500 sequencer. *Gigascience*, **6**, 1–9.

Human Microbiome Project Consortium *et al.* (2012) Structure, function and diversity of the healthy human microbiome. *Nature*, **486**, 207–214.

Li,H. *et al.*; 1000 Genome Project Data Processing Subgroup. (2009) The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, **25**, 2078–2079.

Lu,Y. *et al.* (2017) CAFE: aCcelerated Alignment-FrEe sequence analysis. *Nucleic Acids Res.*, **45**, W554–W559.

Manconi,A. *et al.* (2016) Removing duplicate reads using graphics processing units. *BMC Bioinformatics*, **17**, 346.

Rahman,A. *et al.* (2018) Association mapping from sequencing reads using k-mers. *Elife*, **7**, e32920.

White,A. (2018) *process-watcher: Watch Linux Processes by Polling/Proc and Notify via Email or Desktop Notification When They Complete*. https://github.com/arlowhite/process-watcher (14 June 2019, date last accessed).

Xu,H. *et al.* (2012) FastUniq: a fast de novo duplicates removal tool for paired short reads. *PLoS One*, **7**, e52249.

Zook,J. *et al.* (2019) An open resource for accurately benchmarking small variant and reference calls. *Nat. Biotechnol.*, **37**, 561–566.