ISSN 0973-2063 (online) 0973-8894 (print)
Bioinformation 18(1): 36-40 (2022)

BIOINFORMATION
Discovery at the interface of physical and biological sciences
BIOMEDICAL INFORMATICS
www.bioinformation.net
Volume 18(1)
Research Article

Received November 13, 2021; Revised November 29, 2021; Accepted November 29, 2021, Published January 31, 2022
DOI: 10.6026/97320630018036

Declaration on Publication Ethics:
The author's state that they adhere with COPE guidelines on publishing ethics as described elsewhere at https://publicationethics.org/. The authors also undertake that they are not associated with any other third party (governmental or non-governmental agencies) linking with any form of unethical issues connecting to this publication. The authors also declare that they are not withholding any information that is misleading to the publisher in regard to this article.

Declaration on official E-mail:
The corresponding author declares that official e-mail from their institution is not available for all authors

Comments from readers:
Articles published in BIOINFORMATION are open for relevant post publication comments and criticisms, which will be published immediately linking to the original article without open access charges. Comments should be concise, coherent and critical in less than 1000 words.

Edited by Pandjassarame Kangueane
Citation: Altayyar & Monim Artoli, Bioinformation 18(1): 36-40 (2022)

Fast-HBR: Fast hash based duplicate read remover

Sami Altayyar* **& Abdel Monim Artoli**

Department of Computer Science, College of Computer and Information Sciences, King Saud University, P.O. Box 51178, Riyadh 11543, Saudi Arabia, E-mail:436107303@student.ksu.edu.sa , aartoli@ksu.edu.sa; *Corresponding author

Abstract:
The Next-Generation Sequencing (NGS) platforms produce massive amounts of data to analyze various features in environmental samples. These data contain multiple duplicate reads which impact the analyzing process efficiency and accuracy. We describe Fast-HBR, a fast and memory-efficient duplicate reads removing tool without a reference genome using *de-novo* principles. It uses hash tables to represent reads in integer value to minimize memory usage for faster manipulation. Fast-HBR is faster and has less memory footprint when compared with the state of the art De-novo duplicate removing tools. Fast-HBR implemented in Python 3 is available at https://github.com/Sami-Altayyar/Fast-HBR.

Background:
The number of the publicly available NGS projects tripled from 1200 in 2017 to 3500 in 2020 [1-2]. Therefore, preprocessing of data is essential to reduce the size of the data with an adequate level of data quality [3]. One of the preprocessing steps that reduce the dataset size is removing duplicate reads in the dataset. This step is essential for sequence-based algorithms since duplicate reads affect the algorithm accuracy [4]. Removing duplicate reads may reduce the assembly algorithms consumption of RAM [5]. Duplicate reads removal tools are either reference based or *de novo*. Some examples of de novo tools are CD-HIT [6], FastUniq [7] and Fulcrum [8]. Available *de novo* tools include NGS Reads Treatment [9], Nubeam-

ISSN 0973-2063 (online) 0973-8894 (print)
Bioinformation 18(1): 36-40 (2022)

dedup [5], BioSeqZip [10] and Minirmd [11]. NGS Reads Treatment [9] is a hash-based tool that uses Cuckoo Filter [12] which is a probabilistic data structure. The authors elsewhere [5] developed the Nubeam-dedup tool that uses Nubeam [13] to represent each read as a number by calculating a product of matrices that represent nucleotides in the read. The BioSeqZip [10] tool starts by splitting the reads into small chunks, and then it sorts them alphabetically with memory limiting feature having long processing time. Minirmd [11] with the help of k-minimizer [14] clusters the reads into groups, where each group will contain reads that have the same k-minimizer in the same position. Therefore, it is of interest to describe Fast-HBR, a fast and memory-efficient duplicate reads removing tool without a reference genome using *de-novo* principles.

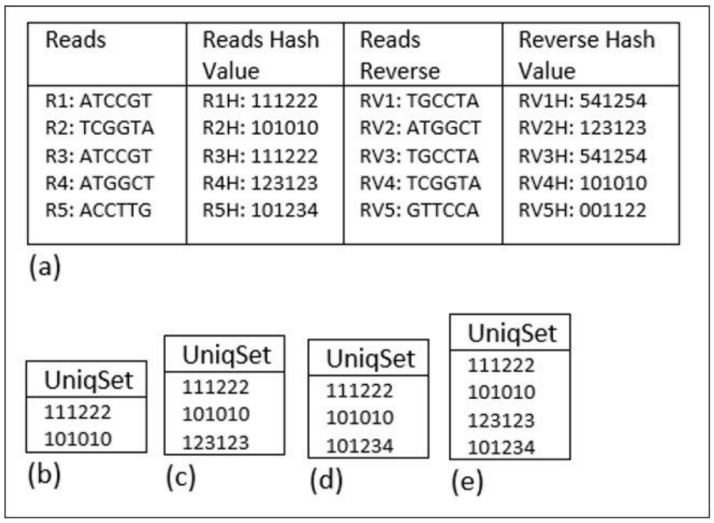

Figure 1: Fast-HBR methodology illustrated using an example.

Methodology:

Fast-HBR is implemented in Python 3. Therefore, it is platform-independent. The source code is available at https://github.com/Sami-Altayyar/Fast-HBR. It uses Python's built-in hash function to represent reads (in nucleotide or amino acid level) as an integer value. The reads hash value is stored in a set and each new read hash value will compare to the set items to decide if it is duplicate or not. The input files are either a single-end or paired-end, and it could process the files with reverse complement removing option or without it.

Single-end files:

In single-end files, each read is independent; therefore its evaluation process will depend only on its hash value. Fast-HBR will starts by creating a set (UniqSet) to store all unique hash values. After that, it extracts from the input file one read at a time and then calculate the hash value (HV1) of the read. Depending on HV1 and the reverse complement removing option, Fast-HBR will have three cases. In the first case, if HV1 is in UniqSet, the read will consider a duplicate and will be discarded. In the second case, if HV1 is not in UniqSet and the reverse complement removing option is not activated, then HV1 will be added to UniqSet and the read will be written in the output file. In the third case, if HV1 is not in UniqSet and the reverse complement removing option is activated, Fast-HBR will calculate the hash value of the reverse complement of the read (HV2). If HV2 is in UniqSet the read will consider a duplicate and will be discarded. Otherwise, the read is unique and then only HV1 will be added to UniqSet and the read will be written in the output file.

We consider the input reads and their reverse and the hash values for the reads and the reverse as shown in **Figure 1A.** In the beginning, the reads R1 and R2 are unique and therefore their hash values would be added to UniqSet as in **Figure 1B**. For R3, its hash value (111222) is in UniqSet therefore R3 would be considered as duplicate read, and it will be discarded. Regarding read R4, the read hash value (123123) is not in UniqSet therefore if the reverse complement option is not active it will be considered a unique read and its hash value would be added to UniqSet as in **Figure 1C,** but if the reverse complement option is active, the hash value of the read reverse complement RV4 (101010) is in UniqSet and it will be considered as duplicate read and discarded.Finally, the read R5 hash value (101234) is not in UniqSet and its reverse complement hash value (001122) is not in UniqSet. Therefore, if the reverse complement option is active or not the read R5 is unique and the hash value of it (101234) would be added to UniqSet. **Figure 1D** shows the final UniqSet if the reverse complement option is active and Figure 1E if the reverse complement option is not active. Fast-HBR will not calculate the reverse complement hash unless it is necessary, which will minimize computational operations to the minimum. On the other hand, since we store only HV1 of unique reads in UniqSet, the number of elements in UniqSet will be less than or equal to the number of reads in the file. Consequently, the memory would be used efficiently, especially because the hash values in UniqSet are integers.

Table 1: Properties of the used datasets

Name	Number of reads	Layout	Size	Published
SRR10315305	99,998,928	SINGLE	3.7GB	6/26/2020
SRR13555429	308,271,670	SINGLE	22.2GB	3/1/2021
SRR13555395	524,201,007	SINGLE	28.3GB	3/1/2021
SRR681003	102,886,046	PAIRED	6.8GB	7/22/2015
SRR837669	213,967,552	PAIRED	27.4GB	4/18/2014
SRR6424061	476,540,265	PAIRED	58GB	1/2/2019

Paired-end files:

For paired-end file processing, Fast-HBR would create a set (UniqSet) to store unique hash values. For each pair of reads i (R i1, R i2), if the reverse complement removing option is not activated, Fast-HBR would calculate the hash value (HV) as the hash of the concatenation of the two reads (Hash (R i1 concatenate R i2)). Then, if HV is present in UniqSet the reads pair (R i1, R i2) would be considered as a duplicate. Otherwise, the reads pair (R i1, R i2) is unique and will be written to the output file and HV would be added to UniqSet. The second case is when the reverse complement removing option is active as shown in **Figure 1**. Here, the change is the calculation of HV. It would be the sum of the hash value of R i1 plus the hash value of R i2. Therefore, if the pair reads in position (i) swapped in other position (j) in the file, they will have the same HV value and should be considered as a duplicate. Either with or

ISSN 0973-2063 (online) 0973-8894 (print)
Bioinformation 18(1): 36-40 (2022)

without the reverse complement removing option, this methodology would guarantee that each pair of reads would represent by only one integer value. Because of that, the number of elements in UniqSet will be less than or equal to the number of pairs of reads, which lets Fast-HBR deal with memory more efficiently.

Table 2: The number of removed reads in each dataset after applying the tools

Tools	Number of removed reads				
Dataset	Fast-HBR	Nubeam-dedup	Minirmd	BioSeqZip	NGSReads-Treatment
SRR10315305	86838679	86838679	69223179	86838679	86838679
SRR13555429	93616179	93616179	NC*	93616179	NC*
SRR13555395	221901599	221901599	NC*	221901599	NC*
SRR681003	21,384,300	21,384,300	21,078,681	21,384,300	NC*
SRR837669	31,981,762	31,981,762	NC*	31,981,762	NC*
SRR6424061	40,215,059	40,215,067	NC*	40,215,059	NC*

*The tool does not complete the processing of the dataset.

Table 3: CPU Time in minutes and Memory in Gigabytes by each tool without reverse complement removing option

Tool		Fast-HBR	Nubeam-dedup	Minirmd	BioSeqZip	NGSReads-Treatment
Dataset						
SRR10315305	Time	4.35	6.54	24.52	27.02	486.03
	Memory	3.24	2.11	130.67	13.03	6.35
SRR13555429	Time	23.21	29.53	NC*	105.2	NC*
	Memory	51.68	34.7		14.27	
SRR13555395	Time	39.12	49.56	NC*	176	NC*
	Memory	84.61	45.14		14.28	
SRR681003	Time	9.72	27.37	6.69	26.48	NC*
	Memory	22.07	28.84	193.2	13.38	
SRR837669	Time	28.34	80.75	NC*	102.38	NC*
	Memory	46.89	61.6		14.39	
SRR6424061	Time	74.45	97.08	NC*	305.55	NC*
	Memory	104.39	141		13.56	

*The tool does not complete the processing of the dataset.

Table 4.CPU Time in minutes and Memory in Gigabytes by each tool with reverse complement removing option

	Tool	Fast-HBR	Nubeam-dedup	Minirmd
Dataset				
SRR10315305	Time	4.91	5.48	27.82
	Memory	3.24	4.23	130.09
SRR13555429	Time	38.83	61.4	NC*
	Memory	51.65	69.63	
SRR13555395	Time	50.52	73.77	NC*
	Memory	84.58	90.53	
SRR681003	Time	9.66	46.29	7.24
	Memory	20.97	21.83	193.18
SRR837669	Time	29.33	88.97	NC*
	Memory	47.18	61.53	
SRR6424061	Time	90.57	NC*	NC*
	Memory	105.21		

*The tool does not complete the processing of the dataset.

Results and Discussion:

Results obtained using Fast-HBR is tabulated in **Table 2, Table 3 and Table 4.** Comparisons with NGS Reads Treatment [9], Nubeam-dedup [5], BioSeqZip [10] and Minirmd [11] similar state of the art *De novo* tools are shown. The Linux bash command time was used to calculate the time spent by each tool and the tool's maximum memory usage. In this comparison, six datasets were used, three are single-end datasets (SRR10315305, SRR13555429 & SRR13555395) and three paired-end datasets (SRR681003, SRR837669, SRR6424061) and Table 1 shows the datasets information. We run the tools on King Abdulaziz University's High Performance Computing Center (Aziz Supercomputer) (http://hpc.kau.edu.sa), where all tools run on normal nodes which equipped with 24 processors and 96GB memory. Because NGS Reads Treatment [9] and BioSeqZip [10] do not support the reverse complement removing option, we had to conduct two comparisons for each dataset. First, all five tools were compared without the reverse complement removing option, and the second comparison is only between Fast-HBR, Nubeam-dedup [5] and Minirmd [11] while with reverse complement removing option is activated.

NGS Reads Treatment [9] with a different number of threads (16, 24, 32) was very slow and was not able to complete the processing of five datasets (SRR13555429, SRR13555395, SRR681003, SRR837669, SRR6424061) because it exceeds the limited time for the job which is 48 hours. Minirmd [11] consumes a huge amount of memory and it failed to complete the processing of four datasets (SRR13555429, SRR13555395, SRR837669, SRR6424061) because of a memory error. Moreover, Nubeam-dedup [5] was able to process all datasets except SRR6424061 when the reverse complement removing option is activated because of memory error. On the

ISSN 0973-2063 (online) 0973-8894 (print)
Bioinformation 18(1): 36-40 (2022)

other hand, Fast-HBR and BioSeqZip [10] were able to process all datasets successfully. We note that BioSeqZip [10] has the ability to limit the memory usage (default 4GB) and we try to increase its memory limit to 16GB, 32GB, and 64GB, but the tool failed to complete the process and cause a memory error, therefore, we run the tool with its default's memory limit.

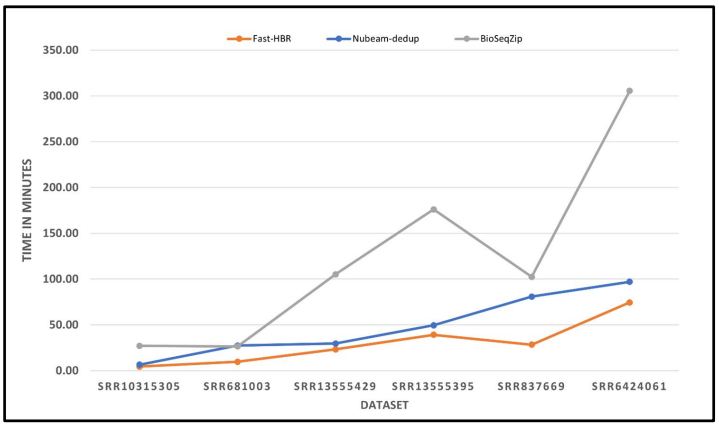

Figure 2: Processing time for the used datasets without reverse complement removing option.

Table 2 shows the number of removed reads in each dataset after applying the tools. Minirmd [11] was the tool that removed the smallest number of duplicated reads. On the other hand, the remaining tools were able to remove the same number of duplicated reads except for Nubeam-dedup [5] in one dataset (SRR6424061) where it considered a slightly a greater number of reads as duplicated reads. The results of the tools regarding CPU time and memory footprint are tabulated in **Table 3** and **Table 4**. **Table 3** shows the results when the tools applied on the datasets without the reverse complement removing option, where **Table 4** contains the results when the reverse complement removing option is activated.

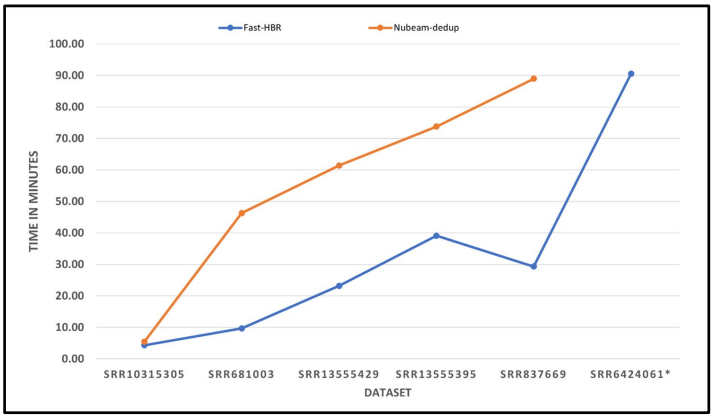

Figure 3: Processing time for the used datasets with reverse complement removing option. It should be noted that Nubeam-dedup was not able to complete processing SRR6424061 dataset.

Fast-HBR was the tool with the least CPU time in all single-end datasets in either case with or without reverse complement removing option. It was able to outperform the tool with the second least CPU time by a percentage that varies from 10% to 37%. In the paired-end datasets, Fast-HBR was the tool with the least CPU time in two of the three datasets and the outperform percentage in these two datasets varies from 23% to 67%. Generally, Fast-HBR was the tool with the least CPU time in ten out of twelve possible cases of processing datasets. Finally, the processing time for the tools when reverse complement is not activated is shown in **Figure 2** while **Figure 3** shows the processing time for the tools when reverse complement activated and here we should mention that NGS Reads Treatment [9] and Minirmd [11] are removed from the figures because they were not able to complete most of the datasets.

BioSeqZip [10] consume almost the same memory amount in all datasets because of its memory limit control. Therefore, it has a smaller memory footprint than Fast-HBR in all datasets except SRR10315305. If we exclude BioSeqZip [10] because it caused memory error when we try to increase the memory limit, Fast-HBR consumes the least memory in all paired-end datasets with or without the reverse complement removing option. Moreover, when the reverse complement removing option is activated, Fast-HBR has the least memory footprint while processing all datasets. By comparing each tool's memory consumption when the reverse complement is not active (**Table 3**) and when the reverse complement is activated (**Table 4**), we noted that the amount of memory used by the Fast-HBR is almost unchanged whether the reverse complement option is enabled or not. On the other hand, when the reverse complement option is enabled the memory footprint of Nubeam-dedup [5] almost doubled.

Conclusion:

We describe a *de novo* tool named Fast-HBR to remove duplicated reads in the meta-genomics data to reduce the dataset size which will benefit the meta-genomics analyzing pipelines. Fast-HBR represents each read to a single integer value by using hashing algorithms and hash tables for memory efficiency and speed. Fast-HBR shows the least computational requirement in validation. The CPU time required by it was less than the second-best tool Nubeam-dedup [5] by at least 10% and up to 67%. Moreover, Fast-HBR is the least memory consumption tool in all paired-end datasets using the reverse complement removing option.

Acknowledgements:

The authors would like to thank Deanship of scientific research in King Saud University for funding and supporting this research through the initiative of DSR Graduate Students Research Support (GSR).

Conflict of Interest:

The authors declare no conflict of interest.

References:

[1] Mitchell AL *et al. Nucleic Acids Research* 2018 **46:**D726 [PMID: 29069476].

[2] Mitchell AL *et al. Nucleic Acids Research* 2020 **48:**D570 [PMID: 31696235].
[3] Exposito RR *et al. Bioinformatics* 2017 **33:**2762 [PMID: 28475668].
[4] Manconi A *et al. BMC Bioinformatics* 2016 **17:**346 [PMID: 28185553].
[5] Dai H & Guan Y, *Bioinformatics* 2020 **36:**3254 [PMID: 32091581].
[6] Li W & Godzik A, *Bioinformatics* 2006 **22:**1658 [PMID: 16731699].
[7] Xu H *et al. PloS One* 2012 **7**:e52249 [PMID: 16731699].
[8] Burriesci MS *et al. Bioinformatics* 2012 **28:**1324 [PMID: 23284954].
[9] Gaia ASC *et al. Scientific Reports* 2019 **9:**1 [PMID: 31406180].
[10] Urgese G *et al. Bioinformatics* 2020 **36:**2705 [PMID: 31999333].
[11] Liu Y *et al. Bioinformatics* 2021 **37:**1604 [PMID: 33112385].
[12] Pagh R & Rodler FF, *Journal of Algorithms* 2004 **51:**122–144.
[13] Dai H & Guan Y, *BioRxiv* 2019 763631.
[14] Roberts M *et al. Bioinformatics* 2004 **20:**3363. [PMID: 15256412]