BIOINFORMATICS ORIGINAL PAPER

Vol. 28 no. 21 2012, pages 2732–2737
doi:10.1093/bioinformatics/bts482

Sequence analysis

Advance Access publication September 1, 2012

Rainbow: an integrated tool for efficient clustering and assembling RAD-seq reads

Zechen Chong[1,2,†], Jue Ruan[1,†,*] and Chung-I. Wu[1,3,*]

[1]Laboratory of Disease Genomics and Individualized Medicine, Beijing Institute of Genomics, Chinese Academy of Sciences, Beijing 100029, People's Republic of China, [2]Graduate University of Chinese Academy of Sciences, Beijing 100049, People's Republic of China and [3]Department of Ecology and Evolution, University of Chicago, Chicago, IL 60637, USA

Associate Editor: Michael Brudno

ABSTRACT

Motivation: The innovation of restriction-site associated DNA sequencing (RAD-seq) method takes full advantage of next-generation sequencing technology. By clustering paired-end short reads into groups with their own unique tags, RAD-seq assembly problem is divided into subproblems. Fast and accurately clustering and assembling millions of RAD-seq reads with sequencing errors, different levels of heterozygosity and repetitive sequences is a challenging question.

Results: Rainbow is developed to provide an ultra-fast and memory-efficient solution to clustering and assembling short reads produced by RAD-seq. First, Rainbow clusters reads using a spaced seed method. Then, Rainbow implements a heterozygote calling like strategy to divide potential groups into haplotypes in a top–down manner. And along a guided tree, it iteratively merges sibling leaves in a bottom–up manner if they are similar enough. Here, the similarity is defined by comparing the 2nd reads of a RAD segment. This approach tries to collapse heterozygote while discriminate repetitive sequences. At last, Rainbow uses a greedy algorithm to locally assemble merged reads into contigs. Rainbow not only outputs the optimal but also suboptimal assembly results. Based on simulation and a real guppy RAD-seq data, we show that Rainbow is more competent than the other tools in dealing with RAD-seq data.

Availability: Source code in C, Rainbow is freely available at http://sourceforge.net/projects/bio-rainbow/files/

Contact: ruanjue@gmail.com

Received on February 29, 2012; revised on July 3, 2012; accepted on July 26, 2012

1 INTRODUCTION

Genetic markers are polymorphic DNA sequences distributed along the chromosomes. They are important for studying population genetics, molecular evolution and inherited diseases. Traditionally, genetic markers include restriction fragment length polymorphisms, simple sequence length polymorphisms, amplified fragment length polymorphisms, microsatellite polymorphisms, single-nucleotide polymorphisms (SNPs), short tandem repeats, etc. (http://en.wikipedia.org/wiki/Genetic_marker). The development of these genetic markers is usually a costly, laborious and time-consuming work and could not easily be parallelized (Davey *et al.*, 2011). The advent of next-generation sequencing (NGS) technology enables sequencing many millions of reads with plummeting cost. Methods for discovering and genotyping genetic markers based on NGS are rapidly springing up. These include reduced-representation libraries, complexity reduction of polymorphic sequences (CRoPS) and restriction site associated DNA sequencing (RAD-seq); for a good review of these methods, see Davey *et al.* (2011). They are very useful for population genetic/genomic studies, especially when the reference genomes are unknown.

RAD-seq could generate a genome-wide density of genetic markers, suggesting a broader application than the other methods. RAD method was initially invented by Michael Miller (Lewis *et al.*, 2007; Miller *et al.*, 2007a, b) using microarrays. Then, high-throughput RAD-seq was developed as an efficient method to identify genetic markers (Baird *et al.*, 2008; Davey and Blaxter, 2011). RAD-seq has been successfully used to discovery SNPs and study adaptive evolution in the stickback (Baird *et al.*, 2008; Hohenlohe *et al.*, 2010), to resolve genetic structure and direction of evolution in the pitcher plant mosquitoes (Emerson *et al.*, 2010), to infer phylogenies (Rubin *et al.*, 2012) and to study interesting traits of butterfly (The Heliconius Genome Consortium, 2012).

As illustrated in Figure 1, the protocol of RAD-seq can be summarized as first digesting the genomic DNAs using restriction enzymes (REs), then the digested restriction fragments are randomly sheared into staggered sequences and sizes suitable for sequencing, e.g. on Illumina Genome Analyzer platform, are selected and amplified using PCR and last by sequencing the ends of selected segments, RAD-seq data are ready for downstream analysis (Baird *et al.*, 2008). The sequenced ends near the restriction site are called RAD tags or 'Tagged' reads and the other ends are '2nd' ends. At a particular restriction site, the Tagged reads contain identical information. Thus, by grouping the RAD Tagged reads and local assembling of the corresponding ends (2nd reads), contiguous sequences of ~600 bp (limited by the insert size of library of the sequencing platform) that represent parts of the genome are generated. Moreover, the Tagged reads can be barcoded, enabling pooled individuals are sequenced in a single reaction. For examples, Etter *et al.* (2011)

*To whom correspondence should be addressed.

†The authors wish it to be known that, in their opinion, the first two authors should be regarded as joint First Authors.

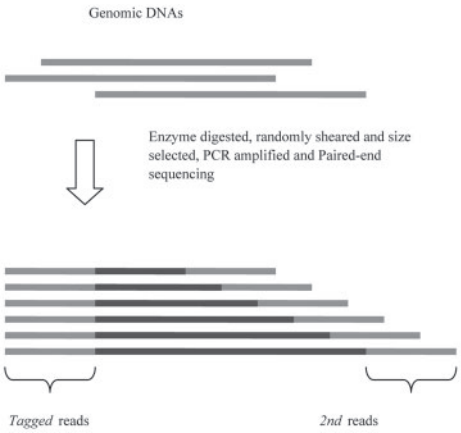

Fig. 1. An illustration of RAD-seq method. Genomic DNAs are digested using at least one enzyme and then randomly sheared into diverse insert sizes. Thus, sequencing the ends of these sequences could generate the 'Tagged' reads with same content (near the enzyme site) and the 2nd reads. By grouping the 'Tagged' reads and assembling 2nd reads, a local contiguous sequence (contig) is generated

and Willing *et al.* (2011) used an enzyme to digest the genomic DNAs of stickleback and guppy, and by locally assembling RAD tags, they tried to identify SNPs and define putative haplotypes in populations.

Although these methods show potential application for local assembly and downstream population genomic and molecular evolution analysis, it lacks an efficient and integrated tool to fulfill the prerequisite clustering the identical Tagged reads and local assembly work. We show here that clustering is the most critical and challenging problem, in that the number of reads to be clustered is huge and they may contain sequencing errors. In addition, the levels of heterozygosity and/or repetitive sequences may be high. A direct hashing method (Hiatt *et al.*, 2010) could quickly group the Tagged reads but cannot tolerate the inherent sequencing errors. Pairwise alignment for all short reads (Willing *et al.*, 2011) could overcome this weakness, but the cost of computation is extremely high.

To avoid the cost of pairwise alignments, heuristic methods that filter out unnecessary comparison of distant sequences are introduced. These methods include CD-HIT (Li and Godzik, 2006), UCLUST (Edgar, 2010) and DNAclust (Ghodsi *et al.*, 2011). All these tools employ a similar *k*-mer based method, i.e. the sequences are represented as a table containing the number of the occurrence of all substrings of length *k*, and insufficient *k*-mer number between two sequences will not be compared (Li and Godzik, 2006; Edgar, 2010; Ghodsi *et al.*, 2011). They are developed for general clustering analysis, but may not be well suitable for NGS data. Recently, SEED (Bao *et al.*, 2011) used a block spaced seed approach to fast clustering NGS reads while tolerating up to three mismatches. It demonstrated the potential usefulness of spaced seed method for clustering huge amounts of next-generation short reads. Another set of algorithms such as Vmatch (Abouelhoda *et al.*, 2004) and BWA (Li and Durbin, 2009) using suffix array to index sequences first, and by backtracking in the indexed structure, similarity between reads can be obtained. Shimizu and Tsuda (2011) developed an exact algorithm SlideSort, which used a recursive pattern growth method to find chains of *k*-mers and arbitrary edit distance could be calculated.

However, these methods may not solve the RAD-seq clustering problem properly. First, the customized similarity threshold is subtle. A higher level of mismatches could collapse too many similar sequences into one cluster, which could cause wrong assembly and will overestimate the diversity of the species. A lower level of mismatches splits one cluster into many smaller ones, which will generate redundant contigs or cannot be correctly assembled because of too few reads could be used. One goal in RAD-seq is to collapse heterozygote while distinguish repeats. Second, these methods can use the information of only the Tagged reads, while the 2nd ends are deserted. The Tagged reads usually consist of barcode, RE site and the RAD tag. Barcodes and RE sites and/or low-quality sequence at the tail must be removed before clustering. Although NGS reads are evolving longer and longer, the total length for clustering is limited. Given the paired-end reads of RAD-seq, should we use both of the ends, which indeed uses longer sequence information. Third, they are either not time efficient or not space efficient and combining with other tools to proceed subsequent analysis might be tough for biologists who are new to bioinformatics.

To achieve these goals, we demonstrate a package called Rainbow, which uses a spaced seed hashing method for primary clustering, a heterozygote calling strategy for further dividing clusters, a bottom–up merging module using paired-end information of RAD-seq and a greedy assembly method for local assembly all at one time. Rainbow can quickly and accurately cluster and assemble the Tagged reads and their 2nd ends with a low footprint of memory.

2 METHODS

2.1 Overview of Rainbow algorithms

The aim of Rainbow is to cluster RAD paired reads into groups, which should come from its unique location in the genome and locally assemble clustered reads and their paired 2nd ends into contigs representing corresponding genomic DNA segments. The implementation of Rainbow mainly consists of the following steps:

(a) Index all Tagged reads using spaced seeds.

(b) Cluster all relative Tagged reads having no more than a given number of mismatches with any other read into one group.

(c) Recursively divide clusters from the previous step until the minor bases of all sites in the cluster are likely to be sequencing errors.

(d) Merge similar clusters (come from the same locus) using paired ends along the guided tree generated by Step (c) in a bottom–up manner.

(e) Assemble the final divided reads and their 2nd partners into contigs using a greedy assembly method.

2.2 Spaced seed hash indexing strategy

First, we index all Tagged reads using spaced seed hash tables. The spaced seed is a non-contiguous seed template for hash indexing, introduced by Ma *et al.* (2002). For example, a template '1011011' indicates that all bases at '1's are indexed, while at '0's are not. The number of '1's

Fig. 2. The illustration of spaced seed strategy. Six templates with a seed length of 32 and a weight of 16 are used. This combination could ensure full search of reads within two mismatches in the 32 bp

in the template is called 'weight'. The use of spaced seeds can significantly increase sensitivity without sacrificing speed (Ma *et al*., 2002).

In this article, we employ a hash indexing procedure like Eland (A.J. Cox, unpublished results), MAQ (Li *et al*., 2008) and SOAP (Li *et al*., 2008). For example, we can index the first 32 bp of each read using six templates (see Figure 2). Using these templates, we are guaranteed to find any pair of reads within two mismatches in their first 32 bp. In Rainbow, we used two sets of spaced seed templates that guarantee to find two or five mismatches between a pair of reads to deal with low heterozygous genomes and normal or high heterozygous genomes, respectively.

2.3 Primary clustering algorithm

The purpose of primary clustering step is to cluster reads indexed in previous step into groups. In principle, we try to put all connective reads in one group. Here, the connectivity is defined as two reads have no more than a certain number of mismatches. Thus, the primary clustering is guaranteed to group all potential 'true' clusters correctly.

To reduce memory, we iteratively process each template instead of loading all templates at once. For each template, we encode each read at position '1's of template into integers. The reads with the same encoded integer are grouped into a list. For the reads in the list, all pairwise alignments are executed without gaps. During alignment, a Hamming distance is calculated. Two reads within a maximum distance are grouped as one cluster.

This sort of clustering may overrepresent the true clusters. For example, given a maximum number of mismatches, *m*, if read A and read B have *m* mismatches and read A and read C also have m mismatches, A, B and C will be clustered together, but we cannot conclude that read B and read C are within *m* mismatches. So we introduced the dividing procedure.

2.4 Top–down dividing clusters

The primary clusters might mix reads from repeats and heterozygous alleles together. The dividing module is introduced to distinguish sequencing errors from heterozygote or variants between repetitive sequences.

The dividing procedure used in Rainbow resembles heterozygote calling. For each primary cluster, reads are piled up. We scan across the entire reads. For every position of the reads, we calculate the count of minor base *K* and its frequency *F*. We recursively divide the cluster at the most significant site with the largest number of *K* until *K* and *F* are lower than the minimum requirement. Minor bases still present in final clusters are considered as sequencing errors. A guided tree is generated during dividing to record the relationship of final divided clusters.

The dividing procedure could generate either true clusters or split clusters with separated heterozygous sites. In other words, each cluster may represent a haplotype.

2.5 Bottom–up merging process

In RAD-seq application, heterozygote should usually be collapsed to define genetic markers. Besides, repetitive sequences should be as distinguishable from each other as possible. Because collapsed repeats could overestimate the diversity of the population.

To collapse heterozygote as well as to discriminate repetitive sequences as far as possible, we merge potential heterozygous clusters in a bottom–up manner.

Indeed, the recursive dividing procedure has generated a guided tree. The leaves of the tree are the finally divided clusters. If two leaves are brothers and are similar enough, then we merge the two leaves to generate a new leaf at their parent node and delete the two leaves.

Since the Tagged reads usually contain barcode information and restriction site sequences, the available sequences may not unambiguously distinguish heterozygote from repetitive sequences. To take full advantage of RAD-seq property, we use the 2nd reads to define the similarity between two brother leaves. Comparing 2nd reads is in deed comparing much longer segments. The longer the sequence, the better it could distinguish from each other.

In Rainbow, when comparing the similarity between two leaves, we index the leaf with relative larger number of reads using a sliding *k*-mer into a hash table and then using the leaf with smaller number of reads and splitting the reads into sliding *k*-mers to query against the hash table. If a threshold number of reads hit the hash table, we regard the two leaves as similar and merge them.

2.6 Local assembly of final clusters

Rainbow employs a greedy algorithm to fulfill the assembly step. Both the Tagged reads and the 2nd reads are pooled. Then, all pairwise alignments are performed. This computation is feasible, since the number of reads in a local assembly is small. Next, given a threshold of overlap length and overlap similarity, two reads or contigs with the largest overlap are chosen and merged into one consensus sequence. This process is repeated until no more reads can be merged. Rainbow reports not only the optimal contig but also suboptimal ones.

2.7 Parameter setting for evaluated clustering softwares

To evaluate the performance of Rainbow, we compared Rainbow with several state-of-the-art tools. For clustering comparison, we simulated three levels of heterozygosity using the 'Human' genome reference. We added DNAclust, SEED, SlideSort and Vmatch for reference. The parameters are listed in Table 1.

3 RESULTS

3.1 Rainbow performance on simulation data

To evaluate the performance of Rainbow, we simulated three datasets with three levels of heterozygosity (Table 2) on Human genome reference (hg19) digested with EcoRI (GAATTC). There are 778 882 RAD sites recognized by EcoRI on the *Human* genome. By using a modified version of 'wgsim' (Li *et al*., 2009), paired-end read sequences of length 100 bp were simulated from wide-range insert size libraries initiated from 120 bp and elongation of 10 steps (each step extends 50 bp). For each step, we simulated a mean depth of 5× of paired-end reads. And a 0.5% sequencing error rate was randomly introduced. Thus, an expected coverage of 482 906 840 bp or a representative of RAD-seq library cover 15.6% of the Human genome reference is simulated. All the simulation and subsequent analysis were performed on an X86 64 bit linux platform with 32 2.13 GHz CPUs and 512 GB main memory. But only one CPU core is used when evaluation.

To test the performance of Rainbow, we separated it into two parts: reads clustering and assembly of clustered reads. The clustering step is critical for subsequent assembly procedure. Hence, we first evaluated the clustering performance.

We compared Rainbow with DNAclust (Ghodsi *et al*., 2011), SEED (Bao *et al*., 2011), SlideSort (Shimizu and Tsuda, 2011) and Vmatch (Abouelhoda *et al*., 2004) to evaluate their performance for clustering on simulation data. The parameters of these programs are listed in Table 1. Similar to the gauge of SEED (Bao *et al*., 2011), we recorded the statistics of the number of clusters, the number of clusters identical with true ones, Jaccard index, false discovery rate (FDR), sensitivity and computing time and memory consumption. For two Clusters C1 and C2, the Jaccard index is defined as $|C1 \cap C2|/|C1 \cup C2|$. Its value lies between 0 and 1, and the higher value of Jaccard index indicates the better similarity of the two clusters.

Table 1. Tools and parameter used when benchmarking clustering

Dataset	programs	Parameter used (not listed using default)
NO.1	Rainbow	cluster -L, merge -p 0.95 -N500
	DNAclust	-s 0.98
	SEED	–mismatch 2 –shift 0 –fast
	SlideSort	-d 2
	Vmatch	-d -l 100 -dbcluster 100 100 -h 1
NO.2	Rainbow	cluster -m6, merge -p 0.85 -N 500
	DNAclust	-s 0.97
	SEED	–mismatch 3 –shift 0 –fast
	SlideSort	-d 3
	Vmatch	-d -l 100 -dbcluster 100 100 -h 2
NO.3	Rainbow	merge -p 0.85 -N 500
	DNAclust	-s 0.96
	SEED	–mismatch 3 –shift 0 –fast
	SlideSort	-d 4
	Vmatch	-d -l 100 -dbcluster 100 100 -h 3

From Table 2, we can see that at low heterozygosity, all programs show similar level of Jaccard Index except DNAclust. But as the heterozygosity increases, Rainbow outperforms all the other programs. Besides, Rainbow almost shows exclusively the highest sensitivity and the lowest FDR, indicating that Rainbow performs well and robustly in dealing with different heterozygous conditions. Meanwhile, on efficiency, Rainbow is always faster and has a much smaller memory footprint than the other tools. The less memory occupation of Rainbow makes possible working on a desktop dealing with RAD-seq data for biologists.

To further evaluate Rainbow's performance for clustering, we locally assembled the reads in each clusters using the Rainbow assembler 'rbasm'. To compare Rainbow with the other tools fairly, we ran rbasm on the clustering results of each program. The N50 (the largest number above which the combined length is at least 50% of the total length of all contigs) and N90 of assembled contigs are quite consistent from 580 to 615 bp among these programs (data not shown), indicating that in most cases these programs could cluster reads accurately and rbasm performed well. We then selected the optimal (longest) assembled contigs and mapped the contigs to the reference using 'BWASW' (Li and Durbin, 2010). The alignments were processed using SAMtools (Li *et al*., 2009). Rainbow covers more genomic regions than the other tools, except DNAclust. It is interesting that although DNAclust cannot exceed the other programs in statistics such as sensitivity, Jaccard Index, it could cover more sites of the genome. From the statistics, we can see that DNAclust tends to cluster more groups with fewer reads. But in reality, this may cause redundant segments, since the similarity threshold is strict. When the depth is low or unevenly distributed, the clustered reads may not be assembled into better contigs.

Table 2. Comparison of clustering programs on simulation data (778 882 clusters and expected coverage: 482 906 840 bp)

Heterozygosity	Program	No. of clusters	No. of clusters identical with true ones	Jaccard Index	FDR	Sensitivity	Optimal contig cov	Optimal and suboptimal contig cov	Time (s)	Memory (MB)
0.001	Rainbow[a]	762 171	743 864	0.966	0.039	0.976	422 738 035	422 740 419	1453.0	2998.6
	DNAclust	829 961	237 920	0.819	0.059	0.884	425 198 931	426 761 646	180 245.4	36 918.7
	SEED	762 104	750 569	0.969	0.041	0.976	420 845 207	422 436 778	1955.3	36 901.3
	SlideSort	753 454	750 519	0.968	0.042	0.974	419 517 465	420 736 926	5018.6	9301.1
	vmatch	766 414	751 443	0.965	0.034	0.974	422 133 498	423 170 445	7421.6	42 194.7
0.010	Rainbow	770 800	736 691	0.953	0.043	0.972	423 389 515	423 395 671	2282.8	4134.2
	DNAclust	977 714	475 382	0.672	0.141	0.852	425 917 530	427 831 938	120 773.1	39 699.9
	SEED	772 433	723 702	0.941	0.053	0.964	419 373 406	421 269 473	1990.9	37 161.0
	SlideSort	767 922	733 347	0.939	0.058	0.961	417 533 066	418 978 003	6617.3	9366.6
	vmatch	818 189	693 952	0.869	0.075	0.939	420 296 809	421 693 200	21 441.8	42 191.3
0.020	Rainbow	784 585	728 876	0.937	0.044	0.968	424 769 039	424 773 322	2358.3	3002.5
	DNAclust	871 718	614 306	0.801	0.093	0.914	423 467 852	425 552 482	170 990.2	40 760.8
	SEED	855 889	626 431	0.815	0.091	0.919	420 936 857	423 051 365	2292.0	37 337.2
	SlideSort	788 787	707 516	0.895	0.079	0.943	416 080 003	417 817 238	9376.1	9398.6
	vmatch	860 882	646 658	0.797	0.107	0.908	418 962 111	420 661 755	51 390.8	21 993.7

[a]The time consumption for rainbow is the sum of clustering, dividing and merging procedures.

Table 3. Comparison of assembly results among Rainbow, Velvet and LOCASopt

Tool	No. of contigs	Max (bp)	Mean (bp)	N50 (bp)	N90 (bp)	covered bases
RainbowOpt	3273	697	562	603	478	1 840 989
RainbowSub	4208	697	471	601	351	1 984 010
VelvetOpt	3273	619	520	592	380	1 703 253
VelvetSub	4379	619	431	586	270	1 887 082
LOCASopt	4115	598	436	552	278	1 793 710

Rainbow has introduced a top–down dividing module and bottom–up merging process, which tries to collapse heterozygote and distinguish repetitive sequences. All the other tools lack this capability and they tend to collapse both heterozygous and repetitive sequences within the maximum mismatch threshold. If reads from two segments are grouped only due to the similarity of the Tagged reads, then the assembly should resolve the two segments. To demonstrate this, we not only selected the optimal contigs but also suboptimal ones. By mapping these contigs to the reference, we can see from Table 2 that the genomic sites covered by Rainbow contigs increase only slightly while other tools increase much more. But even if suboptimal contigs are chosen, optimal contigs of Rainbow usually cover more regions.

To test the assembly performance of Rainbow, we compared its assemblies with that of Velvet version 1.1.05 (Zerbino and Birney, 2008) and LOCASopt (http://ab.inf.uni-tuebingen.de/software/locas/) based on the final clusters from the Dataset 3, which has a heterozygosity level of 0.02 and is processed by Rainbow. The three programs represent state-of-the-art algorithms for *de novo* assembly, i.e. greedy (Rainbow), overlap-layout-consensus (LOCASopt) and De Bruijn graph (Velvet). We selected 3273 clusters as a demonstration. We used VelvetOptimiser (http://bioinformatics.net.au/software.velvetoptimiser.shtml) to run Velvet in order to get the best results. We used the parameter '*-s 11 -e 31 -f '-fasta -short input.txt' -d outputdir -v -t 8*' to run VelvetOptimiser. For LOCASopt, we used the parameter suggested in the article (Willing *et al*. 2011), i.e. overlap = 21, 23, . . . , 67, *k*-mer = 13, 15, 17 and mismatch rate = 0.05, 0.07, 0.09. After assembling, we selected both optimal contigs and optimal plus suboptimal contigs for comparison. We recorded the number of assembled contigs, their max and mean big lengths, N50, N90 and total covered bases. Table 3 implies that Rainbow outperforms Velvet and LOCASopt in terms of almost all statistics, which indicates that greedy algorithm is suitable for local assembly task in RAD-seq.

3.2 Rainbow performance on real data

We also tested Rainbow performance on real data. Willing *et al*. (2011) used RAD-seq assembly method to survey a less complex part of the guppy genome. They digested the genomes using the enzyme EcoRI and assembled 200–500 bp contigs to further perform population genetics analysis in the guppy populations. They provided a pipeline named RApiD to execute bioinformatical analysis. RApiD consists of several scripts. For the main clustering step, it calls Vmatch (http://www.vmatch.de) for pairwise alignment of RAD tags and then processes the alignment results to generate clusters. And during the assembling procedure, it calls LOCAS (http://ab.inf.uni-tuebingen.de/software/locas/) to assemble the second reads.

Table 4. Comparison of assembly result between Rainbow and RApiD

Method	No. of clusters	No. of contigs	Reads used	Mean (bp)	Sum (Mb)
Rainbow	313 728	324 909	57.0	375	121.7
RApiD-fixed	291 149	503 748	75.6	286	143.8
RApiD-opt	291 159	334 215	76.8	349	116.6

We downloaded the above data and tested the performance of Rainbow. We compared Rainbow results with RApiD. We used the default parameter in Rainbow, while RapiD used a fixed parameter and a set of parameters trying to obtain the optimal one suggested in the article. Indeed, we have spent 83.5 h to run the optimal assembly procedure of the pipeline RApiD and got a similar result as shown in their article. In contrast, it took only 3.4 h for Rainbow to accomplish clustering and *de novo* assembly on the same dataset. Here, we filtered the final clusters of Rainbow with at least 10 reads in each cluster. We summarized the assembly results as total number of clusters, total number of contigs, average reads used for a contig, the mean contig size and the total length of contigs for comparison (see Table 4).

From Table 4, we can see that Rainbow could generate longer clusters than RapiD. Because the guppy genome is not available, we could not map the contigs onto the reference. To evaluate the final assembly results, we adopted the method introduced by Willing *et al*. (2011). We downloaded 5223 guppy BAC ends from NCBI and predicted 910 RAD markers with length $\geq$150 bp using EcoRI (GAATTC). We mapped those markers to the assembly results of Rainbow and RApiD using BLAST (Altschul *et al*., 1990). 677 (74.4%) and 657 (72.2%) markers could match to Rainbow assembled contigs and RApiD references, respectively. 576 (85.1%) out of 677 hits for Rainbow and 515 (78.4%) out of 657 hits for RApiD could include the restriction site at one end and have identity over 90%. This indicates the results of Rainbow are as good as RApiD or even slightly better. The mean number of reads used to generate a contig in Rainbow is about 20 reads less than RApiD. We also note that the number of final contigs is much larger than the number of clusters for RApiD, but the corresponding numbers in Rainbow is nearly the same. This suggests that RApiD tended to collapse more sequences (repetitive sequences) while Rainbow only tried to collapses heterozygote. These results show that Rainbow is quite qualified for the analysis of RAD-seq data.

4 DISCUSSION

In this article, we present Rainbow as an efficient and general tool for RAD-seq short reads clustering and assembling. It could quickly group and assemble millions of short reads with a small memory footprint. In addition, it could tolerate a high

heterozygous genome. In particular, by using the top–down dividing and bottom–up merging procedures, we can collapse heterozygote while distinguish repetitive sequences. We have shown that it performs well both on simulation data and real data.

RAD-seq is a developing field and intriguing the interests of more and more people. As the evolution of NGS technology, we believe RAD-seq will improve a lot and become more important for biological research.

In the future, we will improve Rainbow in the following aspects. First, to improve the accuracy of clustering, base quality will also be incorporated into the dividing procedure. Besides, to be compatible with more applications, we will add a seed shifting strategy that SEED proposed to deal with overhanging bases, which could rescue reads due to irregular cuts. Third, although RAD-seq is mainly performed on the Illumina Genome Analyzer, the rapid evolving RAD-seq technology might spread to other platforms. We will prepare to let Rainbow support more platforms.

ACKNOWLEDGEMENTS

We are very grateful to Dr Heng Li and Dr Weiwei Zhai for providing many valuable suggestions on preparing the article. We would also like to thank the anonymous reviewers for their useful comments and suggestions for the work.

Funding: The National Nature Science Foundation of China (31000588).

Conflict of Interest: none declared.

REFERENCES

Abouelhoda,M.I. *et al.* (2004) Replacing suffix trees with enhanced suffix arrays. *J. Discrete Algorithms*, **2**, 53–86.

Altschul,S.F. *et al.* (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.

Baird,N.A. *et al.* (2008) Rapid SNP discovery and genetic mapping using sequenced RAD markers. *PloS One*, **3**, e3376.

Bao,E. *et al.* (2011) SEED: efficient clustering of next-generation sequences. *Bioinformatics*, **27**, 2502–2509.

Davey,J.W. *et al.* (2011) Genome-wide genetic marker discovery and genotyping using next-generation sequencing. *Nat. Rev. Genet.*, **12**, 499–510.

Edgar,R.C. (2010) Search and clustering orders of magnitude faster than BLAST. *Bioinformatics*, **26**, 2460–2461.

Emerson,K.J. *et al.* (2010) Resolving postglacial phylogeography using high-throughput sequencing. *Proc. Natl. Acad. Sci. USA*, **107**, 16196–16200.

Etter,P.D. *et al.* (2011) Local de novo assembly of RAD paired-end contigs using short sequencing reads. *PLoS One*, **6**, e18561.

Ghodsi,M. *et al.* (2011) DNACLUST: accurate and efficient clustering of phylogenetic marker genes. . *BMC Bioinformatics*, **12**, 271.

Hiatt,J.B. *et al.* (2010) Parallel, tag-directed assembly of locally derived short sequence reads. *Nat. Methods*, **7**, 119–122.

Hohenlohe,P.A. *et al.* (2010) Population genomics of parallel adaptation in three-spine stickleback using sequenced RAD tags. *PLoS Genet.*, **6**, e1000862.

Lewis,Z.A. *et al.* (2007) High-density detection of restriction-site-associated DNA markers for rapid mapping of mutated loci in Neurospora. *Genetics*, **177**, 1163–1171.

Li,H. and Durbin,R. (2009) Fast and accurate short read alignment with Burrows–Wheeler transform. *Bioinformatics*, **25**, 1754–1760.

Li,H. and Durbin,R. (2010) Fast and accurate long-read alignment with Burrows–Wheeler transform. *Bioinformatics*, **26**, 589–595.

Li,H. *et al.* (2009) The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, **25**, 2078–2079.

Li,H. *et al.* (2008) Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Res.*, **18**, 1851–1858.

Li,R. *et al.* (2008) SOAP: short oligonucleotide alignment program. *Bioinformatics*, **24**, 713–714.

Li,W. and Godzik,A. (2006) CD-HIT: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, **22**, 1658–1659.

Ma,B. *et al.* (2002) PatternHunter: faster and more sensitive homology search. *Bioinformatics*, **18**, 440–445.

Miller,M.R. *et al.* (2007a) RAD marker microarrays enable rapid mapping of zebrafish mutations. *Genome Biol.*, **8**, R105.

Miller,M.R. *et al.* (2007b) Rapid and cost-effective polymorphism identification and genotyping using restriction site associated DNA (RAD) markers. *Genome Res.*, **17**, 240–248.

Rubin,B.E. *et al.* (2012) Inferring phylogenies from RAD sequence data. *PloS One*, **7**, e33394.

Shimizu,K. and Tsuda,K. (2011) SlideSort: all pairs similarity search for short reads. *Bioinformatics*, **27**, 464–470.

The Heliconius Genome ConsortiumButterfly genome reveals promiscuous exchange of mimicry adaptations among species. *Nature*, advanced online publication.

Willing,E.M. *et al.* (2011) Paired-end RAD-seq for de novo assembly and marker design without available reference. *Bioinformatics*, **27**, 2187–2193.

Zerbino,D.R. and Birney,E. (2008) Velvet: algorithms for *de novo* short read assembly using de Bruijn graphs. *Genome Res.*, **18**, 821–829.