OPEN ACCESS Freely available online

PLOS ONE

A Rank-Based Sequence Aligner with Applications in Phylogenetic Analysis

Liviu P. Dinu[1,2☯], **Radu Tudor Ionescu**[1*☯], **Alexandru I. Tomescu**[3☯]

1 Faculty of Mathematics and Computer Science, University of Bucharest, Bucharest, Romania, **2** Personal Genetics, Bucharest, Romania, **3** Helsinki Institute for Information Technology HIIT, Department of Computer Science, University of Helsinki, Helsinki, Finland

Abstract

Recent tools for aligning short DNA reads have been designed to optimize the trade-off between correctness and speed. This paper introduces a method for assigning a set of short DNA reads to a reference genome, under Local Rank Distance (LRD). The rank-based aligner proposed in this work aims to improve correctness over speed. However, some indexing strategies to speed up the aligner are also investigated. The LRD aligner is improved in terms of speed by storing k-mer positions in a hash table for each read. Another improvement, that produces an approximate LRD aligner, is to consider only the positions in the reference that are likely to represent a good positional match of the read. The proposed aligner is evaluated and compared to other state of the art alignment tools in several experiments. A set of experiments are conducted to determine the precision and the recall of the proposed aligner, in the presence of contaminated reads. In another set of experiments, the proposed aligner is used to find the order, the family, or the species of a new (or unknown) organism, given only a set of short Next-Generation Sequencing DNA reads. The empirical results show that the aligner proposed in this work is highly accurate from a biological point of view. Compared to the other evaluated tools, the LRD aligner has the important advantage of being very accurate even for a very low base coverage. Thus, the LRD aligner can be considered as a good alternative to standard alignment tools, especially when the accuracy of the aligner is of high importance. Source code and UNIX binaries of the aligner are freely available for future development and use at http://lrd.herokuapp.com/aligners. The software is implemented in C++ and Java, being supported on UNIX and MS Windows.

Citation: Dinu LP, Ionescu RT, Tomescu AI (2014) A Rank-Based Sequence Aligner with Applications in Phylogenetic Analysis. PLoS ONE 9(8): e104006. doi:10.1371/journal.pone.0104006

Editor: Charles Y. Chiu, University of California, San Francisco, United States of America

Received December 11, 2013; **Accepted** July 9, 2014; **Published** August 18, 2014

Funding: The work of Alexandru I. Tomescu was supported by the Academy of Finland under grant 250345 (CoECGR). The work of Radu Tudor Ionescu was supported from the European Social Fund under Grant POSDRU/159/1.5/S/137750. The research of Liviu P. Dinu was supported by Personal Genetics. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript. There are no other funding sources for this study.

Competing Interests: Liviu P. Dinu is an employee of Personal Genetics, and received salary from them. There are no patents, products in development or marketed products to declare. This does not alter the authors' adherence to all PLOS ONE policies on sharing data and materials.

* Email: raducu.ionescu@gmail.com

☯ These authors contributed equally to this work.

Introduction

Novel high-throughput sequencing technologies generate up to several millions of short DNA reads (30 to 400 nucleotides long) from random locations in the genome. Putting together these reads into a coherent whole is a significant computational challenge. The first and most expensive step of this process is aligning each read to a known reference genome. Recently, many tools designed to align short reads have been proposed [1]. Sequence alignment tools are designed to optimize the trade-off between correctness and speed, usually sacrificing correctness over speed. This leaves room for new tools for sequence alignment that can better satisfy one of (or both) the two needs, namely efficiency and accuracy. With broad applications from phylogenetic analysis to finding motifs or common patterns in a set of given DNA sequences, new alignment tools are of great interest for the entire community of computational biology researchers.

This paper proposes a method for assigning a set of short DNA reads to a reference genome, under Local Rank Distance (LRD) [2]. Local Rank Distance is an extension of rank distance [3] that is designed to work on overlapping k-mers instead of single characters as rank distance. Despite the fact that LRD was only recently introduced, it has already demonstrated its performance in phylogenetic analysis [2] and native language identification [4].

The rank-based sequence aligner works as follows. Given a set of reads that need to be aligned against a reference genome, the aligner determines the position of each read in the reference genome that gives the minimum Local Rank Distance. The proposed aligner will be referred to as the *LRD aligner* through the rest of this paper. Some strategies of optimizing the search for the best positions of reads are also proposed and investigated. The LRD aligner is improved in terms of speed by storing k-mer positions in a hash table for each read. An approximate LRD aligner that works even faster is obtained through the following strategy. The approximate aligner considers only the positions in the reference that are likely to give the minimum distance, by previously counting the number of k-mers from the read that can be found at every position in the reference.

The LRD sequence aligner is designed to work with genomic data produced by Next-Generation Sequencing technologies. These high-throughput technologies are able to produce up to 200 million DNA reads of length between 30 and 400 base pairs in a single experiment. Despite this abundance of reads, their short

length makes the problem of assembling them into the originating genome a difficult one in practice. Therefore, methods for finding the class, the order, the family or even the species of an unknown organism, given only a set of short Next-Generation Sequencing DNA reads originating from its genome, are of interest. A method that can be used to solve this phylogenetic analysis task is proposed in this work. The method works as follows: given a collection $\mathcal{R}$ of short DNA reads, and a collection $\mathcal{G}$ of genomes, it finds the genome $G \in \mathcal{G}$ that gives a minimum score. This method serves two purposes. First, the method can be used to determine the place of an individual in a phylogenetic tree, by finding the most similar organism in the phylogenetic tree. This can be achieved by using only a set of short DNA reads originating from the genome of the new individual. Second, the method is used to evaluate the performance level of the rank-based aligner and to compare it with other state of the art alignment tools, such as BWA [5], BOWTIE [6], or BLAST [7]. Experimental results on simulated reads were obtained under two scenarios: low and high error rate. In the former scenario, all the aligners besides BWA have full precision. In the latter scenario, the LRD aligner is the only one that attains full precision. It seems that the LRD aligner gives the most accurate results, while being more computationally expensive than the other aligners.

A set of experiments are conducted to determine the precision and the recall of the proposed LRD aligner, in the presence of contaminated reads. The task is to align reads sampled from several mammals on the human mitochondrial DNA sequence genome. The goal is to maximize the number of aligned reads sampled from the human genome (true positives), and to minimize the number of aligned reads sampled from the other mammals (false positives). Again, the LRD aligner seems to have the best performance, followed closely by BOWTIE and BLAST.

The proposed aligner is also tested on three human vibrio pathogens with results that point towards the same conclusion of [8,9]. In all the experiments presented in this work, the rank-based aligner shows results that are better than the state of the art alignment tools, in terms of accuracy. The results obtained in this work can be considered as a strong argument in favor of using rank-based distance measures for computational biology tasks, in order to obtain results that are more accurate from a biological point of view.

It is important to point out that the main focus of the experiments is on the alignment accuracy of the aligner based on LRD. Therefore, the simple strategy of assigning each read to the genomic sequence with the best LRD distance was used. However, in other biological problems, these alignments can be fed to other more elaborate methods. For example, in profiling bacterial species from a metagenomics sample, various tools, such as the MG-RAST server [10], MEGAN [11] and metaBEETL [12], align the reads to a reference taxonomy, but report as hit the Lowest Common Ancestor node of a set of significant hits in this taxonomic tree.

Related Work

Similarity Measures Between Genomes. Since most DNA variations between organisms of the same species consist of point mutations like single nucleotide polymorphisms, or small insertions or deletions, edit distance is the standard string measure in many biomedical analyses, such as the detection of genomic variation, genome assembly [13], identification and quantification of RNA transcripts [14–16], identification of transcription factor binding sites [17], or methylation patterns [18].

In the case of genomic sequences coming from different related species, other mutations are present, such as reversals [19], transpositions [20], translocations [21], fissions and fusions [22]. For this reason, there have been a series of different proposals of similarity between entire genomes, including rearrangement distance [23], k-break rearrangements [24], edit distance with block operations [25].

Some of the other popular distance measures for recent computational biology techniques are the Hamming distance [26,27] and the Kendall-tau distance [28], among others [29]. Rank distance [3] is another such measure of similarity, having low computational complexity, but high significance in phylogenetic analysis [30,31] and in finding common patterns in DNA sequences [32].

Sequence Aligners. One of the most widely used computational biology programs is BLAST [7]. Compared to the previously developed techniques based on dynamic programming [33], BLAST increases the speed of alignment by reducing the search space. An interesting remark is that BLAST calculates the statistical significance for each sequence alignment result.

While BLAST remains an essential tool for biologists, the vast amount of data produced by the high-throughput sequencing technologies led to the development of faster and more accurate sequence aligners. Recently, many tools designed to align short reads have been proposed [1]. The main efforts in the design of such tools are on improving speed and correctness. Fast tools are needed to keep the pace with data production, while the number of correctly placed reads is maximized. Usually tools sacrifice correctness over speed, allowing only few mismatches between the reads and the reference genome. Tools that optimize such trade-off are BOWTIE [6] and BWA [5]. Both the BWA and the BOWTIE aligners work under the edit distance, and they use the Burrows-Wheeler Transform to efficiently align short reads against a large reference sequence, allowing mismatches and gaps. The BOWTIE2 aligner [34] combines the full-text minute index with the flexibility of hardware-accelerated dynamic programming algorithms to achieve both speed and accuracy.

The BFAST [35] tool moves towards favoring correctness over speed, allowing alignments with a high number of mismatches and indels. Another accurate tool able to align reads in the presence of extensive polymorphisms, high error rates and small indels, is rNA [27]. The experiments performed in [27] give an idea about the different approaches of such tools for optimizing the trade-off between correctness and speed. For example, in one experiment BWA is 100 times faster than BFAST, while losing about 8.00% in terms of accuracy.

Results

Data Sets

To evaluate the aligners proposed in this work, several experiments are conducted on two data sets of genome sequences. The first data set contains mitochondrial DNA sequence genomes of 20 mammals. The genomes are available for download in the EMBL database (http://www.ebi.ac.uk/ena/) using the accession numbers given in Table 1. They belong to the following biological orders: Primates, Perissodactylae, Cetartiodactylae, Rodentia, Carnivora.

Mitochondrial DNA (mtDNA) is the DNA located in organelles called mitochondria. The DNA sequence of mtDNA has been determined from a large number of organisms and individuals, and the comparison of those DNA sequences represents a mainstay of phylogenetics, in that it allows biologists to elucidate the evolutionary relationships among species. In mammals, each double-stranded circular mtDNA molecule consists of 15,000 to 17,000 base pairs. DNA from two individuals of the same species

Table 1. The 20 mammals from the EMBL database used in the phylogenetic experiments. The accession number is given on the last column.

Mammal	Latin Name	Accession No.
human	*Homo sapiens*	V00662
common chimpanzee	*Pan troglodytes*	D38116
pigmy chimpanzee	*Pan paniscus*	D38113
gorilla	*Gorilla gorilla*	D38114
orangutan	*Pongo pygmaeus*	D38115
Sumatran orangutan	*Pongo pygmaeus abelii*	X97707
gibbon	*Hylobates lar*	X99256
horse	*Equus caballus*	X79547
donkey	*Equus asinus*	X97337
Indian rhinoceros	*Rhinoceros unicornis*	X97336
white rhinoceros	*Ceratotherium simum*	Y07726
harbor seal	*Phoca vitulina*	X63726
gray seal	*Halichoerus grypus*	X72004
cat	*Felis catus*	U20753
fin whale	*Balaenoptera physalus*	X61145
blue whale	*Balaenoptera musculus*	X72204
cow	*Bos taurus*	V00654
sheep	*Ovis aries*	AF010406
rat	*Rattus norvegicus*	X14848
mouse	*Mus musculus*	V00711

doi:10.1371/journal.pone.0104006.t001

differs by only 0.1%. This means, for example, that mtDNA from two different humans differs by less than 20 base pairs. Because this small difference cannot affect the study, the experiments are conducted using a single mtDNA sequence for each mammal.

The second data set contains chromosomal DNA sequence genomes of three vibrio pathogens available in the NCBI database (http://www.ncbi.nlm.nih.gov): *Vibrio vulnificus* YJ106, *Vibrio parahaemolyticus* RIMD 2210633, and *Vibrio cholerae* El Tor N16961. The genomes of these three organisms consist of two circular chromosomes. Additional information about these chromosomes, including accession number and size (given in Megabase pairs), is given in Table 2. The genomic sequences of these vibrio species have been revealed by different studies [9,36,37]. Several studies report that Vibrio vulnificus shares morphological and biochemical characteristics with other human vibrio pathogens, including Vibrio cholerae and Vibrio parahaemolyticus [8,9].

Alignment in the Presence of Contaminated Reads

In this experiment, reads sampled from the genomes of several mammals are aligned on the human mtDNA sequence genome. The reads were simulated with the *wgsim* tool [38], using the default parameters. More precisely, the reads were generated using an error rate of 0.02, a mutation rate of 0.001, a fraction of indels of 0.15 (out of the total number of mutations) and a probability of extending an indel of 0.30.

The LRD aligner is compared to the BWA, the BOWTIE2 and the BLAST aligners, under two different scenarios. In the first scenario, 10,000 contaminated reads are sampled from the orangutan genome. In the second scenario, 50,000 contaminated reads are sampled from 5 mammals, namely the orangutan, the blue whale, the harbor seal, the donkey, and the house mouse. There are actually 10,000 reads sampled from each of the 5 mammals. In both scenarios 10,000 reads simulated from the human genome are included. The simulated reads are always 100

Table 2. The genomic sequence information of three vibrio pathogens consisting of two circular chromosomes.

Species	Chromosome	Accession No.	Size (Mbp)
V. vulnificus YJ016	I (VV1)	NC_005139	3.4
V. vulnificus YJ016	II (VV2)	NC_005140	1.9
V. parahaemolyticus RIMD 2210633	I (VP1)	NC_004603	3.3
V. parahaemolyticus RIMD 2210633	II (VP2)	NC_004605	1.9
V. cholerae El Tor N16961	I (VC1)	NC_002505	3.0
V. cholerae El Tor N16961	II (VC2)	NC_002506	1.0

doi:10.1371/journal.pone.0104006.t002

bases long. The goal is to maximize the number of aligned reads sampled from the human genome (true positives), and to minimize the number of aligned reads from the other mammals (false positives). Unlike the other experiments presented in this paper, reverse complement reads were not included in this experiment. However, it is important to mention that the aligners are dealing with a hard task, since the contaminated reads were sampled only from organisms that are in the same class as the human. It may be that contaminated reads from other species that are not in the Mammalia class (such as viruses, for example) can be identified and discarded more easily.

The parameters of the aligners were adjusted as described next. For the BOWTIE2 aligner, two variants are evaluated. The first one uses the *local* and the *very-sensitive-local* options. The second variant uses the *end-to-end* and the *very-sensitive* options. For the BLAST aligner, the *megablast* option is used. Two variants of the LRD aligner based on 3-mers and a maximum offset between paired 3-mers of 36 are also evaluated. One is based on the exact search algorithm, while the other one uses the approximate algorithm based on hash tables that runs much faster.

To evaluate and compare the aligners, the precision and recall curve is used. Note that the precision is given by the proportion of aligned reads that are positive, while the recall is given by the proportion of true positive reads that are aligned. In order to obtain the precision-recall curve for each aligner, the idea is to vary the threshold that gives the maximum distance allowed for an aligned read. In the case of the BWA and the BOWTIE aligners, the edit distance threshold takes values from 0 to 30. The score of the BLAST aligner ranges from 185 to 100. The LRD threshold takes values from 50 to 600, for both variants of the LRD aligner. Higher precision is obtained for lower distance thresholds, while higher recall is obtained for higher distance thresholds. The only aligner that works the other way around, and gives higher precision for higher scores, and higher recall for lower scores, is the BLAST aligner.

Several statistical measures, such as the Area Under the ROC Curve (AUC), the F_1 measure, and the F_2 measure, are also presented in order to better compare the aligners. The ROC curve plots the fraction of true positive reads versus the fraction of false positive reads, at various threshold settings. The AUC score represents the area under the ROC curve. The F_1 measure (also known as the F_1 score) can be interpreted as a weighted average of the precision and recall at a certain distance threshold. The F_2 measure is similar to the F_1 measure, only that it weights recall higher than precision. For each aligner, the highest F_1 and F_2 scores can indicate the thresholds that give a good trade-off between precision and recall. The F_β measure is computed as follows:

$$F_\beta = (1+\beta^2)\frac{precision \cdot recall}{\beta^2 \cdot precision + recall}. \quad (1)$$

The F_1 and the F_2 scores are immediately obtained from Equation 1, by replacing β with 1 and 2, respectively.

Human versus Orangutan Experiment. In this experiment, there are 20,000 reads to be aligned on the human mtDNA sequence. Half of them are sampled from the same human mitochondrial genome, while the other half are sampled from the orangutan mitochondrial genome. Thus, the contamination rate is 50%.

The precision-recall curves of the BWA, the BOWTIE, and the BLAST aligners together with the precision-recall curves of the two variants of the LRD aligner are presented in Figure 1. By analyzing Figure 1, it can be observed that the aligners obtain roughly similar results in terms of precision and recall. To better assess the performance of the evaluated aligners, the AUC measure and the best F_1 and F_2 scores for each aligner are presented in Table 3. In terms of the AUC, the BOWTIE and the LRD aligners attain the best results, while the other aligners fall behind. In terms of the F_1 measure, the BOWTIE aligner seems to be slightly better than the LRD aligner, while in terms of the F_2 measure, the LRD aligner achieves the best score, followed closely by the BOWTIE aligner. The BLAST aligner comes in third place after the LRD and the BOWTIE aligners. The results of the BWA aligner are also not too far from the other top scoring aligners.

The results presented in Figure 1 indicate that all the aligners obtain a good trade-off between precision and recall. Indeed, all of them are able to align more than 90% of the human reads with a precision that is higher than 90%. For instance, the hash LRD aligner is able to align 98.6% of the humand reads with 97.02% precision. However, it would be interesting to observe how the LRD aligner behaves at the sequence level. For this purpose, some metrics of the reads simulated from the human mitochondrial genome are provided in Table 4. More precisely, the average Hamming distance and the average edit distance of the human reads that are mapped to the human genome (true positives) are reported at different precision and recall levels. In the same time, the average Hamming distance and the average edit distance of the human reads that are not mapped to the human genome (false negatives) are also reported. Perhaps it would be more interesting to give the average number of errors and mutations in the true positive reads versus the average number of errors and mutations in the false negative reads. Unfortunately, the *wgsim* tool does not output these values for the simulated reads. Nevertheless, the simulation tool does output the exact location from which each read was simulated. Therefore, a standard distance can be computed between a simulated read and its corresponding original substring (of 100 bases) from the human genome, that was used by *wgsim* to generate the read. The Hamming distance and the edit distance together should give some indication of the number of changes in the human reads that are not mapped to the human genome. It can be observed that for each LRD threshold presented in Table 4, the average Hamming distance of the mapped reads is always less than the average Hamming distance of the false negative reads. The same statement is also valid for the edit distance. For both distance measures, the difference between the average distance of true positives and the average distance of false negatives is not very high. Basically, only a few more bases are different from the source substring for the false negatives compared to the true positives. The highest difference is reported for the LRD threshold of 250. Table 4 shows that, on average, the reads that are mapped to the genome have less errors and mutations than the reads that are not mapped. However, the difference is not significant, since the false negative reads have at most 3 more errors, on average, than the mapped reads. An interesting remark is that the LRD aligner accepts more and more errors and mutations in the aligned reads as the LRD threshold increases, but even with the highest threshold of 539 that gives 100% recall rate, the precision of the hash LRD aligner is still very high (90.18%). In other words, the LRD aligner does a good job at discarding most of the reads simulated from the orangutan genome (true negatives), while mapping all the human reads, even those with higher error rates.

Human versus Five Mammals Experiment. In this second experiment, there are 60,000 reads to be aligned on the human mtDNA sequence. Only 10,000 reads are actually sampled from the same human genome. The 50,000 contaminated reads where

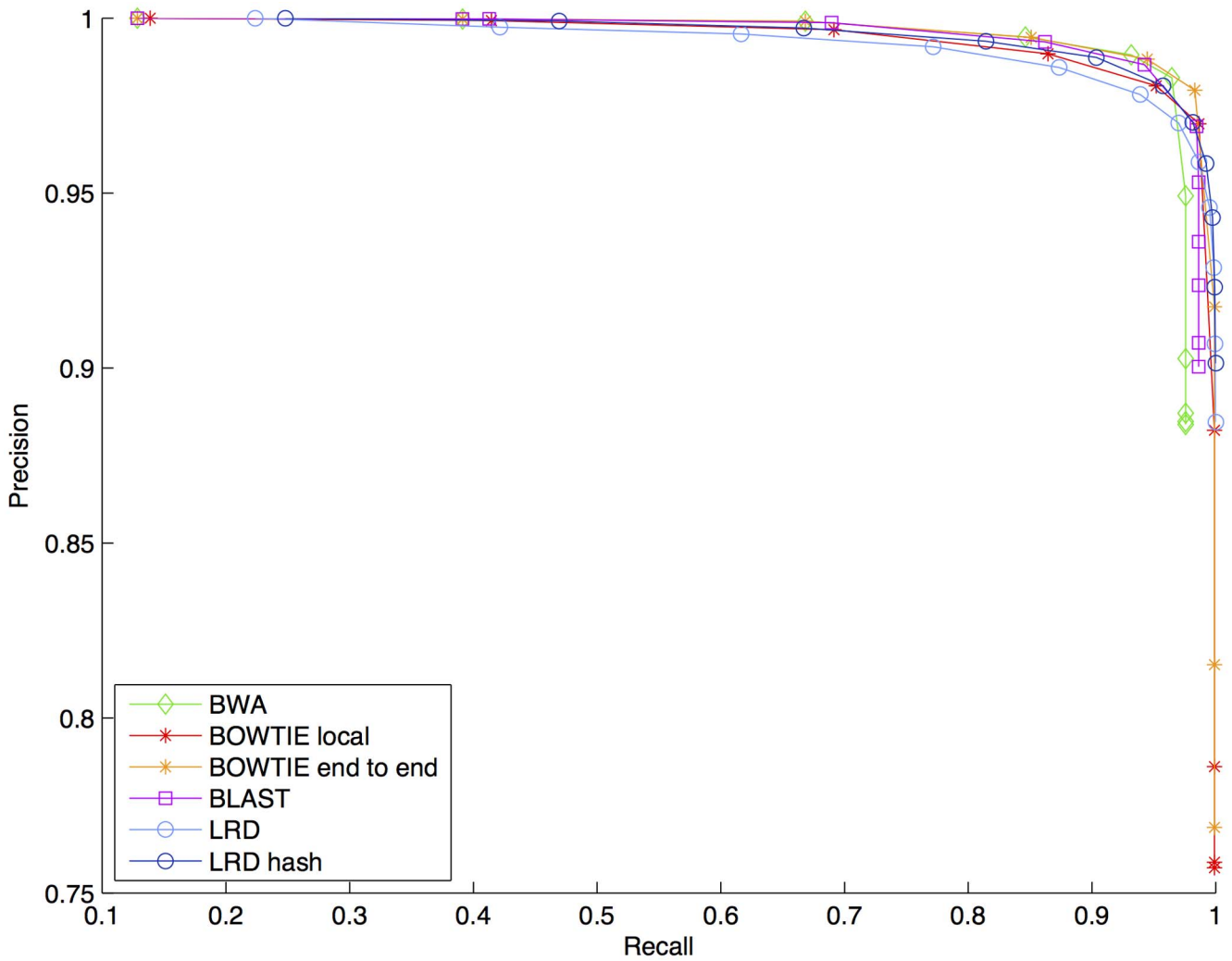

Figure 1. The precision-recall curves of the state of the art aligners versus the precision-recall curve of the two LRD aligners, when 10,000 contaminated reads of length 100 from the orangutan are included. The two variants of the BOWTIE aligner are based on local and global alignment, respectively. The LRD aligner based on hash tables is a fast approximate version of the original LRD aligner.
doi:10.1371/journal.pone.0104006.g001

sampled from 5 different mammals. The mammals where chosen to represent the 5 orders available in the first data set: Primates, Perissodactylae, Cetartiodactylae, Rodentia, and Carnivora. The contamination rate of 83.33% is much higher than in the previous scenario.

The precision-recall curves of the BWA, the BOWTIE, and the BLAST aligners versus the precision-recall curve of the two

Table 3. Several statistics of the state of the art aligners versus the LRD aligner, when 10,000 contaminated reads of length 100 sampled from the orangutan genome are included.

Aligner	AUC	Best F_1 Score	Best F_2 Score
BWA	97.37%	97.38%	97.03%
BOWTIE local	99.46%	97.80%	98.30%
BOWTIE end-to-end	**99.63%**	**98.13%**	98.24%
BLAST	98.38%	97.67%	98.15%
LRD aligner	99.46%	97.25%	98.48%
Hash LRD aligner	**99.63%**	97.58%	**98.61%**

The AUC is computed from the ROC curve, while the best F_1 and F_2 measures where computed using different points on the precision-recall curve. The F_2 measure puts a higher weight on recall.
doi:10.1371/journal.pone.0104006.t003

Table 4. Metrics of the human reads mapped to the human mitochondrial genome (true positives) by the hash LRD aligner versus the human reads that are not mapped to the genome (false negatives).

LRD	Precision	Recall	TP	FN	TP Ham.	FN Ham.	TP edit	FN edit
51	100%	24.79%	2479	7521	40.63	40.95	30.66	31.17
100	99.91%	46.92%	4692	5308	39.98	41.67	30.29	31.72
150	99.72%	66.70%	6670	3330	40.03	42.57	30.37	32.41
200	99.34%	81.43%	8143	1857	40.37	43.09	30.66	32.74
250	98.87%	90.36%	9036	964	40.56	43.82	30.81	33.26
300	98.06%	95.74%	9574	426	40.78	42.99	30.97	32.76
350	97.02%	98.16%	9816	184	40.84	42.97	31.02	32.61
400	95.85%	99.24%	9924	76	40.87	41.83	31.04	31.92
539	90.18%	100%	10000	0	40.87	-	31.05	-

The average Hamming distance and the average edit distance are reported for true positive (TP) and false negative (FN) reads, respectively. The average distances are given for several points on the precision-recall curve of the hash LRD aligner, going from 100% precision to 100% recall. The points are obtained by varying the LRD threshold from 51 to 539.

doi:10.1371/journal.pone.0104006.t004

variants of the LRD aligner are presented in Figure 2. Among the evaluated aligners, the BOWTIE local aligner has the lowest results in terms of precision and recall. Figure 2 seems to indicate that the LRD, the BLAST, and the BOWTIE end-to-end aligner give fairly similar results.

To make a better distinction between the aligners, the AUC measure and the best F_1 and F_2 scores for each aligner are presented in Table 5. The results presented in Table 5, indicate that the LRD aligner achieves the best AUC score, followed closely by the BOWTIE end-to-end aligner. As in the previous experiment, the BOWTIE aligner attains the highest F_1 score, while the LRD aligner attains the highest F_2 score. The BLAST aligner falls in third place.

An advantage of the LRD aligner is that it is the most flexible aligner in terms of precision and recall. The aligner proposed in this work is the only aligner that can be adjusted to go from 100% precision to 100% recall. Even if the other state of the art aligners do reach full recall, it is interesting to show the best recall that can be obtain by each one. The BWA aligner reaches a maximum recall of 97.57%, while the BLAST aligner reaches a maximum recall of 98.63%. Both variants of the BOWTIE aligner go up to 99.91% recall. As mentioned before, the maximum recall obtained by the LRD aligner is 100%.

Another interesting statistics is the recall when 100% precision is achieved. The recall at best precision is recorded in two scenarios. In the first scenario, only the contaminated reads from the orangutan are included, while in the second scenario, the rest of 40,000 contaminated reads from all the other mammals, besides the orangutan, are included. Since the orangutan and the human belong to the Primates order, the first scenario is more difficult.

The recall at best precision for each aligner evaluated in the first scenario is given in Table 6. When 10,000 contaminated reads sampled from the orangutan genome are used, it seems that the LRD aligner obtains the highest recall at 100% precision. The LRD aligner is roughly 10% higher than the state of the art aligners, which give similar recall values to each other.

The recall at best precision for each aligner evaluated in the second scenario is given in Table 7. This time, the recall at 100% precision for each aligner is much higher than in the first scenario. This indicates that if contaminated reads do not belong to an organism that is closely related to the human, the tools are able to align most of the true positive reads with 100% precision. Again, the best aligner is the LRD aligner based on the hash tables implementation. It attains a recall of 81.43%, being roughly 13% better than most of the state of the art aligners. In the second scenario, it seems that the BOWTIE local aligner falls very far behind the other alignment tools.

Overall, the LRD aligner seems to be the best tool among the evaluated aligners, in the presence of contaminated reads. It is closely followed by the BOWTIE end-to-end aligner. The high accuracy of the LRD aligner comes with the cost of being the slowest one among the evaluated aligners.

Clustering an Unknown Organism

The rank-based aligner is evaluated in the context of finding a solution for the task of clustering a new (or unknown) organism, given only a set of short Next-Generation Sequencing DNA reads. More precisely, the task is to find the order, the family, or the species of the unknown organism, without having to sequence its genome first, by aligning its reads into several genomes in order to obtain the nearest neighbor species (or the most similar species). The LRD aligner is compared to the BWA, the BOWTIE2 and the BLAST aligners. In the case of the BOWTIE2 aligner, two variants are evaluated, one based on local alignment and the other based on global alignment. The LRD aligner is based on 3-mers with a maximum offset between paired 3-mers of 36. A maximum distance threshold of 1000 was used in the case on the LRD aligner. The distance threshold for the LRD aligner was adjusted in order to allow more reads to be aligned, especially for the mammals that are more distantly related, more precisely, that are not from the same order. The approximate hash LRD aligner achieves similar results to the basic LRD aligner, when it aligns reads only in the positions that have at most 5 similar k-mers less than the maximum number of k-mers from the read that can be found at any given position in the reference sequence. For this reason, only the results of the approximate LRD aligner are reported in the following experiments.

One by one, each of the 20 mammalian genomes from the EMBL database will be considered to be unknown for the purpose of this experiment. The unknown individual will be represented by a set .. of short DNA reads randomly sampled from its genome. The task is to find the most similar individual (or species) from the remaining 19 individuals, for each unknown individual. In order to solve the task, the collection $\mathcal{R}$ of reads (that represents an unknown individual) is aligned on each of the 19 genomes from

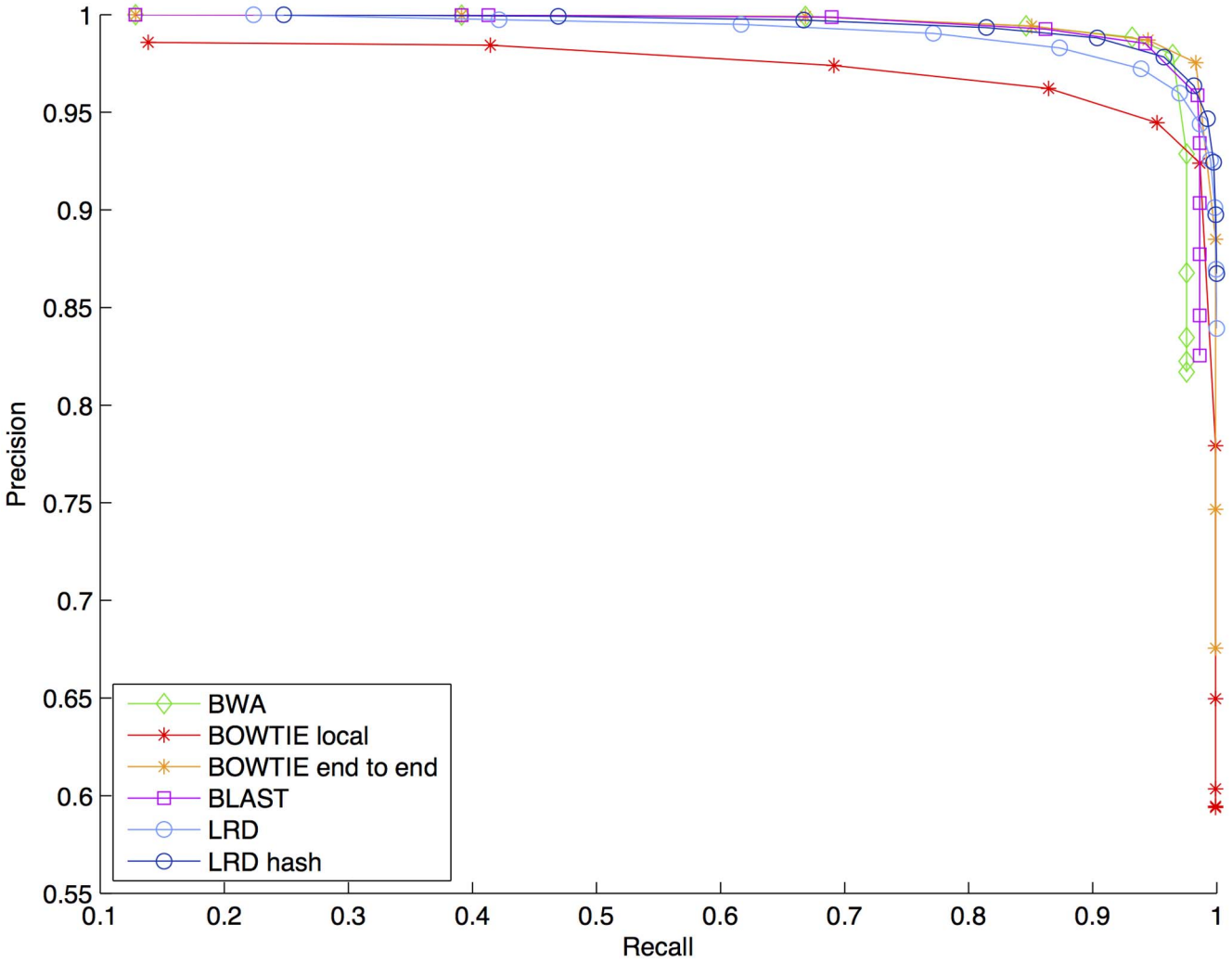

Figure 2. The precision-recall curves of the state of the art aligners versus the precision-recall curves of the two LRD aligners, when 50,000 contaminated reads of length 100 from 5 mammals are included. The two variants of the BOWTIE aligner are based on local and global alignment, respectively. The LRD aligner based on hash tables is a fast approximate version of the original LRD aligner.
doi:10.1371/journal.pone.0104006.g002

the collection $\mathcal{G}$ of genomes. Reads are aligned under a maximum distance threshold. Thus, only a subset $\mathcal{S} \subseteq \mathcal{R}$ of reads is aligned on each genome. An alignment score is computed for each genome in order to obtain the most similar individual. The score is given by the average minimum distances of the reads in $\mathcal{S}$ divided by the number of aligned reads. The minimum distance for a specific read is given by the best positional match in the reference genome. Lower scores indicate greater similarity between species, and higher scores indicate a greater dissimilarity between species. The individual (or the species) with the lowest score is considered to be the most similar one. Finally, the unknown organism is considered to be part of the same order as its most similar individual. The unknown individual is correctly clustered if it is indeed a member of the order predicted by the aligner. Thus, the performance of each aligner on this task is determined by the number of correctly clustered unknown individuals. The evaluation procedure can also be described as the leave-one-out cross-validation procedure. It is important to notice that the procedure described above does not generate a partitioning of the data set, but rather assigns a newly discovered (or unknown) organism to a specific cluster in an existing phylogenetic tree. An evaluation tool to obtain this score has also been added to the software package.

An interesting remark is that the tools evaluated on this task align reads under a given maximum distance threshold and, hence, many reads remain unaligned. The distance measure depends on the aligner. While the BWA and the BOWTIE aligners are based on the edit distance, the BLAST aligner uses a score of its own. The rank-based aligner is based on Local Rank Distance. Therefore, the alignment score is obtained by the average distance divided by the number of aligned reads. In other words, a genome with more aligned reads is more likely to be similar to the unknown individual.

The aligners are evaluated and compared under two different scenarios. In both scenarios, reads of 100 bases long were simulated using the *wgsim* tool [38]. In the first scenario, 20,000 short DNA reads per mitochondrial genome are sampled using the default parameters of the simulation tool. More precisely, the reads were generated using an error rate of 0.02, a mutation rate of 0.001, a fraction of indels of 0.15 (out of the total number of mutations) and a probability of extending an indel of 0.30. With an average base coverage of 100, the number of reads should be

Table 5. Several statistics of the state of the art aligners versus the LRD aligner, when 50,000 contaminated reads of length 100 sampled from the genomes of 5 mammals are included.

Aligner	AUC	Best F_1 Score	Best F_2 Score
BWA	97.52%	97.20%	96.75%
BOWTIE local	99.55%	95.41%	97.32%
BOWTIE end-to-end	99.84%	**97.93%**	98.16%
BLAST	98.57%	97.15%	97.93%
LRD aligner	99.86%	96.49%	98.04%
Hash LRD aligner	**99.92%**	97.25%	**98.29%**

The AUC is computed from the ROC curve, while the best F_1 and F_2 measures where computed using different points on the precision-recall curve. The F_2 measure puts a higher weight on recall.
doi:10.1371/journal.pone.0104006.t005

far than enough to correctly determine the order of unknown organisms. This scenario is designed to simulate a real-world setting where a high number of Next-Generation Sequencing reads is usually available. In the second scenario, only 200 simulated short DNA reads per genome are used in order to make the task harder to solve. The alignment methods should be challenged by the small amount of available reads. The generated reads also have more errors. More precisely, the reads for this second test case were simulated using an error rate of 0.08, a mutation rate of 0.008, a fraction of indels of 0.15 (out of the total number of mutations) and a probability of extending an indel of 0.30. In both test cases, half of the simulated reads from each genome are reverse complements.

Real-World Setting Experiment on Mammals. In the first test case, 20,000 simulated DNA reads of length 100 per genome are used, which corresponds to an average base coverage of 100. Table 8 compares the results of the LRD aligner with the other state of the art aligners.

The BWA aligns only the reads that fall under a certain edit distance threshold. The BWA aligner based on the default threshold 5 is listed in Table 8 under the name of *BWA edit 5*. Another BWA aligner with a threshold of 10 was used in the experiments. Since the latter one aligns more reads, it should be able to give more accurate results than the default BWA aligner.

In this scenario, it seems that the BLAST, the BOWTIE and the LRD aligners achieve perfect results. More precisely, they are all able to identify the most similar individual as being part of the same order as the unknown organism, for the entire set of 20 mammals. On the other hand, the BWA edit 5 aligner is only able to predict the correct order for 17 out of 20 mammals. It clusters the cat as Primates, and the fin whale and the gorilla as part of the Carnivora order. The BWA edit 10 aligner works even worse, correctly predicting the order for 14 mammals.

It is interesting to observe that all the methods are usually able to determine not only the correct order, but also the most similar species in the group. For example, the horse is always clustered near the donkey, rather than the Indian or the white rhinoceros, despite the fact that they are all members of the same order, namely Perissodactylae. The same situation can be observed in the case and the gray seal, which is always considered to be most similar with the harbor seal rather than the other member of the Carnivora order, namely the cat.

The empirical results show that, with the exception of the BWA aligner, all the other methods work very well. This also demonstrates that the evaluation procedure gives a relevant measure of similarity between a set of reads and a reference genome, that can be used for solving the task of clustering unknown organisms.

Hard Setting Experiment on Mammals. The first test case is not enough to make a clear distinction between the compared methods, with respect to the accuracy and the biological relevance. To better assess the performance levels of these aligners, another experiment is conducted using only 200 short DNA reads of length 100 per genome. As described above, the reads also contain more errors and mutations than in the previous test case.

The results of the state of the art aligners together with the results of the LRD aligner are shown in Table 9. Compared to the previous scenario, the results of the state of the art aligners are much lower this time. The BWA aligners predict the correct order for 10 and 16 mammals, respectively. Unlike the previous test case, the BWA edit 10 aligner works better than the BWA edit 5 aligner, probably because it is able to align more reads with high error and

Table 6. The recall at best precision of the state of the art aligners versus the LRD aligner, when 10,000 contaminated reads of length 100 sampled from the orangutan genome are included.

Aligner	Recall at Best Precision	Best precision
BLAST	12.83%	100.0%
BWA	12.84%	100.0%
BOWTIE end-to-end	12.84%	100.0%
BOWTIE local	13.87%	100.0%
LRD aligner	22.36%	100.0%
Hash LRD aligner	**24.79%**	100.0%

doi:10.1371/journal.pone.0104006.t006

Table 7. The recall at best precision of the state of the art aligners versus the LRD aligner, when 40,000 contaminated reads of length 100 sampled from the blue whale, the harbor seal, the donkey, and the house mouse genomes are included, respectively.

Aligner	Recall at Best Precision	Best precision
BLAST	68.95%	100%
BWA	66.84%	100%
BOWTIE end-to-end	66.84%	100%
BOWTIE local	13.87%	98.58%
LRD aligner	52.25%	100.0%
Hash LRD aligner	**81.43%**	100.0%

doi:10.1371/journal.pone.0104006.t007

mutation rates. The BOWTIE aligners obtain results that are roughly similar to the results of the BWA aligners. The BOWTIE local aligner predicts the right order for 13 out of 20 mammals, while the BOWTIE end-to-end aligner is able to correctly cluster two more mammals, reaching a total of 15 correctly clustered mammals. The BLAST aligner works fairly well, predicting the correct order for 17 mammals. It wrongly predicts the order for the cat and for the two members of the Rodentia order, namely the house mouse and the rat. It seems that all the aligners, besides the LRD aligner, have trouble predicting the right order for the Rodentia members. On the other hand, it seems that the aligners find it very easy to predict the correct order for the Primates. Finally, the LRD aligner is able to predict the correct class for the entire set of mammals. The LRD aligner seems to be more robust to high error and mutation rates, as it achieves the best results among all the evaluated aligners.

It is interesting to observe that the BWA with an edit distance threshold of 10 is not able to align any reads at all, for two of the mammals. This is the reason why no similar mammal is found for the cat or for the rat. The same problem occurs in the case of the BWA edit 5 aligner, which is not able to find any similar genomes for 10 mammals, due to the lack of aligned reads. This problem is likely caused by the high error and mutation rates, that were used to sample the reads from the original genomes. It may be concluded that the BWA aligner is the most fragile aligner with respect to high error and mutation rates.

Time Evaluation. The time taken by each aligner to produce the results for the two test cases of the experiment on clustering unknown organisms is shown in Table 10. For both test cases, there are 20,200 short DNA reads that must be aligned for each mammal on the rest of 19 mammalian genomes. In total, each tool must align 7,676,000 short DNA reads of 100 bases long, on a reference mtDNA genome of roughly $15{,}000-17{,}000$ bases. Note that the reference genome is not necessarily always the same, since the reads sampled from a genome are aligned into the remaining 19 genomes. The time was measured on a computer with Intel Core i7 2.3 GHz processor and 8 GB of RAM memory using a single Core.

Among the evaluated aligners, the BWA aligner is the fastest one, taking just over 3 minutes to align all the reads. The BOWTIE2 aligner is also very fast. It takes roughly 7 minutes to align the reads when the *local* option is used, and $9-10$ minutes for the *end-to-end* option. The BLAST aligner takes 30 minutes when the *megablast* option is turned on. Finally, the LRD aligner is the slowest one, but it also has the advantage of being the most accurate on all the test cases. The approximate LRD aligner based on the hash optimization implemented in C++ needs $16-17$ hours to align all the reads. The Java implementation of LRD aligner based on hash tables is roughly 3 times faster, with a total time of $5-6$ hours. The speed gain of the Java implementation is given by the optimized hash table implementation available in the Java API. It is important to mention that the parameters of the approximate LRD aligner are optimized for accuracy, not for speed. Even so, the approximate hash LRD aligner implemented in Java is roughly 50 times faster than the basic LRD aligner. The reported time of the approximate hash LRD aligner is comparable to that of the other tools that favor correctness over speed, such as BFAST [35]. Parallel or GPU processing could be used to further reduce the running time of the LRD aligner and to make it run as fast as BOWTIE2 or BLAST.

An important advantage of the LRD aligner is that it obtains very accurate results even for a very low base coverage. For instance, the LRD predicts the correct order for the entire set of 20 mammals by aligning 200 reads per genome (with high error and mutation rates), while the BWA edit 5 aligner is only able to predict the correct order for 17 mammals using 20,000 reads per genome (with low error and mutation rates). Considering this fact, the LRD aligner obtains better results than the fastest aligner (BWA) in the same amount of time (roughly 3 minutes). This being said, the LRD aligner can produce accurate results in an amount of time which is comparable the other state of the art aligners, simply by aligning considerably less reads than the other tools would require.

Experiment on Vibrio Species

In [9], a comparative study of the *V. vulnificus* YJ106, *V. parahaemolyticus* RIMD 2210633, and *V. cholerae* El Tor N16961 genomes was conducted to compare relative positions of conserved genes and to investigate the movement of genetic materials within and between the two chromosomes of these vibrio species. The study shows that *V. vulnificus* has a higher degree of conservation in gene organization in the two chromosomes relative to *V. parahaemolyticus* rather than to *V. cholerae*. This implies that *V. vulnificus* is closer to *V. parahaemolyticus* than to *V. cholerae* from the evolutionary point of view. This result is also supported by the study of [8], which determines that the block-interchange distance between *V. vulnificus* and *V. parahaemolyticus* is smaller than that between *V. vulnificus* and *V. cholerae*.

The goal of this experiment is to determine if the LRD aligner can achieve similar results to [8,9], using the evaluation procedure for clustering an unknown organism proposed in this work. Thus the experiment consists of aligning simulated reads from the *V. vulnificus* chromosomes into *V. parahaemolyticus* and *V. cholerae*. It is important to note that three test cases were considered. In the first test case, simulated reads of chromosome VV1 are aligned into VP1 and VC1, respectively. In the second case, simulated

Table 8. The results for the real-word setting experiment on mammals.

Mammal	Class (Label)	BWA edit 5	BWA edit 10	BLAST	BOWTIE local	BOWTIE end-to-end	LRDa LRD 1000
cow	Cet (1)	4	**9***	2	2	2	2
sheep	Cet (2)	1	**9***	1	1	1	1
blue whale	Cet (3)	4	4	4	4	4	4
fin whale	Cet (4)	**6***	3	3	3	3	3
cat	Car (5)	**9***	**9***	7	7	7	6
gray seal	Car (6)	7	7	7	7	7	7
harbor seal	Car (7)	6	6	6	6	6	6
human	Pri (8)	11	11	11	11	11	11
gibbon	Pri (9)	8	**2***	8	11	13	11
gorilla	Pri (10)	**5***	13	13	11	11	11
p. chimpanzee	Pri (11)	13	13	13	13	13	13
orangutan	Pri (12)	14	14	14	14	14	14
chimpanzee	Pri (13)	11	11	11	11	11	11
S. orangutan	Pri (14)	12	12	12	12	12	12
horse	Per (15)	16	16	16	16	16	16
donkey	Per (16)	15	15	15	15	15	15
I. rhinoceros	Per (17)	18	18	18	18	18	18
w. rhinoceros	Per (18)	17	17	17	17	17	17
mouse	Rod (19)	20	**17***	20	20	20	20
rat	Rod (20)	19	**17***	19	19	19	19
Accuracy		17/20	14/20	20/20	20/20	20/20	20/20

The results of clustering unknown organisms using the BWA aligner, the BLAST aligner, the BOWTIE aligner and the LRD aligner are presented on columns, respectively. Mammals are labeled with numbers from 1 to 20, given on the second column. The label of the closest species obtained by each aligner is reported for each mammal. Incorrectly clustered mammals are marked in bold and with an asterisk. Classes are actually 3-letter prefixes of order names. Unknown organisms are represented by 20,000 reads of length 100 simulated from the original genomes. Half of the reads are reverse complements.

doi:10.1371/journal.pone.0104006.t008

Table 9. The results for the hard setting experiment on mammals.

Mammal	Class (Label)	BWA edit 5	BWA edit 10	BLAST	BOWTIE local	BOWTIE end-to-end	LRDa LRD 1000
cow	Cet (1)	*	2	2	**19***	2	2
sheep	Cet (2)	*	**5***	1	1	**12***	1
blue whale	Cet (3)	*	4	4	**12***	4	4
fin whale	Cet (4)	3	1	3	3	3	3
cat	Car (5)	*	*	**1***	**9***	**19***	7
gray seal	Car (6)	7	7	7	7	7	7
harbor seal	Car (7)	6	6	6	6	6	6
human	Pri (8)	*	13	11	11	13	11
gibbon	Pri (9)	*	11	13	**16***	14	13
gorilla	Pri (10)	8	11	8	11	8	11
p. chimpanzee	Pri (11)	13	13	13	13	13	13
orangutan	Pri (12)	*	14	14	14	14	14
chimpanzee	Pri (13)	11	11	11	11	11	11
S. orangutan	Pri (14)	12	12	12	12	12	12
horse	Per (15)	16	16	16	16	16	16
donkey	Per (16)	15	15	15	15	15	15
I. rhinoceros	Per (17)	18	18	18	15	**12***	18
w. rhinoceros	Per (18)	*	17	17	**14***	17	17
mouse	Rod (19)	*	**6***	**12***	**14***	**12***	20
rat	Rod (20)	*	*	**12***	**8***	**5***	19
Accuracy		10/20	16/20	17/20	13/20	15/20	20/20

The results of clustering unknown organisms using the BWA aligner, the BLAST aligner, the BOWTIE aligner and the LRD aligner are presented on columns, respectively. Mammals are labeled with numbers from 1 to 20, given on the second column. The label of the closest species obtained by each aligner is reported for each mammal. Incorrectly clustered mammals are marked in bold and with an asterisk. Classes are actually 3-letter prefixes of order names. Unknown organisms are represented by 200 reads of length 100 (half of them being reverse complements) simulated from the original genomes, using an error rate of 0.08 and a mutation rate of 0.008.

doi:10.1371/journal.pone.0104006.t009

reads of chromosome VV2 are aligned into VP2 and VC2, respectively. Finally, the simulated reads from both chromosomes of *V. vulnificus* are aligned into the two chromosomes of *V. parahaemolyticus* on one hand, and into the two chromosomes of *V. cholerae* on the other hand.

In this experiment, reads of 100 bases long were simulated using the default parameters of the *wgsim* tool [38]. More precisely, the reads were generated using an error rate of 0.02, a mutation rate of 0.001, a fraction of indels of 0.15 (out of the total number of mutations) and a probability of extending an indel of 0.30. In this experiment, 30,000 simulated reads per chromosome are used, which corresponds to an average base coverage of 1. As in the previous experiment, half of the simulated reads from each genome are reverse complements. The LRD aligner is based on 3-mers with a maximum offset between paired 3-mers of 36. As in the previous experiments, the maximum distance threshold is set to 1000.

The scores of simulated reads from *V. vulnificus* chromosomes I and II aligned into *V. parahaemolyticus* and *V. cholerae* using the LRD aligner are shown in Table 11. The empirical results for all the three test cases are presented in this table. Each score is given by the average minimum Local Rank Distances of the aligned reads divided by the number of aligned reads on each genome. The results of the LRD aligner are similar to the results obtained in [8,9]. More precisely, the score between *V. vulnificus* and *V. parahaemolyticus* is lower than that between *V. vulnificus* and *V. cholerae* for both chromosomes of the three vibrio species. Even if chromosomes I and II are combined, *V. vulnificus* is found to be more similar to *V. parahaemolyticus*.

Some concern regarding the results obtained in this experiment might be that the results are influenced by the length difference between the reference genomes of *V. parahaemolyticus* and *V. cholerae*. First of all, the difference between the scores obtained by the LRD aligner is much higher than the difference between the lengths of the chromosomes VP1 and VC1. However, the study might be affected by the significant length difference between VP2 and VC2. While the number of simulated reads is fixed, the alignment tool excludes the reads that show a distance that is higher than the maximum threshold of 1000. The threshold should remove most of the reads that are aligned by chance, thus giving a score that is not influenced by the longer length of the VP2 chromosome.

Discussion

The results of the LRD aligner presented in this work are obtained using 3-mers and a maximum offset of 36. The maximum offset depends on the read length, more precisely it should be less or equal to the read length. The *k*-mers length should also be adjusted with regard to the read length. For reads of length 100, 3-mers are a reasonable choice since the chances of finding matching pairs of 3-mers between a read and the genome are very high. But even 4-mers and 5-mers work well, especially if the reads and the reference genome belong to the same species. If longer reads are considered for alignment, even longer *k*-mers can be used for a better accuracy and speed. On the other hand, longer *k*-mers are likely to reduce the accuracy of the aligner when the mutation and error rates are high, since the longer is the *k*-mer the greater is the probability of containing a mutation or error. For instance, if a *k*-mer contains a point mutation, the *k*-mer will not be matched correctly when LRD is computed. Even if LRD is designed to handle such situations, a carefully chosen *k*-mer length can make the most of the aligner proposed in this work. For instance, the work of [2] shows that LRD can be used with *k*-mers ranging from 3 to 20 letters. In the phylogenetic analysis of mammals presented in [2], the best results are obtained with *k*-mers ranging from 6 to 18 letters. When the LRD aligner is used for a specific application, it is recommended to tune the parameters of the aligner on a validation data set first, by considering the guidelines provided in this work.

Overall, the LRD aligner gives the most accurate results and it seems to be very robust for reads that contain many errors or mutations. However, the accurate results of LRD come with a cost. The time that LRD takes to align the same number of reads is higher than the time of the state of the art aligners evaluated in this paper. Nevertheless, the empirical results presented in this work show that the LRD aligner can produce very accurate results in the same amount of time as the other alignment tools, simply by using a lower base coverage. There is still enough room to speed up the LRD algorithm. By implementing it on GPU, the LRD aligner will be comparable (in terms of time) with the other aligners that favor efficiency over correctness. The LRD aligner can be considered as an useful tool for sequence alignment, being highly accurate from a biological (or evolutionary) point of view.

It is worth mentioning that another aligner based on rank distance (RD) was also proposed and evaluated. Despite the fact that the RD aligner is twice as fast as the BLAST aligner, the results obtained by the RD aligner on the set of experiments presented in this paper were not very convincing in terms of

Table 10. The running times of the BWA aligner, the BLAST aligner, the BOWTIE aligner and the LRD aligner.

Method	Time
BWA edit 5	3 minutes 14 seconds
BWA edit 10	3 minutes 50 seconds
BOWTIE local	9 minutes 43 seconds
BOWTIE end-to-end	7 minutes 14 seconds
BLAST	30 minutes
LRDa	285 hours
LRDa + hash (C++ implementation)	16 hours 33 minutes
LRDa + hash (Java implementation)	326 minutes

The aligners are compared on the task of aligning 7,676,000 short DNA reads of 100 bases long on a reference mtDNA genome of roughly $15{,}000-17{,}000$ bases. The aligners were evaluated on a computer with Intel Core i7 2.3 GHz processor and 8 GB of RAM memory using a single Core.
doi:10.1371/journal.pone.0104006.t010

accuracy. More precisely, it seems that the RD is not able to distinguish contaminated reads when a high recall is desired. On the other hand, it was able to identify the order of unknown organisms at a success rate comparable to the state of the art aligners. The RD aligner is included in the software package provided by this work for future development.

The results presented in this work can be considered as a strong argument in favor of using Local Rank Distance for computational biology tasks, in order to obtain results that are often more accurate from a biological point of view. Local Rank Distance [2] is related to the rearrangement distance [39]. The rearrangement distance works with indexed k-mers and is based on a process of converting a string into another, in a similar fashion to the edit distance. Unlike the edit distance or the rearrangement distance, LRD does not impose such global constraints. Instead, LRD tries to capture only the local changes in DNA. This seems to be more natural from an evolutionary point of view, since changes in DNA, such as point mutations or indels, occur at the local level. Perhaps this is the key insight of why Local Rank Distance should be expected to give more accurate results than the other distance measures. For instance, the edit distance counts the minimum number of operations required to transform one string into the other. It is clear that the actual number of DNA changes that did occur may be higher than the minimum number of operations. The Hamming distance sides with Local Rank Distance regarding the local aspect. However, the Hamming distance is greatly affected by indels. A single character that is inserted (or deleted) into one of the two strings will damage the Hamming distance computation for the rest of string. On the other hand, Local Rank Distance is more robust to changes such as indels or duplications, since it sums up the positional offsets of identical k-mers. When two DNA sequences are identical, the positional offsets of identical k-mers sum up to zero. If the two DNA sequences are affected by various types of DNA changes, the positional offsets of identical k-mers increase mostly in the affected DNA regions. Consequently, the Local Rank Distance will be higher, since it finds displaced k-mers. When more point mutations, indels, reversals or other kinds of errors occur in the DNA, LRD will indicate an even higher distance between the DNA sequences. Intuitively, Local Rank Distance reflects the total amount of local changes between two DNA sequences. This intuition can be better observed in Figure 3, which shows how the Local Rank Distance between two DNA sequences changes when one of the two sequences is affected by different types of DNA polymorphisms. Another key insight of why the rank-based approach should work better is that Local Rank Distance can capture very fine differences between strings, unlike the more commonly used edit distance or Hamming distance. More results that support this statement are presented in the empirical study performed in [32], which compares rank distance with Hamming distance and edit distance, respectively.

Conclusion and Further Work

This paper presented a tool for aligning a set of short DNA reads inside a reference genome, under Local Rank Distance. Several strategies for improving the speed of the LRD aligner were proposed. First of all, the k-mer positions were stored in a hash table for each read. Second of all, only the positions in the reference that are likely to give the minimum distance were considered, by previously counting the number of k-mers from the read that can be found at each position in the reference.

A set of experiments were conducted to assess the performance of the rank-based aligner in the presence of contaminated reads. In another set of experiments, the proposed aligner was used to find a solution for the task of clustering an unknown individual, given only a set of short DNA reads. Compared to the other evaluated tools, the LRD aligner has the important advantage of being very accurate even for a very low base coverage. To conclude, the empirical results showed that the LRD aligner can be considered as a viable alternative to standard alignment tools, since it can often be more accurate. Furthermore, the results obtained by the LRD aligner stand to support the studies of vibrio species performed in other studies [8,9], showing that the proposed aligner can indeed obtain conclusive results from an evolutionary point of view.

Methods

This section introduces the sequence aligner that work under Local Rank Distance. First, mathematical preliminaries about the rank-based distance measures are discussed. The LRD aligner and several optimization strategies are presented next.

Preliminaries

Given a string x over an alphabet Σ, and a character $a \in \Sigma$, the length of x is denoted by $|x|$, the number of occurrences of the character a in x is denoted by $|x|_a$. Strings are considered to be indexed starting from position 1, that is $x = x[1]x[2]\cdots x[|x|]$. Moreover, $x[i:j]$ denotes its substring $x[i]x[i+1]\cdots x[j-1]$.

Given two strings x and y over Σ, the *rank distance* (RD) between x and y, denoted by $\Delta_{RD}(x,y)$, is defined through the following algorithmic process: both strings are scanned (from left to right) and for each character a in the first string, and for each of

Table 11. The results of the rank-based aligner on vibrio species.

Reads Source	Reference	LRDa Score
VV1	VP1	606.2
VV1	VC1	643.9
VV2	VP2	773.0
VV2	VC2	849.9
VV1 + VV2	VP1 + VP2	641.7
VV1 + VV2	VC1 + VC2	697.7

The LRD aligner is based 3-mers, a maximal offset of 36, and a LRD threshold of 1000. The scores obtained by the LRD aligner for simulated reads of *V. vulnificus* chromosomes I and II aligned into *V. parahaemolyticus* and *V. cholerae* are presented in this table. The first column indicates the source chromosome of the simulated reads. The second column indicates the reference chromosome. The third and fourth columns show the scores of the two aligners computed with the evaluation tool provided in the software package.

doi:10.1371/journal.pone.0104006.t011

its k-th occurrence in x ($1 \le k \le \min\{|x|_a, |y|_a\}$), the algorithm sums up the absolute difference between the position of its k-th occurrences in x and y. Moreover, for each of the $||x|_a - |y|_a|$ non-matched occurrences of a in one of the two strings, the algorithm adds to the sum the arithmetic mean of $|x|$ and $|y|$, as described in ([30], Definition 2). The total sum computed by this algorithm represents the rank distance. Rank distance [3] is a low computational complexity measure of similarity with various applications in computational biology, from phylogenetic analysis [30,31] to finding common patterns in DNA sequences [32].

A recently introduced distance measure, termed Local Rank Distance [2], comes from the idea of better adapting rank distance to string data, in order to capture a better similarity (or dissimilarity) between strings, such as DNA sequences or text. Local Rank Distance (LRD) has already shown promising results in computational biology [2] and native language identification [4].

Local Rank Distance is inspired by rank distance, the main differences being that it uses k-mers instead of single characters, and that it matches each k-mer in the first string with the nearest equal k-mer in the second string. Given a fixed integer $k \ge 1$, a threshold $m \ge 1$, and two strings x and y over Σ, the *Local Rank Distance* between x and y, denoted by $\Delta_{LRD}(x,y)$, is defined through the following algorithmic process. For each position i in x ($1 \le i \le |x|-k+1$), the algorithm searches for that position j in y ($1 \le j \le |y|-k+1$) such that $x[i : i+k] = y[j : j+k]$ and $|i-j|$ is minimized. If j exists and $|i-j| < m$, then the offset $|i-j|$ is added to the Local Rank Distance. Otherwise, the maximal offset m is added to the Local Rank Distance. An important remark is that LRD does not impose any mathematically developed global constraints, such as matching the i-th occurrence of a k-mer in x with the i-th occurrence of that same k-mer in y. Instead, it is focused on the local phenomenon, and tries to pair equal k-mers at a minimum offset. To ensure that LRD is a (symmetric) distance function, the algorithm also has to sum up the offsets obtained from the above process by exchanging x and y. LRD can be formally defined as follows.

Definition 1 *Let $x,y \in \Sigma^*$ be two strings, and let $k \ge 1$ and $m \ge 1$ be two fixed integer values. The Local Rank Distance between x and y is defined as*:

$$\Delta_{LRD}(x,y) = \Delta_{left}(x,y) + \Delta_{right}(x,y),$$

where $\Delta_{left}(x,y)$ and $\Delta_{right}(x,y)$ are defined as follows:

$$\Delta_{left}(x,y) = \sum_{i=1}^{|x|-k+1} \min\{|i-j| \ : \ 1 \le j \le |y|-k+1 \text{ and } x[i : i+k] = y[j : j+k]\} \cup \{m\},$$

$$\Delta_{right}(x,y) = \sum_{j=1}^{|y|-k+1} \min\{|j-i| \ : \ 1 \le i \le |x|-k+1 \text{ and } y[j : j+k] = x[i : i+k]\} \cup \{m\}.$$

Notice that in order to be a symmetric distance measure, LRD must consider every k-mer in both strings. The symmetric property of LRD is ensured by computing both Δ_{left} and Δ_{right}. It is easy to observe that $\Delta_{left}(y,x) = \Delta_{right}(x,y)$. An interesting remark is that overlapping k-mers are permitted in the computation of LRD, since there is no restriction that tells where k-mers should start or end in a DNA string. Another interesting remark is that the search for matching k-mers is limited within a window of fixed size. The size of this window is determined by the maximum offset parameter m. This parameter must be set a priori and should be proportional to the size of the alphabet, the k-mers, and to the lengths of the DNA strings. Finding similar matches beyond this window is costly and it may also bring unwanted noise in the process. More details about the setting up the parameters of LRD are given in [2].

To better understand how LRD actually works, it is useful to consider the following example where LRD is computed between two strings using 2-mers.

Example 1 *Given two strings $x = abcaa$ and $y = cabca$, a fixed maximal offset $m = 3$, and a fixed size of k-mers $k = 2$, Δ_{left} and Δ_{right} are computed as follows*:

$$\Delta_{left}(x,y) = |1-2| + |2-3| + |3-4| + 3 = 6,$$
$$\Delta_{right}(x,y) = |1-3| + |2-1| + |3-2| + |4-3| = 5.$$

By summing up the two partial sums, Local Rank Distance is obtained

$$\Delta_{LRD}(x,y) = \Delta_{left}(x,y) + \Delta_{right}(x,y) = 11.$$

LRD Aligner

The aligner proposed in this paper is based on Local Rank Distance. It aligns a read of length l against a reference DNA sequence of length n. For efficiency reasons, it actually computes only Δ_{left} from Definition 1 between the read and a certain substring from the reference genome. It is perfectly reasonable to use only one of the two partial sums, Δ_{left} or Δ_{right}, since the symmetric property of LRD is no longer needed in the context of sequence alignment.

The basic alignment algorithm compares the read with the first substring in the reference and remembers the offset of each k-mer in the read. As it continues to compare the read with the following substrings at position 2,3,...,$n-l+1$ in the reference genome, respectively, the algorithm only needs to update the offset of each k-mer to obtain the new Δ_{left} distance at a certain position. The read is aligned in the position that gives the minimum Δ_{left} distance, but only if the obtained distance is less than a certain threshold. This basic LRD aligner is provided in the software package. It is worth mentioning that the algorithm described above is also applied for the short DNA string obtained by reverse complementing the original read. Several efficiency improvements are described next. In the end, they lead to the development of the faster LRD aligner presented in Table 12.

Indexing Strategies and Efficiency Improvements. The main efficiency improvement brought to the LRD aligner is to store k-mer positions in a hash table for each read. More precisely, the hash table h constructed from a short DNA read r will contain an array for each k-mer in the read. The array will contain all the positions of that k-mer in the read r. This hash table is actually a positional inverted index structure that is very popular in information retrieval. When LRD is computed for the read at a certain position i in the reference genome s, it is no longer necessary to do an extensive search within a window of fixed size to find equal k-mers between the read and the substring $s[i : i+|r|]$. The alternative solution is to take every k-mer in $s[i : i+|r|]$ and to look it up in the hash table h. Let j denote the position of the currently considered k-mer in $s[i : i+|r|]$. If the k-

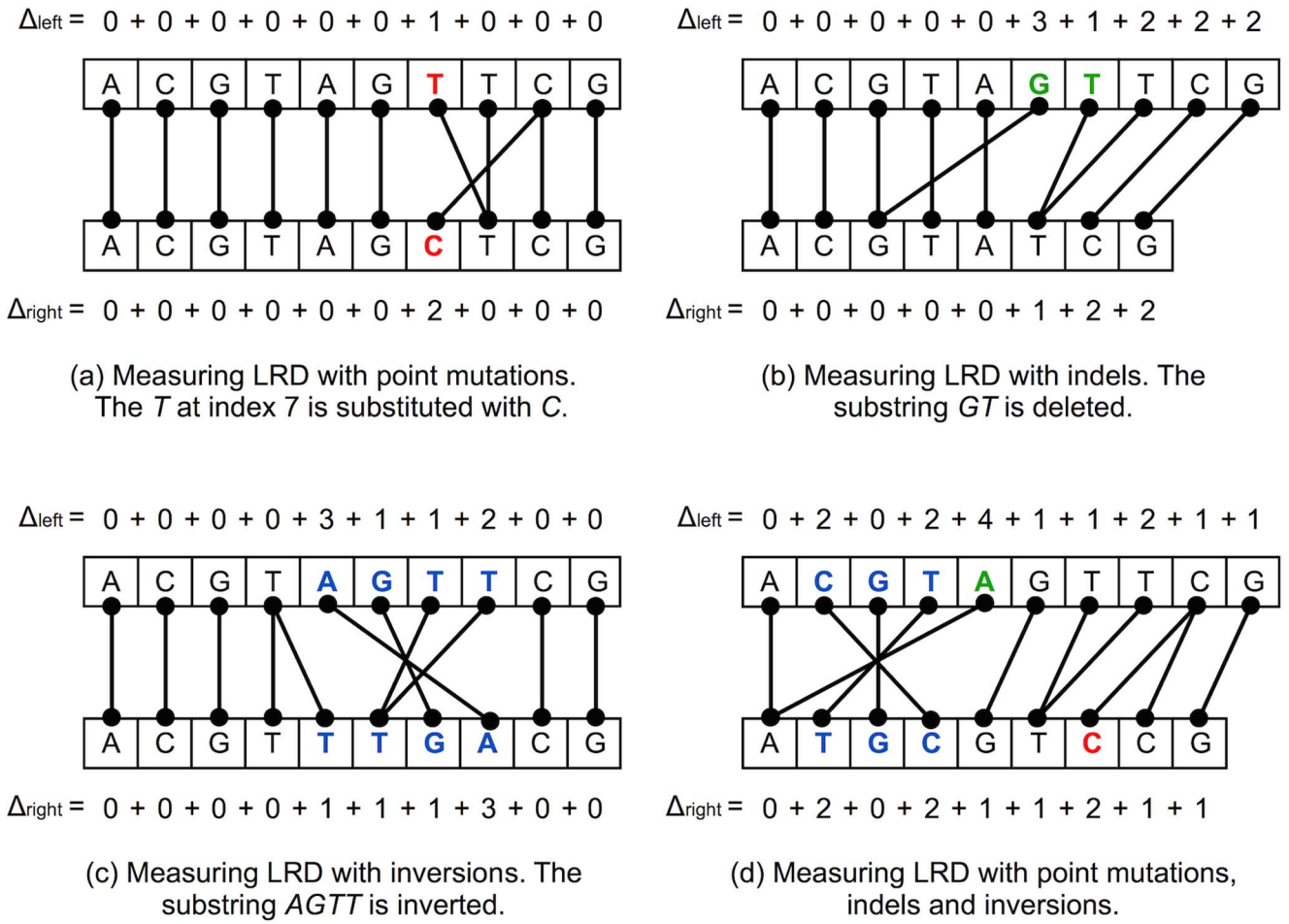

Figure 3. Local Rank Distance computed in the presence of different types of DNA changes such as point mutations, indels and inversions. In the first three cases (a), (b) and (c), a single type of DNA polymorphism is included in the second (bottom) string. The last case (d) shows how LRD measures the differences between the two DNA strings when all the types of DNA changes occur in the second string. The nucleotides affected by changes are marked with bold. To compare the results for the different types of DNA changes, the first string is always the same in all the four cases. Note that in all the four examples, LRD is based on 1-mers. In each case, $\Delta_{LRD} = \Delta_{left} + \Delta_{right}$.
doi:10.1371/journal.pone.0104006.g003

mer is found in h, the next step is to try a binary search in the positional array that is stored in $h(s[i+j-1 : i+j-1+k])$, in order to find the nearest position p that minimizes $|j-p|$. The offset $|j-p|$ is added to the distance sum only if $|j-p|$ is less than the maximal offset m, otherwise, m is added. If the k-mer is not found in h, m is added to the distance sum. The final sum obtained by this algorithm is the Δ_{right} partial sum from Definition 1. As mentioned before, one of the two partial sums can be left out for efficiency reasons, without affecting the accuracy. Thus, the hash LRD implementation is based only on Δ_{right}, as opposed to the basic implementation that uses only Δ_{left}. Consequently, there are some minor differences in the results obtained by the two implementations, but the accuracy levels are very similar and in some cases almost the same. For instance, in the experiments performed to evaluate the aligners in the presence of contaminated reads, the hash LRD aligner is only slightly better, while for the task of clustering unknown organisms, the results of the two LRD aligners are exactly the same.

The following strategies are designed to further improve the hash LRD aligner, in terms of speed. First of all, a boolean array f of size $|s|-k+1$ is used. Each element $f[i]$ indicates if the k-mer $s[i : i+k]$ is in the hash table h. When the algorithm tries to align the read at every position i in the reference sequence s, by computing the distance from the read r to the substring $s[i : i+|r|]$, it will have to look up some of the k-mers in h, several times (more precisely, $|r|$ times). Despite the fact that the hash table look up takes $O(1)$ time in theory, it is still faster to check the value of $f[i]$ instead of doing a hash table look up. Another improvement is to stop the alignment at a certain position i, if the distance sum computed between r and $s[i : i+|r|]$ becomes greater than the minimum Δ_{right} obtained so far.

The next efficiency improvement is to count the number of k-mers that are found in h, for every substring $s[i : i+|r|]$ in the reference genome. These counts are stored in an array c of length $|s|-|r|$. The algorithm can now consider the alignment only in the positions in the reference that are more likely to give the minimum Δ_{right} distance. It is fairly easy to observe that the more equal k-mers r and $s[i : i+|r|]$ have in common, the lower $\Delta_{right}(r,s[i : i+|r|])$ should be, since LRD is first based on finding equal k-mers between the two strings and, then, on minimizing the offsets between these k-mers. However, there is no guarantee that this is always the case. Therefore, the approach to skip the alignment for some positions i with low $c[i]$, in order to speed up the hash LRD aligner, gives approximate alignment results. More

Table 12. Algorithm 1. The hash LRD aligner algorithm.

Input:

r – a short DNA string of length l;

s – a reference DNA sequence of length n;

k – the size of the k-mers to be compared;

m – the maximum offset;

th – the maximum rank distance threshold accepted for the aligned read;

d – the threshold that can be adjusted to skip the alignment at some positions.

Initialization:

1. $\Delta_{min} = th$;
2. $bestPos = 0$;

Computation:

3. **for** $i \in \{1,...,l-k+1\}$
4. add i in the array stored at $h(r[i:i+k])$;
5. **for** $i \in \{1,...,n-k+1\}$
6. **if** $|h(s[i:i+k])| > 0$ **then** $f[i] = \text{true}$;
7. **else** $f[i] = \text{false}$;
8. $count = 0$;
9. **for** $i \in \{1,...,l-k+1\}$
10. **if** $f[i] == \text{true}$ **then** $count++$;
11. $c[1] = count$;
12. **for** $i \in \{2,...,n-l+1\}$
13. **if** $f[i-1] == \text{true}$ **then** $count--$;
14. **if** $f[i+l-k+1] == \text{true}$ **then** $count++$;
15. $c[i] = count$;
16. **for** $i \in \{1,2,...,n-l+1\}$
17. **if** $c[i] \geq \max\{c\} - d$ **and** $(|r|-k-c[i])\cdot m < \Delta_{min}$ **then**
18. $\Delta = 0$;
19. **for** $j \in \{1,...,l-k+1\}$
20. **if** $\Delta > \Delta_{min}$ **then**
21. abort and proceed to the next value of i in the loop from step 12;
22. **else**
23. **if** $f[i+j-1] == \text{true}$ **then**
24. do a binary search in the array stored at $h(s[i+j-1:i+j-1+k])$ to obtain the position p that minimizes $|j-p|$
25. $\Delta = \Delta + \min\{|j-p|, m\}$;
26. **else**
27. $\Delta = \Delta + m$;
28. **if** $\Delta < \Delta_{min}$ **then**
30. $\Delta_{min} = \Delta$;
31. $bestPos = i$;

Output:

$bestPos$ – the position were the read r was aligned;

Δ_{min} – the minimum LRD (or Δ_{right} to be more precise) obtained at position $bestPos$.

doi:10.1371/journal.pone.0104006.t012

precisely, lower distances can probably be obtained for some of the disregarded positions. These positions are disregarded by the two rules described next. The first rule is to eliminate the position i if $c[i] < \max\{c\} - d$, where d is a new input parameter of the aligner. This parameter can take values in the interval $[0, max\{c\}]$. When $d = 0$, more positions are disregarded. When $d = \max\{c\}$, no positions are disregarded at all, since $c[i]$ is always greater than 0. If the parameter d is set to eliminate more positions during the alignment, the algorithm will be faster, but it will also give less accurate alignment results. However, choosing $d = 5$ for reads of length 100 gives similar results to the basic LRD aligner in terms of accuracy, while drastically reducing the computational time, as the empirical results presented in this paper show. In all the experiments, the results of the approximate hash LRD aligner

are obtained with $d=5$. The second rule used by the approximate aligner is to eliminate the position i, if $(|r|-k-c[i])\cdot m$ is greater than the minimum Δ_{right} distance obtained until position i. The difference $|r|-k-c[i]$ gives the number of k-mers in $s[i:i+|r|]$ that are not found in h. For each of these missing k-mers, the maximal offset m is added to the Δ_{right} sum. Thus, $\Delta_{right}(r,s[i:i+|r|])$ is always greater than $(|r|-k-c[i])\cdot m$. But, if $(|r|-k-c[i])\cdot m$ is already greater than the minimum Δ_{right} distance obtained so far, there is no point in aligning the read at position i.

All the improvements described above are actually combined together to obtain an efficient LRD aligner. It is fairly obvious that these efficiency improvements and indexing strategies produce a different yet more efficient algorithm than the basic LRD aligner. The approximate hash LRD aligner algorithm is described in Table 12. As for the basic LRD aligner, a read is only aligned if the minimum LRD (or Δ_{right}, to be more precise) obtained by the algorithm is less than a certain threshold.

The algorithm described in Table 12 is also applied for the short DNA string obtained by reverse complementing the original read. But, another speed improvement is considered here. The alignment tool tries to align the reverse complement only if the minimum distance for the original read is not acceptable. An internal threshold is used to determine if the minimum Δ_{right} is acceptable (lower than the threshold) or not. This threshold is computed as follows:

$$t=\min\{th,\min\{k,m\}\cdot(|r|-k+1)\}.$$

The threshold t is low enough to ensure, with a certain probability, that if $\Delta_{right}<t$ then the read is aligned in the right place. This parameter speeds up the alignment tool especially when the reads and the reference genome belong to the same species. If the reads belong to other species (as in the case of contaminated reads, for example), the aligner will most likely try to align the reverse complements too.

Finally, the computational complexity of the algorithm described in Table 12 is $O(n\times l\times \log\frac{l}{|\Sigma|^k})$. Unlike the basic LRD aligner, the computational time of the approximate hash LRD aligner is no longer limited by the maximal offset m of LRD. This is a clear advantage of this faster implementation. However, in the experiments, the results of both the basic LRD aligner and the approximate hash LRD aligner are obtained with $m=36$ in order to compare the results of the two aligners and to show that they produce almost the same results.

In practice, the input parameters of the algorithm described in Table 12 should be carefully adjusted with respect to length of the DNA reads and to the amount of mutations and errors in DNA. For example, setting $k=10$ to use 10-mers for reads of 100 or 200 bases is not reasonable, since finding similar 10-mers in such short DNA strings is rare, if not almost impossible. But 3 to 5-mers are probably more suitable for aligning short DNA reads. Notice that the maximum offset parameter m should be adjusted accordingly. Using 5-mers and a maximum offset that is too small (less than 10, for example) might result in finding almost no similar 5-mers in the search window. The best practice to choose the parameters of the aligner is to tune them on a validation data set first.

Acknowledgments

The authors also thank Djamal Belazzougui and Fabio Cunial for helpful discussions. The authors thank the reviewers for their comments which lead to great improvements of this work.

Author Contributions

Conceived and designed the experiments: RTI AIT. Performed the experiments: AIT RTI. Analyzed the data: RTI AIT LPD. Contributed reagents/materials/analysis tools: AIT. Wrote the paper: LPD RTI AIT. Designed the software used in analysis: RTI AIT.

References

1. Li H, Homer N (2010) A survey of sequence alignment algorithms for next-generation sequencing. Briefings in Bioinformatics 11: 473–483.
2. Ionescu RT (2013) Local Rank Distance. Proceedings of SYNASC: 221–228.
3. Dinu LP (2003) On the classification and aggregation of hierarchies with different constitutive elements. Fundamenta Informaticae 55: 39–50.
4. Popescu M, Ionescu RT (2013) The Story of the Characters, the DNA and the Native Language. Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications: 270–278.
5. Li H, Durbin R (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics 25: 1754–1760.
6. Langmead B, Trapnell C, Pop M, Salzberg S (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biology 10: R25–10.
7. Altschul S, Gish W, Miller W, Myers E, Lipman D (1990) Basic local alignment search tool. Journal of Molecular Biology 215: 403–410.
8. Lin YC, Lu CL, Chang HY, Tang CY (2005) An efficient algorithm for sorting by block-interchanges and its application to the evolution of vibrio species. Journal of Computational Biology 12: 102–112.
9. Chen CY, Wu KM, Chang YC, Chang CH, Tsai HC, et al. (2003) Comparative genome analysis of Vibrio vulnificus, a marine pathogen. Genome Research 13: 2577–2587.
10. Meyer F, Paarmann D, D'Souza M, Olson R, Glass E, et al. (2008) The metagenomics RAST server - a public resource for the automatic phylogenetic and functional analysis of metagenomes. BMC Bioinformatics 9: 386.
11. Huson DH, Auch AF, Qi J, Schuster SC (2007) MEGAN analysis of metagenomic data. Genome Research 17: 000.
12. Ander C, Schulz-Trieglaff O, Stoye J, Cox AJ (2013) metaBEETL: high-throughput analysis of heterogeneous microbial populations from shotgun DNA sequences. BMC Bioinformatics 14: S2.
13. Zerbino DR, Birney E (2008) Velvet: Algorithms for de novo short read assembly using de Bruijn graphs. Genome Research 18: 821–829.
14. Trapnell C, Pachter L, Salzberg SL (2009) TopHat: discovering splice junctions with RNA-Seq. Bioinformatics 25: 1105–1111.
15. Trapnell C, Williams B, Pertea G, Mortazavi A, Kwan G, et al. (2010) Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. Nature Biotechnology 28: 511–515.
16. Tomescu AI, Kuosmanen A, Rizzi R, Mäkinen V (2013) A Novel Min-Cost Flow Method for Estimating Transcript Expression with RNA-Seq. BMC Bioinformatics 14: S15.
17. Levy S, Hannenhalli S (2002) Identification of transcription factor binding sites in the human genome sequence. Mammalian Genome 13: 510–514.
18. Prezza N, Fabbro CD, Vezzi F, Paoli ED, Policriti A (2012) ERNE-BS5: aligning BS-treated sequences by multiple hits on a 5-letters alphabet. In: BCB. pp. 12–19.
19. Bader DA, Moret BME, Yan M (2001) A Linear-Time Algorithm for Computing Inversion Distance between Signed Permutations with an Experimental Study. In: Proceedings of the 7th International Workshop on Algorithms and Data Structures. London, UK, UK: Springer-Verlag, WADS '01, pp. 365–376.
20. Bafna V, Pevzner PA (1998) Sorting by transpositions. SIAM Journal on Discrete Mathematics 11: 224–240.
21. Hannenhalli S (1996) Polynomial-time algorithm for computing translocation distance between genomes. Discrete Applied Mathematics 71: 137–151.
22. Hannenhalli S, Pevzner PA (1995) Transforming men into mice (polynomial algorithm for genomic distance problem. In: 36th Annual IEEE Symposium on Foundations of Computer Science. pp. 581–592.
23. Belda E, Moya A, Silva F (2005) Genome rearrangement distances and gene order phylogeny in gamma-proteobacteria. Molecular Biology and Evolution 22: 1456–1467.
24. Alekseyev MA, Pevzner PA (2008) Multi-break rearrangements and chromosomal evolution. Theoretical Computer Science 395: 193–202.
25. Shapira D, Storer JA (2003) Large edit distance with multiple block operations. In: SPIRE. pp. 369–377.
26. Chimani M, Woste M, Bocker S (2011) A Closer Look at the Closest String and Closest Substring Problem. Proceedings of ALENEX: 13–24.

27. Vezzi F, Fabbro CD, Tomescu AI, Policriti A (2012) rNA: a fast and accurate short reads numerical aligner. Bioinformatics 28: 123–124.
28. Popov VY (2007) Multiple genome rearrangement by swaps and by element duplications. Theoretical Computer Science 385: 115–126.
29. Felsenstein J (2004) Inferring Phylogenies. Sunderland, Massachusetts: Sinauer Associates.
30. Dinu LP, Sgarro A (2006) A Low-complexity Distance for DNA Strings. Fundamenta Informaticae 73: 361–372.
31. Dinu LP, Ionescu RT (2012) Clustering based on Rank Distance with Applications on DNA. Proceedings of ICONIP 7667: 722–729.
32. Dinu LP, Ionescu RT (2012) An Efficient Rank Based Approach for Closest String and Closest Substring. PLoS ONE 7: e37576.
33. Smith T, Waterman M (1981) Comparison of biosequences. Advances in Applied Mathematics 2: 482–489.
34. Langmead B, Salzberg SL (2012) Fast gapped-read alignment with Bowtie 2. Nature Methods 9: 357–359.
35. Homer N, Merriman B, Nelson SF (2009) BFAST: An Alignment Tool for Large Scale Genome Resequencing. PLoS ONE 4: e7767+.
36. Heidelberg JF, Eisen JA, Nelson WC, Clayton RA, Gwinn ML, et al. (2000) DNA sequence of both chromosomes of the cholera pathogen Vibrio cholerae. Nature 406: 477–483.
37. Makino K, Oshima K, Kurokawa K, Yokoyama K, Uda T, et al. (2003) Genome sequence of Vibrio parahaemolyticus: A pathogenic mechanism distinct from that of V cholerae. Lancet 361: 743–749.
38. Li H. wgsim - Read simulator for next generation sequencing. Available: http://github.com/lh3/wgsim.
39. Amir A, Aumann Y, Benson G, Levy A, Lipsky O, et al. (2006) Pattern matching with address errors: rearrangement distances. Proceedings of SODA: 1221–1229.