Fast and sensitive protein alignment using DIAMOND

Benjamin Buchfink[1], Chao Xie[2,3] & Daniel H Huson[1,2]

The alignment of sequencing reads against a protein reference database is a major computational bottleneck in metagenomics and data-intensive evolutionary projects. Although recent tools offer improved performance over the gold standard BLASTX, they exhibit only a modest speedup or low sensitivity. We introduce DIAMOND, an open-source algorithm based on double indexing that is 20,000 times faster than BLASTX on short reads and has a similar degree of sensitivity.

In metagenomics studies, millions of sequence reads are analyzed to determine the functional or taxonomic content of microbial samples from the environment[1]. An important computational step is to determine which genes are present, usually by aligning translated DNA sequences against a reference database of protein sequences such as the NCBI nonredundant (NCBI-nr) database[2] or KEGG[3]. BLASTX[4] has long been considered the gold standard tool for this owing to its high sensitivity. However, BLASTX is much too slow for routine application in a high-throughput context.

A number of faster approaches have been proposed, such as BLAT[5] and USEARCH[6]. Notably, RAPSearch2 (ref. 7) improves speed by a factor of up to 100 over BLASTX while maintaining a similar level of sensitivity. However, as the size and number of samples continue to grow, even faster methods are required. PAUDA[8] provides a 10,000-fold increase in speed over BLASTX but has very low sensitivity on the level of individual alignments, reporting only 2–3% of all BLASTX alignments.

Here we present DIAMOND (double index alignment of next-generation sequencing data), an open-source program that is ideally suited for replacing BLASTX in a high-throughput setting (http://ab.inf.uni-tuebingen.de/software/diamond and **Supplementary Software**). When targeting significant alignments against the NCBI-nr database with an expected value below 10^{-3}, DIAMOND aligns short sequence reads at approximately 20,000 times the speed of BLASTX and has a similar level of sensitivity. Like BLASTX, DIAMOND is an 'all mapper' that attempts to determine exhaustively all significant alignments for a given query.

Most sequence comparison programs, including BLASTX, follow the seed-and-extend paradigm. In this two-phase approach, users search first for matches of seeds (short stretches of the query sequence) in the reference database, and this is followed by an 'extend' phase that aims to compute a full alignment.

Sequence comparison programs typically precompute an index that holds all seed locations in the reference sequences. A file of queries is then linearly scanned, and the seeds of a given query are matched to seeds in the reference sequences by random-access lookups in the index. In contrast, DIAMOND uses double indexing, an approach that determines the list of all seeds and their locations in both the query and reference sequences (Online Methods and **Supplementary Software**). The two lists are sorted lexicographically and traversed together in a linear manner to determine all matching seeds and their corresponding locations. Double indexing takes advantage of the cache hierarchy by increasing data locality, thus reducing the demands on main memory bandwidth.

Most sequence comparison programs, including BLASTX and RAPSearch2, use single consecutive seeds, which need to be short (length 3–6 amino acids) to ensure sensitivity. To increase speed without losing sensitivity, DIAMOND uses spaced seeds—that is, longer seeds in which only a subset of positions are used[9,10]. The number and exact layout of those positions are called the weight and shape of the spaced seed, respectively. To achieve high sensitivity, DIAMOND uses a set of four carefully chosen shapes[11] of length 15–24 and weight 12 by default. The most sensitive version of DIAMOND uses 16 shapes of weight 9. In addition, DIAMOND uses a reduced amino acid alphabet of size 11 to enhance sensitivity[12]. A simple exact match criterion determines which seeds are passed on to the extension phase, in which a Smith-Waterman alignment[13] is computed.

In a recent metagenomic study of 12 permafrost samples[14], a BLASTX comparison of 176 million high-quality DNA reads against the KEGG reference database[3] was reported to require 800,000 CPU hours at a supercomputing center[15]. When we used DIAMOND with its default settings, the analysis of all 246 million reads took 2.3 h on a single workstation, producing a total of 568.9 million alignments on 43 million reads.

To systematically compare the performance of DIAMOND (version 0.4.7) with BLASTX (version 2.2.28+) and RAPSearch2 (version 2.18), we downloaded publicly available metagenome data sets produced with Illumina (permafrost[14] and Human Microbiome Project[16] data), Ion Torrent (ERP004234), 454 Titanium (SRR1298978 and SRR1298979) and Sanger[17] sequencing technologies as well as open reading frames (ORFs) predicted from a microbial assembly[18]. We used all three programs to align all data sets against NCBI-nr (version May 2013) on a single workstation using 48 cores. We ran both DIAMOND and RAPSearch2

[1]Department of Computer Science and Center for Bioinformatics, University of Tübingen, Tübingen, Germany. [2]Singapore Centre on Environmental Life Sciences Engineering, School of Biological Sciences, Nanyang Technological University, Singapore. [3]Life Sciences Institute, National University of Singapore, Singapore. Correspondence should be addressed to B.B. (buchfink@gmail.com) or D.H.H. (daniel.huson@uni-tuebingen.de).

RECEIVED 29 APRIL; ACCEPTED 20 OCTOBER; PUBLISHED ONLINE 17 NOVEMBER 2014; DOI:10.1038/NMETH.3176

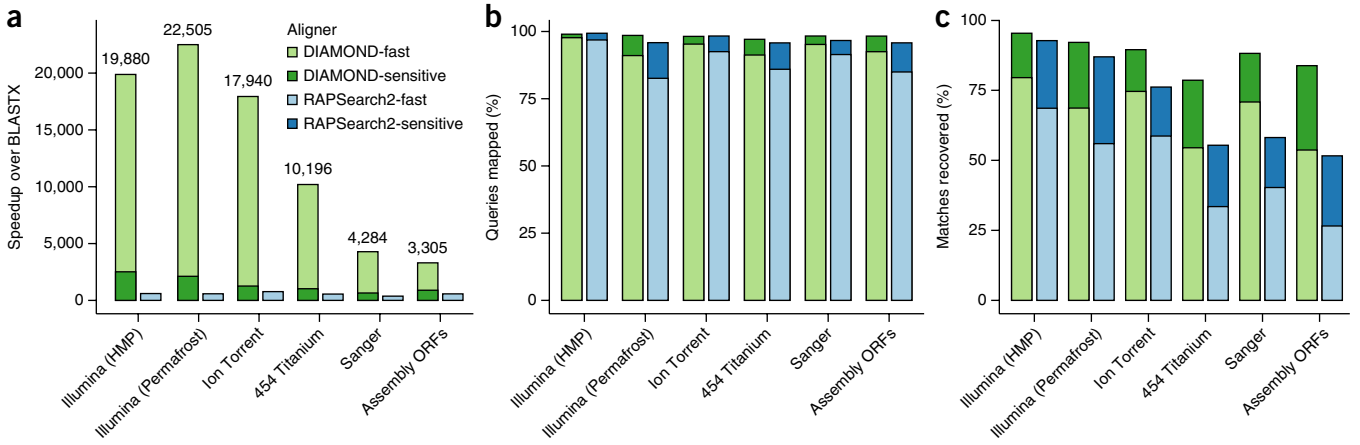

Figure 1 | Comparison of DIAMOND and RAPSearch2 against BLASTX for four sequencing technologies and for ORFs predicted from a bacterial assembly. (**a**) Fold speedup of each program over BLASTX. (**b**) Percentage (relative to BLASTX) of queries for which each program reports at least one alignment. (**c**) Percentage (relative to BLASTX) of matches recovered by each program. Only alignments with an expected value of ≤0.001 are considered. Programs were set to report alignments for up to 250 target sequences per read. Times are wall-clock times on a server using 48 cores and exclude one-time program startup overhead, which was <1 min for BLASTX and 5 min for DIAMOND-fast. HMP, Human Microbiome Project.

using a fast setting (–fast) and a sensitive setting (–sensitive) (**Fig. 1** and **Supplementary Table 1**).

DIAMOND-fast uses four seed shapes and runs around 20,000 times faster than BLASTX. DIAMOND-sensitive uses 16 seed shapes and runs about 2,000 times faster than BLASTX, aligning 99% of reads that have a BLASTX alignment and obtaining over 92% of all BLASTX alignments, on Illumina reads. DIAMOND-fast was 40 times faster than RAPSearch2-fast, with greater sensitivity, and was up to 500 times faster than RAPSearch2-sensitive, with similar sensitivity. DIAMOND-fast and DIAMOND-sensitive consistently outperformed RAPSearch2-fast and RAPSearch2-sensitive, respectively, on all data sets. The peak memory usage of DIAMOND-default was 100 GB, but the program can be configured to require less than 32 GB of memory, at the expense of a 30% reduction in speed.

Whereas the functional analysis of metagenome data sets has in the past been restricted by the requirement for supercomputing services, researchers can now use DIAMOND to perform functional analysis routinely on all their data sets. An analysis that takes 1 month with BLASTX takes only a few minutes with DIAMOND. Functional analysis of all 35 billion Illumina reads generated by the Human Microbiome Project, the largest published metagenome data set to date, would take about 2 weeks on a single server with DIAMOND.

METHODS

Methods and any associated references are available in the online version of the paper.

Note: Any Supplementary Information and Source Data files are available in the online version of the paper.

ACKNOWLEDGMENTS

This research was partially supported by the National Research Foundation and Ministry of Education Singapore under its Research Centre of Excellence Programme, and by the A*STAR Computational Resource Centre through the use of its high-performance computing facilities.

AUTHOR CONTRIBUTIONS

B.B. designed and implemented the algorithm. C.X. performed the experimental study. C.X. and D.H.H. initiated and guided the project. D.H.H. and B.B. wrote the manuscript.

COMPETING FINANCIAL INTERESTS

The authors declare no competing financial interests.

1. Handelsman, J., Rondon, M., Brady, S., Clardy, J. & Goodman, R. *Chem. Biol.* **5**, R245–R249 (1998).
2. Benson, D.A., Karsch-Mizrachi, I., Lipman, D., Ostell, J. & Wheeler, D. *Nucleic Acids Res.* **33**, D34–D38 (2005).
3. Kanehisa, M. & Goto, S. *Nucleic Acids Res.* **28**, 27–30 (2000).
4. Altschul, S.F., Gish, W., Miller, W., Myers, E.W. & Lipman, D.J. *J. Mol. Biol.* **215**, 403–410 (1990).
5. Kent, W.J. *Genome Res.* **12**, 656–664 (2002).
6. Edgar, R.C. *Bioinformatics* **26**, 2460–2461 (2010).
7. Zhao, Y., Tang, H. & Ye, Y. *Bioinformatics* **28**, 125–126 (2012).
8. Huson, D.H. & Xie, C. *Bioinformatics* **30**, 38–39 (2014).
9. Burkhardt, S. & Kärkkäinen, J. *Fundamenta Informaticae* **23**, 1001–1018 (2003).
10. Ma, B., Tromp, J. & Li, M. *Bioinformatics* **18**, 440–445 (2002).
11. Ilie, L., Ilie, S., Khoshraftar, S. & Bigvand, A.M. *BMC Genomics* **12**, 280 (2011).
12. Murphy, L.R., Wallqvist, A. & Levy, R.M. *Protein Eng.* **13**, 149–152 (2000).
13. Smith, T.F. & Waterman, M.S. *J. Mol. Biol.* **147**, 195–197 (1981).
14. Mackelprang, R. *et al. Nature* **480**, 368–371 (2011).
15. Jansson, J. *Microbe* **6**, 309–315 (2011).
16. Turnbaugh, P.J. *et al. Nature* **449**, 804–810 (2007).
17. Venter, J.C. *et al. Science* **304**, 66–74 (2004).
18. Wilson, M.C. *et al. Nature* **506**, 58–62 (2014).

ONLINE METHODS

Overview of DIAMOND. DIAMOND is a high-throughput alignment program that compares a file of DNA sequencing reads against a file of protein reference sequences, such as NCBI-nr[19] or KEGG[3]. It is implemented in C++ and is designed to run on multicore servers. The software can be obtained at http://ab.inf.uni-tuebingen.de/software/diamond.

DIAMOND is 4 orders of magnitude faster than BLASTX[4] at comparing short DNA reads against the NCBI-nr database and maintains a comparable level of sensitivity on alignments with an e-value $<10^{-3}$. The program is explicitly designed to make use of modern computer architectures that have large memory capacity and many cores. It follows the seed-and-extend approach. Additional algorithmic ingredients are the use of a reduced alphabet, spaced seeds and double indexing.

Seed and extend. The program is based on the traditional seed-and-extend paradigm for sequence comparison, in which exact occurrences of seeds (that is, short words of a given fixed length) contained in query sequences are located in the reference sequences, and these seed matches are then extended, if possible, to full alignments between the queries and references. The seed length used by an alignment program has a substantial impact on performance; shorter seeds increase sensitivity, whereas longer seeds increase speed.

Reduced alphabet. To increase speed without losing sensitivity, one approach is to use a reduced alphabet when comparing seeds. Using this, RAPSearch2 (ref. 7) is 40–100 times faster than BLASTX with minimal loss of sensitivity. For DIAMOND, we investigated the use of published reductions to four, eight and ten letters[12]. By analyzing a large number of BLASTX alignments, we developed a new reduction to an alphabet of size 11 that achieves slightly better sensitivity (brackets indicate one letter): [KREDQN] [C] [G] [H] [ILV] [M] [F] [Y] [W] [P] [STA].

Spaced seeds. A second improvement of the seed step is to use spaced seeds—that is, longer seeds in which only a subset of positions are used. The number and exact layout of those positions are called the weight and shape of the spaced seed, respectively. Theoretical analysis shows that a single spaced seed can perform better than a contiguous seed of the same weight, if its shape is suitably chosen[10]. Moreover, sensitivity can be increased further by using additional seed shapes, each resulting only in a sublinear increase in running time[10]. By default, DIAMOND uses a set of four shapes[11] of length 15–24 and weight 12 (**Supplementary Fig. 1**).

Seed index. The main bottleneck in aligning a large number of reads against a large reference database is not CPU performance but rather memory latency and bandwidth. The limiting factor is the amount of time required to load seed locations from main memory for comparison. Moving data from main memory into the cache takes hundreds of CPU clock cycles. Thus, a fast algorithm should take the cache hierarchy of computers into account so as to maximize data locality and minimize the number of main memory accesses. This can be done by decomposing the problem into small subproblems that fit into the cache.

In most seed-and-extend programs, an index structure (such as hash table or FM index) is built on the reference sequences to facilitate the location of seeds in the references. Queries are processed in the order that they occur in the input file. For a given query, all seeds are determined and the index is then used to look up all matching locations in the reference sequences. The reference locations are loaded from main memory into the cache. When another read is later encountered that contains some of the same seeds, the corresponding index and sequence data will usually have been evicted from the cache, so this data will have to be loaded from main memory again. Hence, using a single index in this way does not make good use of the cache. The problem is compounded when using a full-text index, such as a suffix array or a compressed FM index, as these require multiple individual memory accesses per index lookup.

Double indexing. DIAMOND uses a double-indexing approach in which both the queries and the references are indexed. A DIAMOND index is a list of seed-location pairs that is lexicographically ordered on a compressed representation of the seed. By traversing the two index lists linearly in parallel, all matching seeds between the queries and the references are identified, allowing the local alignment computation at the corresponding seed locations. The index memory access pattern of this approach is linear and can be efficiently handled by the hardware prefetcher, which will fill the cache with the indexing information before it is needed.

The double-indexed approach also improves data locality with respect to accessing the sequences. To see this, let S_q and S_r denote the set of seeds contained in the set of queries and references, respectively. For a given seed s, let m_s and n_s be the number of occurrences of the seed in the queries and the references. Using the standard index approach, the number of memory access operations is approximately $K = \Sigma_{s \in Sq \cap Sr} m_s n_s$, as for each occurrence of a reference seed in the queries, all corresponding reference locations will be loaded into memory. Using the double-indexing approach, the number of memory access operations is approximately $K' = \Sigma_{s \in Sq \cap Sr}(m_s + n_s)$, assuming that the combined size of query and reference locations for one seed is small enough to fit into the cache. This number is much smaller than K unless the sum is dominated by singleton seeds. To demonstrate this effect on real data, we plot the ratio of memory accesses for the two approaches depending on the length of the query sequence in letters as observed on our benchmark data (**Supplementary Fig. 2**).

The double-indexing algorithm used by DIAMOND is based on the well-known database sort-merge join algorithm, applied to the two seed sets of the queries and references. The main computation, the compilation and sorting of the two lists of seeds, can be efficiently addressed in parallel using a radix clustering step[20] in combination with a fast sorting algorithm. The total amount of time required to sort all the seeds in the given set of queries is smaller than what is required to access all seeds in a hash-table approach. Further, sorting all seeds in the reference sequences 'on the fly' takes much less time than loading a precomputed index. For example, the complete seed index for the current version of the NCBI-nr database is about 100 GB in size. This takes 100 s to generate in memory and about six times as long to read from disk at a typical read rate of 150 MB per second.

The alignment program mrsFast uses sorted lists as an index structure[21]. The authors of that tool spent a lot of effort on making their algorithm cache oblivious. Much of the challenge discussed there stems from the fact that short nonoverlapping seeds were used, which causes the set of all occurrences of a seed to exceed the cache capacity. Owing to the more elaborate seed strategy used by DIAMOND, in our program the amount

of data associated with a given seed will always be small enough to fit into the cache.

Memory efficiency. A drawback of using multiple spaced seeds is that this uses a lot of memory, which is a main reason why this approach has not been widely used, despite its proven advantages. The naïve implementation of a multiple spaced seed index builds a hash-table index for each of the seed shapes. With one hash table index for the NCBI-nr database being about 100 GB in size, four seed shapes would consume 400 GB, and 16 shapes would consume 1.6 TB of memory.

DIAMOND constructs and processes its indexes for one shape at a time, freeing up the memory used by one shape before moving on to the next. Thus, DIAMOND can perform alignment tasks with its sensitive 16-shape configuration using only as much memory as one shape index requires, which is an additional advantage of our approach. Moreover, using the radix cluster technique, the seed space is decomposed into 1,024 disjoint partitions. By building and processing indexes for only a subset of these partitions at the same time, the memory usage will be limited to the size of the subset index.

Seed extension. For each seed match found, DIAMOND determines whether it can be extended to an ungapped alignment of ten or more amino acids. If this is the case, then the seed match triggers the extend phase of the algorithm, which involves computing a Smith-Waterman alignment[13]. DIAMOND uses its own streaming SIMD extension (SSE)-accelerated Smith-Waterman implementation that extends previous algorithms[22] to allow the computation of banded and anchored alignments. By default, the program uses the BLOSUM62 matrix[23], a gap score of 11 and an extension score of 1, however, other BLOSUM matrices and scoring parameters can be used. The program determines the bit score and expected value of the computed alignment as in BLASTX. By default, alignments with a bit score <50 are not reported.

Because DIAMOND proceeds seed by seed rather than read by read, a key issue is how to avoid computing the same local alignment between a read and a reference more than once at different times during the search phase. To address this, DIAMOND allows a seed match to trigger an extension only if it is the left-most seed match in the corresponding ungapped alignment.

Experimental study. We downloaded the NCBI-nr database, which consists of 25.9 million sequences and 8.9 billion letters, in May 2013 to use as reference database.

We downloaded ten files of Illumina reads from the Human Microbiome Project website (http://www.hmpdacc.org/) covering samples from a range of different human-associated microbiomes (SRA SRS011134, SRS013687, SRS013951, SRS015578, SRS042628, SRS011239, SRS013800, SRS015369, SRS016753 and SRS053335). We extracted 500,000 random reads from each file so as to obtain a total set of 5 million reads, of average length 101. We refer to this as the Illumina (HMP) data set. We downloaded 12 Illumina data sets associated with permafrost cores[14] from the US Department of Energy Joint Genome Institute website. We extracted 500,000 random reads from each file so as to obtain a total set of 6 million reads, of average length 114, to use as our benchmark set of read sequences. We refer to this as our Illumina (permafrost) data set.

We downloaded a single data set of Ion Torrent reads from a study entitled 'Metagenome from artisanal cheeses from Tucuman' from NCBI (ERP004234). We downloaded two data sets of 454 Titanium reads from a study entitled 'Microbial community gene content and expression in the Central North Pacific Gyre, Station ALOHA, HOT186' from NCBI, with accession numbers SRR1298978 and SRR1298979. We downloaded a single data set of Sanger reads from the Sargasso Sea project[17]. We download a set of contigs from a microbial assembly[18] and used MetaGeneMark[24] to predict a total of 30,000 open reading frames (ORFs).

These data sets were used to compare the performance of DIAMOND (version 0.4.7), RAPSearch2 (version 2.18) and BLASTX (version 2.2.28+). All three programs were run on the same computer using 48 cores of a 64 core AMD Opteron server with 512 GB of main memory, running Ubuntu 12.04. The following parameter settings were used for the programs. BLASTX: blastx _num_threads 48 -evalue 0.1 -max_target_seqs 250 -comp_based_stats 0; DIAMOND-fast: diamond blastx -t 48 -k 250 -min-score 40; DIAMOND-sensitive: diamond blastx -t 48 -k 250 -sensitive -min-score 40; RAPSearch2-fast: rapsearch -a T -z 48 -v 250 -e -1 -b 1; RAPSearch2-default: rapsearch -z 48 -v 250 -e -1 -b 1.

To measure run times, BLASTX and RAPSearch2 were each run on three random subsets of each data set, and the run times were then extrapolated to the full data sets and averaged. DIAMOND was run on the full data sets (**Fig. 1** and **Supplementary Table 1**). For **Figure 1**, we considered only alignments with a BLASTX e-value below 10^{-3}. For **Supplementary Table 1** we also considered only the subset of all alignments with a BLASTX e-value below 10^{-5}. Alignments of poorer quality are usually not considered in metagenome analysis.

The reported run times are the total wall-clock time required on 48 cores, minus the program's overhead time. The overhead time of a program is measured by the wall clock time that the program requires to process an input file that contains only one read. This is a constant contribution to the run time that we ignore in our analysis because DIAMOND is intended to run on data sets that are much larger than our test data sets. For DIAMOND-fast, this is 5 min, and for DIAMOND-sensitive, this is 20 min. The reason we tested data sets that are small by practical standards was to allow us to run a full BLASTX analysis of each data set in a reasonable amount of time.

Sensitivity on the level of queries (percentage of queries mapped) is based on the number of reads for which BLASTX and the test method both find at least one common alignment against some reference sequence, divided by the total number of reads for which BLASTX finds at least one alignment. Sensitivity on the level of matches (percentage of matches recovered) is based on the number of alignments found both by BLASTX and the test method, divided by the total number of alignments found by BLASTX.

Principal coordinates (PCoA) analysis. To illustrate that minor differences between the output of BLASTX and DIAMOND do not affect the results of higher-level analyses, we used 12 random subsamples, each containing 200,000 reads, from 12 published permafrost samples[14]. Each subsample was aligned against NCBI-nr using both BLASTX and DIAMOND-fast,

doi:10.1038/nmeth.3176

keeping alignments with a minimum bit score of 50. Reads were mapped to KEGG Orthology (KO) numbers[3], for each read using the best alignment for which a KO number is known. We then computed Bray-Curtis distances from the resulting profiles and used PCoA to generate diagrams (**Supplementary Fig. 3**).

Memory usage and compatibility. The memory management of DIAMOND is designed to allow for an adaptable memory footprint that does not depend on the total size of the input. DIAMOND breaks down the input query and reference data into fixed size blocks of B sequence letters to be compared against each other at a time. With a seed index entry being 8 bytes long and the index being processed in C chunks, the total memory usage is then given by $2(B + 8B/C + const)$, where const represents a constant amount of overhead memory. With a default value of $C = 4$, the memory usage of the program is bounded by $6B + const$ irrespective of the total size of the database and queries. The block size B can be arbitrarily chosen by the user to fit the target machine. The command line options for B and C are -b and -c, respectively.

To explore the effect of the block size parameter on the performance, we aligned a query set of 35 million Illumina reads from permafrost against the NCBI-nr database containing 9 billion letters using different values of B (**Supplementary Table 2**). This operation requires a high memory server for maximum performance but can be efficiently handled by a machine with 16 GB of memory at about half the speed. This amount of RAM is readily available at a price of $160 on a standard desktop computer.

To ensure the compatibility of the program, we conducted tests on various systems using the NCBI-nr database, downloaded in September 2014, and a query set of 61 million Illumina reads from the Human Microbiome Project (**Supplementary Table 3**).

Code availability. DIAMOND v0.4.7 source code is available in **Supplementary Software** and at http://ab.inf.uni-tuebingen.de/software/diamond.

19. Wheeler, D.L. *et al. Nucleic Acids Res.* **36**, D13–D21 (2008).
20. Boncz, P., Manegold, S. & Kersten, M.L. *Proc. VLDB Conf.* **99**, 54–65 (1999).
21. Hach, F. *et al. Nat. Methods* **7**, 576–577 (2010).
22. Rognes, T. *BMC Bioinformatics* **12**, 221 (2011).
23. Henikoff, J.G. & Henikoff, S. *Methods Enzymol.* **266**, 88–105 (1996).
24. Zhu, W., Lomsadze, A. & Borodovsky, M. *Nucleic Acids Res.* **38**, e132 (2010).