*Published online 1 February 2021*

*NAR Genomics and Bioinformatics, 2021, Vol. 3, No. 1* **1**
*doi: 10.1093/nargab/lqab001*

# *Identity*: rapid alignment-free prediction of sequence alignment identity scores using self-supervised general linear models

**Hani Z. Girgis [1,*], Benjamin T. James [2] and Brian B. Luczak [3]**

[1]Bioinformatics Toolsmith Laboratory, Department of Electrical Engineering and Computer Science, Texas A&M University-Kingsville, 700 University Boulevard, Kingsville, TX 78363, USA, [2]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 32 Vassar Street, Cambridge, MA 02139, USA and [3]Department of Mathematics, Vanderbilt University, 1326 Stevenson Center Lane, Nashville, TN 3721, USA

Received May 14, 2019; Revised December 07, 2020; Editorial Decision January 04, 2021; Accepted January 08, 2021

## ABSTRACT

**Pairwise global alignment is a fundamental step in sequence analysis. Optimal alignment algorithms are quadratic—slow especially on long sequences. In many applications that involve large sequence datasets, all what is needed is calculating the identity scores (percentage of identical nucleotides in an optimal alignment—including gaps—of two sequences); there is no need for visualizing how every two sequences are aligned. For these applications, we propose *Identity*, which produces global identity scores for a large number of pairs of DNA sequences using alignment-free methods and self-supervised general linear models. For the first time, the new tool can predict pairwise identity scores in linear time and space. On two large-scale sequence databases, *Identity* provided the best compromise between sensitivity and precision while being faster than BLAST, Mash, MUMmer4 and USEARCH by 2–80 times. *Identity* was the best performing tool when searching for low-identity matches. While constructing phylogenetic trees from about 6000 transcripts, the tree due to the scores reported by *Identity* was the closest to the reference tree (in contrast to andi, FSWM and Mash). *Identity* is capable of producing pairwise identity scores of millions-of-nucleotides-long bacterial genomes; this task cannot be accomplished by any global-alignment-based tool. Availability: https://github.com/BioinformaticsToolsmith/Identity.**

## INTRODUCTION

We live in an era when sequences are generated at an unprecedented rate. Analyzing these countless sequences requires efficient computational methods. Algorithms for comparing sequence similarity are among the most widely applied ones to analyzing DNA, RNA and protein sequences. Pairwise alignment algorithms (1,2) have been the standard methods for assessing sequence similarity over the past 40–50 years. Multiple software tools for alignment are available (3,4). Applications of alignment algorithms include gene discovery (5), genome assembly (6–8), function prediction (9,10) and metagenomics (11), just to name a few. Many advancements have been made since the Needleman–Wunsch alignment algorithm was devised (2,12–14), but these new algorithms still depend on slow, quadratic, dynamic programming. This limitation is well manifested when comparing two very long sequences or scanning a very large sequence database. Almost all of the speed-ups are based on heuristic methods and may reduce the theoretical runtime down to $\mathcal{O}(n \log n)$ instead of $\mathcal{O}(n^2)$.

This shortcoming of alignment algorithms has led the field to develop plenty of faster, alignment-free methods. Multiple reviews of alignment-free methods have been published (15–23), indicating the importance and the abundance of such methods. One particular class of these methods depends on comparing two histograms of short words called $k$-mers, i.e. words of fixed length $k$. Building the histograms and comparing them can be done very efficiently. Although these alignment-free methods are very efficient, they have not become the standard in the field because their scores are not as intuitive or biologically relevant as the identity scores generated by alignment algorithms (an identity score is defined as the percentage of identical nucleotides between two optimally aligned sequences). However, $k$-mer statistics are often used by alignment tools as heuristics, such as in BLAST and USEARCH.

Motivated by the lack of biological relevance of the early alignment-free scores, a number of tools that produce phylogenetic distances, e.g. number of substitutions per position and pairwise mutation distance, have been proposed (24–31). However, alignment-free tools that can

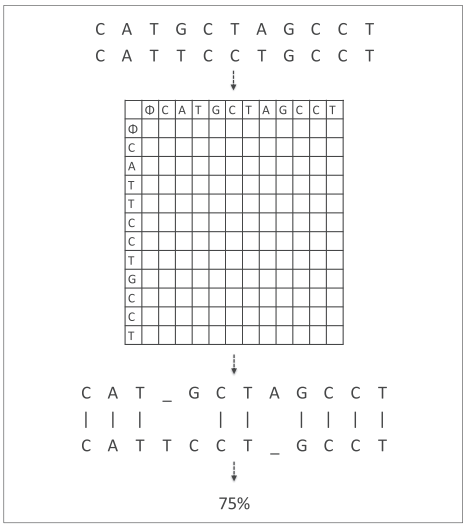
*To whom correspondence should be addressed. Tel: +1 361 5932 094; Fax: +1 361 5932 131; Email: hani.girgis@tamuk.edu

efficiently calculate global identity scores—the standard metric—still need to be developed.

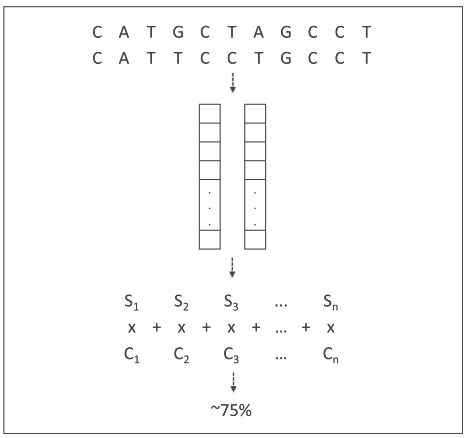
Often times, the identity score alone is enough; generating the alignment itself is not needed. For example, consider scanning GenBank for similar sequences to a particular gene. For another example, consider the task of clustering a large number of sequences. Or consider building a phylogenetic tree of the genomes of related species. In these three applications there is no need to generate alignments; only identity scores are required.

We propose a new tool, which we call *Identity*, for predicting pairwise global identity scores of DNA sequences. *Identity* can predict global identity scores in linear time—$\mathcal{O}(n)$—and space. The tool utilizes self-supervised machine learning algorithms in predicting global identity scores using a small number of alignment-free, *k*-mer statistics. *Identity* overcomes the weaknesses of alignment algorithms (slow, especially on long sequences) and those of alignment-free methods (their scores are not biologically relevant). It produces identity scores, which are intuitive, biologically relevant and the standard metric in the field while taking advantage of the efficiency of calculating *k*-mer statistics. *Identity* is designed for large-scale datasets; currently, it is unsuited to be applied to two sequences only. *Identity* can be applied to scanning a large sequence database for sequences similar to a query. The new tool can also generate all-versus-all identity scores, which can be used in clustering sequences, aligning multiple sequences and building phylogenetic trees. However, *Identity* does not generate a pairwise alignment, in which each nucleotide in one sequence is matched to a nucleotide or a gap in the other sequence. Because the new tool produces identity scores efficiently, it has the potential to save thousands of computational hours.

The core of *Identity* is an adaptive, linear model for predicting the identity scores above a user provided threshold. This design was inspired by our earlier research. We have successfully implemented adaptive software tools using self-supervised learning algorithms for locating cis-regulatory modules (32), identifying DNA repeats (33,34), and for clustering DNA sequences (35,36). Multiple software tools we developed earlier utilize general linear models (GLM) (34–40). Alignment-assisted methods that can classify similar and dissimilar sequences and predict identity scores were developed (34,35); such methods use alignment-free statistics to predict the alignment identity scores, on which they are trained, i.e. they are not alignment-free completely. These earlier tools justify our design choice of the adaptive, linear model as the core of *Identity*.




**A** Global alignment algorithm




**B** *Identity*

**Figure 1.** A high-level comparison between (**A**) the Needleman–Wunsch alignment algorithm and (**B**) the underlying method of *Identity*. The alignment algorithm calculates a table of size $n \times m$, where $n$ and $m$ are the lengths of the two sequences. In contrast, *Identity* converts each sequence to a *k*-mer histogram whose length is proportional to the average length of the two sequences. Then *Identity* calculates some statistics on the two histograms and the final score as the weighted average of the statistics.

## MATERIALS AND METHODS

### Main idea

In Figure 1, we diagram the main idea of *Identity* and contrast it to the Needleman–Wunsch alignment algorithm.

### Definition of identity scores

First, we define identity scores. An identity score is the ratio or the percentage of all aligned columns with identical nucleotides in the two sequences. In other words, it is the ratio of columns with identical nucleotides to columns with different nucleotides, same nucleotides or including gaps. For example, the alignment shown in the lower part of Figure 1A has 12 columns, nine of which have identical nucleotides in the two sequences; therefore, the identity score is 0.75 (9 ÷ 12) or 75%.

### Input and output

*Identity* is a tool for calculating pairwise global alignment identity scores in linear time. The input to the tool is a database of sequences and a query list if the user desires to search this database for sequences similar to the query sequence(s). Alternatively, the input can be a database only if the user desires to obtain all-versus-all identity scores. Alignment algorithms can be applied to two sequences only.

*Identity* requires training sequences, which are selected from the input database. Therefore, *Identity* should be applied to medium or large datasets. *Identity* requires a threshold score; sequence pairs with identity scores below the threshold are not reported (the user may chose to report all pairs). The tool outputs the headers of sequence pairs along with their identity scores if they are above the threshold.

### Method overview

*Identity* is an instance of self-supervised learning; such learning algorithms generate their own labeled training data. They then train themselves on the generated data. *Identity* performs the following steps:

1. Select a sequence subset—the templates—of input data.
2. Mutate copies of the templates to generate training and testing semi-synthetic sequence pairs along with their alignment-free identity scores.
3. Convert each sequence to a *k*-mer histogram.
4. Calculate statistics on histogram pairs.
5. Select features for the regression model, which is a GLM, using the best-first feature selection algorithm.
6. Train and test the GLM on sequence pairs in the training and the testing sets.
7. Read all input sequences, convert them to histograms, compute the selected statistics and use the trained GLM to calculate the identity scores.

We start with explaining how the templates are selected.

### Template selection

When a database of DNA sequences is given, the first step is to use the first 1000 sequences or to use all sequences in the database if they are fewer than 1000 to generate semi-synthetic data. We call the selected sequences templates. The next step depends on the application mode (database search or all versus all). If *Identity* is applied in the search mode, the 1000 templates are filtered based on length to eliminate too-short or too-long sequences that are impossible to produce the desired minimum identity score. For example, suppose that the query sequence is 1000 base pair (bp) long and the identity threshold is 0.5, sequences shorter than 500 ($1000 \times 0.5$) bp and longer than 2000 ($1000 \div 0.5$) bp are removed. After that, the query sequence(s) is added to the remaining templates. If *Identity* is applied in the all-versus-all mode, no filtering is performed. Next, we explain how the semi-synthetic data are generated from the templates.

### Semi-synthetic data generation

Semi-synthetic data are generated by mutating real sequences taken from the input database—the templates. Each template is copied multiple times. The number of copies per template is calculated by dividing 10 000 (the desired size of data utilized in training and testing) by the number of the templates. These copies are then mutated using the following mutation types: (i) single point mutation or (ii) block mutation. In single point mutation, a single nucleotide is mismatched, deleted, or inserted. In block mutation, a block of random nucleotides is inserted; or a block of consecutive nucleotides is deleted; or a block of nucleotides is duplicated and placed in tandem to the original block. The size of the block is chosen at random (the default range is 2–5 bp). Each mutation type, e.g. single insertion, single deletion or block duplication, has the same chance to be applied. To ensure that the original nucleotide composition is conserved, random nucleotides to be inserted or to be changed are generated from the same distribution of the templates' nucleotides. For example, suppose that the templates have the following nucleotide distribution: A: 0.4, C: 0.1, G: 0.1, T: 0.4. When a random nucleotide to be inserted, A or T has the highest probability of 0.4 each and C or G has the lowest probability of 0.1 each.

We invented this generative process to avoid using alignment algorithms for three reasons. First, alignment algorithms are slow. Second, the input database may not have enough sequence pairs with specific identity scores to train the regression model. A successful regression model should be trained on sequence pairs with identity scores representing the whole range from 0 to 1. We have no guarantee that the input sequences include pairs with identity scores covering the whole range. Third, alignment algorithms are almost infeasible on very long sequences.

Because mutated sequences are generated with specific mutation types and rates, the identity scores can be calculated without using any alignment algorithms. To calculate an identity score, we need to know the length of the alignment and the number of matches—identical nucleotides—between two sequences. Each mutation type affects these two numbers in a unique way. If we keep track of the mutations applied and update the alignment length and the number of matches accordingly, *the corresponding identity score can be obtained without actually aligning the two sequences.* Table 1 lists the mutation types used in our study and their effects on the alignment length and the number of matches.

For a very simple example, consider a 10-nt long sequence. We wish to mutate 30% of this sequence. For simplicity, assume that the three mutations are mismatch, insertion and deletion. Initially, the length of the alignment and

**Table 1.** The effects of each mutation type on identity scores

| Mutation type | Alignment length | Number of matches |
|---|---|---|
| Mismatch | No effect | Subtract 1 |
| Deletion | No effect | Subtract the number of the deleted nucleotide(s) |
| Insertion | Add the number of the inserted nucleotide(s) | No effect |
| Duplication | Add the number of the duplicated nucleotides | No effect |

The identity score is the ratio of identical nucleotides between two sequences to the total length of the alignment, which may include gaps. Because *Identity* generates a mutated sequence from an original sequence, it can calculate their identity score without aligning them. Initially, the alignment length and the number of matches are equal to the length of the original sequence that is the template to be mutated. As the mutation process proceeds, a mutation type is selected randomly. Each mutation type affects the alignment length and the number of matches in a unique way. For example, a mismatch has no effect on the length of the alignment; it decreases the number of matches by 1.

the number of matches are equal to 10—the length of the original sequence. A mismatch does not affect the length of the alignment; however, it decreases the number of matches. After the mismatch is introduced, the length of the alignment is 10, and the number of matches is 9. An insertion would result in a gap in the original sequence if the two sequences were to be aligned versus each other, i.e. it increases the length of the alignment by 1 and does not affect the number of matches. After the insertion is introduced, the length of the alignment is 11 and the number of matches is 9. Deleting a nucleotide would result in a gap in the mutated sequence if it was to be aligned versus the original sequence. This gap does not affect the alignment length; but the number of matches is decreased by 1. After the deletion is introduced, the length of the alignment is 11, and the number of matches is 8. These three mutations lead to an identity score of 0.73 ($8 \div 11$). Next, this procedure is applied to generating two datasets.

### Training and testing sets

We apply the generative procedure discussed earlier to generating two datasets for training and testing. Each dataset has a maximum of 5000 sequence pairs. Recall that multiple mutated sequences are generated from each template. Each of these mutated sequences is generated according to a desired identity score, which is selected uniformly from two segments: (i) between 0 and the user-provided threshold and (ii) the threshold and 1. The number of sequence pairs with identity scores above the threshold is equal to the number of pairs with scores below the threshold. This way identity scores in the important segment (above the threshold) will have equal say to those with scores below the threshold regardless of the threshold. To appreciate this point, consider a threshold of 0.95. If identity scores were selected uniformly between 0 and 1, the ratio of pairs with identity scores above the threshold to the pairs with identity scores below the threshold would be 1:19. This skewed ratio makes the optimization algorithm pay more attention to the segment below the threshold; the user is not interested in sequence pairs with scores in this range. However, under the proposed method, the ratio of pairs with identity scores above the threshold to those with scores below the threshold is 1:1. This equal ratio makes the optimization algorithm pay equal attention to the segment above the threshold (sensitivity) and the segment below the threshold (specificity).

Next, we illustrate how these datasets are represented to the regression model as few statistics calculated on pairs of $k$-mer histograms.

### Calculating the $k$-mer statistics

Each sequence is represented as a $k$-mer histogram. Then statistics are calculated on each pair of histograms. The choice of $k$—the size of $k$-mers—guarantees that the histogram size is linear with respect to an average input sequence. We calculate $k$ according to Equation 1 (21,35).

$$k = \lceil \log_4 \text{average length in a database} \rceil - 2 \qquad (1)$$

Using our survey of alignment-free methods (21), we chose the following 26 statistics: Manhattan, Euclidean, $\chi^2$, Chebyshev, Hamming, Minkowski, Cosine, Correlation, Bray Curtis, Squared chord, Hellinger, Conditional KL divergence, K divergence, Jeffrey divergence, Jensen-Shannon divergence, Revised relative entropy, Intersection, Kulczynski 1, Kulczynski 2, Covariance, Harmonic mean, Similarity ratio, Markov, SimMM, $D_2^S$ and $D_2^*$.

We compute the 26 statistics then normalize each of them between 0 and 1. Some statistics represent distances and others represent similarities. We convert each distance value to a similarity score by subtracting the normalized distance from 1. We call these 26 statistics single statistics. One of the primary results of our evaluation study was that squared versions or multiplicative combinations can often times outperform single statistics. For this reason, we square each of the single statistics to create 26 additional statistics. Finally, the paired statistics are generated by multiplying each unique combination of the 52 single and squared statistics (we do not pair a statistic with itself), resulting in 1326 pairs. The total number of the statistics is 1378 (26 singles + 26 squares + 1326 pairs). These statistics are the features, on which the GLMs are trained. Next, we illustrate GLMs briefly.

### GLMs

The general form of the linear model is $\boldsymbol{y} = \boldsymbol{F}\boldsymbol{w}$ where $\boldsymbol{y}$ is the target we wish to predict. In this work, $\boldsymbol{y}$ represents identity scores. $\boldsymbol{F}$ is a feature matrix; each of its columns represents a particular statistic except the first column is all ones. The coefficients in the $\boldsymbol{w}$ vector are found using the pseudoinverse solution (Equation 2).

$$\boldsymbol{w} = (\boldsymbol{F}^T\boldsymbol{F})^{-1}\boldsymbol{F}^T\boldsymbol{y} \qquad (2)$$

Now that we have the coefficients of the GLM, Equation 3 is used for making predictions.

$$\hat{\boldsymbol{y}} = \boldsymbol{F}\boldsymbol{w} \qquad (3)$$

Here, $\hat{y}$ represents the predicted identity score for a given sequence pair.

Using a small number of features is necessary to the success of training the GLM because it prevents it from overfitting the training data. In the next step, we utilize the best-first feature selection algorithm (41) in selecting few features, i.e. statistics.

### Feature selection

Features are selected automatically on each input dataset. The best-first algorithm is used for selecting a strong group of features without trying every possible combination. *Identity* utilizes a GLM in selecting features that minimize the mean squared error. This step is performed on the training set only. The main idea of the algorithm is to expand a feature set by adding one new feature to the set or removing one of those already in it. For example, suppose that the best performing set consists of three features: A, B and C, and two more features are not in the set: D and E. The expansion step produces the following children sets: {B, C}, {A, C}, {A, B}, {A, B, C, D} and {A, B, C, E}. Each of these children is used in training a GLM. If any of the children results in a better performance than its parent, the child set

becomes the best performing set and the expansion process is repeated on it and the parent set is not considered anymore. The other children are marked open and are added to other open sets that resulted from previous iterations. If none of the children outperforms the parent set, the parent set remains the best performing set and another set of the open ones—with the best performance among them—is expanded. If the best set does not change for few iterations, the algorithm stops. We modified the algorithm slightly to make sure it selects a minimum number of features. This minimum is equal to $k$, which is the length of the $k$-mer.

Up to this point, we discussed how the training and the testing datasets are generated and how the features are extracted and selected. Next, we discuss how the final GLM is trained and tested.

### Training and testing the GLM

The final GLM is trained on the best performing feature set selected at the previous step. Once trained, the mean squared error and the mean absolute error (MAE) of the GLM are evaluated on the training and the testing sets. Now, the GLM is ready to calculate the pairwise identity scores; we next discuss how it can accomplish this task.

### Applying the trained GLM

At this stage, sequences are read and converted to histograms. If two sequences are not too short or not too long with respect to each other (unless the user chooses to report all pairs), then the selected statistics are calculated on their histograms and passed to the trained GLM. The GLM calculates the identity score as the weighted average of the statistics; these weights are determined during training. Sequence pairs that have identity scores above the relaxed threshold are reported. *Identity* relaxes the user-provided threshold by subtracting the testing MAE from it. Relaxing the threshold is disabled by default if the user-provided threshold is 0.90 or higher; the user has the option to use the provided threshold strictly without relaxation using a parameter.

We have completed the description of the steps that *Identity* performs in order to calculate pairwise identity scores. After that, we give the details on how we executed other related tools in order to evaluate and compare their performances to *Identity*'s performance.

### Executing *Identity* and the related tools

We chose three widely used, alignment-based tools—BLAST (13), MUMmer4 (42) and USEARCH (12)—to compare to *Identity* in searching sequence databases. Ground truth sets, on which the tools can be evaluated, were assembled. For this purpose, we chose needleall from EMBOSS (3) for performing global alignments and calculating the ground truth identity scores. The Needleman–Wunsch algorithm—without any heuristics—is implemented in this program; thus, it is slow.

While BLAST is designed for local alignment, it may also generate global alignments, as they are a special case of local alignments. It is possible to get global alignment scores by manipulating parameters and filtering out BLAST results. To obtain the global identity score, we multiply the identity score due to local alignment by the query coverage and by the subject coverage (Equation 4).

$$ID_{\text{global}} = ID_{\text{local}} \times \frac{|q\text{end} - q\text{start}|}{\text{query length}} \times \frac{|s\text{end} - s\text{start}|}{\text{subject length}} \quad (4)$$

Here, $ID_{\text{global}}$ and $ID_{\text{local}}$ are BLAST's global and local identity scores; $q$start and $q$end are the start and the end of the aligned region in the query sequence; $s$start and $s$end are the start and the end of the aligned region in the subject sequence. Table 2 shows few examples of local identity scores and their corresponding global scores due to Equation 4 and the global alignment algorithm. The adjusted scores produced by Equation 4 are very close to the scores calculated by the global alignment algorithm.

**Table 2.** Conversion from BLAST's local identity scores to global identity scores

| Subject sequence | BLAST's local ID (%) | Query coverage | Subject coverage | BLAST's global ID (%) | Global ID (%) |
|---|---|---|---|---|---|
| NM_001098570.1 | 99.1 | 0.98 | 0.98 | 95.9 | 95.2 |
| XM_003939184.2 | 92.2 | 0.99 | 0.96 | 88.8 | 85.5 |
| XM_010628605.1 | 82.4 | 0.95 | 0.94 | 74.1 | 73.6 |

The query sequence is NM_002283.3. Equation 4 is applied to BLAST's local identity score, the query coverage and the subject coverage to produce the corresponding global identity score. BLAST's global identity scores (BLAST's Global ID) are very similar to the ones produced by the global alignment algorithm (Global ID).

To get BLAST to print many alignments, the parameter 'num_alignments' can coax BLAST into printing out more alignments (1 000 000) than just the few best local alignments. The maximum adjusted alignment score of several alignments between the same sequence pair is considered as the global identity score. BLAST was executed with 16 threads. The parameters used for BLAST were '-task blastn -strand plus -perc_identity t -num_threads 16 -num_alignments 1000000 -reward 1 -penalty -1 -gapopen 2 -gapextend 1', where t is the threshold score provided by the user.

The MUMmer4's package contains a program for similar sequences (nucmer) and a program for degenerate sequences (promer). We applied nucmer when the threshold identity score was 0.7 or higher and promer otherwise. Additionally, these two programs were run using the '−−maxmatch' parameter. The program nucmer has a parameter to adjust the number of threads, therefore it was run using 16 threads. The program promer does not provide a parameter for the number of threads. We processed the outputs of nucmer and promer using a third program—show-coords—which is also included in the package. MUMmer4's output was processed in a similar fashion to BLAST's output to obtain global identity scores.

USEARCH (the free version) has limitation on the size of data it can process. To circumvent this issue, we created a multi-process module (using the GNU Parallel utility (43)) around USEARCH to allow it to use 16 threads to process

**Table 3.** Statistics of the datasets used in evaluating *Identity* and the related tools

| Dataset | Sequence count | Nucleotide count | Maximum length | Minimum length | Mean length | Median length |
|---|---|---|---|---|---|---|
| Keratin | 5 220 536 | 12 517 803 991 | 3250 | 1353 | 2398 | 2350 |
| Keratin-small | 6670 | 14 495 374 | 3249 | 1700 | 2173 | 2078 |
| P27 | 7 990 947 | 19 919 189 291 | 4000 | 1500 | 2493 | 2373 |
| Viral | 5089 | 37 326 825 | 18 406 | 2964 | 7335 | 7003 |
| Bacterial | 8725 | 33 698 914 838 | 7 304 136 | 1 826 467 | 3 862 340 | 4 016 947 |
| 16S rRNA | 1 071 335 | 269 374 512 | 372 | 171 | 251 | 256 |

small portions. Large datasets were divided into 16 equally sized, small datasets, each of which was scanned using USEARCH. The following command was executed in parallel: '-search_global -strand plus -id t -threads 1 -blast6out,' where t is the threshold identity score. Small datasets were scanned without partitioning by executing the following command: '-search_global -strand plus -id t -threads 16 -blast6out,' where t is the threshold identity score.

We chose three widely used tools—andi (26), FSWM (28) and Mash (27)—to compare to *Identity* with regard to constructing phylogenetic trees. On some sequence pairs, andi or FSWM may fail and report non-numeric values. For those pairs, we assign the maximum distance plus 0.1. The three tools were executed with their default parameters and 16 threads (except Mash reported sequence pairs with distances below a threshold distance when it was applied to searching databases). We applied Clustal Omega (44) to produce a multiple-sequence alignment that can be utilized in calculating an all-versus-all distance matrix to be used in building a reference phylogenetic tree with the fneighbor program.

*Identity* was run with 8 threads in the search mode and with 16 threads in the all-versus-all mode. Relaxing the user-provided threshold is disabled by default when the threshold identity score was 0.9 or higher.

All tools were run on the same computer (Dell Precision 3630 Tower), which has 8 cores (allowing for 16 hyper-threads), Nvidia Quadro RTX 4000 graphics card, 64 GB of RAM, 1-TB solid-state disk and two 2-TB spinning disks. This machine runs Ubuntu 18.04.

We have just finished discussing how *Identity* and the related tools were executed. Afterward, we explain how we constructed six datasets, on which the tools were evaluated.

## Datasets

Six datasets were used in evaluating the tools. Searching a sequence database for similar sequences to a query sequence is an important and common application of alignment tools and the proposed tool. We utilized three datasets as databases to be searched. Table 3 shows statistics about these sets.

The Keratin set consists of 5 220 536 sequences (the average length is 2398 bp). The P27 set consists of 7 990 947 sequences (the average length is 2493 bp). Query sequences were selected for the Keratin and the P27 sets. The Keratin query sequence is the NM_002283.3—*Homo sapiens* keratin 85 (KRT85), transcript variant 1, mRNA sequence. The P27 query sequence is the NM_004064.4—*H. sapiens* cyclin-dependent kinase inhibitor 1B. Both the P27 and the Keratin ground truth sets, i.e. similar sequences to the query sequence, were found by searching the NCBI database (45). The search parameters for Keratin were: 'srcdb_refseq[PROP] AND Keratin NOT Homo Sapiens.' This search was restricted to animal sequences between 1700 and 3250 bp in length. When gathered on 26 June 2018, this query resulted in 6669 sequences (the Keratin small dataset). To find similar sequences to the P27 query, we searched: 'srcdb_refseq[PROP] AND cyclin dependent kinase inhibitor 1B NOT *H. sapiens*.' This search was restricted to animal sequences 2000–3000 bp long, resulting in 131 sequences when gathered on 26 June 2018. After that, sequences that have <70% identity with the query sequences were removed from the ground truth sets. Finally, the query sequences were added, resulting in 59 and 68 sequences similar to the Keratin and the P27 query sequences.

The third search was performed on the viral dataset. The genome of Banana Streak CA Badnavirus (7408 bp long) was used as the query sequence. Initially, 11 988 viral genomes were obtained from virusSITE (46). Sequences that are too long or too short were removed because they can never produce the desired minimum identity score, which is 0.5, if they were to be aligned versus the query sequence. The final dataset consists of 5089 sequences (the average length is 7335 bp).

The bacterial and the 16S-rRNA datasets were used in calculating all-versus-all identity scores. We downloaded all bacterial genomes from the NCBI. Then, we selected genomes that consist of one chromosome only. After that, we calculated the average length. To produce a homogeneous dataset, sequences that are shorter than half the average length or longer than double the average length were removed, resulting in 8725 genomes with average length of 3 862 340 bp. The last dataset was obtained from a microbial study (11), focusing on the 16S rRNA gene. This dataset consists of 1 071 335 sequences. The average length is 251 bp.

Up to here, we described the computational principles behind *Identity*. Then the details of the related tools and the evaluation datasets have been given. After that, we discuss the evaluation measures.

## Evaluation measures

We evaluated the tools using the following seven measures:

- **Sensitivity:** Sensitivity is the rate of true positives (TP) to the combined TP and false negatives. It measures the ability of a tool to identify TP in a large dataset—the more TP found, the better.
- **Precision:** Precision is the ratio of TP to TP and false positives (FP). Precision measures the relevancy of returned results, since it rates TP to the total predicted positive la-

bels. This measure is very important when experimental validations of results are considered.

- **F-measure:** F-measure combines sensitivity and precision by taking the harmonic mean of them (Equation 5).

$$\text{F-measure} = 2 \times \frac{\text{Precision} \times \text{Sensitivity}}{\text{Precision} + \text{Sensitivity}} \quad (5)$$

- **MAE:** This measure is used in regression analysis to measure how close, on average, the predicted value is to the actual value. The MAE is the average absolute difference between the predicted value and the actual value. Using this metric allows a comparable benchmark for the error in estimating the identity score in relation to the one due to global alignment algorithms. Additionally, it can be used as an expected margin of error.
- **Normalized Robinson-Foulds (nRF) distance:** One of the important applications where *Identity* can be applied is building phylogenetic trees. Calculating the distance between two phylogenetic trees is usually done via the RF and the nRF metrics. The RF (Equation 6) is defined as the number of edges present in the first tree only plus the number of edges present in the second tree only (47). It represents the edit distance to change one tree into the other. The nRF (Equation 7) is obtained by dividing the RF distance by the maximum possible edit distance (the total number of edges in the two trees) for the two trees (48).

$$\text{RF} = |E_1 - E_2| + |E_2 - E_1| \quad (6)$$

$$n\text{RF} = \frac{\text{RF}}{|E_1| + |E_2|} \quad (7)$$

Here, $E_1$ and $E_2$ are the edge sets of the first tree and the second tree. The nRF distance is in the range 0–1.

- **Time:** Time reported is the wall clock time, as multithreaded applications are best estimated using real time.
- **Memory:** Maximum memory used by a tool is measured, as the memory requirement is set by the maximum amount.

We report sensitivity, precision, F-measure and MAE as percentages. The nRF is unitless. Time is measured in seconds and memory in Gigabytes (GB). Next, we evaluate the performances of the tools using these criteria.

## RESULTS

### Main contributions

The main contributions of this research are: (i) calculating identity scores in linear time and space for the first time, (ii) calculating identity scores for pairs of very long sequences for the first time and (iii) the *Identity* software tool. We are about to discuss multiple evaluation experiments, which demonstrate our contributions.

### Evaluation on large-scale datasets

Using two of the datasets described earlier, BLAST, *Identity*, Mash, MUMmer4, and USEARCH were evaluated by searching for one query sequence in a database (Table 4). We start by looking at the sensitivities of the five tools. On the Keratin set, Mash and USEARCH were the most sensitive tools (100.0%), followed by BLAST (83.1%), followed by *Identity* (74.6%) and MUMmer4 (28.8%). On the P27 set, Mash and USEARCH achieved perfect sensitivities of 100%, whereas BLAST and *Identity* achieved comparable sensitivities of 94.1 and 92.7% and MUMmer4 achieved 58.8%.

The number of FP is quite important because of the large number of sequences to be searched. *Identity* and MUMmer4 had the lowest numbers of FP on the Keratin and the P27 sets (*Identity*: 0 and 0, and MUMmer4: 0 and 2), followed by BLAST (2 and 7). Mash and USEARCH resulted in a large number of FP on these two datasets (Mash: 2112 and 183, and USEARCH: 1274 and 6641). These numbers represent very small percentages of the databases, but they are much larger than the sizes of the ground truth sets (59 and 68). For this reason, we discuss the precision metric.

*Identity* and MUMmer4 had perfect precision scores (100.0%) on the Keratin set. BLAST came second (96.1%). On the P27 set, *Identity* was the most precise tool (100.0%) followed by MUMmer4 (95.2%) and BLAST (90.1%). Because of the large numbers of FP detected by Mash and USEARCH, they were the least precise tools (Mash: 2.7 and 27.1%, and USEARCH: 4.4 and 1.0%).

On the Keratin dataset, BLAST and *Identity* came first and second in terms of the F-measure scores (89.1 and 85.4%); MUMmer4 came third (44.7%), USEARCH came fourth (8.5%) and Mash came fifth (5.3%). On the P27 dataset, *Identity* achieved the highest F-measure score (96.2%); BLAST came second (92.1%), followed by MUMmer4 (72.7%), Mash (42.6%) and USEARCH (2.0%).

In addition, we evaluated how close the identity scores produced by each of the tools were to those produced by the global alignment algorithm. *Identity* had 2–4% MAE. The error due to MUMmer4 ranged 3–5% and that due to BLAST ranged 4–5%. USEARCH error was about 7%. Mash had the highest error (about 13%) because it reports mutation distances that were converted to identity scores simply by subtracting them from 1. Keep in mind that these errors were calculated on the TP only, excluding FP.

One of the main advantages of the proposed method is its speed. *Identity* was the fastest tool on the first two sets; it was at least twice as fast as BLAST (68 versus 146 s and 89 versus 221 s). *Identity* was faster than USEARCH by 15–23 times, Mash by 20–65 times and MUMmer4 by 31–80 times. This speed was achieved with reasonable memory requirements (3.1–3.9 GB), which are readily available on average personal computers.

When it comes to searching large databases, one should pay attention to precision in addition to sensitivity, both of which are combined in F-measure. Mash and USEARCH are the most sensitive tools, whereas *Identity* and MUMmer4 are the most precise tools. *Identity* had the best F-measure score on one set and the second best on the other set. Therefore, *Identity* provides the best compromise between sensitivity and precision while being the fastest tool with reasonable memory requirements.

Here, we reported the results of searching for similar sequences in two large datasets. In the next experiment, we

**Table 4.** Evaluations of BLAST, *Identity*, Mash, MUMmer4 and USEARCH

| Dataset | Tool | TP | FP | Sensitivity | Precision | F-measure | MAE | Time (s) | Memory (GB) |
|---|---|---|---|---|---|---|---|---|---|
| Keratin | BLAST | 49 | 2 | 83.1 | 96.1 | 89.1 | 4.7 | 146 | 3.1 |
| | *Identity* | 44 | 0 | 74.6 | 100.0 | 85.4 | 4.1 | 68 | 3.9 |
| | Mash | 59 | 2112 | 100.0 | 2.7 | 5.3 | 12.8 | 1384 | 61.9 |
| | MUMmer4 with nucmer | 17 | 0 | 28.8 | 100.0 | 44.7 | 2.9 | 1451 | 7.9 |
| | USEARCH with Parallel | 59 | 1274 | 100.0 | 4.4 | 8.5 | 6.7 | 696 | 12.4 |
| P27 | BLAST | 64 | 7 | 94.1 | 90.1 | 92.1 | 3.8 | 221 | 4.8 |
| | *Identity* | 63 | 0 | 92.7 | 100.0 | 96.2 | 1.6 | 89 | 3.1 |
| | Mash | 68 | 183 | 100.0 | 27.1 | 42.6 | 12.7 | 5820 | 61.7 |
| | MUMmer4 with nucmer | 40 | 2 | 58.8 | 95.2 | 72.7 | 5.2 | 5265 | 12.6 |
| | USEARCH with Parallel | 68 | 6641 | 100.0 | 1.0 | 2.0 | 7.2 | 1546 | 19.4 |
| Viral | BLAST | 4 | 0 | 6.1 | 100.0 | 11.4 | 5.1 | 1 | 0.1 |
| | *Identity* | 33 | 1 | 50.0 | 97.1 | 66.0 | 3.1 | 18 | 0.1 |
| | Mash | 12 | 0 | 18.2 | 100.0 | 30.8 | 14.5 | 1 | 0.1 |
| | MUMmer4 with promer | 12 | 0 | 18.2 | 100.0 | 30.8 | 3.1 | 83 | 0.8 |
| | USEARCH | 17 | 1 | 25.8 | 94.4 | 40.5 | 2.4 | 172 | 0.1 |

We tested the tools' abilities to search databases for similar sequences to a query sequence. Alignment scores were generated with needleall, which is a tool for global alignment. These evaluations were conducted on three datasets: Keratin, P27 and the viral datasets. There is only one query sequence for the Keratin dataset. We searched for sequences that are at least 70% identical to the query sequence, resulting in a ground truth set consisting of 59 sequences. The P27 set has one query sequence. Sequences with at least 70% identity scores to the P27 query sequence—68 sequences—are considered TP. The viral query set includes one sequence and its ground truth set includes 66 viral sequences that have at least 50% identity scores with the query sequence. Sequences detected by a tool that do not belong to the ground truth set are considered FP. Sensitivity, precision and F-measure scores are reported as percentages. MAE is displayed as a percentage; it is measured on the TP only. Time is reported in seconds (s), and memory requirements are reported in GB.

report the results of searching for similar degenerate sequences.

### Evaluation on low-identity sequence pairs

Although alignment-free *k*-mer statistics are efficient regardless of the identity between two sequences, they often do not perform very well in comparing sequences with low alignment identity values (21). We evaluated *Identity* and the related tools on the viral dataset, which has low identity scores even among similar sequences. As expected, the sensitivities of the five tools were low. The most sensitive tool—*Identity*—achieved 50.0% followed by USEARCH, which achieved 25.8%. All tools were very precise because of the very few false positives. *Identity* achieved the highest F-measure score of 66.0%, whereas USEARCH achieved 40.5%, Mash achieved 30.8% and MUMmer4 achieved 30.8%. BLAST came last on this dataset (11.4%). With respect to the MAE, *Identity*, MUMmer4 and USEARCH had 2–3% errors and BLAST had about 5% error. As expected, Mash had the highest error of approximately 15%. With respect to speed, *Identity* came second after BLAST and Mash (18 versus 1 s); however, *Identity* was faster than MUMmer4 (18 versus 83 s) and USEARCH (18 versus 172 s). All tools required modest amounts of memory (0.1–0.8 GB). To put these results in context, the global alignment algorithm—needleall—running in parallel using GNU Parallel with 16 threads took 2 h and 8 min and required 33 GB of memory. Although there is a big room for improvement in sensitivity, the speed advantage of *Identity* is clear.

In this experiment, we focused on searching for similar degenerate sequences. In the next experiment, we evaluated *Identity* in a different application—constructing phylogenetic trees.

### Evaluation on phylogenetic trees

The purpose of this experiment is to compare the qualities of phylogenetic trees due to distance matrices produced by *Identity* and three alignment-free tools (andi, FSWM and Mash). The Keratin-small dataset was utilized in this experiment. This dataset has 6670 sequences (see Table 3 for more details). We started by removing sequences that have less than 1000 unambiguous nucleotides because andi and FSWM cannot process these sequences, reducing the number of sequences to 6384. Next, we applied Clustal Omega to obtaining a multiple-sequence alignment of these sequences. All-versus-all identity scores were calculated from the multiple alignment and converted to an all-versus-all distance matrix, which was utilized in building a reference phylogenetic tree by the neighbor-joining algorithm. After that, we applied each of the four tools to calculating all-versus-all matrices, and used these matrices in building phylogenetic trees using the same algorithm. The fives trees are provided as Supplementary Data 1. Finally, we calculated the nRF distances between each of these trees and the reference tree.

The tree due to *Identity* had the lowest nRF of 0.63 followed by those of Mash (0.75), FSWM (0.76) and andi (0.90). With respect to speed, Mash was the fastest (41 s) followed by *Identity* (119 s), FSWM (407 s) and andi (482 s). With respect to memory requirements, Mash had the least (0.15 GB), followed by *Identity* (0.16 GB), FSWM (0.87 GB) and andi (3.9 GB). To put the time and memory requirements in context, Clustal Omega took 11 127 s and utilized 5.28 GB of memory. These results demonstrate the successful application of *Identity* to constructing phylogenetic trees. *Identity* was able to produce the highest quality tree in this experiment with modest time and memory requirements.

After this experiment, we applied *Identity* to generating an all-versus-all distance matrix of more than 8000 bacterial genomes.

### Evaluation on long sequences

Comparing long sequences—millions of nucleotides long—using alignment algorithms requires a prohibitively long time. In this experiment, we show that it is possible

to calculate pairwise identity scores on bacterial genomes. We applied *Identity* to calculating the all-versus-all identity scores on the bacterial dataset, which consists of 8725 genomes (38 058 450 pairwise sequence comparisons). *Identity* finished this task in 48 h, 18 min and 17 s while requiring 37.8 GB of RAM.

We demonstrated *Identity*'s ability to compare long sequences in this experiment. In the next one, we demonstrate *Identity*'s ability to calculate billions of pairwise identity scores.

### Evaluation on billions of pairwise sequence comparisons

In order to show the scalability of this tool, we computed all-versus-all pairwise similarities on the entire 16S rRNA dataset, which consists of 1 071 335 sequences with 251 bp average length. This experiment showed that *Identity* is able to compute over 573 billion global pairwise comparisons in 13 h and 33 min (only pairs with identity scores above the threshold were reported). In sum, this experiment demonstrates the scalability of this tool to large datasets.

Now, we have demonstrated *Identity*'s accuracy, efficiency and scalability. Next, we discuss how correlated *Identity*'s scores to those calculated by the global alignment algorithm and the related widely used tools.

### *Identity*'s scores are highly correlated with the related tools' scores

We studied how similar (different) the identity scores calculated by our tool to (from) the scores calculated by the original global alignment algorithm. Using the P27 ground truth dataset (68 sequences), we calculated the Pearson's correlation coefficient between *Identity*'s scores and the scores calculated by needleall (an implementation of the Needleman–Wunsch algorithm for global alignment). We also plotted the scores calculated by the two tools versus each other (Figure 2). *Identity*'s scores were highly correlated (0.97 Pearson's correlation coefficient) to the scores calculated by the original global alignment algorithm. Additionally, we calculated the correlation coefficients between *Identity*'s scores and those calculated by BLAST, Mash, MUMmer4 and USEARCH. Again, *Identity*'s scores were highly correlated (0.94–0.95 Pearson's correlation coefficient) with those calculated by these tools. These results demonstrate that identity scores calculated by *Identity* are accurate and very similar to those obtained by widely used tools. Therefore, *Identity* represents an efficient alternative to these tools.

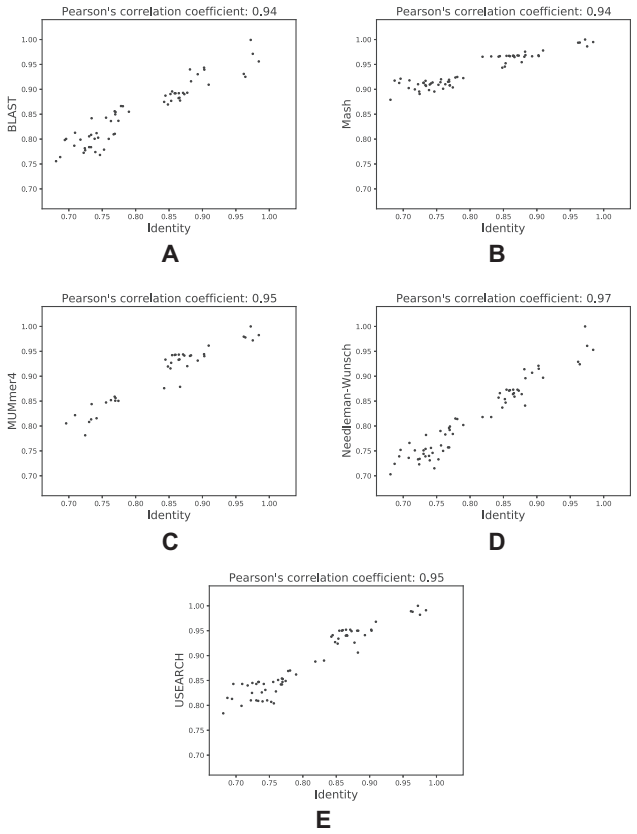



**Figure 2.** Correlations between *Identity*'s scores and those calculated by (**A**) BLAST, (**B**) Mash, (**C**) MUMmer4, (**D**) the Needleman–Wunsch global alignment algorithm and (**E**) USEARCH. These scores were obtained on the P27 ground truth dataset. *Identity*'s scores are highly correlated with the identity scores due to the global alignment algorithm and with those calculated by the other tools.

## DISCUSSION

In this section we discuss the following points: (i) the rationale of using GLMs as the core of *Identity*; (ii) analysis of time and memory requirements; (iii) potential applications of the new software tool; (iv) the effect of the semi-synthetic data size on *Identity*'s performance; (v) comments on related tools and (vi) directions for future research.

### Rationale of choosing GLMs

Several machine learning algorithms for regression are available. These algorithms include Support Vector Machines (SVMs) and Artificial Neural Networks (ANNs). We are attracted to GLMs because they are parameter-free models, which are well suited to the idea of adaptive training. We chose GLMs primarily because of the absence of parameters to optimize. Both of SVMs and ANNs can be highly accurate if given enough time to train. However, they require several parameters that need to be optimized, making these algorithms incompatible with our adaptive training idea. On the other hand, GLMs only require calculating the pseudo-inverse solution to find the linear coefficients. This operation is much cheaper than searching for optimal parameters required by the other algorithms. Our experiments show that GLMs can obtain comparable results to SVMs and ANNs—without parameter optimization however.

### Runtime and space analysis

The algorithm behind *Identity* is a linear algorithm. We start by analyzing the time required by the training stage. Then we analyze the time required by the scanning stage.

First, the size of k-mers is determined in constant time by scanning a fixed number of sequences—1000. The training stage involves: (i) generating semi-synthetic sequences, (ii) selecting features and (iii) training the final GLM. Generating a fixed number—10 000—of semi-synthetic sequences

takes constant time. Although there is a huge number of feature subsets to be selected and examined from the 1378 features, the best-first algorithm examines 15 000–30 000 subsets in practice. Currently, we do not use the number of examined sets as a convergence criterion (we use no change in mean squared error instead); however, it is straight forward to add it. For simplicity, we assume that 30 000 subsets are the maximum number to be examined; therefore, the feature-selection operation takes constant time. Training the final model takes time equivalent to examining one of the 30 000 subsets; this operation takes constant time too. Thus, the entire training process takes constant time. Next, we discuss the time requirement for the scanning stage.

To predict the identity score of a pair of sequences, the two sequences are read (linear time). Then the histograms are generated (linear time). Note that a histogram is 16 times smaller than the average sequence (see Equation 1). Calculating one of the single statistics takes linear time with respect to the size of the histogram, whereas calculating a squared statistic or a paired statistic takes constant time. Assuming that all of the 26 single statistics were selected (worst-case scenario), calculating these 26 features takes linear time with respect to the average sequence (average length $\times$ 26 $\div$ 16). Because the training stage takes constant time and the scanning stage takes linear time, the entire algorithm is a linear algorithm ($\mathcal{O}(n)$).

As a practical example, executing *Identity* in the all-versus-all mode on the bacterial dataset with threshold score of 0.8 took 48 h, 18 min and 17 s. The breakdown of the execution time is as follows: (i) generating semi-synthetic sequence pairs took 3 min and 15 s, (ii) feature selection and training the final GLM took 4 s and (iii) calculating the all-versus-all identity scores took the remaining time (48 h and 15 mins approximately). These results confirm our theoretical analysis that *Identity*'s training stage takes constant time. *Identity*'s training time should not affect the total execution time markedly when a scanned database is very large or includes very long sequences.

With regard to the space requirement, the training stage requires loading 1000 sequences in memory. Semi-synthetic sequence pairs are generated one by one; each pair is converted to two histograms, on which the 1378 features are calculated. After that the mutated sequence and the two histograms are discarded, only the features are kept. Therefore, the entire training process requires constant space to store a fixed number of sequences and a feature matrix. The space requirement for predicting an identity score of two sequences is linear because we need space only for the two sequences and their histograms.

### Potential applications

Because pairwise global alignment is such a pervasive algorithm in bioinformatics, there are several applications, in which this approximative method would be well suited. The advantage of *Identity* over both alignment-based tools and traditional alignment-free methods is that it bridges together many of the aspects that make them great: an alignment identity score is calculated, providing a meaningful similarity, and the speed at which these identities are calculated.

A common application of this method is the database search that was demonstrated earlier. When a query sequence is used for finding similar matches in a database, the search returns matches above the cutoff and their corresponding identity scores. For example, consider finding similar gene transcripts in a database of transcripts of related organisms. Secondly, an all-versus-all similarity matrix can be produced. Using this matrix, many different applications are possible. Phylogenetic tree construction is one example, which we explored earlier. Using such trees is useful in studying how different species are related to each other. Another application of the all-versus-all matrix is adjacency graph construction, where similar sequences are represented as connected nodes in the graph. An edge connecting two nodes is weighted by the identity score of the sequence pair represented by the two nodes. Adjacency graphs are useful in analyzing networks—an important research area in computational biology. Generating multiple-sequence alignments utilizes all-versus-all matrices. Further, clustering algorithms, e.g. *k*-means and mean shift (35,36), can utilize *Identity* in grouping similar sequences. Finally, this methodology was also utilized in Look4TRs (34) where it was applied to finding repeated motifs in tandem repeats. In Look4TRs, alignment-free identity scores replaced those produced by global alignment algorithms.

In sum, wherever pairwise global identity scores are required and whenever speed is a concern, *Identity* can be readily utilized.

### The effect of the semi-synthetic data size on *Identity*'s performance

*Identity* is an instance of self-supervised learning, in which an algorithm generates its own labeled training data. The tool can generate any number of semi-synthetic sequence pairs for training and testing purposes. We conducted a simple experiment to find out a reasonable size for the training and the testing datasets. In this experiment, we executed *Identity* on the Keratin-small dataset (6670 sequences) while generating semi-synthetic datasets of increasing sizes. Then we plotted the F-measure score of *Identity* versus the size of each of these datasets; we calculated the F measure scores on the real sequences not on the semi-synthetic ones. Figure 3 shows the results of this experiment. From the figure, we concluded that 10 000 (5000 for training and 5000 for testing) sequence pairs led to a reasonably high F-measure score (92.7%). Not only that, but also increasing or decreasing this number by 2000 led to comparable F-measure scores (90.7–94.6%), suggesting performance stability on different datasets.

### Related tools

Local alignment tools (other than BLAST) were not compared. Tools that are purely computational improvements such as SWIPE (a SIMD parallel optimized version of BLAST) (49) were not considered since the results other than time should be very similar to BLAST. Our rationale is that these improvements—using specialized instructions or hardware—could be applied to other tools (such as *Identity*

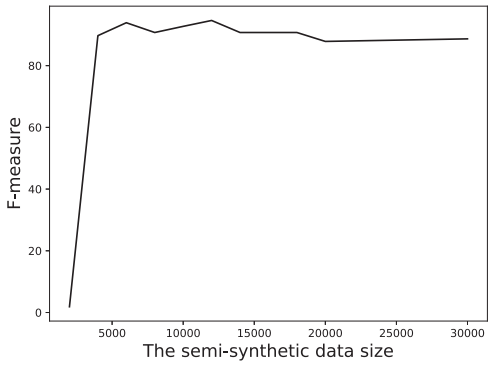



**Figure 3.** The effect of the semi-synthetic data size on *Identity*'s performance: to determine the appropriate number of the semi-synthetic sequence pairs, the combined sizes of the training and the testing datasets were plotted versus the F-measure. This curve suggests that 10 000 semi-synthetic pairs should result in high F-measure scores and performance consistency.

and USEARCH) to similarly speed up these tools with no change in output. In this way, our comparisons are focused on highlighting improvements due to algorithmic advancements. We could not compare to CaBLAST (14); although novel, the currently available proof-of-concept is too slow. CaBLAST applies BLAST to compressed representations of the sequences rather than to the sequences themselves.

### Future research directions

Currently, *Identity* only supports nucleotide sequences; however, the same *k*-mer statistics can be further applied to protein sequences. Therefore, in the next release, we plan on supporting protein sequences using the same methodology. Additionally, we plan to extend *Identity* to be able to compare large genomes consisting of multiple chromosomes as well as unassembled genomes. Finally, we will conduct additional research on how to apply *Identity* to small datasets or even a single sequence pair.

## CONCLUSION

A very important algorithm in bioinformatics, pairwise global alignment, is slow. Fast alternatives such as *k*-mer distances produce scores that do not have relevant biological meanings as the identity scores produced by alignment algorithms. We developed a novel software tool—*Identity*—for estimating identity scores of DNA sequence pairs. On an input database, *Identity* trains a self-supervised regression model to predict identity scores using few, efficient, *k*-mer statistics. Training this model is done with a novel method for generating sequences with known identity scores, *allowing for alignment-free prediction of alignment identity scores.* This is the first time identity scores are obtained in linear time using linear space.

## SUPPLEMENTARY DATA

Supplementary Data are available at NARGAB Online.

## ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their comments and suggestions, which improved the manuscript and the software.

## FUNDING

Texas A&M University-Kingsville (in part); University of Tulsa (in part); Oklahoma Center for the Advancement of Science and Technology [PS17-015, in part].
*Conflict of interest statement.* None declared.

## REFERENCES

1. Needleman,S.B. and Wunsch,C.D. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, **48**, 443–453.
2. Gotoh,O. (1982) An improved algorithm for matching biological sequences. *J. Mol. Biol.*, **162**, 705–708.
3. Rice,P., Longden,I. and Bleasby,A. (2000) EMBOSS: the European molecular biology open software suite. *Trends Genet.*, **16**, 276–277.
4. Rizk,G. and Lavenier,D. (2010) GASSST: global alignment short sequence search tool. *Bioinformatics*, **26**, 2534–2540.
5. Korf,I. (2004) Gene finding in novel genomes. *BMC Bioinformatics*, **5**, 59.
6. Butler,J., MacCallum,I., Kleber,M., Shlyakhter,I.A., Belmonte,M.K., Lander,E.S., Nusbaum,C. and Jaffe,D.B. (2008) ALLPATHS: de novo assembly of whole-genome shotgun microreads. *Genome Res.*, **18**, 810–820.
7. Luo,R., Liu,B., Xie,Y., Li,Z., Huang,W., Yuan,J., He,G., Chen,Y., Pan,Q., Liu,Y. *et al.* (2012) SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler. *Gigascience*, **1**, 18.
8. Eaton,D.A.R. (2014) PyRAD: assembly of de novo RADseq loci for phylogenetic analyses. *Bioinformatics*, **30**, 1844–1849.
9. Peled,S., Leiderman,O., Charar,R., Efroni,G., Shav-Tal,Y. and Ofran,Y. (2016) De-novo protein function prediction using DNA binding and RNA binding proteins as a test case. *Nat. Commun.*, **7**, 13424.
10. Carradec,Q., Pelletier,E., Da Silva,C., Alberti,A., Seeleuthner,Y., Blanc-Mathieu,R., Lima-Mendez,G., Rocha,F., Tirichine,L., Labadie,K. *et al.* (2018) A global ocean atlas of eukaryotic genes. *Nat. Commun.*, **9**, 373.
11. Costello,E.K., Lauber,C.L., Hamady,M., Fierer,N., Gordon,J.I. and Knight,R. (2009) Bacterial community variation in human body habitats across space and time. *Science*, **326**, 1694–1697.
12. Edgar,R.C. (2010) Search and clustering orders of magnitude faster than BLAST. *Bioinformatics*, **26**, 2460–2461.
13. Altschul,S.F., Gish,W., Miller,W., Myers,E.W. and Lipman,D.J. (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.
14. Loh,P.-R., Baym,M. and Berger,B. (2012) Compressive genomics. *Nat. Biotechnol.*, **30**, 627–630.
15. Vinga,S. and Almeida,J. (2003) Alignment-free sequence comparison–a review. *Bioinformatics*, **19**, 513–523.
16. Vinga,S., Carvalho,A.M., Francisco,A.P., Russo,L.M. and Almeida,J.S. (2012) Pattern matching through Chaos Game Representation: bridging numerical and discrete data structures for biological sequence analysis. *Algorithms Mol. Biol.*, **7**, 10.
17. Bonham-Carter,O., Steele,J. and Bastola,D. (2014) Alignment-free genetic sequence comparisons: a review of recent approaches by word analysis. *Brief. Bioinform.*, **15(6)**, 890–905.
18. Song,K., Ren,J., Reinert,G., Deng,M., Waterman,M.S. and Sun,F. (2014) New developments of alignment-free sequence comparison: measures, statistics and next-generation sequencing. *Brief. Bioinform.*, **15**, 343–353.
19. Vinga,S. (2014) Editorial: alignment-free methods in computational biology. *Brief. Bioinform.*, **15**, 341–342.
20. Chattopadhyay,A.K., Nasiev,D. and Flower,D.R. (2015) A statistical physics perspective on alignment-independent protein sequence comparison. *Bioinformatics*, **31**, 2469–2474.

21. Luczak,B.B., James,B.T. and Girgis,H.Z. (2017) A survey and evaluations of histogram-based statistics in alignment-free sequence comparison. *Brief. Bioinform.*, **20**, 1222–1237.
22. Zielezinski,A., Vinga,S., Almeida,J. and Karlowski,W.M. (2017) Alignment-free sequence comparison: benefits, applications, and tools. *Genome Biol.*, **18**, 186.
23. Zielezinski,A., Girgis,H.Z., Bernard,G., Leimeister,C.-A., Tang,K., Dencker,T., Lau,A.K., Röhling,S., Choi,J.J., Waterman,M.S. *et al.* (2019) Benchmarking of alignment-free sequence comparison methods. *Genome Biol.*, **20**, 144.
24. Haubold,B., Pfaffelhuber,P., Domazet-Losˇo,M. and Wiehe,T. (2009) Estimating mutation distances from unaligned genomes. *J. Comput. Biol.*, **16**, 1487–500.
25. Yi,H. and Jin,L. (2013) Co-phylog: an assembly-free phylogenomic approach for closely related organisms. *Nucleic Acids Res.*, **41**, e75.
26. Haubold,B., Klötzl,F. and Pfaffelhuber,P. (2014) andi: fast and accurate estimation of evolutionary distances between closely related genomes. *Bioinformatics*, **31**, 1169–1175.
27. Ondov,B.D., Treangen,T.J., Melsted,P., Mallonee,A.B., Bergman,N.H., Koren,S. and Phillippy,A.M. (2016) Mash: fast genome and metagenome distance estimation using MinHash. *Genome Biol.*, **17**, 132.
28. Leimeister,C.-A., Sohrabi-Jahromi,S. and Morgenstern,B. (2017) Fast and accurate phylogeny reconstruction using filtered spaced-word matches. *Bioinformatics*, **33**, 971–979.
29. Klötzl,F. and Haubold,B. (2019) Phylonium: fast estimation of evolutionary distances from large samples of similar genomes. *Bioinformatics*, **36**, 2040–2046.
30. Sarmashghi,S., Bohmann,K., Gilbert,M.T. P., Bafna,V. and Mirarab,S. (2019) Skmer: assembly-free and alignment-free sample identification using genome skims. *Genome Biol.*, **20**, 34.
31. Röhling,S., Linne,A., Schellhorn,J., Hosseini,M., Dencker,T. and Morgenstern,B. (2020) The number of k-mer matches between two DNA sequences as a function of k and applications to estimate phylogenetic distances. *PLoS One*, **15**, e0228070.
32. Girgis,H.Z. and Ovcharenko,I. (2012) Predicting tissue specific cis-regulatory modules in the human genome using pairs of co-occurring motifs. *BMC Bioinformatics*, **13**, 25.
33. Girgis,H.Z. (2015) Red: an intelligent, rapid, accurate tool for detecting repeats de-novo on the genomic scale. *BMC Bioinformatics*, **16**, 227.
34. Velasco Alfredo,I., James,B.T., Wells,V.D. and Girgis,H.Z. (2019) Look4TRs: a *de novo* tool for detecting simple tandem repeats using self-supervised hidden Markov models. *Bioinformatics*, **36**, 380–387.
35. James,B.T., Luczak,B.B. and Girgis,H.Z. (2018) MeShClust: an intelligent tool for clustering DNA sequences. *Nucleic Acids Res.*, **46**, e83.
36. James,B.T. and Girgis,H.Z. (2018) MeShClust2: application of alignment-free identity scores in clustering long DNA sequences. bioRxiv doi: https://doi.org/10.1101/451278, 24 October 2018, preprint: not peer reviewed.
37. Girgis,H.Z. and Corso,J.J. (2008) In: *Stp: the sample-train-predict algorithm and its application to protein structure meta-selection*. Technical report 16, The State University of New York at Buffalo, Department of Computer Science and Engineering.
38. Girgis,H.Z. (2008) In: *Machine-learning-based meta approaches to protein structure prediction*. Ph.D. Thesis, The State University of New York at Buffalo, Department of Computer Science and Engineering.
39. Girgis,H.Z., Corso,J.J. and Fischer,D. (2009) On-line hierarchy of general linear models for selecting and ranking the best predicted protein structures. In: *Conf Proc IEEE Eng Med Biol Soc*. pp. 4949–4953.
40. Girgis,H.Z. and Sheetlin,S.L. (2013) MsDetector: toward a standard computational tool for DNA microsatellites detection. *Nucleic Acids Res.*, **41**, e22.
41. Kohavi,R. and John,G.H. (1997) Wrappers for feature subset selection. *Artif. Intell.*, **97**, 273–324.
42. Marçais,G., Delcher,A.L., Phillippy,A.M., Coston,R., Salzberg,S.L. and Zimin,A. (2018) MUMmer4: a fast and versatile genome alignment system. *PLoS Comput. Biol.*, **14**, e1005944.
43. Tange,O. (2011) GNU parallel: the command-line power tool. *USENIX Mag.*, **36**, 42–47.
44. Sievers,F., Wilm,A., Dineen,D., Gibson,T.J., Karplus,K., Li,W., Lopez,R., McWilliam,H., Remmert,M., Söding,J. *et al.* (2011) Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Mol. Syst. Biol.*, **7**, 539–539.
45. O'Leary,N.A., Wright,M.W., Brister,J.R., Ciufo,S., Haddad,D., McVeigh,R., Rajput,B., Robbertse,B., Smith-White,B., Ako-Adjei,D. *et al.* (2016) Reference sequence (RefSeq) database at NCBI: current status, taxonomic expansion, and functional annotation. *Nucleic Acids Res.*, **44**, D733–D745.
46. Stano,M., Beke,G. and Klucar,L. (2016) viruSITE—integrated database for viral genomics. *Database*, **2016**, baw162.
47. Robinson,D. and Foulds,L. (1981) Comparison of phylogenetic trees. *Math. Biosci.*, **53**, 131–147.
48. Kupczok,A., Schmidt,H.A. and von Haeseler,A. (2010) Accuracy of phylogeny reconstruction methods combining overlapping gene data sets. *Algorithms Mol. Biol.*, **5**, 37–37.
49. Rognes,T. (2011) Faster Smith-Waterman database searches with inter-sequence SIMD parallelisation. *BMC Bioinformatics*, **12**, 221.