SWSPM: A Novel Alignment-Free DNA Comparison Method Based on Signal Processing Approaches

Evolutionary Bioinformatics
Volume 15: 1–11
© The Author(s) 2019
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/1176934319849071

Tomáš Farkaš[1], Jozef Sitarčík[1], Broňa Brejová[2] and Mária Lucká[1]

[1]Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava, Bratislava, Slovakia. [2]Faculty of Mathematics, Physics and Informatics, Comenius University in Bratislava, Bratislava, Slovakia.

ABSTRACT: Computing similarity between 2 nucleotide sequences is one of the fundamental problems in bioinformatics. Current methods are based mainly on 2 major approaches: (1) sequence alignment, which is computationally expensive, and (2) faster, but less accurate, alignment-free methods based on various statistical summaries, for example, short word counts. We propose a new distance measure based on mathematical transforms from the domain of signal processing. To tolerate large-scale rearrangements in the sequences, the transform is computed across sliding windows. We compare our method on several data sets with current state-of-art alignment-free methods. Our method compares favorably in terms of accuracy and outperforms other methods in running time and memory requirements. In addition, it is massively scalable up to dozens of processing units without the loss of performance due to communication overhead. Source files and sample data are available at https://bitbucket.org/fiitstubioinfo/swspm/src

KEYWORDS: DNA, alignment-free comparison, spectral transform, fast Fourier transform, distance matrix

RECEIVED: April 12, 2019. **ACCEPTED:** April 12, 2019.

TYPE: Original Research

FUNDING: The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was partially supported by the Scientific Grant Agency of Slovak Republic grant VEGA 1/0458/18 and the Slovak Research and Development Agency grant APVV-16-0484.

DECLARATION OF CONFLICTING INTERESTS: The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

CORRESPONDING AUTHOR: Mária Lucká, Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava, Ilkovičova 2, 842 16 Bratislava, Slovakia. Email: maria.lucka@stuba.sk

Introduction

Computing similarities or distances between 2 nucleotide sequences is one of the basic steps in many areas of bioinformatics. One of its possible applications is constructing phylogenetic trees that represent evolution of a group of species. For large data sets, such trees are often constructed by distance-based methods, such as Neighbor Joining,[1] UPGMA,[2] or SLINK.[3] These methods apply variants of hierarchical clustering to a distance matrix, which expresses dissimilarity between each pair of species. The distance matrix is typically computed by comparing nucleotide or protein sequences, which represent individual species. It is therefore important to develop efficient methods that can evaluate pairwise similarity or distance between pairs of sequences to provide data for phylogenetic tree reconstruction.[4] However differences between individual sequences acquired during evolution can have many forms, including substitutions of individual nu k cleotides or amino acids, insertions or deletions of shorter sequences, as well as duplications and rearrangements of sequence parts.[4,5] As a result, sequences need to be compared in a way that tolerates such changes.

Related Work

Sequence comparison methods in bioinformatics can be divided into 2 major groups: (1) alignment methods, which create local pairwise alignments of sequences and evaluate the number of substitutions in such alignments, and (2) alignment-free methods,[4,6–8] which are heuristics based mainly on statistical characteristics of data.

Established alignment-free methods include, for instance, *Word Frequencies Method*[8] (WFM) that counts the number of occurrences for all possible sequences of fixed length (*words* or *k*-mers) across the genome and then compares the resulting frequency vectors using some similarity metrics. Based on WFM, *Spaced Word Frequencies Method*[9] (SWFM) uses a different frequency vector computation. In this case, the *word* is not a short contiguous sequence, but a sequence of characters with fixed preset distances in the original genome. The word frequency vectors can be compared by various statistical approaches.[10] For example, the D_2 measure counts the number of pairs of shared word occurrences, while its variants D_2^S, D_2^*, and $D2z$ are normalized to have approximately normal distribution.[11,12] Alternatively, the word frequency vectors can be enriched with chemical properties of the nucleotides and positional information within sequence.[13]

Another well-known group of methods searches for the longest common substring[14] or common substring with fixed k mismatches[15] at each position between 2 sequences. While some approaches consider only lengths of such common matches, others inspect sequences surrounded by shared matches to estimate the number of substitutions between 2 sequences. Variations of this approach are used, for example, in *Andi*[16] and in *Filtered Spaced Words Matches* (FSWM).[17]

Another group of methods to be mentioned here is based on *Locality-Sensitive Hashing* (LSH). *Locality-Sensitive Hashing* is a type of hashing, which unlike general hash functions, aims to map similar keys to the same hash values, thus aiding in finding similar objects. Algorithms based on LSH[18] usually compute a set of hashes for each sequence and search for similarities. Recently, approaches based on MinHash technique became popular,[19] where each word of length k from the input sequence is hashed by some hash function and one or several words with the smallest hash value are used to represent the string.

The methods mentioned so far concentrate on the presence or absence of various substrings, but do not consider positions of such substrings in a sequence. The natural vector method encodes each input sequence as a vector encompassing both the frequency and positional distribution of individual nucleotides,[20] amino acids[21] or even *k*-mers.[22] The method can be further extended to consider covariances of occurrence positions.[23]

A group of emerging alignment-free methods proposed recently[24,25] uses spectral transforms instead of simple statistical quantities. Application of spectral transforms, mainly the fast Fourier transform[26] (FFT), to DNA sequences is not a new technique. Well-established multiple-sequence alignment method *MAFFT*[27] uses the Fourier transform and convolution principle for finding local matches. The process is applied to every pair of sequences, which is quite slow. Since 2014, Yin et al[24] have been using spectral transforms to relax strict dependency on positional information contained in sequences. Their methods compare raw signal spectra acquired from original sequences or compute numerical characteristics out of raw signal spectra, referred as *(Yin) Signal Moments Method*[25] (YSMM). Spectra can be also computed from hydrophobicity profiles of protein sequences[28] or further transformed by computing prefix sums.[29]

The Yin Signal Moments Method[25] is claimed to be very accurate, but it can produce misleading results when applied to sequences with varying lengths. Namely the spectral coefficients used in this method are not normalized by the input sequence length. According to Parseval theorem, the computed coefficients depend on the sum of input sequence indicators, and therefore without a proper normalization, its results reflect predominantly the input sequence length (see Figure 3 in the "Evaluation and Results" section).

Our method differs from YSMM[25] in several important aspects. First, we use a different encoding of an input DNA sequence into a vector of complex numbers, as proposed by Cheever et al.[30] As we will show, our encoding is more efficient, in some cases more accurate and has attractive theoretical properties. Second, we compute the transform in sliding windows, rather than on the whole sequence, which allows us to compare sequences that underwent rearrangements. Third, we summarize each DNA sequence into a vector of the same length as the sliding window, rather than a short vector of size 12, as in Yin method. Finally, we normalize the resulting vector by the input sequence length, which allows us to compare sequences of unequal lengths. Note that closely related sequences may differ in their lengths due to large-scale evolutionary changes (deletions, duplications, transposon insertions), but also due to incompleteness of one of the sequences. Before describing our method in more detail, we review basic principles of spectral transforms.

Background

Spectral transforms

A spectral transform is a transformation of an input numerical series of the length N representing the time or positional domain into a signal spectrum—N new numerical values in the frequency domain. Transformation itself discovers exact or approximate repeats of any size and evaluates their accuracy level. In the resulting frequency domain, the repeat distance is represented by the index in the output vector and the cumulative accuracy level by the corresponding number located in the output vector at that index. For instance, if we are interested in repeats with the period of $1/10$ of the original length, we will consider the 10th element in the frequency domain; its absolute value will reflect the exactness of the repeats. In this manner, spectral transforms can detect and represent all kinds of repetitive signals, from 3-periodic patterns in protein coding regions due to codons[24] up to differences in statistical nucleotide distribution among long sequence parts.

The discrete Fourier transform (DFT) is a well-known spectral transform defined as follows

$$F(u) = \sum_{x=0}^{N-1} f(x) e^{-i2\pi ux/N} \tag{1}$$

In this formula, $F(u)$ for $u \in \{0,\ldots,N-1\}$ is the uth element of the signal spectrum and N is the length of the input vector f. The DFT power spectrum $PS(u)$ for $u \in \{0,\ldots,N-1\}$ is then defined as

$$PS(u) = \frac{\sqrt{Re(F(u))^2 + Im(F(u))^2}}{N} \tag{2}$$

The transform approximates the input signal using a series of sine and cosine functions with periods in interval $<0, N-1>$ or, using another point of view, computes a convolution of series of sine and cosine waves with different periods and the input vector, resulting in a series of numerical values. Transformation can be efficiently computed in $O(N \log N)$ time using the FFT algorithm.[26]

Similarly, the Walsh-Hadamard transform (WHT)[31] approximates the input signal by rectangular basic functions and therefore can be computed by less computationally expensive addition and subtraction operations. A version optimized for sliding windows was also developed,[31] which may further reduce computational requirements. We have investigated WHT as a faster replacement for FFT in our experiments, but it has a lower accuracy.

Spectral transforms in discovering motifs in sequences

A DNA sequence is a nonperiodic signal with some periodic repetitive parts.[32] Because spectral transforms are intended to transform periodic signals, transforming nonperiodic signals

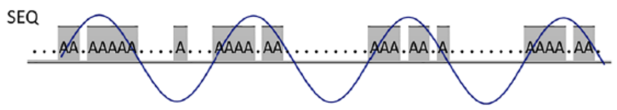

Figure 1. Visualization of a single spectral coefficient calculation from a signal sequence. In this case, the spectral coefficient for the period of 16 characters $(F(N/16))$ will have significant nonzero value as positive basis function parts are convolved with nonzero signal parts.

into signal spectra may resemble hashing one representation to another without understanding its internal structure. The intuition is that the transform helps to find variations in the density of particular characters inside a sequence. Every spectral coefficient reflects differences considering a period related to its index $(\lambda = N/u)$. Zeroth coefficient $(u = 0)$ reflects the overall sum of the input vector elements, lower coefficients (low u) consider variations with long periods, and higher coefficients analyze short periods.

A typical spectral transform is typically used for periodic signals, and therefore, a single signal spectrum is computed for the whole input data. Spectral coefficients are computed using a series of successive basis function periods; see Figure 1 showing 3½ such periods of the length $N/16$. Short motifs are, however, typically separated by random distances, not necessarily by the distance reflecting the same even period lengths. These random distributions can cause canceling interference and harm the results. If, for example, the 4 regions in Figure 1 with densely occurring A's were positioned with unequal gaps between them, which is a typical real-life scenario, the spectral analysis would give no significant results. Therefore, it is more informative to handle the sequence not as a whole, but rather by sliding windows. Sliding windows also help to detect similarities in sequences with rearrangements, which present problems for some of the other approaches.

Method Design

In this section, we propose a novel method designed to estimate similarity of DNA sequences, which is able to handle input data with varying characteristics, such as identity levels, sequence length, and the presence of rearrangements. We have also investigated several related spectral-based methods. Some of them have achieved superior accuracy on some data sets, however, with no general applicability. Therefore, the following sections are dedicated to the most robust method, *Sliding window spectral projection method* (SWSPM).

Linear-time architecture as a key to performance gains

Let us consider comparing sequences of n species, each sequence having length at most m. Our method works also when each species is represented by a set of sequences, corresponding to different genes or chromosomes; value m will then be the upper bound on the combined size of all sequences for one species. The size of the input is thus $O(nm)$ and the output distance matrix has size $O(n^2)$. Ideally, we would like to have a method with running time linear in the combined size of the input and output, that is, running in $O(nm+n^2)$ time.

Computing a full dynamic programming alignment for each pair of sequences results in $O(n^2m^2)$ running time, and even some alignment-free methods perform a separate analysis for each pair of species, resulting in $O(n^2m)$ time. Methods achieving $O(nm+n^2)$ running time typically project DNA sequence of each species into numerical vectors of a fixed size and then compare such vectors for each pair of sequences. We will refer to such approach as *Sequence projection architecture*. For example, *WFMs*[8] produce a vector of length 4^k for k typically ranging between 4 and 8. *Yin Signal Moments method*[25] uses just 12 representative numbers for each input sequence. Our method uses a longer, but still fixed-sized, vector. In this case, the time-consuming sequence analysis is performed only once per each entity.

Sliding window spectral projection method

Sliding Window Spectral Projection Method (SWSPM) is a transformation of a nucleotide sequence to a representative numerical vector of a reduced dimensionality. We start by an algorithm outline and explain individual steps in more detail in the following.

- **Input:**
- The DNA sequence $S[0:m-1] \in \{A,C,G,T\}^m$
  - Length w_l of the sliding window
  - Step w_s defining the number of positions the sliding window is shifted in each iteration

- **Output**: Vector PS_{res} of length w_l containing spectral projection of sequence S
- **Algorithm**:

```
Step 1: Vector $PS_{res}$ of length $w_l$ with zeroes
Step 2: Repeat steps 3 to 5 for each window $W_i$ of
        length $w_l$ given by sequence $S$ and step size $w_s$
Step 3: The alphabetical window $W_i$ into its numerical
        representation $num(W_i)$
Step 4: FFT, compute spectral transform $F_i$ and
        power spectrum $PS_i$ of the numerical vector
        $num(W_i)$ as defined by formulas (1) and (2).
Step 5: $PS_{res}$ by adding $PS_i$
        $PS_{res}[0:w_l-1] := PS_{res}[0:w_l-1]$
        $+PS_i[0:w_l-1]$
Step 6: all iterations, each element of the resulting
        vector $PS_{res}$ is divided by the number of
        windows $n_w$, and zeroth component
        $PS_{res}[0]$ is discarded to obtain the resulting
        spectral projection of the input sequence $S$
        of length $w_l-1$.
```

We now explain the nontrivial steps of the algorithm in more detail.

Window creation (Step 2). Given a sequence $S[0:m-1]$ of length m, window length w_l, and step size w_s, the sequence is divided into $n_w = \lfloor 1+(m-w_l)/w_s \rfloor$ windows. For $i \in \{0,\ldots,n_w-1\}$, the ith window W_i is a subsequence $S[i \cdot w_s, i \cdot w_s + w_l - 1]$. Instead of step size w_s, we will later specify overlap relative to the windows length, defined as $o = 1 - w_s / w_l$.

Numerical representation (Step 3). Spectral transforms can be computed from vectors containing real or complex numbers; therefore, a conversion from alphabetical sequence to a numerical vector is needed. Yin et al use *Binary Sequence Indicators* (BSI), creating C sequences, where C is the alphabet size of the input sequence (for DNA, $C = 4$). Sequence for character x contains a positive integer on the positions where x occurs in the original sequence and zero elsewhere. The vectors are processed separately, implying C-times higher running time.

Cheever et al[30] represent each nucleotide by a different complex root of unity, and we use the same method and we denote it as RU. In particular, we use the following mapping: $(A,C,G,T) \rightarrow (1,-1,i,-i)$. According to our experiments, permutations of this mapping give similar results. Character N, representing an unknown nucleotide, is mapped to value 0. This representation creates a single vector of length m, possibly saving time compared with 4 vectors of BSI. In our experiments, this representation also gave more accurate results than BSI.

Spectral transform (Step 4). Each window is processed separately by the selected spectral transform. The method is designed so that it works with any spectral transform; however, FFT proposed by Yin et al has performed better than both WHT and Wavelet transform with various wavelet functions.

Final processing (Steps 5 and 6). Vectors containing power spectra, one per window, are summed to a single resulting vector of length w_l, where w_l is the length of a sliding window and therefore also the length of spectral transform's result. Zeroth component $PS_{res}[0]$ reflects the overall sum of the elements in the input sequence, which may dominate, causing a false dissimilarity. This $PS_{res}[0]$ component is therefore not used and so the resulting vector has length $w_l - 1$.

To avoid problems caused by different sequence lengths, the output vectors are normalized by the number of windows processed: $PS_{res}[1:w_l-1] := PS_{res}[1:w_l-1]/n_w$.

Pairwise comparison. To produce the final distance matrix, we compare each 2 resulting vectors (spectral projections) using one of the generic distance functions.[8] Our implementation allows both Euclidean and Cosine distances, which have almost identical accuracy. Our preliminary experiments with other distance functions reduced the accuracy.

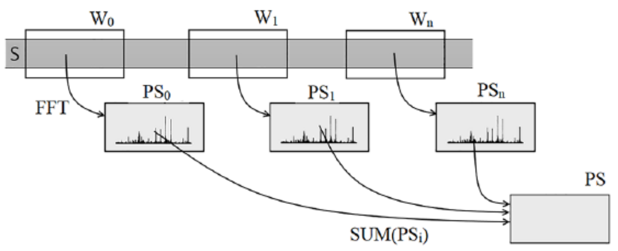

Figure 2. Sliding window spectral projection process illustrated by 3 nonoverlapping windows W_0, W_1, W_n leading to 3 signal spectra PS_0, PS_1, PS_n summed to a single PS_{res}.

Figure 2 shows the outline of the SWSPM process, excluding the pairwise comparison step. For better readability, the sliding windows are displayed without overlaps. In practice, better results are achieved using overlaps up to 87.5% of the window length.

Theoretical properties of SWSPM

In the next section, we will demonstrate the performance of our method on both simulated and real data. Here, we make some theoretical observations on our method.

The sliding window approach uses the sum of transforms computed over shorter windows. If the sequence undergoes large-scale rearrangements, most windows will only change their position in the sequence and thus will contribute equally to the sum. As a result, the distance can stay low. Nonetheless, the difference can be detected, thanks to windows containing rearrangement breakpoints.

Another important property of our method is that the distance of a sequence and its reverse complement is zero, thanks to our root of unity encoding. This is desirable, because the orientation of sequences in a dataset can be arbitrary, and longer sequences often harbor inversions of parts of the sequence.

In particular, for a single window, different sequences always have different DFT transforms, because the DFT transform is invertable. However, the same is not true for the power spectrum, which uses the magnitude of each component of the DFT. For example, if we start with sequence S, and mutually substitute nucleotides A and C in the whole sequence and also nucleotides G and T, the RU encoding yields a vector multiplied by −1 compared with the encoding of S. However, the power spectrum of the original and modified sequence will be the same. This may seem as a flaw, because such sequences would obtain distance zero, but such pairs of sequences are highly unlikely to occur in real data.

In the more relevant case, when one sequence is the reverse complement of another one, we also get the same power spectrum. Reverse complement as an operation consists of reversing the order of bases and then substituting symbols $A \Leftrightarrow T$, $C \Leftrightarrow G$. It is a well-known property of the DFT,[33] that if the input signal is reordered into reversed order, the power

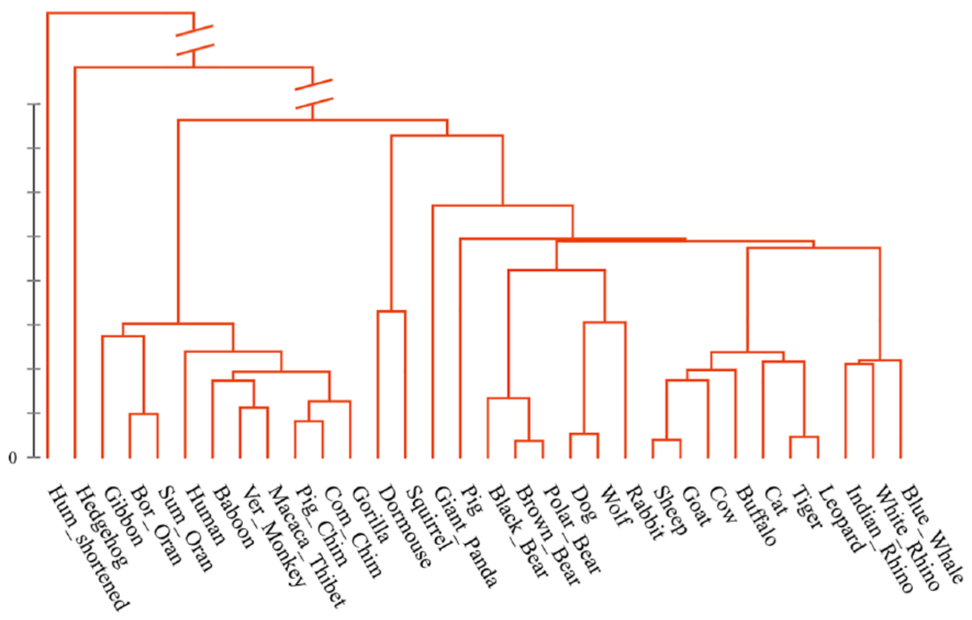

Figure 3. Phylogenetic tree constructed by YSMM on a modified Mammals dataset with a shortened human sequence added. YSMM indicates (Yin) signal moments method.

spectrum also has reversed order (except for the component $PS(0)$). Substitution of the symbols $A \Leftrightarrow T$, $C \Leftrightarrow G$ in our encoding causes that input complex numbers $a+ib$ are changed into $-b-ai$. This has the effect of reversing the order of the spectrum (except for the component $PS(0)$) and also exchanging the imaginary components and real components while inverting the signs of numbers. However, this does not influence the spectrum because the individual components are squared and then summed. The combination of these 2 operations results in the same spectrum and in the original order. This is a great advantage of our encoding in comparison to the BSI encoding, where each type of nucleotide is encoded by a separate sequence.

Computational complexity of our method

Let n be the number of species and m the sequence length per species. The spectral projection is computed once for each sequence by sliding a window with a defined length w_l and step size w_s. The running time of this step is $O(mw_s \log(w_l)/w_s)$ per species. Considering the window size to be a constant, the projection is computed in $O(m)$ time per species and $O(nm)$ overall.

The actual distance matrix computation requires $O(w_l) = O(1)$ time per pair of species or $O(n^2)$ overall. These pairwise comparisons do not consider actual DNA sequences, but rather short numerical vectors which are compared using generic distance functions. Therefore, the $O(n^2)$ part does not dominate the running time in a situation with $m >> n$.

Evaluation and Results

The proposed algorithm has been tested and compared with several existing solutions. These include the following:

- WFM[8]
- SWFM[9]
- FSWM[17]
- YSMM[25]
- Kr[14]
- Andi[16]

SWSPM was implemented in Java within ELKI framework.[34] For Kr and Andi, we have used published implementations. All other methods were reimplemented in the ELKI framework for fairer comparison of running times. Where original published implementations were available, we have verified that our implementation yields the same results. Trees were computed based on the distance matrices using the SLINK algorithm.[3] Unless noted otherwise, in most tests, we use SWSPM with window size $w_l = \min(2048, l_{seq}/8)$ and overlap $o = 87.5\%$. For WFM, we used word length equal to 6 and computed Euclidean distance between frequency vectors. For SWFM, we used filter word 1100000110000011, unless noted otherwise. These parameters had the best overall performance in initial tests on a subset of data.

Evaluation metrics

We have evaluated the accuracy using 3 metrics described in this section.

Distance matrix difference. Distance matrix difference metric compares a reference distance matrix further referred as $M^{dist,ref}$ with a distance matrix $M^{dist,res}$ given by a method being tested. Because the alignment-free methods typically do not compute a standardized measure, such as the number of substitutions per position, the magnitude of the values in output matrices varies from method to method. To overcome this problem, all matrices have been standardized to the equal mean and standard deviation (z-index). The final error score for a method is then computed as the absolute scalar difference of the distance matrix provided by a method being tested and the reference distance matrix:

$$error_{distmx} = \sum_{i=1}^{n}\sum_{j=1}^{n} | M_{i,j}^{dist,res} - M_{i,j}^{dist,ref} |,$$

where n is the number of species.

Robinson and Foulds distance. Our next goal is to evaluate similarity between the reference phylogenetic tree and the one produced by a clustering algorithm based on the distance matrix computed using the proposed spectral transform approaches. As distance matrices are often used to create phylogenetic trees, it is important to measure the influence of the matrix on the tree reconstruction. The first of 2 tree-based metrics is the Robinson and Foulds distance as described by its original authors;[35] here denoted as $error_{R-F}$. To calculate $error_{R-F}$, we used a *treedist* from *PHYLIP* package.[36]

Tree traversal length metric. The Robinson and Foulds distance does not reflect branch lengths of the tree, only the topology of the tree. Our approach is therefore to compute a new pairwise matrix , based on tree traversal of the reference tree; similarly $M^{trav,res}$ is based on the tree provided by the tested method. Each element $M_{i,j}^{trav}$ in one of these matrices corresponds to the length of the unique path connecting leaf nodes representing species i and j. The length of the path is computed as the sum of branch length provided by the phylogeny reconstruction program. However, when the branch length information is not available for the reference tree, we set all branch lengths to 1, that is, for each pair of species (i, j), we compute the number of edges on the path connecting them. Similar to the *Distance matrix difference*, the branch lengths in trees vary depending on the original distance matrix scale. Therefore, both $M^{trav,ref}$ and $M^{trav,res}$ are treated by z-index standardization. The final error score is again the absolute scalar difference of the matrices

$$error_{tree} = \sum_{i=1}^{n}\sum_{j=1}^{n} | M_{i,j}^{tree,res} - M_{i,j}^{tree,ref} |.$$

Results on simulated data

Exploration of parameter settings. We have used INDELible software[37] to generate 10 inputs, each consisting of a phylogeny with 10 species with sequences of length 10 kB. The sequences were generated under the Hasegawa-Kishino-Yano (HKY) model of substitutions with $\kappa = 2$ and 60% GC content, without indels. The tree depth was set to 0.05. We have used the correct tree from the simulation as the reference tree. Individual error metrics were recorded for each input and then averaged over all inputs. Table 1 shows the results for different parameter settings of our method and also of other methods. For WFM, we have used word lengths 4, 6, and 8, denoted in Table 1 by WFM$_a$, WFM$_b$, and WFM$_c$, respectively. For SWFM, we used words 1001001001, 1100000110000011, and 1100000110000011000011, denoted by SWFM$_a$, SWFM$_b$, and SWFM$_c$, respectively. The settings for our method are denoted by SWSPM$_a$-RU (BSI), SWSPM$_b$-RU(BSI), SWSPM$_c$-RU(BSI), and SWSPM$_d$-RU(BSI), where RU or BSI denotes nucleotide encoding and letters $a-d$ denote setting for window size w_l (set to 1024, 2048, 2048, and 4096, respectively) and for step size w_s (set to 256, 256, 512, and 1024, respectively).

By comparing SWSPM$_a$-RU, SWSPM$_c$-RU, and SWSPM$_d$-RU, we see that for these particular inputs, increasing of w_l parameter while not changing the o parameter

Table 1. The measured error values on an artificial dataset for different parameter settings.

METHOD	METRIC		
	$error_{distmx}$	$error_{tree}$	$error_{R-F}$
WFM$_a$	16.66	16.26	**0.4**
WFM$_b$	13.87	13.29	0.6
WFM$_c$	13.80	13.26	**0.4**
SWFM$_a$	16.46	17.06	1.4
SWFM$_b$	13.15	12.54	0.8
SWFM$_c$	13.65	13.09	**0.4**
FSWM	12.97	11.81	0.6
YSMM	18.87	18.74	1.6
SWSPM$_a$-RU	12.61	11.97	0.8
SWSPM$_a$-BSI	15.22	14.37	0.8
SWSPM$_b$-RU	12.09	12.04	0.8
SWSPM$_b$-BSI	14.42	13.84	0.6
SWSPM$_c$-RU	12.06	12.16	0.6
SWSPM$_c$-BSI	14.17	13.53	**0.4**
SWSPM$_d$-RU	**11.97**	12.00	0.8
SWSPM$_d$-BSI	14.53	13.72	1
Andi	12.85	**11.78**	0.6
KR	13.62	12.62	0.8

Lower values are better.
Abbreviations: BSI, binary sequence indicator; FSWM, filtered spaced word matches; SWFM, spaced word frequencies method; SWSPM, sliding window spectral projection method; WFM, word frequencies method; YSMM, (Yin) signal moment method.
The best results in each column are highligted in boldface.

Table 2. The measured error values on several artificial datasets.

DATASET	$error_{distmx}$		$error_{R-F}$	
	SWSPM	SWFM	SWSPM	SWFM
Fast 10 kB	**18.86**	19.33	**0**	0.4
Medium 10 kB	**12.06**	13.15	**0.6**	0.8
Slow 10 kB	33.35	**32.77**	1.4	**1**
Fast 1 MB	**14.52**	16.08	1	1
Medium 1 MB	14.75	**14.62**	0.6	**0.4**
Slow 1 MB	16.69	**16.44**	**0**	0.4
Indels 10 kB	37.68	**20.80**	2.6	**0.2**
Indels 1 MB	36.78	**14.10**	2	**0.4**

Lower values are better.
Abbreviations: SWFM, spaced word frequencies method; SWSPM, sliding window spectral projection method.

achieves better results. However, increasing of o parameter while not changing w_l parameter does not result in a significant difference (compare SWSPM$_b$-RU and SWSPM$_c$-RU). However, these trends are not present in all datasets.

Table 1 also shows that our method with RU nucleotide encoding achieved lower $error_{distmx}$ and $error_{tree}$ in each parameter setting than BSI nucleotide encoding. In $error_{R-F}$ measure, RU had a better or equal results than BSI in all settings except one (SWSPMc-BSI). In comparison with other methods, our method achieved very good results, having the smallest $error_{distmx}$ overall and also the third smallest $error_{tree}$ followed by Andi and FSWM. Our method achieved relatively small $error_{R-F}$ comparable to Andi, FSWM or Kr, whereas the running time of our method was several times smaller, as we note further in the following.

Influence of dataset properties. We have also investigated the accuracy of our method on datasets generated by varying settings in the INDELible software. We ran our method with default settings as described previously (window size $w_l = \min(2048, l_{seq}/8)$ and overlap $o = 87.5\%$). We compared our method to SWFM b.

Table 2 shows results on 8 datasets, each consisting of 10 inputs. The first part of the dataset name denotes the overall tree depth, with slow using 0.02, medium 0.05, and fast 0.1. Indels is a dataset with tree depth 0.05 and insertion and deletion rate set to 10% of the substitution rate. The second part of the dataset name denotes the length of the ancestral sequence, 10 kB or 1 MB. Results are averaged over 10 inputs in each dataset.

On datasets without indels, the results of our method are very similar to SWFM. On datasets with slow mutation rates, our method has a slightly higher $error_{distmx}$ than SWFM, but on datasets with medium and fast mutation rates, it achieves lower $error_{distmx}$, except on 1 MB dataset with medium mutation rate where the $error_{distmx}$ is a bit worse. Results of $error_{R-F}$ on these datasets are a bit lower for our method overall; however, this error measure can vary depending on the algorithm used to construct the tree. On datasets containing deletions and insertions, our method with default parameter setting achieved worse results for both $error_{distmx}$ and $error_{R-F}$ than SWFM.

Overall, our method has performed well on the artificial datasets, although these are in some sense most difficult for our method, because the simulation starts from a random ancestral sequence, and as a result, does not contain nonrandom patterns and periodicities typical for real biological sequence. Therefore, even if our results on datasets with indels were not as good as for SWFM, we will see that on real datasets, which include indels, our method usually performs better.

Running time and memory. We have measured the running time and memory of various methods on artificial inputs consisting of sequences of length 1 MB with substitution frequency varying between 0.04 and 0.3 without indels and rearrangements. Tables 3 and 4 show the results for the number of species n varying from 10 to 50. The experiments confirm that the running time of methods based on the sequence projection architecture grows linearly with n. These include WFM, SWFM, YSMM, and our proposed SWSPM. However, the methods performing pairwise sequence comparison, such as FSWM, proved to run in $O(n^2m)$ time. Only WFM was consistently running faster than our method, but its accuracy is usually worse.

Some of the tested methods cannot run in parallel. To provide a fair comparison, each method was tested in a forced single-core mode. Our SWSPM is based on the sequence projection architecture and therefore is scalable up to n cores, where n is the number of input sequences. Each input sequence can be handled independently of other sequences with no communication overhead, and so, the method is suitable for gigabyte-large datasets and massively parallel systems.

Results on real data

Mammalian mitochondrial genomes. This dataset contains mitochondrial genomes of 31 mammals, and it was obtained from the supplementary materials of the work by Hoang et al.[25] To meaningfully compare results with the YSMM, which is sensitive to sequence length, we have truncated all input sequences to the same length of 16 000 bp. The reference results were obtained by conventional alignment methods. In particular, we have constructed a multiple-sequence alignment using the MUSCLE algorithm[38] hosted as a Web service at the European Molecular Biology Laboratory (https://www.ebi.ac.uk/Tools/msa/muscle/) and then we have applied *distmat* from *EMBOSS* package to create a distance matrix based on the number of substitutions in the multiple-sequence alignment. The reference tree was constructed from this matrix by

Table 3. Time (s) of compared methods to process datasets of various sizes.

METHOD	THE NUMBER OF SPECIES *n*				
	10	20	30	40	50
WFM	**0.7**	**0.8**	**1.2**	**1.7**	**1.9**
SWFM	1.3	2.8	4.2	5.6	7.0
FSWM	9.4	27.0	56.7	95.1	149.7
YSMM	25.8	52.5	80.5	109.4	136.4
Andi	10.3	27.7	57.6	91.5	131.1
Kr	13.2	31.3	53.4	78.1	110.9
SWSPM	1.9	2.4	3.2	4.2	5.2

Abbreviations: FSWM, filtered spaced words matches; SWFM, spaced word frequencies method; SWSPM, sliding window spectral projection method; WFM, word frequencies method; YSMM, (Yin) signal moments method.

Table 4. Memory requirements (MB) of compared methods to process datasets of various sizes.

METHOD	THE NUMBER OF SPECIES *n*				
	10	20	30	40	50
WFM[a]	<100	<100	<100	<100	<100
SWFM[a]	<100	<100	<100	<100	<100
FSWM	191	353	525	697	869
YSMM[a]	<100	<100	<100	<100	<100
Andi	130	191	201	210	220
Kr	211	421	630	841	1051
SWSPM[a]	<100	<100	<100	110	128

Abbreviations: FSWM, filtered spaced words matches; SWFM, spaced word frequencies method; SWSPM, sliding window spectral projection method; WFM, word frequencies method; YSMM, (Yin) signal moments method.
[a]Implementation runs under JVM. Actual memory usage can be lower.

the S-LINK method. The results, shown in Table 5, demonstrate very low error rates of the SWSPM method.

Figures 3 and 4 show phylogenetic trees based on the results of YSMM and SWSPM methods on a modified Mammals dataset, which was enriched by a shortened 13 kbp long human sequence. Note that other tests and accuracy comparisons on this dataset do not use this shorter human sequence. The trees clearly show the effect of missing sequence length normalization in YSMM. The YSMM tree in Figure 3 differs from the tree from the original paper[25] because in our set, the genomes were truncated to common length of 16 kbp, whereas in the original publication, uneven lengths of sequences helped to arrange the species into the expected structure. The shortened human sequence is clustered together with the longer human sequence in SWSPM tree (Figure 4), but not in the YSMM tree.

Table 5. The measured error values on Mammals dataset.

		METRIC		
		$error_{distmx}$	$error_{tree}$	$error_{R-F}$
Method	WFM	584.95	949.25	18
	SWFM	613.11	970.92	18
	FSWM	246.57	806.27	**14**
	YSMM	351.28	822.19	30
	Andi	251.88	801.76	16
	Kr	**197.48**	756.88	**14**
	SWSPM	205.54	**734.40**	**14**

Lower values are better.
Abbreviation: FSWM, filtered spaced words matches; SWFM, spaced word frequencies method; SWSPM, sliding window spectral projection method; WFM, word frequencies method; YSMM, (Yin) signal moments method.

Fungal mitochondrial and nuclear genomes. We have considered 2 datasets consisting of sequences from fungal genomes. The first dataset contains DNA sequences of 7 genes (namely *ATP6, ATP8, ATP9, COB, COX1, COX2*, and *COX3*) from mitochondrial genomes of 10 fungal species. The species were selected from a wider dataset used in the work of Valach et al[39] to be as dissimilar as possible. In particular, we have used *Aspergillus niger, Candida zemplinina, Yarrowia lipolytica, Candida subhashii, Candida albicans, Candida neerlandica, Candida frijolesensis, Candida parapsilosis, Saccharomyces pastorianus*, and *Saccharomyces cerevisiae*. The reference phylogenetic tree was built by PhyML[40] under the JTT model from protein sequences of these species, as described by Valach et al.[39]

The second dataset again uses a subset of species from Valach phylogenetic tree,[39] but each species was represented by its full nuclear genome of the length 14 to 13 Mbp. Sequences were obtained from the National Center for Biotechnology Information (NCBI) database (https://www.ncbi.nlm.nih.gov/). We have used *Aspergillus niger, Debaryomyces hansenii, Candida maltosa, Candida albicans, Candida dubliniensis, Candida sojae, Candida tropicalis, Candida parapsilosis, Saccharomyces pastorianus*, and *Saccharomyces cerevisiae*. On this dataset, we have decreased the overlap parameter to -125%. For the tree traversal metric, we have replaced all branch lengths by 1, because the reference tree is based on mitochondrial genes, which may evolve at a different rate.

As is shown in Tables 6 and 7 and Figure 5, our method again performs very well on these datasets. Note that these 2 datasets differ widely: while the mitochondrial data set contains only short relatively well-conserved protein-coding sequences, the nuclear dataset contains full-length genomes with a wide variety of evolutionary changes including large-scale rearrangements. Figure 5 shows the phylogenetic tree constructed on the nuclear dataset. This tree almost perfectly

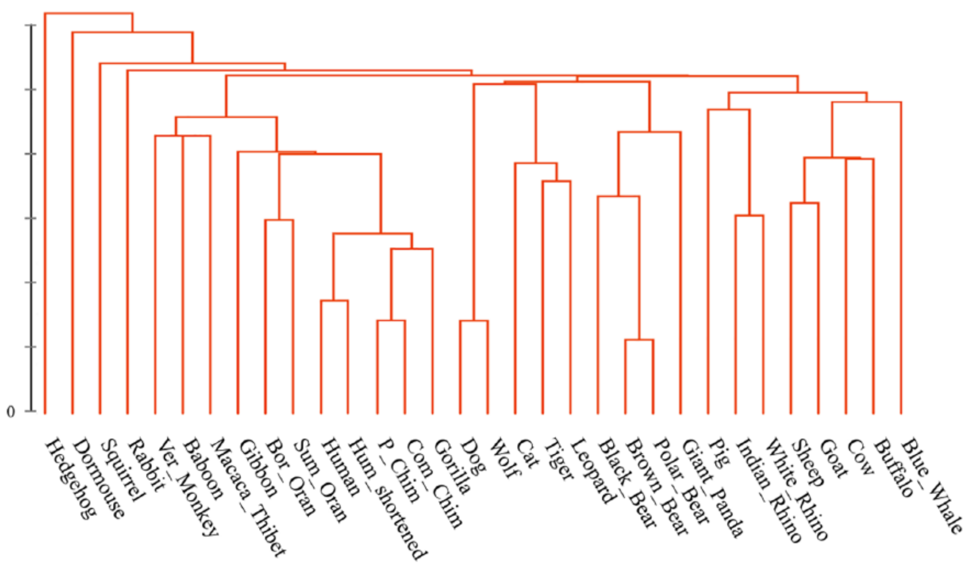

Figure 4. Phylogenetic tree constructed by SWSPM on a modified Mammals dataset with a shortened human sequence added. SWSPM indicates sliding window spectral projection method.

Table 6. The measured error values on the fungal mitochondrial genes.

		METRIC		
		$error_{distmx}$	$error_{tree}$	$error_{R-F}$
Method	WFM	n/a	114.61	8
	SWFM	n/a	109.98	6
	FSWM	n/a	113.59	8
	YSMM	n/a	144.99	12
	Andi	n/a	143.31	14
	Kr	n/a	142.55	8
	SWSPM	n/a	**88.73**	**2**

Lower values are better.
Abbreviations: FSWM, filtered spaced words matches; SWFM, spaced word frequencies method; SWSPM, sliding window spectral projection method; WFM, word frequencies method; YSMM, (Yin) signal moments method.

Table 7. The measured error values on nuclear fungal genomes.

		METRIC		
		$error_{distmx}$	$error_{tree}$	$error_{R-F}$
Method	WFM	n/a	108	10
	SWFM	n/a	134	10
	FSWM	n/a	98	**2**
	YSMM	n/a	170	14
	Andi	n/a	190	14
	Kr	n/a	160	10
	SWSPM	n/a	**92**	4

Lower values are better.
Abbreviations: FSWM, filtered spaced words matches; SWFM, spaced word frequencies method; SWSPM, sliding window spectral projection method; WFM, word frequencies method; YSMM, (Yin) signal moments method.

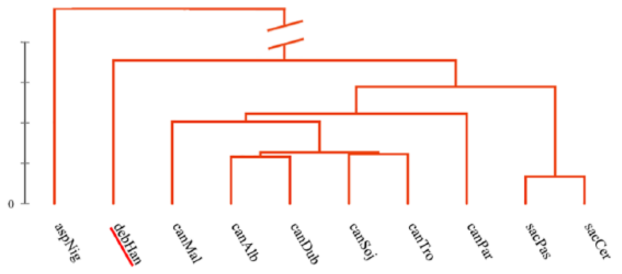

Figure 5. Phylogenetic tree constructed by SWSPM on dataset of nuclear fungal genomes. A single badly placed species is underlined. SWSPM indicates sliding window spectral projection method.

matches the published tree obtained by traditional alignment-based methods, with the exception of one species.

Bacterial genomes. This dataset was previously used by Domazet-Lošo and Haubold[14] from where we also get the reference phylogenetic tree. Input sequences were obtained from the NCBI database. Table 8 shows that our method achieves results similar to WFM, SWFM, and better than YSMM. Compared to our method, FSWM and Andi achieved higher accuracy; however, their running time is much higher than our method. Also note that Kr could not be run on our machine due to some input error, so we could not calculate exact $error_{tree}$; however, we got $error_{R-F}$ from their paper.[14]

HIV genomes. This dataset consists of 825 HIV strains from Wu et al[41] and it was also used by Domazet-Lošo and Haubold[14] For our method. we used default parameter settings (window size $w_l = \min(2048, l_{seq}/8)$ and overlap $o = 87.5\%$). However, we have found that on this dataset, characterized by a high number of relatively short sequences, it is better to use a different tree construction method, namely hierarchical clustering based on Ward[42] linkage and Manhattan distance. The

Table 8. The measured error values on the bacterial dataset.

METHOD	METRIC		
	$error_{distmx}$	$error_{tree}$	$error_{R-F}$
WFM	n/a	108.13	10
SWFM	n/a	110.67	10
FSWM	n/a	83.70	**0**
YSMM	n/a	111.95	14
Andi	n/a	**64.47**	**0**
Kr	n/a	—	2
SWSPM	n/a	128.22	10

Lower values are better.
Abbreviations: FSWM, filtered spaced words matches; SWFM, spaced word frequencies method; SWSPM, sliding window spectral projection method; WFM, word frequencies method; YSMM, (Yin) signal moments method.

Table 9. The measured error values on the Drosophila dataset.

METHOD	METRIC		
	$error_{distmx}$	$error_{tree}$	$error_{R-F}$
WFM	n/a	**76.49**	12
SWFM	n/a	77.96	**10**
FSWM	n/a	—	—
YMMM	n/a	—	—
Andi	n/a	101.01	**10**
Kr	n/a	—	—
SWSPM	n/a	100.4	14

Lower values are better.
Abbreviations: FSWM, filtered spaced words matches; SWFM, spaced word frequencies method; SWSPM, sliding window spectral projection method; WFM, word frequencies method; YSMM, (Yin) signal moments method.

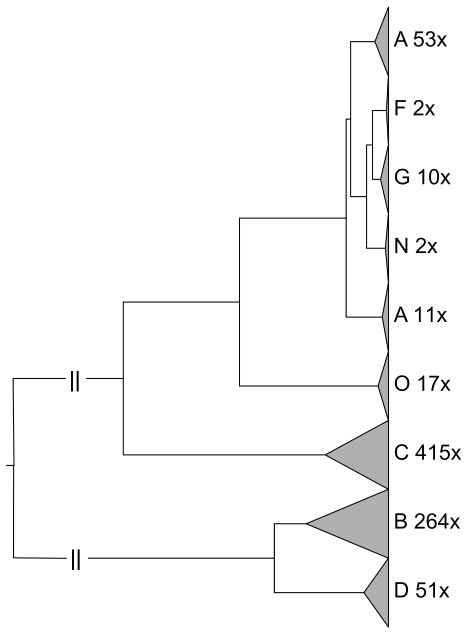

Figure 6. Phylogenetic tree constructed by SWSPM on 825 HIV genomes. SWSPM indicates sliding window spectral projection method.

resulting phylogenetic tree is shown in Figure 6. Strains from each HIV subtype were clustered together, except for subtype *A*, which was split into 2 groups.

Drosophila genomes. This dataset contains 12 different genomes of Drosophila.[43] Dataset was obtained from the download section of The UCSC Genome Browser (https://genome.ucsc.edu/index.html). The reference phylogenetic tree was taken from Haubold et al.[44] All contigs from one species were concatenated into one sequence and unknown bases were deleted. Due to high memory requirements and restricted memory of our testing machine (16 GB RAM), FSWM, YSMM, and Kr could not be run. Thus, the results of this dataset emphasize low memory requirements of our method. We could run Andi on this dataset only with usage of parameter $-l$, which sacrifices speed for lower memory requirements resulting in much longer running time than that of our method and WFM. Lowest $error_{tree}$ was achieved by WFM, whereas our method with our default parameter settings was better than Andi; however, Andi achieved lower $error_{R-F}$ (Table 9).

Conclusions

We propose a novel method for evaluation of pairwise similarity of DNA sequences. The method is based on spectral transforms and proves viability of this concept by outperforming current state-of-art alignment-free methods in most scenarios, as shown in our experiments.

In the process of designing this method, we had considered numerous variants involving spectral transform type, DNA sequence representation type, window lengths, and overlap as well as completely different windowing schemes. The final combination performs well on a variety of data sets and takes into account various domain-specific data properties, such as varying sequence lengths and genome rearrangements.

Author Contributions

ML and TF have conceived the study and designed the algorithms. BB have provided her biological knowledge and experience in designing experiments. TF and JS have implemented the software and conducted the experiments. All authors have participated in manuscript preparation and design of experiments. All authors read and approved the final manuscript.

REFERENCES

1. Saitou N, Nei M. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol Biol Evol*. 1987;4:406–425.
2. Gronau I, Moran S. Optimal implementations of Upgma and other common clustering algorithms. *Inf Process Lett*. 2007;104:205–210.
3. Sibson R. SLINK: an optimally efficient algorithm for the single-link cluster method. *Comput J*. 1973;16:30–34.

4. Haubold B. Alignment-free phylogenetics and population genetics. *Brief Bioinform*. 2014;15:407–418.
5. Borrayo E, Mendizabal-Ruiz EG, Velez-Perez H, Romo-Vazquez R, Mendizabal AP, Morales JA. Genomic signal processing methods for computation of alignment-free distances from DNA sequences. *PLoS ONE*. 2014;9:e110954.
6. Zielezinski A, Vinga S, Almeida J, Karlowski WM. Alignment-free sequence comparison: benefits, applications, and tools. *Genome Biol*. 2017;18:186.
7. Schwende I, Pham TD. Pattern recognition and probabilistic measures in alignment-free sequence analysis. *Brief Bioinform*. 2014;15:354–368.
8. Vinga S, Almeida J. Alignment-free sequence comparison-a review. *Bioinformatics*. 2003;19:513–523.
9. Leimeister CA, Boden M, Horwege S, Lindner S, Morgenstern B. Fast alignment-free sequence comparison using spaced-word frequencies. *Bioinformatics*. 2014;30:1991–1999.
10. Song K, Ren J, Reinert G, Deng M, Waterman MS, Sun F. New developments of alignment-free sequence comparison: measures, statistics and next-generation sequencing. *Brief Bioinform*. 2013;15:343–353.
11. Reinert G, Chew D, Sun F, Waterman MS. Alignment-free sequence comparison (I): statistics and power. *J Comput Biol*. 2009;16:1615–1634.
12. Kantorovitz MR, Robinson GE, Sinha S. A statistical method for alignment-free comparison of regulatory sequences. *Bioinformatics*. 2007;23:i249–255.
13. Bao J, Yuan R, Bao Z. An improved alignment-free model for DNA sequence similarity metric. *BMC Bioinformatics*. 2014;15:321.
14. Domazet-Lošo M, Haubold B. Efficient estimation of pairwise distances between genomes. *Bioinformatics*. 2009;25:3221–3227.
15. Leimeister CA, Morgenstern B. KMACS: the k-mismatch average common substring approach to alignment-free sequence comparison. *Bioinformatics*. 2014;30:2000–2008.
16. Haubold B, Klotzl F, Pfaffelhuber P. Andi: fast and accurate estimation of evolutionary distances between closely related genomes. *Bioinformatics*. 2015;31:1169–1175.
17. Leimeister CA, Sohrabi-Jahromi S, Morgenstern B. Fast and accurate phylogeny reconstruction using filtered spaced-word matches. *Bioinformatics*. 2017:971–979.
18. Buhler J. Efficient large-scale sequence comparison by locality-sensitive hashing. *Bioinformatics*. 2001;17:419–428.
19. Ondov BD, Treangen TJ, Melsted P, et al. Mash: fast genome and metagenome distance estimation using MinHash. *Genome Biol*. 2016;17:132.
20. Yu C, Hernandez T, Zheng H, et al. Real time classification of viruses in 12 dimensions. *PLoS ONE*. 2013;8:e64328.
21. Li Y, Tian K, Yin C, He RL, Yau SST. Virus classification in 60-dimensional protein space. *Mol Phylogenet Evol*. 2016;99:53–62.
22. Zhang YY, Wen J, Yau SST. Phylogenetic analysis of protein sequences based on a novel k-mer natural vector method [published online ahead of print September 5, 2018]. *Genomics*. doi:10.1016/j.ygeno.2018.08.010.
23. Zhao X, Tian K, He RL, Yau SST. Convex hull principle for classification and phylogeny of eukaryotic proteins [published online ahead of print December 5, 2018]. *Genomics*. doi:10.1016/j.ygeno.2018.11.033.
24. Yin C, Chen Y, Yau SST. A measure of DNA sequence similarity by Fourier transform with applications on hierarchical clustering. *J Theor Biol*. 2014;359:18–28.
25. Hoang T, Yin C, Zheng H, Yu C, Lucy He R, Yau SS. A new method to cluster DNA sequences using Fourier power spectrum. *J Theor Biol*. 2015;372: 135–145.
26. Cooley JW, Tukey JW. An algorithm for the machine calculation of complex Fourier series. *Math Comput*. 1965;19:297–301.
27. Katoh K, Toh H. Recent developments in the MAFFT multiple sequence alignment program. *Brief Bioinform*. 2008;9:286–298.
28. Yin C, Yau SST. A coevolution analysis for identifying protein-protein interactions by Fourier transform. *PLoS ONE*. 2017;12:e0174862.
29. Dong R, Zhu Z, Yin C, He RL, Yau SST. A new method to cluster genomes based on cumulative Fourier power spectrum. *Gene*. 2018;673:239–250.
30. Cheever EA, Overton GC, Searls DB. Fast Fourier transform-based correlation of DNA sequences using complex plane encoding. *Comput Appl Biosci*. 1991;7:143–154.
31. Ouyang W, Cham WK. Fast algorithm for Walsh Hadamard transform on sliding windows. *IEEE Trans Pattern Anal Mach Intell*. 2010;32:165–171.
32. Yin Changchuan . Identification of repeats in DNA sequences using nucleotide distribution uniformity. *J Theor Biol*. 2017;412:138–145.
33. Wong WM. *Discrete Fourier Analysis*. Basel, Switzerland: Springer; 2011.
34. Schubert E, Koos A, Emrich T, Züfle A, Schmid KA, Zimek A. A framework for clustering uncertain data. *Proc VLDB Endow*. 2015;8:1976–1979.
35. Robinson DF, Foulds LR. Comparison of phylogenetic trees. *Math Biosci*. 1981;53:131–147.
36. Felsenstein J. PHYLIP—phylogeny inference package (Version 3.2). *Cladistics*. 1989;5:164–166.
37. Fletcher W, Yang Z. INDELible: a flexible simulator of biological sequence evolution. *Mol Biol Evol*. 2009;26:1879–1888.
38. Edgar RC. MUSCLE: a multiple sequence alignment method with reduced time and space complexity. *BMC Bioinform*. 2004;5:113.
39. Valach M, Farkas Z, Fricova D, et al. Evolution of linear chromosomes and multipartite genomes in yeast mitochondria. *Nucleic Acids Res*. 2011;39: 4202–4219.
40. Guindon S, Gascuel O. A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. *Syst Biol*. 2003;52:696–704.
41. Wu X, Cai Z, Wan XF, Hoang T, Goebel R, Lin G. Nucleotide composition string selection in HIV-1 subtyping using whole genomes. *Bioinformatics*. 2007;23:1744–1752.
42. Ward JH Jr. Hierarchical grouping to optimize an objective function. *J Am Stat Assoc*. 1963;58:236–244.
43. Clark AG, Eisen MB, Smith DR, et al. Evolution of genes and genomes on the Drosophila phylogeny. *Nature*. 2007;450:203–218.
44. Haubold B, Pfaffelhuber P, Domazet-Loso M, Wiehe T. Estimating mutation distances from unaligned genomes. *J Comput Biol*. 2009;16:1487–1500.