# *msktuple: An Integrated R Library for Alignment-Free Multiple Sequence k-Tuple Analysis*

Saeid Amiri*

*University of Wisconsin-Green Bay, Department of Natural and Applied Sciences, Green Bay, WI, USA*

Ivo D. Dinov

*Statistics Online Computational Resource (SOCR), Health Behavior and Biological Sciences, Computational Medicine and Bioinformatics, University of Michigan, Ann Arbor, MI, USA*

## Abstract

Recently alignment-free sequence comparison methods based on promoter-frequency distance measures have gained popularity. This paper reports on the implementation and validation of several alignment-free sequence analysis methods for representing and quantifying between-sequence distances and sequence variability. The *msktuple* library includes the following sequence comparison techniques: locational k-tuple, naive k-tuple, CV-Tree, and their ensemble variants. These metrics are used to determine the dissimilarity between sequences using k-letter words. In support of *open-science*, we provide open-source software, R-scripts, and protocols implementing the new techniques. These tools will support collaboration, enable independent validation, promote result reproducibility and enable tool interoperability.

*Keywords:* Clustering, Dissimilarity, Ensembling methods, Multiple Alignment-free, k-tuple.

*2010 MSC:* 62Gxx, 62H10

*Corresponding author

*Email addresses:* saeid.amiri1@gmail.com (Saeid Amiri*), dinov@umich.edu (Ivo D. Dinov)

## 1. Introduction

The use of genomics data has rapidly increased over the past decade partly due to the enormous computational advances, technological modernization, substantial cost-reductions, and the broad proliferation of modern and Cloud services [1], [2] and [3]. The recent data explosion reflects substantial increases in the size, complexity, heterogeneity, scale, incongruence of observed, streamed, and multi-source digital information. The growth of data parallels the enormous methodological, analytic and visualization developments [4] and [5]. Big Healthcare Data serve as a powerful proxy for various observed phenotypes and diagnostic traits, enable forecasting, modeling and prediction of disease prevalence, and impact our understanding of human health and disease. Typically, such big biomedical and health data are high-dimensional and finding underlying patterns, associations, latent relations, causal effects, etc. are challenging. In addition, the computational burden of processing complex multisource data is challenging and often encounters degenerative-statistics, violations of parametric assumptions, lack of convergence, and ultimately biased scientific inference. Novel, efficient, and reliable techniques are necessary to cope with the increase of the data volume and complexity.

Sequence comparison is an essential step in many genomics studies because genetic sequences are expected to be highly homologous, however, they are observed, recorded and analyzed as large and fragmented bundles of short reads that need to be preprocessed prior to their interpretation. Traditional sequence analysis techniques use classical cleaning, preprocessing and alignment to reference (atlas) sequences, which eventually enables measuring, quantification, and characterization of genetic differences, variability, and complexity, as well as genome-wide association studies.

Many biomedical, translational and clinical studies involve genetic sequence comparison as the paramount initial preprocessing step. Over a dozen such sequence comparison techniques have been proposed, implemented and validated, see [6], [7], and [8], among the others. Applications include functional annotation, phylogenetic studies, and assessments of disease risk. However, it is hard to agree on a unique representation of a canonical distance metric because sequence distances are not salient features and may depend on different targets. This problem has attracted significant attention in the computational sciences, mainly driven by problems associated with the deep understanding of the structure of biological sequences such as DNA, RNA, and proteins. Such biological structures can be represented by unidimensional sequences defined over a specific alphabet. Structural homologies between genomics sequences correspond to similar features as well as the functionality of the enzymes or proteins they represent. Many common genetic features may be shared across different species, which reflects common evolutionary and functional mechanisms. Hence, researchers are looking for definitions of robust and efficient distance metrics defined on genetic sequences that are able to identify, quantify and profile these common analogues. Categorical statistical methods are necessary to obtain reliable statistical inference based on local genomics data typically stored as linked lists of categorical values, {A, T, C, G}.

Globally, it is hard to find a short signature vector representing the overall genomics sequence characteristics. Hence, examining the empirical statistical distribution of sequence data is important and may provide complementary information to local base-pair measurements. The traditional methods for comparing biological sequences are mostly based on an initial sequence alignment process, which fragments the data sequence to make it homologous to a reference (target) sequence using various string matching algorithms and specific cost functions. Some alignment-free sequence comparison methods have recently been introduced based on promoter frequency dissimilarity (distance) measures [3]. Most sequence aligners consider only local variations in the genome, which may not be suitable for measuring events, motifs or mutations that involve longer segments of genomics arrangements. Furthermore, the aligning algorithms are very time-consuming for large-scale data. For these reasons, alignment-free dissimilarity measures are potentially more attractive. Sequences are often represented by word-count vectors, and subsequent statistical inference relies on similarity scores defined for such feature vectors.

Since these bases are not randomly distributed, it is natural to count the number of specific k-letter words (k-tuples) or any possible patterns that a pair of sequences have in common. Most alignment-free methods are based on word-counting, which considers the frequency of joint neighboring words without any correction of their locations in the sequence. Our approach is based on tracking the position of nucleotides in the sequence and using the underlying distribution of the k-tuples to recognize homologous genomics segments.

The organization of this paper is as follows: Section 2 presents the alignment-free analysis technique using neighborhood base pair frequency and sequence dissimilarity(distance)s, including naive k-tuple, CV-Tree, and locational k-tuple. Section 3 described the software toolkit implementation details included in the *msktuple* R library. It also contains some results of using nine different mammalian genomes obtained from NCBI database [9]. Finally, Section 4 presents conclusions and discussions of the broader impacts of this approach.

## 2. Alignment-Free Sequence Analysis

Consider two sequences, $S_1$ and $S_2$, with different lengths, $L_1$ and $L_2$, that can be represented

$$\begin{aligned} S_1 &= s_{11} \ldots s_{1L_1}, \\ S_2 &= s_{21} \ldots s_{2L_2}, \end{aligned}$$

where $s_{1i}, s_{2j} \in \{ATCG\}$, $i = 1, \ldots, L_1, j = 1, \ldots, L_2$. We are looking for dissimilarity (distance) metrics $D(S_1, S_2)$ capable of discriminating and identifying the similarities and differences between these two sequences. For $n$ sequences, we can compute a paired distance matrix where each cell entry represents the corresponding pairwise dissimilarity (distance) $D(.,.)$. This *dissimilarity* matrix represents the dissimilarities between the $n$ sequences. We relax the distance-metric requirement for the measure to satisfy the triangle inequality, and thus, it may be more appropriate to refer to these metrics as *dissimilarity measures*, although it's shorter and more convenient to call them distances.

The direct application of $D(.,.)$ is in the phylogenetic tree is often presented using dendrogram clustering methods. These clustering methods represent unsupervised techniques for identifying natural classes within a set of data. The main idea is to group unlabeled data into subsets where the within-group subsequences are fairly homogeneous (in terms of their paired dissimilarity measures), whereas the between-group sequences exhibit more heterogeneity. By using a dissimilarity (distance) matrix, we can design a hierarchical algorithm that may be used to combine clusters and thereby obtain a phylogenetic tree.

The result of this algorithm is a dendrogram, which provides a way to explore the resulting hierarchical classification. In order to achieve a hierarchical clustering, linkage-based algorithms (e.g., average linkage) may be used, see [10].

### *2.1. Dissimilarity via k-tuple*

Let $p_{S_1i}$ and $p_{S_2i}$, $i = 1, \ldots, 4$ be the relative frequency of $A$, $T$, $C$ and $G$ in the sequences $S_1$ and $S_2$, respectively. The dissimilarity between two sequences can be calculated based on the Euclidean distance:

$$D(S_1, S_2) = \sqrt{\sum_{i=1}^{4} (p_{S_1i} - p_{S_2i})^2}.$$

Alternative metrics can be used as well, e.g., the more general Minkowski distance, $D(S_1, S_2) = \left(\sum_{i=1}^{4} (p_{S_1i} - p_{S_2i})^c\right)^{\frac{1}{c}}$.

Instead of using single nucleotides, one may consider a short word length $\ell$ and map each sub-sequence of length $L_1$, length of $S_1$ into vectors of length $\ell$ to assess the similarity of sequences, which is referred to as k-tuple. For the k-tuple with 2 sliding windows, there are $4^2$ combinations, i.e., $\{AA, AT, \ldots, GG\}$; and for $\ell$ sliding windows, there are $4^\ell$ combinations. Let's define a string of length $\ell$ at the location $i$ as $S_1[i, i + \ell - 1]$. Then all possible (or appropriate) k-tuples are defined by:

$$\mathcal{K}^\ell = \{\mathcal{K}_1, \ldots, \mathcal{K}_{\mathfrak{L}}\}, \tag{1}$$

where $\mathfrak{L} = 4^{\ell}$, define the count of them by

$$\begin{aligned} \nu_{1i} &= \{j : S_1[j \ldots j + \ell - 1] = \mathcal{K}_i\}, \qquad i \in \{1, \ldots, \mathfrak{L}\}, \\ \upsilon_{1i} &= |\, \nu_{1i} \,|, \end{aligned}$$

where $|\,.\,|$ is the cardinality of a set, in this case the number of elements of $\nu_i$. Thus, the relative frequency can be expressed as:

$$p_{S_1 i} = \frac{\upsilon_{1i}}{L_1 - \ell}, \qquad i \in \{1, \ldots, \mathfrak{L}\},$$

where $L_1$ is the length of sequence. The $p_{S_2 i}$ can be obtained using the same procedure. Using these relative frequencies as proxies of the corresponding probabilities, the between-sequence dissimilarity can be obtained as follows:

$$D(S_1, S_2) = \sum_{i=1}^{\mathfrak{L}} \left(p_{S_1 i} - p_{S_2 i}\right)^2. \tag{2}$$

This is referred to as the naive k-tuple. We use the (normalized) average frequency dissimilarity measure

$$D(S_1, S_2) = \frac{1}{\mathfrak{L}} \sum_{i=1}^{\mathfrak{L}} \left(p_{S_1 i} - p_{S_2 i}\right)^2. \tag{3}$$

### *2.2. Dissimilarity via CV-Tree*

Unlike the approach proposed in the previous section, [11] calculates the paired sequence correlation,

$$\rho(S_1, S_2) = \frac{\sum_{i=1}^{\mathfrak{L}} p_{S_1 i} p_{S_2 i}}{\sqrt{\sum_{i=1}^{\mathfrak{L}} (p_{S_1 i})^2 \sum_{i=1}^{\mathfrak{L}} (p_{S_2 i})^2}}.$$

Yet another dissimilarity $D(S_1, S_2)$ between the two sequences may be defined as

$$D(S_1, S_2) = \frac{1 - \rho(S_1, S_2)}{2}.$$

### *2.3. Dissimilarity via Location based k-tuple*

Prior work [3] discussed a new alignment-free methods in terms of the location of nucleotides or elements of k-tuple in the sequences. Let us consider a sequence of length $L$ where each nucleotide $A$, $T$, $C$ and $G$ can appear in different spots within sequence. Denote the locations of these nucleotides as $L_A = \{a_1, a_2, \ldots, a_{n_A}\}$, $L_T = \{t_1, t_2, \ldots, t_{n_T}\}$, $L_C = \{c_1, c_2, \ldots, c_{n_C}\}$ and $L_G = \{g_1, g_2, \ldots, g_{n_G}\}$, respectively. The information of the location can be used in the k-tuple elements as shown below.

To calculate the sequence dissimilarity, let's consider the nucleotide $A$ and its location, $L_A$ and define:

$$\mathcal{L}_r = \frac{1}{a_r - a_{r-1}}, \quad r = 1, \ldots, n_A, \tag{4}$$

where $\mathcal{L}_r \in (0,1)$, $a_0 = 0$, and clearly $a_r - a_{r-1} \neq 0$, $r = 1, \ldots, n_A$. $\mathcal{L}_r$ is the difference of sequential locations. Similarly, $\mathcal{L}_r$ can be defined for $T$, $C$ and $G$.

The key issue of the alignment-free methods is to consider various sequence characteristics, extract appropriate features and generate appropriate statistics that capture sufficient information representing the intrinsic characteristics of the genome. [3] explored several choices of dissimilarities and suggested using the reciprocal metric $\mathcal{L}'_r$

$$\mathcal{L}'_r \quad = \quad a_r - a_{r-1} \in (1, L_A).$$

To get complete information about the sequence, we should find $\mathcal{L}'_r$ for $i \in \mathcal{K}^\ell$. Therefore, it may be more appropriate to present it as

$$\mathcal{L}'_{i,+}(S_1) \quad = \quad \{\mathcal{L}'_{i,1}(S_1), \mathcal{L}'_{i,2}(S_1), \ldots\}, \tag{5}$$

where

$$\mathcal{L}'_{i,j}(S_1) = x_{i,j} - x_{i,j-1}, \quad i \in \mathcal{K}^\ell, j \in (1, \mathcal{L}_i), x \in S_1.$$

This operation converts the sequence into a signature-vector of discrete values. Instead of using the direct value, the empirical distribution function of $\mathcal{L}'_{i,+}(S_1)$ can be used. A version of the Kullback-Leibler (KL) divergence may be employed to measure sequence distribution dissimilarities. For $S_1$, $S_2$, $\ell$, $i \in \mathcal{K}^\ell$, the KL divergence on $\mathcal{L}'_{i,+}(S_1)$ and $\mathcal{L}'_{i,+}(S_2)$ is

$$D_\ell(\widehat{p}_{S_1 i} \mid\mid \widehat{p}_{S_2 i}) = \sum_z \widehat{p}_{S_1 i}(z) \log\left(\frac{\widehat{p}_{S_1 i}(z)}{\widehat{p}_{S_2 i}(z)}\right),$$

where $p_{S_1 i}(z)$ and $p_{S_2 i}(z)$ represent the empirical probabilities obtained from $\mathcal{L}'_{i,+}(S_1)$ and $\mathcal{L}'_{i,+}(S_2)$, respectively. The summation over all elements of $\mathcal{K}^\ell$ is referred to as KL dissimilarity

$$\begin{aligned} \widehat{KL}(S_1, S_2, \ell) &= \sum_{i \in \mathcal{K}^\ell} D_\ell(\widehat{p}_{S_1 i} \,||\, \widehat{p}_{S_2 i}) \\ &= \sum_{i \in \mathcal{K}^\ell} \sum_z \widehat{p}_{S_1 i}(z) \log\left(\frac{\widehat{p}_{S_1 i}(z)}{\widehat{p}_{S_2 i}(z)}\right). \end{aligned} \tag{6}$$

To have more robust dissimilarity, we actually consider the symmetrized KL measure

$$\begin{aligned} DKL(S_1, S_2, \ell) &= \widehat{KL}(S_1, S_2, \ell) + \widehat{KL}(S_2, S_1, \ell) \\ &= \sum_{i \in \mathcal{K}^\ell} D_\ell(\widehat{p}_{S_1 i} \,||\, \widehat{p}_{S_2 i}) + \sum_{i \in \mathcal{K}^\ell} D_\ell(\widehat{p}_{S_2 i} \,||\, \widehat{p}_{S_1 i}). \end{aligned} \tag{7}$$

Let $DKL$ be dissimilarity matrix of $n$ sequences obtained via $\widehat{KL}(S_1, S_2, \ell)$ of $\mathcal{L}'_r$, the algorithm of $DKL$ is given in Algorithm 1.

**Algorithm 1:** Calculate dissimilarity of two sequences $S_1$ and $S_2$ via $DKL$

```
1. For a given $\ell$, sliding windows, find $\mathcal{K}^\ell$ in (1).
2. For $i \in \mathcal{K}^\ell$, obtain $\mathcal{L}'_{i,+}(S_1)$ and $\mathcal{L}'_{i,+}(S_2)$ in (5).
3. Find the empirical distribution function of $\mathcal{L}'_{i,+}(S_1)$ and $\mathcal{L}'_{i,+}(S_2)$
   and calculate the empirical probabilities $p_{S_1 i}$ and $p_{S_2 i}$.
4. Repeat the Steps 2-3 for all $i \in \mathcal{K}^\ell$.
5. Calculate $DKL(S_1, S_2, \ell)$ in (7).
```

The sliding window sizes may need to be carefully selected. Earlier studies, [12] and [15] suggested the k-tuple variants with $\ell = 2$ and $\ell = 3$, however they only considered short sequences and the study of proposed models on a large scale DNA data and DNA sequence similarity are postponed to future. [3] also discussed the importance of sliding window – wider sliding windows sizes might not be appropriate especially for short sequences because a lot of elements of

A would have zero frequencies and the proposed dissimilarity may degenerate – it may not distinguish between closer and farther sequences. Furthermore, the calculation complexity increases rapidly in these situations.

Suppose there are $m$ different appropriate sliding window sizes and we are considering a kind of ensembling approach for dissimilarity aggregation. The idea of ensembling is that assume there is a fixed large model and the modeling (prediction) can be done by pooling the results from carefully chosen smaller models, the final model can be found by a consensus value of sub-models and provide a better performance than any one of the component used to form it. The application of ensembling is considered in classification or clustering process, see [14] and references therein. Let's assume $\ell_1, \ldots, \ell_m$ represent $m$ appropriate windows lengths corresponding to the following dissimilarities measures $DKL(S_1, S_2, \ell_1), \ldots, DKL(S_1, S_2, \ell_m)$. To generate a single aggregate dissimilarity measure for $n$ sequences, we standardize all of the measures first:

$$\frac{DKL(S_t, S_z, \ell)}{\sum_{t_0 \leq z_0} DKL(S_{t_0}, S_{z_0}, \ell)}, t \leq z,\ t, z = 1, \ldots, n, \tag{8}$$

ensuring that the summation over all paired dissimilarities is equal to one, the sum of standardized dissimilarities is referred to as ensemble dissimilarity:

$$EnDKL(S_t, S_z) = \frac{DKL(S_t, S_z, \ell_1)}{\sum_{t_0 \leq z_0} DKL(S_{t_0}, S_{z_0}, \ell_1)} + \ldots + \frac{DKL(S_t, S_z, \ell_m)}{\sum_{t_0 \leq z_0} DKL(S_{t_0}, S_{z_0}, \ell_m)}. \tag{9}$$

Algorithm 2 describes the Ensemble dissimilarity, $EnDKL(S_t, S_z)$, of two sequences $S_t$ and $S_z$ via $DKL$, it is referred to as $EnDKL$.

## 3. Alignment-free Toolkit Implementation

Here we show how the discussed methods are implemented, can be invoked using R. We begin by describing the sequences downloaded from NBCI, R functions, and the web deployment.

### 3.1. Dataset

To discuss the computational platform, we consider the Mitochondrial genome sequences of different mammalian species that were downloaded from the Na-

**Algorithm 2:** Calculate Ensemble dissimilarity of $n$ sequences via $EnDKL$

```
1. Let assume $\ell_1, \ldots, \ell_m$ be appropriate windows lengths.
2. For a given $\ell_1$, sliding windows length, find $\mathcal{K}^{\ell_1}$ in (1).
3. For $i \in \mathcal{K}^{\ell_1}$, obtain $\mathcal{L}'_{i,+}(S_1)$ and $\mathcal{L}'_{i,+}(S_2)$ in (5).
4. Calculate the empirical density function of $\mathcal{L}'_{i,+}(S_1)$ and $\mathcal{L}'_{i,+}(S_2)$
   and denote as $p_{S_1 i}$ and $p_{S_2 i}$.
5. Repeat the Steps 3-4 for all $i \in \mathcal{K}^{\ell_1}$.
6. Calculate $DKL(S_1, S_2, \ell_1)$ in (7).
7. Repeat Steps 2-6 for $\ell_2, \ldots, \ell_m$.
8. Repeat Steps 2-7 for all pairs $(t, z), t \leq z$, of sequences and
   standardize them, (8).
9. Calculate $EnDKL(S_t, S_z)$ in (9).
```

tional Center for Biotechnology Information (NCBI), see Table 1 for details. Such data might be appropriate to describe the proposed methods, because the Mitochondrial DNA is highly conserved and has a rapid mutation rate, which makes it useful for examining transorganismic evolutionary relationships, see [13].

### *3.2. msktuple R library*

We implemented and released a new R library msktuple that provides access to several alignment-free methods in one package that is continuously updated to support state-of-the-art alignment-free sequence modeling. The msktuple library is packed under R 3.4.0, so it will be compatible with R 3.4.0 (or later versions), the source code is provided on GitHub [1]. To install on MAC OS and Windows platforms, download the msktuple.tgz and msktuple.zip files locally. MAC OS builds can be installed directly from GitHub.

[1] `https://github.com/saeidamiri1/msktuple/`

Table 1: Mitochondrial genome of species used in the clustering method

| Species | GenBank | Length |
|---|---|---|
| Ape | NC_002764.1 | 16586 |
| Baboon | Y18001.1 | 16521 |
| Chimpanzee | D38113.1 | 16554 |
| Gibbon | X99256 .1 | 16472 |
| Gorilla | D38114.1 | 16364 |
| Human | V00662.1 | 16569 |
| Mouse | J01420.1 | 16295 |
| Rat | AC_000022.2 | 16300 |
| Sumatran Orangutan | NC_002083.1 | 16499 |

```
install.packages("https://github.com/saeidamiri1/msktuple/blob/master/
msktuple_1.0.tgz?raw=true", repos = NULL, type="source")
library('msktuple')
```

Also load the following libraries (dependencies) that are necessary to import sequences into R and to carry out some of the matrix manipulations, statistical analysis, modeling techniques and optimization calculations:

```
lr<-c('seqinr','ape','MASS','stats','Biostrings','foreach','doParallel','parallel')
lapply(lr,library,character.only=T)
```

Read the data and save as a list,

```
XDATA0<-list()
XDATA0[[1]]<-  read.GenBank("NC_002764.1",as.character = TRUE)[1]
XDATA0[[2]] <- read.GenBank("Y18001.1",as.character = TRUE)[1]
XDATA0[[3]] <- read.GenBank("D38113.1",as.character = TRUE)[1]
XDATA0[[4]] <- read.GenBank("X99256.1",as.character = TRUE)[1]
XDATA0[[5]] <- read.GenBank("D38114.1",as.character = TRUE)[1]
XDATA0[[6]] <- read.GenBank("V00662.1",as.character = TRUE)[1]
XDATA0[[7]] <- read.GenBank("NC_002083.1",as.character = TRUE)[1]
```

```
XDATA0[[8]] <- read.GenBank("J01420.1",as.character = TRUE)[1]
XDATA0[[9]] <- read.GenBank("AC_000022.2",as.character = TRUE)[1]
```

To read a file in the fasta format use the `read.fasta()`. R works faster with the numeric values and requires less resources, hence it is appropriate to convert the data from $\{A, T, C, G\}$ to $\{1, 2, 3, 4\}$, the following codes carry out this conversion

```
library(msktuple)
XDATA<-CtoN(XDATA0)
```

At the moment three methods are implemented, naive k-tuple, CV-Tree and lk-tuple. They can be invoked using the function `dktuple` with option of `nktuple`, `cvtktuple` and `lktuple`, the default method is `lktuple`. To expedite the calculations, one can engage multiple cores, if available. The following scripts calculate the *dissimilarity* using lktuple with $\ell = 3$ and multiple cores:

```
nc0 <- detectCores() - 1
st0<-tuples(ell=3)
dis1<-dktuple(sdata=XDATA,stuples=st0,Ncores=nc0,method="lktuple")
```

where `tuples(ell)` generates all possible combinations of $\ell$ nucleotides. Dendrogram plots can be generated with the following script, see Figure 1,

```
hc1<-hclust(as.dist(t(dis1)), method = "average", members = NULL)
cluslab<-c(’Ape’,’Baboon’,’Chimpanzee’,’Gibbon’,’Gorilla’,’Human’,
 ’SuOrangutan’, ’Mouse’,’Rat’)
plot(hc1,label=cluslab, xlab="",sub="")
```

The dissimilarity matrix can be calculated for a specific combinations of nucleotides instead considering all possible combinations. The `dktuple` function can be used for such situation. For instance, the following scripts consider the dissimilarity by using three different combinations of nucleotides.

```
st0<-c("1212","12324","1122")
dis1<-dktuple(sdata=XDATA,stuples=st0,Ncores=nc0,method="lktuple")
```

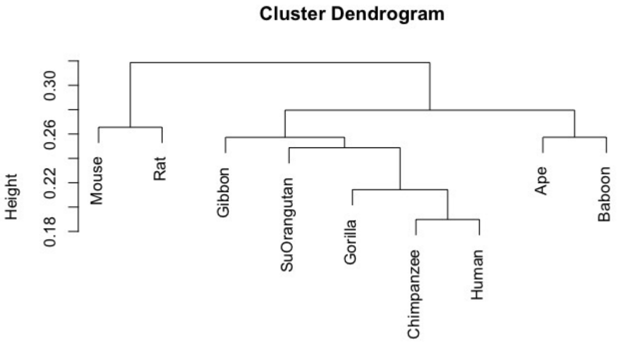

Figure 1: Dendrogram for the alignment and locational k-tuple with $\ell = 3$.

When $\ell$ is big, the procedure is time consuming because function goes through all possible combinations ($4^{\ell}$), the solution is to use the exiting tuples of length *ell* in the sequences data, the following scripts find the tuples of size 5 in `XDATA` and use them to calculate the dissimilarity

```
st0<-tuples(ell=5,sdata=XDATA,Ncores=nc0)
dis1<-dktuple(sdata=XDATA,stuples=st0,Ncores=nc0,
method="cvtktuple")
hc1<-hclust(as.dist(t(dis1)), method = "average", members = NULL)
plot(hc1,label=cluslab, xlab="",sub="")
```

The function `edktuple` can be used to invoke the *ensemble* dissimilarity given in Algorithm 2. Here is an example of computing the ensemble dissimilarity using $\ell_1 = 2$ and $\ell_2 = 3$, see Figure 2,

```
rt0<-list()
rt0[[1]]<-tuples(ell=2)
rt0[[2]]<-tuples(ell=3)
dis<-edktuple(sdata=XDATA,rtuples=rt0,Ncores=nc0,method="lktuple")
hc<-hclust(as.dist(t(dis)), method = "average", members = NULL)
plot(hc,label=cluslab, xlab="",sub="")
```

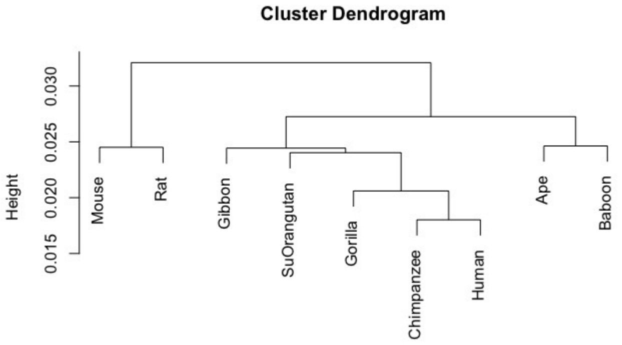

Figure 2: Dendrogram for the dissimilarity obtained using Ensemble locational k-tuple.

## 4. Conclusions

In this paper, we discuss different alignment-free sequence analysis methods. Unlike, classical alignment alternatives based on cost functions and string matching algorithms, alignment-free techniques utilize frequency measures to provide fast, accurate, and scalable solutions to various sequence comparison and profiling problems. There are many alignment and alignment-free methods implemented in different languages, deployed on variety of computational infrastructures, and targeting a broad spectrum of specific challenges. Our approach is based on developing alignment-free techniques that are open-source, promote distributed, interoperable, and multi-core computing, and can be invoked to easily explore and compare sequence characteristics.

Previously [3], we showed the accuracy of local k-tuple for mutated sequences, however, there was no uniform generic approach to select an optimal alignment-free method that is consistently superior. Therefore, we recommend that users test some of the alternative techniques and select an optimal strategy for their specific problem or case-study. Practitioners should always try several alternative methods to examine the underlying patterns in their specific genomic data.

*Validation*

Prof. Reza Modarres, Department of Statistics, George Washington University, USA: This work is interesting, because they implemented several alignment-free methods to achieve the biological sequence comparison: locational k-tuple, regular k-tuple, and a variant of CV-tree. All the sources are available in GitHub. Furthermore, the manuscript explains how to upload a dataset via NCBI to R and run the clustering. The library can be used by other researchers and R users. I found that the library can be installed easily because it is implemented under R. R is a popular programming language and platform used by many researchers and scientists, because of its functionality, reliability, scalability, open-source, and crowd source support.

Dr. Mehdi R. Bidokhti, Department of Molecular and Structural Biochemistry, Collage of Agriculture and Life Science (CALS) North Carolina State University, USA. The paper deals with the implementation of alignment-free methods in an R Library, such method is of interest for who works with the genomic data. I see they put all sources in GitHub and provided a wiki for it in GitHub. I installed this library on my PC, and it works perfectly. The authors explained how to import the sequence from the NCBI which is very helpful to working with genomic data.

*Github repository*

As dedicated supporters of open-science, we share the code, and tools in GitHub `https://github.com/saeidamiri1/msktuple`. We provide the latest release, open-issues, and support for the complete end-to-end software implementing all steps of the proposed analytic framework. This project employs other open-source code including R libraries and scripts. In addition to the source code and computational services, we provide end-to-end documentation and tutorials, from the data upload to the sequence dissimilarity estimation and graphical results.

*Acknowledgment*

We gratefully acknowledge the constructive comments and suggestions of Reza Modarres and Mehdi R. Bidokhti, the article referees, and the associate editor.

## References

[1] Montenegro-Burke, J. Rafael, Thiery Phommavongsay, Aries E. Aisporna, Tao Huan, Duane Rinehart, Erica Forsberg, Farris L. Poole et al. Smartphone Analytics: Mobilizing the Lab into the Cloud for Omic-Scale Analyses. Analytical Chemistry 88, no. 19 (2016): 9753-9758.

[2] Cammen, Kristina M., Kimberly R. Andrews, Emma L. Carroll, Andrew D. Foote, Emily Humble, Jane I. Khudyakov, Marie Louis, Michael R. McGowen, Morten Tange Olsen, and Amy M. Van Cise. Genomic methods take the plunge: Recent advances in high-throughput sequencing of marine mammals. Journal of Heredity 107, no. 6 (2016): 481-495.

[3] Amiri, Saeid, and Ivo D. Dinov. Comparison of genomic data via statistical distribution. Journal of theoretical biology 407 (2016): 318-327.

[4] Dinov, Ivo D. Volume and value of big healthcare data. Journal of medical statistics and informatics 4 (2016).

[5] Dinov, Ivo D. Methodological challenges and analytic opportunities for modeling and interpreting Big Healthcare Data. Gigascience 5, no. 1 (2016): 12.

[6] Cattaneo, Giuseppe, Umberto Ferraro Petrillo, Raffaele Giancarlo, and Gianluca Roscigno. An effective extension of the applicability of alignment-free biological sequence comparison algorithms with Hadoop."The Journal of Supercomputing 73, no. 4 (2017): 1467-1483.

[7] Lu, Yang Young, Kujin Tang, Jie Ren, Jed A. Fuhrman, Michael S. Waterman, and Fengzhu Sun. CAFE: aCcelerated Alignment-FrEe sequence analysis. Nucleic Acids Research (2017).

[8] Baichoo, Shakuntala, and Christos A. Ouzounis. Computational complexity of algorithms for sequence comparison, short-read assembly and genome alignment. Biosystems (2017).

[9] O'Leary, Nuala A., Mathew W. Wright, J. Rodney Brister, Stacy Ciufo, Diana Haddad, Rich McVeigh, Bhanu Rajput et al. Reference sequence (RefSeq) database at NCBI: current status, taxonomic expansion, and functional annotation. Nucleic acids research (2015): gkv1189.

[10] Amiri, Saeid, Bertrand Clarke, Jennifer Clarke, and Hoyt A. Koepke. A general hybrid clustering technique. arXiv preprint arXiv:1503.01183 (2015).

[11] Xu, Zhao, and Bailin Hao. CVTree update: a newly designed phylogenetic study platform using composition vectors and whole genomes. Nucleic acids research 37, no. suppl 2 (2009): W174-W178.

[12] Wei, Dan, Qingshan Jiang, Yanjie Wei, and Shengrui Wang. A novel hierarchical clustering algorithm for gene sequences. BMC bioinformatics 13, no. 1 (2012): 174.

[13] Brown, Wesley M., Ellen M. Prager, Alice Wang, and Allan C. Wilson. Mitochondrial DNA sequences of primates: tempo and mode of evolution. Journal of molecular evolution 18, no. 4 (1982): 225-239.

[14] Amiri, Saeid, Bertrand S. Clarke, and Jennifer L. Clarke. Clustering categorical data via ensembling dissimilarity matrices. Journal of Computational and Graphical Statistics just-accepted (2017).

[15] Bao, Junpeng, Ruiyu Yuan, and Zhe Bao. An improved alignment-free model for dna sequence similarity metric. BMC bioinformatics 15, no. 1 (2014): 321.