Computers in Biology and Medicine 114 (2019) 103439

Contents lists available at ScienceDirect

Computers in Biology and Medicine

journal homepage: http://www.elsevier.com/locate/compbiomed

SpCLUST: Towards a fast and reliable clustering for potentially divergent biological sequences

Johny Matar [a,c], Hicham EL. Khoury [c], Jean-Claude Charr [a,*], Christophe Guyeux [a], Stéphane Chrétien [b]

[a] Université de Bourgogne Franche-Comté, UMR 6174 CNRS, 16 route de Gray, Besançon, France
[b] National Physical Laboratory, Hampton Road, Teddington, United Kingdom
[c] LaRRIS, Faculty of Science, Lebanese University, Fanar, Lebanon

ARTICLE INFO

Keywords:
Sequences clustering
Genomics
Laplacian eigenmaps
Gaussian mixture model
Parallel computation
Spectral clustering

ABSTRACT

This paper presents SpCLUST, a new C++ package that takes a list of sequences as input, aligns them with MUSCLE, computes their similarity matrix in parallel and then performs the clustering. SpCLUST extends a previously released software by integrating additional scoring matrices which enables it to cover the clustering of amino-acid sequences. The similarity matrix is now computed in parallel according to the master/slave distributed architecture, using MPI. Performance analysis, realized on two real datasets of 100 nucleotide sequences and 1049 amino-acids ones, show that the resulting library substantially outperforms the original Python package. The proposed package was also intensively evaluated on simulated and real genomic and protein data sets. The clustering results were compared to the most known traditional tools, such as UCLUST, CD-HIT and DNACLUST. The comparison showed that SpCLUST outperforms the other tools when clustering divergent sequences, and contrary to the others, it does not require any user intervention or prior knowledge about the input sequences.

1. Introduction

Sequence clustering refers to the act of partitioning an input group of sequences into clusters, each containing a group of somehow related sequences. It can involve either nucleotide or protein based sequences and is mainly used to identify sequences that are potentially derived, by mutations or substitutions, from each others or from a common ancestor. Reaching a good speed and accuracy in nucleotide or protein sequences clustering, involving large numbers of divergent sequences, is a very challenging problem.

Mutations and substitutions, in the nucleotide sequences, happens in different rates and for many reasons [1–3]. While some substitutions and mutations happen as a natural environment adaptation process, e.g. the crisis-causing bacterial adaptation to antibiotics [4,5] that is emerging as a reason of excess of mortality [6], others might be linked to some artifacts, such as the exposure to some pollutants, and result in diseases and anomalies [7], e.g. cancerous cells.

As a result of the increasing number of mutations' causes and the large number of new sequences discoveries, linking these sequences to their siblings and ancestors becomes more complex and the use of clustering tools is essential to tackle this problem. Many clustering tools, based on hierarchical or greedy algorithms, relying on a user input similarity threshold, and targeting high speed clustering of highly similar sequences, currently exist and some of them became widely used. Some of these tools use parallel computing to provide even higher clustering speeds. In [8], an innovative clustering module for genomic sequences that uses Laplacian Eigenmaps and a Gaussian Mixture Model, was presented. The first implementation of the algorithm gave very promising results when compared to other existing tools, especially in terms of clustering accuracy of potentially divergent sequences. However, since it computes the similarity matrix between all the input sequences which is a computationally intensive operation, its execution time significantly increases when clustering large sets of input sequences.

In the present paper, a new implementation of the clustering algorithm is presented. A special attention was given to improve the overall performance and scalability of this new version. The new hybrid C++/Python clustering package, called SpCLUST, computes in parallel the

* Corresponding author.
E-mail addresses: jean-claude.charr@univ-fcomte.fr, jc.charr@gmail.com (J.-C. Charr).

https://doi.org/10.1016/j.compbiomed.2019.103439
Received 26 April 2019; Received in revised form 5 September 2019; Accepted 5 September 2019
Available online 10 September 2019

similarity matrix using the Message Passing Interface (MPI) which drastically reduces the execution time of this stage. This new package was integrated into a GALAXY platform [9] and is freely available online. Many experiments were conducted to evaluate the accuracy and the performance of SpCLUST while using simulated and real data sets. It was also compared to other clustering packages, such as UCLUST, CD-HIT and DNACLUST.

The rest of this paper is organized as follows. In Section 2, the motivations for this research work are presented with regard to the state of the art: the most known clustering packages are described, with particular emphasis put on a new unsupervised learning-based package [8]. In Section 3, the improvements added to the initial Python package are detailed. In Section 4, a performance comparison between the different versions of the package is presented. It is followed by a comparative study between SpCLUST and other packages. The article ends with a discussion that recapitulates the contributions of this work with regard to other available clustering tools and provides some future perspectives.

2. Motivation with regard to the state of the art

The variable and unpredictable degree of mutations of biological sequences contained in increasingly large databases, makes it harder to perform an accurate clustering using traditional tools that rely either on a user input (e.g., the similarity threshold) or on such increasing databases. This is why mixture models were recently introduced to tackle these hard problems. They are based on unsupervised learning techniques, differentiate from classical ones by performing grouping based on probability distributions [10]. With this in mind, mixture models have been used in [8] to perform biosequence clustering. While the number of clusters has been decided using a statistical criterion, namely the so-called BIC [11], leading to a process that does not rely on any user input (e.g., similarity threshold). This model exhibited competitive results in the case of really divergent sequences, but its speed needed to be improved.

Indeed, several packages for high speed clustering of nucleotide and/or protein sequences are publicly available, such as CD-HIT [12], UCLUST [13], DNACLUST [14] and SUMACLUST [15]. All these packages rely on greedy algorithms for clustering the sequences. Conversely, DACE [16] relies on the scalable Dirichlet Process means (DP-means) algorithm, making it an algorithm similar to K-means. HPC-CLUST [17], for its part, is based on a hierarchical algorithm. The main features and characteristics of these tools are presented below:

- CD-HIT is a suite of tools for biological sequence handling, including modules for clustering both nucleotide and protein sequences using word counting for computing the similarities, in order to avoid costly pairwise sequences alignments. It processes the input sequences by their order of length, starting from the longest and considers the first one as the first cluster representative. It then classifies the following sequences subsequently, based on the input similarity threshold, as either a new cluster representative or part of a cluster for a previously classified representative. CD-HIT is very fast and can handle extremely large databases [12].
- UCLUST uses a module named USEARCH for assigning the sequences to their clusters, based on a given identity threshold and, optionally, an input of the centroid units. In contrast with CD-HIT, UCLUST does not sort the input sequences by length prior starting the clustering, thus the order of the sequences can impact the result since most clusters representatives (or centroids) are chosen from the first sequences. UCLUST can also cluster both nucleotide and protein sequences, and is able to produce a better clustering quality than its competitor CD-HIT [13] while consuming less memory.
- DNACLUST was designed for quick clustering of highly similar DNA sequences, but it does not handle protein sequences. Similarly to CD-HIT, it first sorts the sequences in their decreasing order of length. If the sequences have equal length, they are sorted in their decreasing order of abundance. The first sequence is considered as the current cluster centroid and all the input sequences having a distance with the centroid inferior than the user input distance threshold are added to that cluster. This procedure is repeated until all the sequences are clustered. Based on its authors study, DNACLUST outperforms UCLUST when high similarity thresholds, above 0.95, are chosen.
- SUMACLUST, along with SUMATRA, is a package aiming for a fast and exact DNA sequences comparison. It first compares the pairwise similarities between sequences using SUMATRA, then sorts the sequences by abundance, and finally it clusters the sequences with a greedy algorithm similar to CD-HIT and UCLUST. The main difference between SUMACLUST and its competitors, CD-HIT and UCLUST, is that it uses a pairwise sequence alignment algorithm before clustering the sequences. Moreover, the alignment step is preceded by a filtering step which enables to only align couples of sequences that potentially have an identity greater than the chosen threshold. Finally to improve the performance of the package, the filtering and the alignment steps are parallelized according to the SIMD model.
- DACE is a parallel high performance clustering tool for very large data sets. It iteratively partitions the input data set into several non intersecting subsets before using the DP-means algorithm for clustering the subsets in parallel. Based on [16], DACE runs up to 80 times faster than its competitors including CD-HIT and UCLUST. However, this is valid only for very large data sets while for small data sets it might perform slower than the mentioned competitors.
- HPC-CLUST is a tool featuring a distributed hierarchical clustering algorithm. It enables HPC-CLUST to cluster large data nucleotide sequences at high speed. In contrast with the previously described tools, HPC-CLUST takes aligned sequences as input, thus saving the computation time of the alignment task. The hierarchical algorithm starts by grouping close sequences, then it forms larger clusters by iteratively merging the closest groups.
- Gclust [18] is a trans-kingdom (or trans-domain) classification package for proteins, using automatically selected similarity threshold for individual protein groups. The similarity selection relies on the entropy-optimized organism count method, which is a heuristic involving E-value, overall similarity and organism count. The homologous sequences are then assembled using single-linkage clustering. Finally, any produced duplication is solved by either merging the smaller groups to larger ones or decomposing the smaller groups based on the overlaps and cross-links they share with the larger groups.

A comparative study of clustering methods was proposed in [19] where these methods were applied to the clustering of 16S rRNA sequences. They were divided into three classes: hierarchical, heuristical, and model-based clustering. The study included widely used tools such as CD-HIT [12], UCLUST [13], DNACLUST [14], Mothur [20], ESPRIT [21], and CROP [22]. It concludes that these methods will continue to play a vital role in microbial analysis because taxonomy dependent algorithms strongly rely on the completeness of existing databases, while the majority of microbe species are unknown. Moreover, the study shows that the estimation of the number of operational taxonomic units (OTUs)[1] is severely affected by the choice of the dissimilarity threshold. Therefore, supervised tuning of this parameter is crucial for accurate clustering of nucleotide sequences.

In [8], a clustering Python module that uses Laplacian Eigenmaps and a Gaussian Mixture Model, was presented. Unlike most clustering packages, that largely utilize greedy approaches and just aim to improve the speed of clustering highly similar sequences, this one focuses mainly on improving the accuracy of the clustering for nucleotide sequences.

[1] Clusters of closely related organisms grouped by DNA sequence similarity.

Only few intelligent clustering tools use machine learning approaches. For instance, MeShClust [23] is a multi-threading enabled package based on a mean shift algorithm. But based on its usage instructions, the input identity of the sequences parameter is the most important parameter. To our knowledge, the proposed package, namely SpCLUST, is the first clustering method that uses unsupervised learning [24] and does not require any sequences identity or centroid sequences user input. Such methods can help researchers make progress in a field where good balance between accuracy in clustering, even for potentially distant and divergent biological sequences, and high computational speed is required. Reaching this balance is unfortunately very difficult in practice. In particular, although the computational speed of the method proposed in [8] was demonstrated to be only moderately worse than the competitors on the ND3 test set of 100 sequences, its performance is significantly degraded when applied to larger sets composed of a few hundreds of sequences.

The clustering package presented in [8] is made of three main stages:

1. Sequences alignment using MUSCLE [25]: in this phase, the input sequences are sent to the MUSCLE package in order to obtain an aligned set. MUSCLE is available as an external and independent executable module, in addition to its existence as a function in Python's COmparative GENomics Toolkit [26] (cogent). (See for more details on how MUSCLE can be called from a Python package in [8].)
2. Similarity matrix calculation: this phase relies on pairwise sequence comparison. For a set of N sequences, a NxN square matrix is computed. The value of the (i,j) element in this matrix is the similarity index between the pair of sequences i and j. This similarity index is based on the distance between these two sequences, which is calculated using the scoring matrix EDNAFULL.
3. Sequences clustering: this last stage is the core of the clustering method. It takes the similarity matrix as input, and clusters the sequences using the Laplacian Eigenmaps and the Gaussian Mixture Modelling.

The first and third stages depend on third party modules and existing libraries' functions, whereas the second stage, is an internally developed code to construct the similarity matrix. This latter is an intensive computation step with a complexity of order $O\left(\frac{N^2-N}{2}\right)$, where N is the number of input sequences.[2] To be able to cluster large sequences in a reasonable time, the execution time of this step must therefore be improved.

3. SpCLUST: an improved clustering package

In this section, we recall the main ideas underlying the construction of the Python module in [8] and describe the main proposed changes for improving its performance and expanding its functionality. We also present a description of each sub-module of the code.

3.1. Analysis of the original Python package

The main objective of the original package was to provide a good clustering method. For a given relatively divergent sets of sequences and without a previous knowledge of the number of clusters, it should produce high quality clusters with high intra-class similarity and low inter-class similarity. It solely focused on the quality of the clusters and little measures were taken to improve the performance of the method. Therefore, in this paper great emphasis was placed on improving the performance of this clustering method. A thorough analysis of the execution time of each stage of the package was performed in order to detect the main bottlenecks.

This analysis was done using two data sets consisting of 100 and 1024 sequences respectively, and using two computers equipped with different processors: an i3-5005U 2.0 GHz dual-core (4 core threads) processor and an i7-6700 3.4 GHz quad-core (8 core threads) processor. The profiling results are displayed in Table 1.

As shown in Table 1, the similarity matrix calculation, which is mostly composed of the distance matrix computation, is by far the most time consuming task in the overall pipeline. Therefore, optimizing the computation of the similarity matrix should drastically reduce the overall execution time of the clustering package.

In consequence, most of the modifications presented in this paper concern this stage. In particular, it was re-implemented in the C++ programming language which, although it is a more complex programming language than Python [27], is clearly faster [28]. Moreover, to furthermore reduce the execution time of the computation of the similarity matrix which computes independently the similarity indexes between all the sequences, it was parallelized using MPI according to the Master/Slave model. The content of each phase of the package and the added modifications to each one of them are detailed in the next subsections.

3.2. Alignment phase

This phase consists of obtaining an aligned version of the input sequences. Many alignment packages, such as MUSCLE [25], T-Coffee [29], MAFFT [30], PASTA [31], and ClustalW [32], are available but have different accuracy and performance. Notredame states in [29] that T-Coffee provides a dramatic improvement in accuracy with a modest sacrifice in speed. On the other hand, ClustalW is widely used, has cross-platform releases and offers both command line and a graphical user interface version. Kuo-Bin Li also worked on improving its performance by introducing in [33] a parallel version, called ClustalW-MPI.

However, MUSCLE remains better supported than ClustalW for command line calls, requiring no user intervention, and can on average achieve both higher accuracy and lower execution time than ClustalW or T-Coffee, depending on the chosen options [34]. For instance, in the case of aligning large data sets which is an extremely time consuming task, after just two iterations, MUSCLE gives an alignment with a precision equal to the one computed by T-Coffee and takes less time than ClustalW [34]. Moreover a study, published in [35], proposes a parallel computation version of MUSCLE that should theoretically improve further its performance. However, the implementation of the proposed parallel version of MUSCLE was not found online.

Further benchmarks [36–38] showed that MAFFT outperforms the other tools, including MUSCLE, in terms of alignment's speed and quality. Moreover, MAFTT has a multi-threaded version where the alignment can be computed in parallel using all the available core threads in a workstation. However, unlike the results presented in [36–38], our alignment test, performed on a set of random nucleotides sequences and conducted over a workstation equipped with a dual-core (4 core threads) 2.0 GHz processor, gave the following results: MUSCLE's official and sequential module computed the alignment in 11 s

Table 1
Execution time of the original Python package.

	i3-5005U 2.0 GHz processor		i7-6700 3.4 GHz processor	
	100 sequences	1049 sequences	100 sequences	1049 sequences
Alignment phase	8 s	73 min	5 s	36 min
Similarity matrix calculation phase	18 min	5696 minutes	9 min	2848 minutes
Clustering phase	7 s	33 min	4 s	15 min
Total	18 min	5802 min	9 min	2899 min

[2] This matrix is symmetric and thus only the upper or lower diagonal must be computed.

while the multi-threaded version of MAFFT took 13 s and the sequential one required 18 s. Moreover, MUSCLE had the following advantages:

- It is a single 4 MB executable file, whereas MAFFT's module is a package containing over 100 files with a total size higher than 60 MB after extraction.
- It does not generate any interfering outputs when called with the "-quiet" parameter, unlike MAFFT that only omits the alignment progress output using this parameter.

For these reasons, the proposed SpCLUST package continues on using MUSCLE for aligning the sequences.

3.3. Similarity matrix computation phase

In the original package, for each pair of aligned sequences, the distance between them is computed using the EDNAFULL [39] scoring matrix. They are stored in the distance matrix, where the element of index (i,j) contains the distance between the i^{th} and j^{th} sequences. The similarity matrix is then computed from the distance matrix. In the new version, this procedure was re-coded in C++ to reduce its execution time and it was extended to use two additional scoring matrices: BLOSUM62 and PAM250 [40] which can be specified as a parameter. The added scoring matrices extends SpCLUST's operational scope to include protein sequences clustering.

Since the computation of the distance matrix is the most time consuming phase of the clustering package and it is quadratically proportional to the size of the input sequences, it was also parallelized to reduce its execution time. The parallel version uses MPI to distribute the computations. For each available core thread on the used workstation, a slave process is created and the master process assigns to each slave process an equal number of sequences pairs. The distances between the assigned pairs of sequences are computed in parallel on P slave processes and sent back to the master which stores them in the distance matrix. This inter-process interactions are illustrated in Fig. 1. Since the computed distance between two sequences is a commutative operation, the resulting distance matrix is symmetric. Therefore, only the upper triangular matrix is computed and the lower one is the transpose of the upper triangular matrix, such as $d_{(i,j)} = d_{(j,i)}$ for $j < i$.

3.4. Clustering phase

This phase uses the previously calculated similarity matrix to cluster the sequences. It also relies on the use of the Laplacian Eigenmaps and the Gaussian Mixture Model, as a machine learning model, in order to produce the clustering. In the new version, obsolete functions were replaced and some parameter calibration were necessary to get the same clusters as in [8], for 100 DNA sequences taken from the mitochondrially encoded NADH dehydrogenase 3 (ND3) gene.

3.5. Package availability

SpCLUST is available in command line versions for both Windows and Linux. A basic graphical user interface allows the user to browse input and output files, and to view the obtained clustering. The source code and the executable files of the SpCLUST package are available online.[3] They could be customized by passing parameters that can modify the quality and the execution time of the alignment performed by MUSCLE. The latest version is also integrated to a publicly accessible GALAXY server.[4] It can be found under the menu item "SpCLUST".

[3] https://github.com/johnymatar/SpCLUST.
[4] http://galaxy.ul.edu.lb.

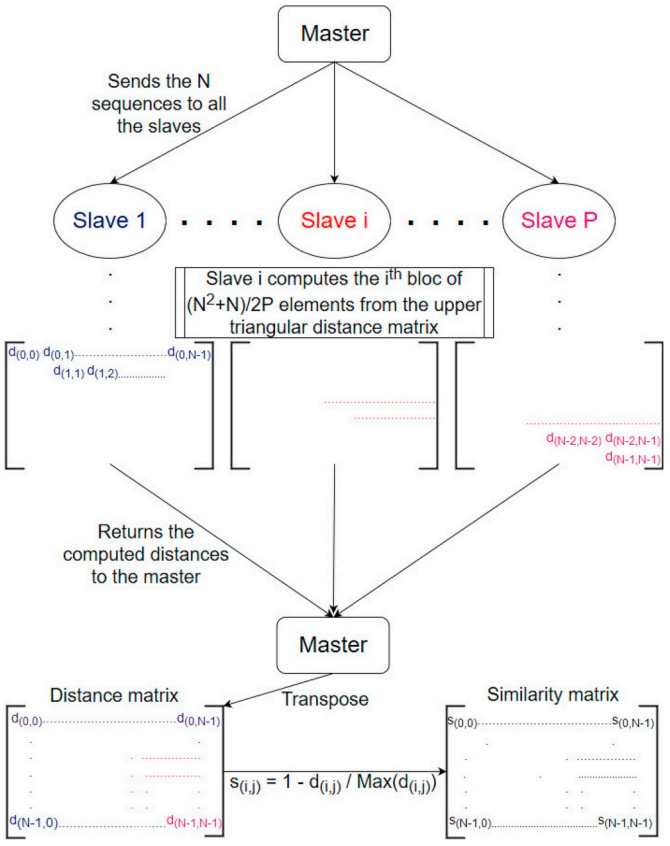

Fig. 1. Processes Master-Slave architecture.

4. Performance evaluation of SpCLUST

After describing the workflow and the improvements offered in the new clustering module, the performance of the different versions will be investigated. The following experiments will detail the performance of each phase in the proposed module.

4.1. Alignment phase

In this section, the execution time of the alignment phase, performed by MUSCLE, is presented for different sizes of data sets, different number of iterations, and while running on different workstations. Figs. 2 and 3 show the time taken to align, in different numbers of iterations, two data sets using the 2.0 GHz i3-5005U processor and the 3.4 GHz i7-6700 processor.

For large data sets, the choice of the number of iterations drastically impacts the alignment time. Therefore, a trade-off between the quality of the alignment and its execution time should be considered. The impact of this choice on the clustering quality will be discussed in Section 5.2.3.

4.2. Similarity matrix calculation phase

As mentioned in Section 3, the similarity matrix calculation was re-implemented in C++ then parallelized using MPI to reduce its execution time. Fig. 4 illustrates the execution times taken by each version to compute the similarity matrix of the 1049-sequences set with the three scoring matrices. These experiments were conducted on a workstation equipped with the I3–5005U 2.0 GHz dual-core Intel processor. The parallel version was using the four core threads of that processor.

The results in Table 1 show a huge performance improvement when compared to the execution time needed by the initial Python module. The modulated version is a C++ version where the main module and the

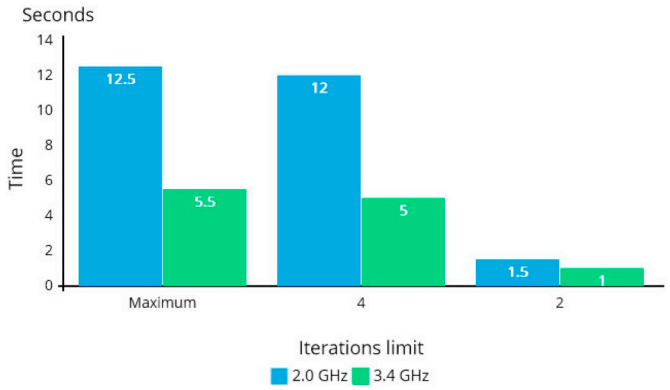

Fig. 2. Alignment time for the 100-sequences ND3 set.

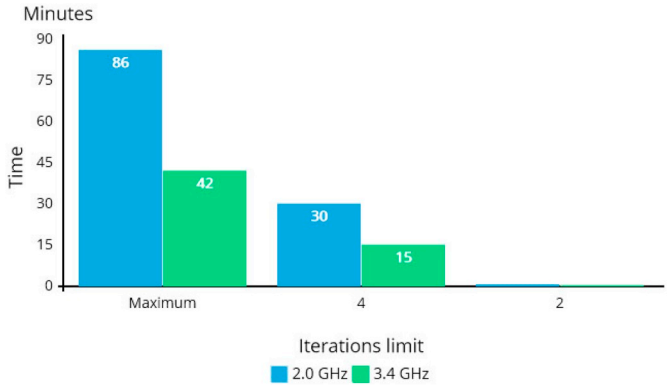

Fig. 3. Alignment time for the 1049-sequences set.

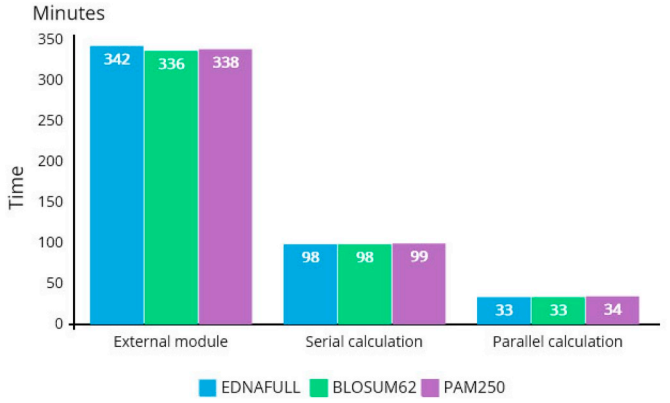

Fig. 4. Similarity matrix calculation time using the I3–5005U 2.0 GHz dual-core (4 core threads) processor.

sequence pairwise distance calculation module were two separate executables. It took 342 min to compute the similarity matrix using the EDNAFULL scoring matrix while the original Python module required 5696 min, resulting in around 16.6X speed-up. The serial one executable version took just 99 min, showing a speed-up of 57.5X. Finally, the parallel version, running on 4 core threads, took 34 min, a 167.5X speed-up over the Python package and a 2.91X speed-up over the serial C++ version.

The results also show that for all the versions, the similarity matrix calculation time is nearly the same, regardless of the chosen scoring matrix: the use of the newly integrated matrices does not affect the performance of the package.

Moreover, to study the scalability of the parallel similarity matrix computation, its normalized execution time and its strong scaling efficiency[5] were measured. Fig. 5 shows the normalized execution time and the strong scaling efficiency of this matrix computation using different number of slave processes running on the 3.4 GHz I7-6700 quad-core (8 core threads) processor for the same data set. As shown in Fig. 5, the strong scaling efficiencies with 2, 4 and 8 slave processes were equal to 96%, 80% and 59% respectively. Those numbers demonstrate that although the parallel version do not scale linearly due to communication and I/O overheads, its execution time is continuously reduced while increasing the number of used slave processes up to the number of available core threads.

A similar experiment was conducted on a workstation equipped with 4 slower 1.87 GHz quad-core XEON E7520 processors and using up to 16 core threads. Fig. 6 shows the normalized execution time and the strong scaling efficiency of the parallel similarity matrix computation. As in the previous experiment, and as shown in Fig. 6, it can be noticed that the strong scaling efficiency of this parallel application slowly decreases from 99% with 8 slave processes to 59% with 16. It can also be noticed that the optimal number of slave processes to use is 8, and beyond this number the IO overhead from reading sequences in parallel significantly reduces, on this workstation, the scalability of the parallel similarity matrix computation. This second workstation is currently hosting the publicly accessible GALAXY server with the latest SpCLUST version.

To conclude the scalability study, the similarity matrix computation time was recorded for five larger sets of sequences. These sets contained 2000, 5000, 10000, 20000 or 30000 sequences, extracted form a set of aligned archaea nucleotide sequences that was retrieved from the Linux package of HPC-CLUST.[6] These experiments were performed over a cluster of 34 nodes, where each node has a 3.4 GHz processor with 4 cores and 2 GB RAM. Fig. 7 shows the recorded computation time for each input number of sequences, ranging from 0.66 min for the 2000-sequence set to 42 min for the 30000-sequence set. As previously shown, the complexity of the similarity matrix computation is of order $O\left(\frac{N^2-N}{2}\right)$, where N is the number of input sequences. Therefore, the number of operations should scale quadratically to N. However, the experiments show that the computation time of the similarity matrix scales on a slower rate. For example, between the 5k and the 10k sets the computation time scales by a ratio of 3 while the number of operations was multiplied by 4. Similarly, between the 10k and the 20k sets, the computation time only scales by a ratio of 3.5. These ratios show that the

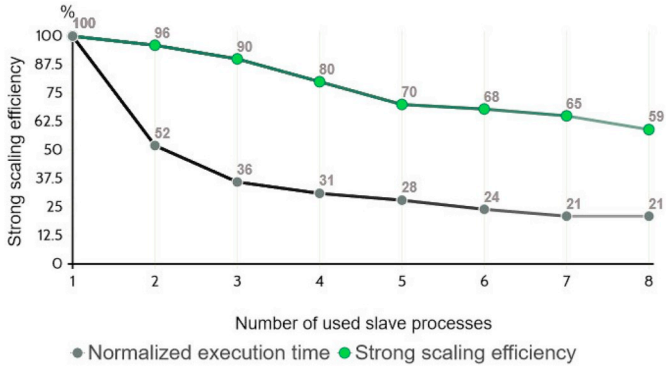

Fig. 5. Scalability on a 3.4 GHz processor with 8 core threads.

[5] If the amount of time to complete a work unit with 1 processing element is $t1$, and the amount of time to complete the same unit of work with N processing elements is tN, the normalized execution time is $(tN/t1)*100$ and the strong scaling efficiency is.$t1/(N*tN)*100$

[6] https://www.meringlab.org/software/hpc-clust/hpc-clust-1.2.1-bin.tar.gz.

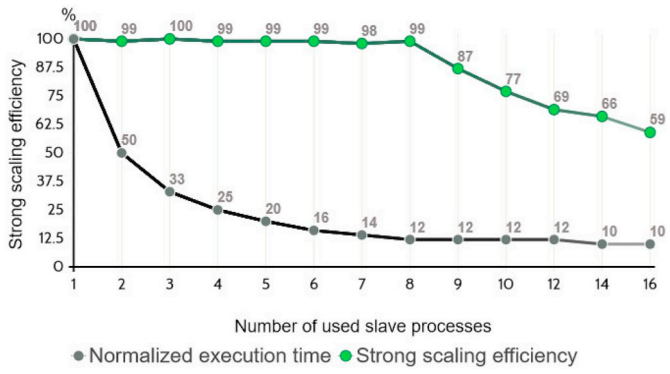

Fig. 6. Scalability on four 1.87 GHz processors using 16 core threads.

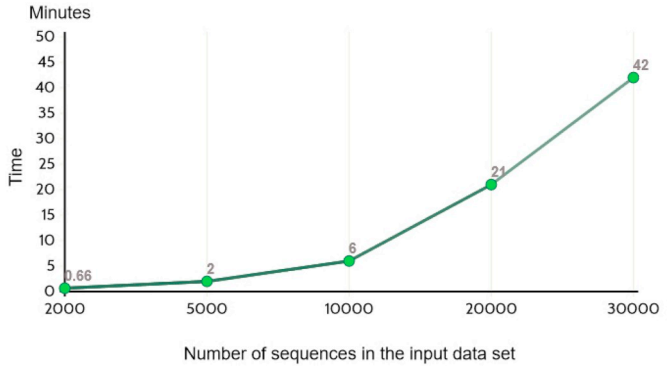

Fig. 7. Scalability on a cluster of 34 nodes having 3.4 GHz processors.

proposed solution is indeed scalable and the difference of proportionality comes from communication time needed to distribute the increasing data set to the cluster's slave processes over the network.

4.3. Clustering phase and overall performance

The clustering phase's sub-module, as stated in Section 3, was updated and uses the latest Gaussian Mixture model package [41]. The recorded runtime values reflect the clustering phase's performance after the replacement of the obsolete functions. These tests allow us to conclude the profiling analysis and assess the overall performance improvement of the full package. Fig. 8 shows the detailed profiling results of SpCLUST while running on the 2.0 GHz dual-core (4 core threads) i3-5005U processor and for a set of 1049 sequences. It took 153 min while using a maximum precision alignment and just 67 min with a fast alignment. These values show a speed-up of 37.9X and 86.5X, respectively, when compared to the original Python module. Moreover, the total runtime, on the quad-core (8 core threads) i7-6700 3.4 GHz and for the same set of 1049 sequences, was equal to 65 min using a maximum precision alignment and 23 min using a fast alignment, giving a 44.6X and 126X speed-up, respectively.

5. A comparative study between SpCLUST and four competing clustering tools

This section presents a comparative study between SpCLUST and four competing clustering tools. The experimental protocol is first described, then the clustering results are compared in terms of number of clusters and their contents. Finally, the effect of the alignment quality on the clustering is discussed.

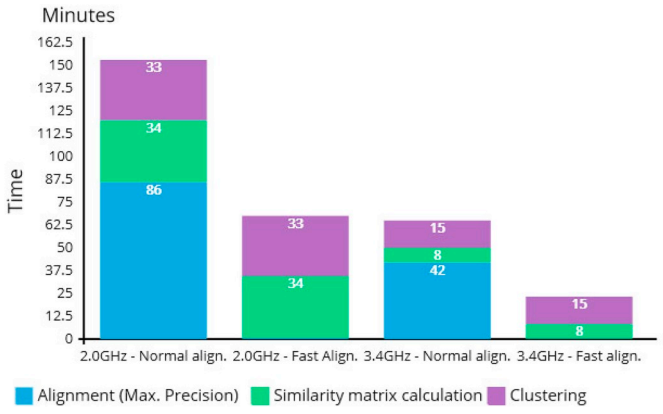

Fig. 8. Run time for all phases execution.

5.1. Experimental protocol

The comparative study, interpreted in this section, used eighteen sets of simulated data (12 genomic sets and 6 protein sets) and eight sets of real data (4 genomic sets and 4 protein ones). The simulated data sets derive from the following reference sequences:

- four different NADH dehydrogenase 3 (ND3) sequences, extracted from a collection of Platyhelminthes and Nematoda species. The mutated sequences, generated from these four reference sequences, should produce four clusters.
- five different protein sequences (should produce five clusters).
- six different gene sequences extracted from chloroplastic genomes (should produce six clusters).

From each group of reference sequences, new mutated sets which contains 30 mutations from each initial sequence, were generated. Each set had different properties in terms of mutation criteria and divergence rate between its sequences. The mutated sets were divided into two categories based on the used transition. The transition is performed in either of the following two ways:

1. The nucleotide or protein transition is performed on a random base.
2. The nucleotide transition is performed according to a real computed rate for URA3, published in [42], whereas the protein transition is performed according to the rates of the PAM1 substitution matrix.

Each category includes three mutated sets from each group of reference sequences, namely {S1, S2, S3} and {S′1, S′2, S′3} for each group. The mutation is performed using the following criteria:

- the S1 and S′1 are the result of four generations of mutation with 15% mutation rate, 10 maximum random insertions of size inferior to 9 nucleotides or proteins and a gap rate equal to 30% of the number of insertions with a maximum gap size of 10.
- the S2 and S′2 are the result of four generations of mutation with 10% mutation rate, 7 maximum random insertions of size inferior to 6 nucleotides or proteins, and a gap rate equal to 20% of the number of insertions with a maximum gap size of 7.
- the S3 and S′3 are the result of two generations of mutation with 5% mutation rate, 4 maximum random insertions of size inferior to 4 nucleotides or proteins, and a gap rate equal to 10% of the number of insertions with a maximum gap size of 4.

Based on the mentioned criteria, the sets S1 and S′1 contain the most divergent sequences when compared to the initial ones, whereas S3 and S′3 contain the least divergent ones. Fig. 9 shows part of a ND3 sequence on which four generations of simulated mutations were performed. The

first row contains the original sequence content and the next four rows show the added mutations, gaps and insertions from one generation to the other.

The real data sets were formed from mixes of genomic or protein sequences gathered and downloaded using NCBI's HomoloGene online tool[7] and NCBI's virus resources,[8,9] The starting genomic or protein sequences are from:

- homologous genes to the human genes FCER1G, S100A1, S100A6, S100A8, S100A12 and SH3BGRL3 which all belong to the first chromosome, and to the human gene MC1R which belongs to the 16^{th} chromosome. These homologous genes are extracted from a collection of mammal species.
- variants for segments PB2 and PB1 from the most fatal type A influenza's serotypes [43]. These serotypes are H1N1, H2N2, H3N2, H5N1, and H7N9.
- variants for the C-E genome region of the Zika virus.

The sequences in each data set were randomly shuffled. The following describes the content of each data set and how many clusters are expected in the clustering results:

- Set 1 consists of six series of homologous genes to the FCER1G, S100A1, S100A6, S100A8, S100A12, and SH3BGRL3 genes which belong to the human's first chromosome. The sequences should be separated into six clusters.
- Set 2 contains a series of homologous genes to the MC1R gene, found in the human's sixteenth chromosome, in addition to the series in the first set. The clustering of set 2 should therefore produce seven clusters.
- Set 3 contains variants of the segment PB2 of the influenza type A serotype and should produce five clusters.
- Set 4 contains variants of the Zika's C-E segment, influenza's AH1N1 PB2 segment, and influenza's AH2N2 PB1 segment and should produce three clusters.

All these data sets are available on SpCLUST's GitHub repository[10].

In this comparative study, besides SpCLUST, six other clustering tools were considered: CD-HIT, UCLUST, DNACLUST, SUMACLUST, DACE, and HPC-CLUST. However, since SUMACLUST failed to run correctly on many data sets and HPC-CLUST generated clusters containing only singletons on most of the data sets, only the remaining four tools were evaluated and compared to SpCLUST.

To cover multiple levels of sequence divergence, we used 3 similarity thresholds: 0.9, 0.6, and 0.4. These values have been chosen according to [44], an article investigating homologous sequences similarity for a wide range of organisms. According to this research work, 0.9 identity allows to identify highly similar groups of sequences, while identities ranging from 0.4 to 0.6 cover the majority of highly to moderately divergent groups of sequences.

5.2. Results comparison and interpretation

The interpretation of the experiment's results is organized in three tracks. The number of clusters, in each clustering, is discussed in the first track, the clusters' contents are assessed in the second one, whereas SpCLUST's results, using a high quality (slow) or fast alignment, are compared in the third track.

5.2.1. Analysis of the number of returned clusters

The number of clusters in the clustering results of each evaluated tool, reflects a first aspect of the clustering accuracy. Obtaining a number of clusters equal or relatively very close to the real value, is an essential but not a sufficient condition for considering it to be a good quality clustering. The content of the clusters must corresponds to the reality too.

The data presented in Table 2 show the resulting clusters' numbers for the simulated sets that were produced from the sequences previously mentioned in the experimental protocol. In Table 2, "Prot." and "Clp" are the abbreviations of "Protein" and "Chloroplast". On the other hand, an "E" value indicates that the concerned tool was not able to cluster the specified data set with the given options, e.g. DNACLUST and DACE failed to handle the large chloroplast sequence sets while CD-HIT-Est failed to run with similarity thresholds equal to 0.6 or 0.4. Moreover, a "-" indicates that the current tool or parameter is not designed to handle this type of sequences, i.e. this tool is designed for genomic sequences while the set contains protein sequences, or vice versa. Finally, two tools of CD-HIT were used: for the genomic sets CD-HIT-Est was used whereas for the protein sets it was replaced by CD-HIT-Protein.

As described in the experimental protocol, the expected number of clusters for each data set are:

- 4 clusters for the ND3 simulated sets.
- 5 clusters for the protein simulated sets.
- 6 clusters for the simulated sets derived from chloroplast genes.

A general overview of Table 2 shows that in the cases of the highly and moderately divergent sets, S1, S′1, S2 and S′2, all the tools except SpCLUST returned a number of clusters far from what was expected. Conversely, for the least divergent sets, S3 and S′3, these same tools, except DACE, gave the exact number of clusters or a close number to the expected one. In particular, it can be noticed that DNACLUST, while using a similarity threshold equal to 0.6, returned the expected number of clusters for the ND3 simulated sets, S3 and S′3. Moreover, UCLUST, while using a similarity threshold equal to either 0.6 or 0.4, produced the expected number of clusters for the ND3 and chloroplast genes simulated sets, S3 and S′3. In addition, CD-HIT-Protein, while using a similarity threshold equal to 0.6, was able to find the expected number of clusters for the protein simulated sets, S3 and S′3. Finally, SpCLUST gave either the exact number of clusters or a close one for all the simulated sets, including the most divergent ones.

The above observation shows that CD-HIT, DNACLUST and UCLUST are efficient in determining a correct or closely correct number of clusters in the case where the clusters' member sequences are convergent. For instance, for the least divergent simulated sets, these tools were efficient in most cases when applied with a similarity threshold equal to 0.6 or 0.4. But, unlike SpCLUST, these tools failed to cluster the relatively divergent sets.

Table 3 shows the numbers of clusters, produced by each tool while clustering the real data sets. The expected numbers of clusters are equal to 6 clusters for the first set, 7 for the second set, 5 for the third set and 3 for the fourth set. CD-HIT-Est, and similarly to the experiments performed on the simulated sets, failed to run using the similarity thresholds 0.6 or 0.4. CD-HIT-Protein also failed while processing the protein sequences of the third set. DNACLUST failed to process the genomic sequences of the first and second sets, while DACE failed to process all the genomic sets. DACE also failed processing three out of the four protein sets using the similarity thresholds 0.9 or 0.6.

It is important to highlight that both High Performance Computing tools, DACE and HPC-CLUST, were successfully tested using a data set provided by one of their authors and containing tens of thousands of sequences. The main difference between this data set and ours is the length of the sequences. Therefore, although these tools might be well optimized to cluster large number of sequences, it seems they cannot handle lengthy sequences which are very common in real life cases.

[7] https://www.ncbi.nlm.nih.gov/homologene.

[8] https://www.ncbi.nlm.nih.gov/genomes/FLU/Database/nph-select.cgi#mainform.

[9] https://www.ncbi.nlm.nih.gov/genomes/VirusVariation/Database/nph-select.cgi.

[10] https://github.com/johnymatar/SpCLUST.

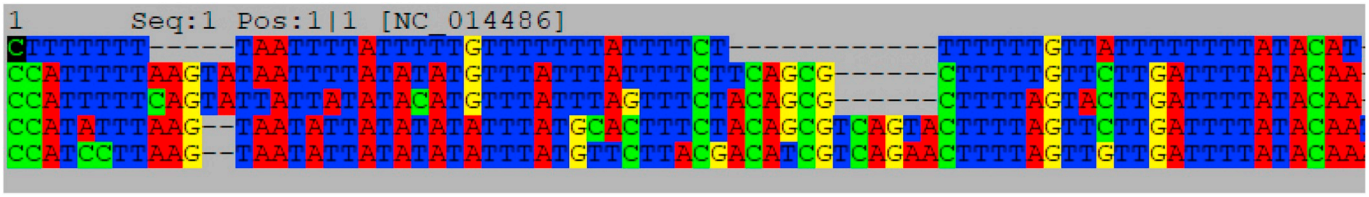

Fig. 9. ND3 simulated mutation.

Table 2
The number of clusters returned by each clustering tool for the simulated sets.

Similarity threshold or scoring matrix	CD-HIT			DNACLUST			UCLUST			DACE		SpCLUST		
	0.9	0.6	0.4	0.9	0.6	0.4	0.9	0.6	0.4	0.6	0.4	DNAF.	PAM.	BLOS.
ND3 simul. set S1	124	E	E	124	36	1	124	30	13	E	E	2	–	–
ND3 simul. set S2	94	E	E	110	23	1	106	18	10	E	E	5	–	–
ND3 simul. set S3	76	E	E	99	4	1	65	4	3	8	8	4	–	–
ND3 simul. set S′1	124	E	E	124	36	1	124	31	20	E	E	2	–	–
ND3 simul. set S′2	85	E	E	109	18	1	110	14	13	E	E	4	–	–
ND3 simul. set S′3	68	E	E	97	4	1	74	5	4	22	22	5	–	–
Prot. simul. set S1	155	54	26	–	–	–	155	66	40	E	E	–	4	5
Prot. simul. set S2	110	36	13	–	–	–	151	46	17	E	E	–	6	6
Prot. simul. set S3	89	5	4	–	–	–	114	8	4	40	22	–	6	6
Prot. simul. set S′1	155	58	34	–	–	–	155	51	34	E	E	–	5	5
Prot. simul. set S′2	113	34	15	–	–	–	150	36	13	E	18	–	6	5
Prot. simul. set S′3	102	5	4	–	–	–	112	7	4	21	E	–	5	5
Clp. simul. set S1	186	E	E	E	E	E	186	63	32	E	E	6	–	–
Clp. simul. set S2	133	E	E	E	E	E	142	40	24	E	E	5	–	–
Clp. simul. set S3	78	E	E	E	E	E	86	6	6	E	E	4	–	–
Clp. simul. set S′1	186	E	E	E	E	E	186	65	43	E	E	5	–	–
Clp. simul. set S′2	133	E	E	E	E	E	146	45	33	E	E	5	–	–
Clp. simul. set S′3	72	E	E	E	E	E	89	6	6	E	E	5	–	–

Table 3
The number of clusters returned by each clustering tool for the real data sets.

Similarity threshold or scoring matrix	CD-HIT			DNACLUST			UCLUST			DACE			SpCLUST		
	0.9	0.6	0.4	0.9	0.6	0.4	0.9	0.6	0.4	0.9	0.6	0.4	DNAF.	PAM.	BLOS.
Set 1 genomic	33	E	E	E	E	E	33	17	8	E	E	E	4	–	–
Set 2 genomic	40	E	E	E	E	E	40	19	6	E	E	E	4	–	–
Set 3 genomic	5	E	E	10	43	44	5	1	1	E	E	E	5	–	–
Set 4 genomic	3	E	E	10	10	24	3	3	1	E	E	E	4	–	–
Set 1 protein	19	7	6	–	–	–	19	7	6	E	E	5	–	4	4
Set 2 protein	25	9	7	–	–	–	25	9	7	E	E	5	–	3	4
Set 3 protein	1	E	E	–	–	–	1	1	1	3	1	1	–	3	4
Set 4 protein	3	3	3	–	–	–	3	3	3	E	E	1	–	4	4

The results displayed in Table 3 show that, for the genomic sets, SpCLUST succeeded in producing the exact number of clusters for the third set and a close number for the others. CD-HIT-Est and UCLUST produced the exact number for the third and fourth sets while using a similarity threshold equal to 0.9, and UCLUST returned close numbers of clusters for the first and second sets while using a similarity threshold of 0.4. For the protein sets, the results displayed in Table 3 show that SpCLUST, for all the sets, only returned close number of clusters to the expected ones while CD-HIT-Protein and UCLUST produced the exact number for the first, second, and fourth sets with a similarity threshold equal to 0.4, while DACE produced close numbers for the first three sets only.

Given these results, it can be noticed that for the real data sets some tools in some cases give equal or better quality clustering than SpCLUST. However, it is also important to highlight the high impact of the similarity threshold's choice on the quality of the clustering. For example, for the genomic sequences of the second set, UCLUST returned 6 clusters instead of 7 when given a similarity threshold equal to 0.4. On the other hand, it returned 19 and 40 clusters for similarity thresholds equal to 0.6 and 0.9 respectively. Therefore, for sequence sets with an unknown degree of similarity and an unknown expected number of clusters, SpCLUST remains a better choice because it returns exact or close number of clusters in all cases and without any prior knowledge about the input sequences. Moreover, for the highly similar protein sequences of the third set SpCLUST returned 4 clusters instead of 5, using BLOSUM62's matrix while CD-HIT-Protein and UCLUST put all the sequences in one cluster even with a similarity threshold equal to 0.9.

In summary, for the most divergent sets of the artificially mutated data sets, only SpCLUST gave the exact or close number of clusters, the results of the other tools were very far from the expected ones. In the less divergent simulated and real sets, SpCLUST gave better or at least good results when compared to the other tools. In the next section, the clusters' contents in these experiments are compared and discussed.

5.2.2. *Analysis of the clusters' contents*

In this section, the contents of the clusters, returned by the selected clustering tools, are compared in order to evaluate their accuracy. To evaluate the degree of similarity between a given clustering and the expected one, a metric should be defined. In the literature, many metrics are proposed, and some are presented in [45]. In this paper the Adjusted

Rand Index was selected. It computes a similarity measure between two clusterings, the predicted and the true clusterings, by considering all pairs of samples and counting pairs that are assigned in the same or different clusters. The Adjusted Rand Index requires the knowledge of the correct cluster which is the case in the previous described experiments. This index ranges between 0 for two completely different clusterings to 1 for identical clusterings. The compared clusterings do not have to have the same number of clusters. Moreover, the Adjusted Rand Index is symmetric: swapping or changing the clusters' labels, in a certain set, does not affect the calculation, e.g. the sets [a,a,a,b,b,c,c] and [c,c,c,a,a,d,d] are still detected identical although the labelling of the clusters' elements was changed. Therefore, the Adjusted Rand Index is a good fit in our case since it only requires, for its calculation, the labeling of the clusters' elements according to which cluster they belong in the perfect clustering.

Table 4 displays for each simulated data set, the Adjusted Rand Index calculated between the known exact clustering and the one returned by each tool. Only the clusterings that had the correct number of clusters or a very close one are considered because the remaining clusterings have an Adjusted Rand Index close to 0. Based on the values of the computed Adjusted Rand Index, and beside the fact that SpCLUST was the only tool to cluster well all the data sets, the average Adjusted Rand Index for the clustering of the 16 sets was equal to 0.805. Therefore, it can be stated that the proposed module, SpCLUST, performed well, compared to the other tools, and delivered a good overall clustering quality. In contrast, the other clustering tools returned good results in only 1/ 3 of the studied cases. Therefore, Even if for 1/3 of the cases the average of the displayed Adjusted Rand Indexes for certain tools is better than SpCLUST's average index, when considering the remaining 2/ 3 cases, where this index falls to nearly zero for the other tools, their overall index averages are way below SpCLUST's average index.

In addition, it can be noticed, from the data in Table 4, that CD-HIT-Protein was the only tool to return the exact clustering for one of the least divergent sequences' sets of protein, S3. However SpCLUST gave a nearly perfect clustering for this set, with a scored Adjusted Rand Index of 0.992, and using either scoring matrices. As for the other proteins sets, SpCLUST returned the exact clustering for one of them and for the rest of the data sets it gave good quality clustering results having an Adjusted Rand Index varying between 0.7 and 0.992 using either PAM250 or BLOSUM62.

For the genomic sets, on the one hand, DNACLUST and UCLUST succeeded in returning a perfect clustering for the least divergent sets, S3 and S′3. But on the other hand, SpCLUST also performed well and returned clusterings with Adjusted Rand Indexes varying between 0.783 and 1 for these same sets. But although the tools returned very good results, in the case of the least divergent genomic or protein sets they require a user intervention to choose the adequate similarity threshold.

Moreover, since SpCLUST does not rely on any user input identity parameter, it outperforms the other tools in the case of clustering highly divergent data sets. Conversely, the other tools proved to be highly accurate in clustering convergent sets, and at least one of these tools succeeded in finding the true clustering for each one of the least divergent sets.

The mentioned sequence divergence is illustrated in Table 5, using Levenshtein distance [46] which counts the number of characters insertions or substitutions between two strings. A random starting sequence was chosen from each group of reference sequences and the Levenshtein distance was calculated between this sequence and a randomly chosen mutation of this same sequence from each set, S1 to S′3. The values displayed in Table 5 show how the Levenshtein distance decreases from sets S1 and S′1 to sets S3 and S′3. The distances for the other reference sequences in each set should be close to the calculated ones since they were generated using the same mutation criteria.

Finally, although SpCLUST's results do not seem good in the case of the ND3 sets S1 and S′1 (Adjusted Rand Indexes equal to 0.082 and 0.282), a logical reason might explain this phenomenon: in fact, contrary to the other sets which originate from sequences of different genes, the ND3 sets derive from 4 initially divergent sequences of the same ND3 gene. Therefore, the simulated mutations might randomly cause the descendent sequences to converge or make some of the descendent sets highly diverge from the others. In this last case, the distance between the resulting mutated genes might make them look like being, more likely, part of two different clusters instead of four: one cluster containing the descendents that either re-converged towards each other or reached a closely equal distance between each other, and another cluster containing the remainder of the sequences that diverged more from the others. Indeed, the clusters contents, of the clustering of ND3's S1 and S′1, supports the presented theory: for both sets, one cluster contains a certain number of descendents from the same original sequence and the other cluster contains all the remaining sequences.

Table 5
Levenshtein distance between the original and mutated sequences.

	S1	S′1	S2	S′2	S3	S′3
ND3	167	178	147	119	41	38
Prot	186	167	158	137	36	44
Clp	120	81	77	56	20	17

Table 4
Adjusted Rand index for simulated data sets clustering.

Similarity threshold or scoring matrix	CD-HIT		DNACLUST	UCLUST		SpCLUST		
	0.6	0.4	0.6	0.6	0.4	DNAFULL	PAM250	BLOSUM62
ND3 simul. set S1	–	–	–	–	–	0.082	–	–
ND3 simul. set S2	–	–	–	–	–	0.637	–	–
ND3 simul. set S3	–	–	1	1	0.476	1	–	–
ND3 simul. set S′1	–	–	–	–	–	0.282	–	–
ND3 simul. set S′2	–	–	–	–	–	0.734	–	–
ND3 simul. set S′3	–	–	1	0.778	0.741	0.913	–	–
Prot. simul. set S1	–	–	–	–	–	–	0.406	0.7
Prot. simul. set S2	–	–	–	–	–	–	0.933	0.933
Prot. simul. set S3	1	0.778	–	0.876	0.778	–	0.992	0.992
Prot. simul. set S′1	–	–	–	–	–	–	0.438	0.984
Prot. simul. set S′2	–	–	–	–	–	–	0.933	0.643
Prot. simul. set S′3	1	0.778	–	0.945	0.778	–	0.968	1
Clp. simul. set S1	–	–	–	–	–	0.406	–	–
Clp. simul. set S2	–	–	–	–	–	0.797	–	–
Clp. simul. set S3	–	–	–	1	1	0.783	–	–
Clp. simul. set S′1	–	–	–	–	–	0.368	–	–
Clp. simul. set S′2	–	–	–	–	–	0.517	–	–
Clp. simul. set S′3	–	–	–	1	0.987	0.82	–	–

Table 6 presents, for each real data set, the Adjusted Rand Index calculated between the true and the predicted clusterings. SpCLUST was the only tool that was able to return acceptable clusterings for all the real data sets. It produced good overall clustering quality for all the real data sets whereas the other clustering tools failed to return acceptable results for at least one of the sets.

Moreover, it can be noticed that SpCLUST outperforms CD-HIT and UCLUST in the first 3 genomic sets and returns better quality clusterings. For the protein sets, CD-HIT and UCLUST gave better quality clusterings than SpCLUST and DACE gave the lowest quality clusterings. Another remarkable phenomenon is that, in contrary to the tests results on the simulated data sets, CD-HIT and UCLUST performed, in general, better than SpCLUST on the real data sets. In fact, a closer look at the data sets shows that 2 out of the 4 sets are very convergent. In fact, CD-HIT and UCLUST both delivered either a perfect or a good clustering for the genomic sets 3 and 4 as well as the protein set 4 with a similarity threshold equal to 0.9. Moreover, for the protein sets 1 and 2, CD-HIT and UCLUST also produced better results than SpCLUST, and for a similarity threshold of 0.6. This shows that the first couple of sets' clusters contents are also not very divergent like it was the case in most of the simulated data sets.

Appendix A presents some tables illustrating SpCLUST's clustering contents. Tables 10 and 11 show SpCLUST's clustering contents for the genomic real data sets 2 and 4 that scored Adjusted Rand Indexes equal to 0.47 and 0.869 respectively. Each column in these tables holds an SpCLUST cluster's content. The cells sharing the same color hold sequences that should be in the same cluster. Thus, looking to Table 10 shows that SpCLUST successfully isolated in one cluster, the MC1R sequences that belongs to a gene coming from a different chromosome than all the other sequences is this set. The other clusters contain mainly a mix of 2 or 3 true clusters, and not a random shuffle of elements. This is caused by the fact that elements of these clusters may share by chance a certain percentage of similarity, without presenting the same characteristics or reflecting a real homology. Similarly, looking to Table 11 shows that SpCLUST perfectly isolated Zika virus' sequences from Influenza's. The other clusters also tend to be perfect and only 3 Influenza's H2N2 sequences, that might be a bit too divergent from their other relatives, were clustered separately. It is the same for the set of protein sequences, as shown in Table 12.

Since SpCLUST scored the lowest indexes, 0.386 and 0.354, thus the worst clustering quality, for the first real genomic set and the second real protein set, a further investigation was undertaken for these two sets. The quality of their clustering was evaluated again using another metric, the purity of a cluster [47] which is computed for each cluster as in equation (1). The purity of a given cluster might vary between $[0, 1]$. It will be close to zero if the cluster's elements are found in different clusters in the perfect clustering and on the other hand it will increase up to 1, if the cluster have all its elements from the same cluster in the perfect clustering. The quality of a single cluster C_i is computed according to the maximum of p_j^i which is the number of elements that C_i has in common with the cluster j in the perfect clustering.

$$purity(C_i) = \frac{1}{|C_i|} * max_j\left(p_j^i\right) \quad (1)$$

Table 7 shows the purity of the clusters obtained for the first real genomic set and the second real protein set: the clustering of the first set had three pure clusters out of four and the clustering of the second one had two pure clusters out of four and a third cluster with a high purity. Therefore, the resulting low adjusted Rand Indexes do not come from having a bad quality clustering but rather from the merger of two or three clusters with similar elements from the perfect clustering. These results are consistent with the previous analysis of the clustering results of these sets.

Finally, it can be concluded that based on the clusters' contents quality interpreted in this section, SpCLUST presents a better tool for clustering highly divergent data sets, while the other tested tools remain a good choice for more convergent sets, but only in the case of a good similarity threshold choice. However, choosing a good similarity threshold is usually not trivial and requires some knowledge about the data set contents, or the expected number of clusters, which is almost never available in practice for newly discovered sequences. Meanwhile, SpCLUST does not need any user intervention and it offers an innovative tool for clustering new or unknown, genomic or protein, sequence sets.

5.2.3. *Effect of sequences alignment quality on clustering results*

Since, in the proposed module, the result of the input sequences alignment represents the starting point of the clustering process pipeline, the quality of that alignment, as mentioned in Section 4, might impact both the process running time and the clustering accuracy. In this section, the effect of the alignment quality on SpCLUST's results is analyzed on the previously described real data sets.

The Levenshtein distance will be used to compute the distance between the same sequence in two alignments: a fast and a normal one. The ratio between the computed Levenshtein distance and the length of the sequence represents a normalized distance value going from 0 (for exactly similar sequences) to 1 (for completely distinct sequences). Thus, the distance between two alignments can be defined as the average of the distances ratios for all the sequences in the alignment. Table 8 shows the distance between the normal alignment and the fast alignment for the genomic and protein mixes of the real data sets. While the second genomic set has the biggest distance, the third genomic and protein sets have identical alignments with the fast and normal alignment.

Starting with the cases where the distance between the alignments did not affect the clustering results. In the case of the third mix from the real data sets, containing sequences from Influenza viruses variants, the

Table 7
Clusters purity.

	C1	C2	C3	C4
Set 1 Genomic	1	0.38	1	1
Set 2 Protein	1	1	0.32	0.82

Table 6
Adjusted Rand index for real data sets clustering.

Similarity threshold or scoring matrix	CD-HIT			UCLUST			DACE		SpCLUST		
	0.9	0.6	0.4	0.9	0.6	0.4	0.9	0.4	DNA.	PAM.	BLOS.
Set 1 genomic	–	–	–	–	–	0.359	–	–	0.386	–	–
Set 2 genomic	–	–	–	–	–	0.356	–	–	0.47	–	–
Set 3 genomic	0.68	–	–	0.68	–	–	–	–	0.786	–	–
Set 4 genomic	1	–	–	1	1	–	–	–	0.869	–	–
Set 1 protein	–	0.974	0.58	–	0.958	0.712	–	0.409	–	0.625	0.493
Set 2 protein	–	0.928	0.658	–	0.914	0.772	–	0.327	–	0.241	0.354
Set 3 protein	–	–	–	–	–	–	0.205	–	–	0.206	0.456
Set 4 protein	1	1	1	1	1	1	–	–	–	0.869	0.869

Table 8
Distance between normal and fast alignments.

	Set 1	Set 2	Set 3	Set 4
Genomic	0.1352	0.3	0	0.1473
Protein	0.1424	0.0878	0	0.0368

fast and the normal alignment were identical and they both produced the same clustering. In the case of the fourth mix from the real data sets, containing mixed sequences from Influenza and Zika viruses, the fast and the normal alignments were slightly different, their normalized distance was equal to 0.1473. However, the SpCLUST's clustering result, using a fast alignment, was similar to the one using a normal alignment, for both genomic and protein sets of this mix. This shows that even with a small normalized distance between two alignments, the quality of the clustering was not affected for these sets. Conversely, for the other sets, and although some distances between their alignments were smaller than 0.1473, it made a difference in the clustering quality.

Table 9 displays the Adjusted Rand Index between the clustering produced from a fast alignment and those produced from a normal alignment. It also shows the Adjusted Rand Index between the clustering produced from a fast alignment and the true clustering. Fast alignment produced a moderately dissimilar clustering in 3 out of 4 cases when compared to the clustering produced using a normal alignment. Moreover, the calculated Adjusted Rand Index between the clustering, produced from a fast alignment, and the true clustering reflects, in 3 cases out of 4, a slight deterioration in this clustering quality, compared to the one done with a normal clustering. This can be seen by comparing the current values of the Adjusted Rand Index with those in Table 6. Appendix A, Table 13 shows, for the second genomic real data set, the SpCLUST clustering produced from a fast alignment and can be compared to the clustering produced from a normal alignment and displayed in Table 10.

Based on the presented results, and knowing that the used viruses' genes sequences (in sets 3 and 4) are much smaller in size than the used mammals genes sequences (in sets 1 and 2), it can be said that a small distance, between a normal alignment and a fast one, does not affect the clustering results in the case of relatively small sequences. Conversely this distance, although being small, slightly impacts the quality of the clustering for larger sequences.

6. Discussion and future directions

In this work, an efficient and fast clustering package for potentially divergent nucleotide sequences is proposed. This package is based on the Python module presented in [8] which uses an unsupervised learning method to produce the clustering. However, the new package offers many improvements over the old one such as enhanced performance due to its implementation in C++ and its parallelization with MPI. A performance comparison between the original package and the new one shown a speed-up ranging from 37.9X to 44.6X when performing a high quality alignment and up to 126X when performing a fast alignment. Moreover, two additional substitution matrices for the distance matrix calculation, PAM250 and BLOSUM62, were added to the package which extends its capabilities to cluster protein sequences. The proposed package compiles and runs on both Linux and Windows and can be easily integrated to a GALAXY server.

A comparative study between SpCLUST and some existing and widely used clustering tools, such as UCLUST [13], CD-HIT [12], DNACLUST [14] and DACE [16], was conducted over different sets of simulated and real, genomic and protein, sequences. In contrast with these state of the art tools, SpCLUST does not mainly aim for higher clustering speeds, of highly similar sequences, than its competitors. SpCLUST aims for fast clustering of data sets containing potentially divergent elements, and without any a priori knowledge of the similarity threshold or the number of clusters. The experiments shown that in the most cases SpCLUST gave better or fairly good results, compared to the other tools, in terms of number of clusters and their contents. Moreover, for highly divergent sequences that the other tools were not able to cluster, SpCLUST gave good clustering quality compared to the expected clustering. Finally, unlike the other tools that need a highly influencing similarity threshold parameter input, SpCLUST does not require any user intervention.

Despite proving that the use of the Gaussian Mixture Model (GMM) along with the Bayesian Information Criterion (BIC) provides a good clustering tool for potentially divergent sequences that does not require any user intervention, and despite the huge performance improvement introduced in SpCLUST when compared to the previous implementation of the algorithm, the complexity of this model remains by far higher than the complexity of the traditional greedy or hierarchical ones. Therefore, the proposed parallel tool remains much slower than the traditional High Performance Computing tools based on hierarchical algorithms. The traditional tools are better suited to cluster highly similar sequences with a well known similarity threshold, while SpCLUST successfully serves its intended purpose of fast clustering in the case of highly divergent sequences or when a typical similarity threshold is hard to specify.

Possible future extensions to the proposed package include the development of a custom and a faster parallel alignment sub-module instead of using MUSCLE. Further performance improvement could be gained by implementing in C++ the clustering sub-module, in addition to increasing its convergence rate. Moreover, introducing a graphical user interface for the proposed tool and enabling a graphical visualization of the clustering could improve the accessibility of the package and the analysis of the clustering results. Finally, a comparison with other tools, based on machine learning models might be interesting, along with the evaluation of the effect of different affinity matrices, as well as some parameters in the GMM model, on the performance and accuracy of SpCLUST.

Table 9
Adjusted Rand Index - Fast alignment.

	Set 1 gen.	Set 1 prot.	Set 2 gen.	Set 2 prot.
SpCLUST normal align.	0.053	0.568	0.41	0.38
Perfect clustering	0.289	0.404	0.383	0.386

Appendix A. Clusters contents

Table 10
Clustering - Real data genomic set 2.

C1	C2	C3	C4
H.sapiens_MC1R	H.sapiens_SH3BGRL3	H.sapiens_S100A6	G.gallus_S100A6
P.troglodytes_MC1R	C.lupus_SH3BGRL3	P.troglodytes_S100A6	H.sapiens_S100A8
C.lupus_MC1R	B.taurus_SH3BGRL3	M.mulatta_S100A6	M.mulatta_S100A8
B.taurus_MC1R	M.musculus_Sh3bgrl3	C.lupus_S100A6	C.lupus_S100A8
M.musculus_Mc1r	R.norvegicus_Sh3bgrl3	B.taurus_S100A6	B.taurus_S100A8
R.norvegicus_Mc1r		M.musculus_S100a6	M.musculus_S100a8
G.gallus_MC1R		R.norvegicus_S100a6	R.norvegicus_S100a8
D.rerio_mc1r		G.gallus_SH3BGRL3	H.sapiens_S100A1
		H.sapiens_FCER1G	P.troglodytes_S100A1
		P.troglodytes_FCER1G	M.mulatta_S100A1
		C.lupus_FCER1G	C.lupus_S100A1
		B.taurus_FCER1G	B.taurus_S100A1
		M.musculus_Fcer1g	M.musculus_S100a1
		R.norvegicus_Fcer1g	R.norvegicus_S100a1
			G.gallus_S100A1
			D.rerio_s100a1
			H.sapiens_S100A12
			P.troglodytes_S100A12
			C.lupus_S100A12
			B.taurus_S100A12

Table 11
Clustering - Real data genomic set 4

C1	C2	C3	C4
KU758869 ZIKA	CY083917 H1N1 PB2	CY021939 H2N2 PB1	CY021027 H2N2 PB1
KU312313 ZIKA	CY063613 H1N1 PB2	CY020323 H2N2 PB1	AY210016 H2N2 PB1
KU758873 ZIKA	CY083782 H1N1 PB2	CY022019 H2N2 PB1	CY020419 H2N2 PB1
KU758868 ZIKA	CY073732 H1N1 PB2	CY021811 H2N2 PB1	
KU312314 ZIKA	CY062698 H1N1 PB2	CY021795 H2N2 PB1	
KU758872 ZIKA	CY062706 H1N1 PB2		
KU758876 ZIKA			
KU758871 ZIKA			
KU758870 ZIKA			
KU758875 ZIKA			

Table 12
Clustering - Real data protein set 4

C1	C2	C3	C4
AML81020 ZIKA	ADX98969 H1N1 PB2	ABQ01363 H2N2 PB1	ABO52255 H2N2 PB1
ALX35660 ZIKA	ADH01967 H1N1 PB2	ABO38106 H2N2 PB1	AAO46332 H2N2 PB1
AML81024 ZIKA	ADX98798 H1N1 PB2	ABQ44468 H2N2 PB1	ABO38742 H2N2 PB1
AML81019 ZIKA	ADN05235 H1N1 PB2	ABP49467 H2N2 PB1	
ALX35661 ZIKA	ADG42162 H1N1 PB2	ABP49445 H2N2 PB1	
AML81023 ZIKA	ADG42172 H1N1 PB2		
AML81027 ZIKA			
AML81022 ZIKA			
AML81021 ZIKA			
AML81026 ZIKA			

Table 13
Clustering - Real data genomic set 2 - Fast alignment

C1	C2	C3	C4
H.sapiens_MC1R	H.sapiens_SH3BGRL3	H.sapiens_S100A6	H.sapiens_S100A8
P.troglodytes_MC1R	C.lupus_SH3BGRL3	P.troglodytes_S100A6	M.mulatta_S100A8
C.lupus_MC1R	B.taurus_SH3BGRL3	M.mulatta_S100A6	C.lupus_S100A8
B.taurus_MC1R	M.musculus_Sh3bgrl3	C.lupus_S100A6	B.taurus_S100A8
M.musculus_Mc1r	R.norvegicus_Sh3bgrl3	B.taurus_S100A6	M.musculus_S100a8
R.norvegicus_Mc1r		M.musculus_S100a6	
G.gallus_MC1R		R.norvegicus_S100a6	
D.rerio_mc1r		G.gallus_S100A6	
		G.gallus_SH3BGRL3	
		R.norvegicus_S100a8	
		H.sapiens_S100A1	
		P.troglodytes_S100A1	
		M.mulatta_S100A1	
		C.lupus_S100A1	
		B.taurus_S100A1	
		M.musculus_S100a1	
		R.norvegicus_S100a1	
		G.gallus_S100A1	
		D.rerio_s100a1	
		H.sapiens_FCER1G	
		P.troglodytes_FCER1G	
		C.lupus_FCER1G	
		B.taurus_FCER1G	
		M.musculus_Fcer1g	
		R.norvegicus_Fcer1g	
		H.sapiens_S100A12	
		P.troglodytes_S100A12	
		C.lupus_S100A12	
		B.taurus_S100A12	

References

[1] S. Duffy, L.A. Shackelton, E.C. Holmes, Rates of evolutionary change in viruses: patterns and determinants, Nat. Rev. Genet. 9 (4) (2008) 267.

[2] S. Wielgoss, J.E. Barrick, O. Tenaillon, M.J. Wiser, W.J. Dittmar, S. Cruveiller, B. Chane-Woon-Ming, C. Médigue, R.E. Lenski, D. Schneider, Mutation rate dynamics in a bacterial population reflect tension between adaptation and genetic load, Proc. Natl. Acad. Sci. 110 (1) (2013) 222–227.

[3] A. Oliver, A. Mena, Bacterial hypermutation in cystic fibrosis, not only for antibiotic resistance, Clin. Microbiol. Infect. 16 (7) (2010) 798–808.

[4] E. Gullberg, S. Cao, O.G. Berg, C. Ilbäck, L. Sandegren, D. Hughes, D.I. Andersson, Selection of resistant bacteria at very low antibiotic concentrations, PLoS Pathog. 7 (7) (2011) p. e1002158.

[5] C.L. Ventola, The antibiotic resistance crisis: part 1: causes and threats, Pharm. Therapeut. 40 (4) (2015) 277.

[6] C. Lim, E. Takahashi, M. Hongsuwan, V. Wuthiekanun, V. Thamlikitkul, S. Hinjoy, N.P. Day, S.J. Peacock, D. Limmathurotsakul, Epidemiology and burden of multidrug-resistant bacterial infection in a developing country, Elife 5 (2016) p. e18082.

[7] M. Sørensen, H. Autrup, P. Møller, O. Hertel, S.S. Jensen, P. Vinzents, L.E. Knudsen, S. Loft, Linking exposure to environmental pollutants with biological effects, Mutat. Res. Rev. Mutat. Res. 544 (2) (2003) 255–271.

[8] M. Bruneau, T. Mottet, S. Moulin, M. Kerbiriou, F. Chouly, S. Chretien, C. Guyeux, A clustering package for nucleotide sequences using laplacian eigenmaps and Gaussian mixture model, Comput. Biol. Med. 93 (2018) 66–74.

[9] E. Afgan, D. Baker, B. Batut, M. Van Den Beek, D. Bouvier, M. Čech, J. Chilton, D. Clements, N. Coraor, B.A. Grüning, A. Guerler, J. Hillman-Jackson, S. Hiltemann, V. Jalili, H. Rasche, N. Soranzo, J. Goecks, J. Taylor, A. Nekrutenko, D. Blankenberg, The galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update, Nucleic Acids Res. 46 (W1) (2018) W537–W544.

[10] P. Larrañaga, B. Calvo, R. Santana, C. Bielza, J. Galdiano, I. Inza, J.A. Lozano, R. Armañanzas, G. Santafé, A. Pérez, V. Robles, Machine learning in bioinformatics, Briefings Bioinf. 7 (03 2006) 86–112.

[11] S.I. Vrieze, Model selection and psychological theory: a discussion of the differences between the akaike information criterion (aic) and the bayesian information criterion (bic), Psychol. Methods 17 (2) (2012) 228.

[12] W. Li, A. Godzik, " Cd-hit, A fast program for clustering and comparing large sets of protein or nucleotide sequences, Bioinformatics 22 (13) (2006) 1658–1659.

[13] R.C. Edgar, Search and clustering orders of magnitude faster than blast, Bioinformatics 26 (19) (2010) 2460–2461.

[14] M. Ghodsi, B. Liu, M. Pop, Dnaclust: accurate and efficient clustering of phylogenetic marker genes, BMC Bioinf. 12 (Jun 2011) 271.

[15] C. Mercier, F. Boyer, A. Bonin, E. Coissac, Sumatra and sumaclust: fast and exact comparison and clustering of sequences, in: Programs And Abstracts Of the SeqBio 2013 Workshop, Citeseer, 2013, pp. 27–29. Abstract.

[16] L. Jiang, Y. Dong, N. Chen, T. Chen, DACE: a scalable DP-means algorithm for clustering extremely large sequence data, Bioinformatics 33 (12 2016) 834–842.

[17] J.F. Matias Rodrigues, C. von Mering, Hpc-clust: distributed hierarchical clustering for large sets of nucleotide sequences, Bioinformatics 30 (2) (2013) 287–288.

[18] N. Sato, Gclust: trans-kingdom classification of proteins using automatic individual threshold setting, Bioinformatics 25 (01 2009) 599–605.

[19] W. Chen, C.K. Zhang, Y. Cheng, S. Zhang, H. Zhao, A comparison of methods for clustering 16s rrna sequences into otus, PLoS One 8 (8) (2013) p. e70837.

[20] P.D. Schloss, S.L. Westcott, T. Ryabin, J.R. Hall, M. Hartmann, E.B. Hollister, R. A. Lesniewski, B.B. Oakley, D.H. Parks, C.J. Robinson, et al., Introducing mothur: open-source, platform-independent, community-supported software for describing and comparing microbial communities, Appl. Environ. Microbiol. 75 (23) (2009) 7537–7541.

[21] Y. Sun, Y. Cai, L. Liu, F. Yu, M.L. Farrell, W. McKendree, W. Farmerie, Esprit: estimating species richness using large collections of 16s rrna pyrosequences, Nucleic Acids Res. 37 (10) (2009) pp. e76–e76.

[22] X. Hao, R. Jiang, T. Chen, Clustering 16s rrna for otu prediction: a method of unsupervised bayesian clustering, Bioinformatics 27 (5) (2011) 611–618.

[23] H.Z. Girgis, B.T. James, B.B. Luczak, MeShClust: an intelligent tool for clustering DNA sequences, Nucleic Acids Res. 46 (05 2018) pp. e83–e83.

[24] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, Ser, 2001.

[25] R.C. Edgar, Muscle: multiple sequence alignment with high accuracy and high throughput, Nucleic Acids Res. 32 (5) (2004) 1792–1797.

[26] cogent.pypi, Accessed: 2018-10-11, https://pypi.org/project/cogent/.

[27] T.E. Oliphant, Python for scientific computing, Comput. Sci. Eng. 9 (May 2007) 10–20.

[28] M. Fourment, M.R. Gillings, A comparison of common programming languages used in bioinformatics, BMC Bioinf. 9 (Feb 2008) 82.

[29] C. Notredame, D.G. Higgins, J. Heringa, T-coffee: a novel method for fast and accurate multiple sequence alignment1, J. Mol. Biol. 302 (1) (2000) 205–217.

[30] K. Katoh, D.M. Standley, Mafft multiple sequence alignment software version 7: improvements in performance and usability, Mol. Biol. Evol. 30 (4) (2013) 772–780.

[31] S. Mirarab, N. Nguyen, S. Guo, L.-S. Wang, J. Kim, T. Warnow, Pasta: ultra-large multiple sequence alignment for nucleotide and amino-acid sequences, J. Comput. Biol. 22 (5) (2015) 377–386.

[32] J.D. Thompson, T.J. Gibson, D.G. Higgins, Multiple sequence alignment using clustalw and clustalx, Curr. Protoc. Bioinform. 00 (1) (2002) 2.3.1–2.3.22.

[33] K.-B. Li, Clustalw-mpi: clustalw analysis using distributed and parallel computing, Bioinformatics 19 (12) (2003) 1585–1586.

[34] Muscle user guide, Accessed: 2018-09-27, http://www.drive5.com/muscle/muscle.html.

[35] X. Deng, E. Li, J. Shan, W. Chen, Parallel implementation and performance characterization of muscle, in: Proceedings 20th IEEE International Parallel Distributed Processing Symposium, April 2006, p. 7, pp.–.

[36] A. Wilm, I. Mainz, G. Steger, An enhanced rna alignment benchmark for sequence alignment programs, Algorithm Mol. Biol. 1 (1) (2006) 19.

[37] V. Ahola, T. Aittokallio, M. Vihinen, E. Uusipaikka, A statistical score for assessing the quality of multiple sequence alignments, BMC Bioinf. 7 (1) (2006) 484.

[38] P.A. Nuin, Z. Wang, E.R. Tillier, The accuracy of several multiple sequence alignment programs for proteins, BMC Bioinf. 7 (1) (2006) 471.

[39] Rosalind — glossary — dnafull, Accessed: 2018-09-27, http://rosalind.info/glossary/dnafull/.

[40] Substitution matrix, Accessed: 2018-09-27, https://en.wikipedia.org/wiki/Substitution_matrix.

[41] sklearn.mixture.gaussianmixture, Accessed: 2018-10-10, http://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html.

[42] G.I. Lang, A.W. Murray, Estimating the per-base-pair mutation rate in the yeast saccharomyces cerevisiae, Genetics 178 (1) (2008) 67–82.

[43] Influenza, Accessed: 2019-02-12, https://en.wikipedia.org/wiki/Influenza.

[44] W.R. Pearson, An introduction to sequence similarity ("homology") searching, Curr. Protoc. Bioinform. 42 (1) (2013), pp. 3–1.

[45] S. Wagner, D. Wagner, Comparing Clusterings: an Overview, Universität Karlsruhe, Fakultät für Informatik Karlsruhe, 2007.

[46] Levenshtein distance, Accessed: 2019-03-07, https://en.wikipedia.org/wiki/Levenshtein_distance.

[47] S. Schulte im Walde, *Experiments On the Automatic Induction Of German Semantic Verb Classes*. PhD Thesis, Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart, 2003. Published as AIMS Report 9(2).

Christophe Guyeux obtained his agrégation in mathematics in 2001 and a thesis in computer science in 2010. He was a lecturer at the University of Burgundy Franche-Comté, in computer science from 2011 to 2014, and is now a professor at the same university. He is particularly interested in big data and bioinformatics issues.

Hicham El Khoury is an associate professor in the computer science and statistics department at the Lebanese University, Lebanon. He got his master's degree in Modelling from Lebanese University, and his Ph.D. degree in Computer Science and Telecommunications from Paul Sabatier – Toulouse III, France. His research interests include Formal Methods, Recommender Systems, and Science of Education.

Stephane Chretien holds a Ph.D. in Signal Processing and Automatic Control from the University of Paris XI-Orsay (1996) and was a student at the Ecole Normale Superieure de Cachan from 1989 to 1993, France. He was Assistant and associate professor in the mathematics department at the University of Franche-Comte, Besançon from 2000 to 2015. He is now Senior Research Scientist at the National Physical Laboratory, the UK's national metrology institute, located in Teddington.

Johny Matar is a bioinformatics PhD student at the University of Bourgogne Franche-Comte and a computer science laboratory assistant in the computer science and statistics department of the Lebanese University, Faculty of Science II. He got his master's degree in computer science from American University of Culture and Education, and his applied math diploma, option: computer science from the Lebanese University. His research interest includes bioinformatics, IoT and networking.

Jean-Claude Charr holds a PhD in computer science from the University of Franche-Comte (2009). He got his master's degree in computer science from the university of Paul Sabatier in Toulouse (2006). Since 2010, He is an associate professor in the DISC department of the FEMTO-ST institute. His research interest includes distributed computing and high performance computing.