Recursive Binary Divisor: a new tool to study the diversity of alpha-satellite DNA in Old World Monkeys

Louis-Mael Gueguen
louis-mael.gueguen@etu.univ-lyon1.fr
Master en Bioinformatique

Laboratory: **Diversité et instabilité des génomes**
Natural History Museum of Paris, INSERM U1154 ; CNRS UMR7196
43, rue Cuvier
75005 PARIS

Laboratory director: Dr. **Jean-Baptiste Boulé**
Team: **Repeated DNA, Chromatin and Evolution**
Team leader: **Dr. Christophe Escudé**
Supervisor: Dr. **Loïc Ponger**

Acknowledgments :

I would like to thank my supervisor, **Dr. Loïc Ponger**, for guiding me through my work, providing technical coding assistance, and discussing with me the algorithm from different angles. I would also like to thank **Dr. Christophe Escudé** for helping me to understand the underlying biological concepts and meanings of the subject and data. Both helped me interpreting the biological results of the new tool.

I would like to give my acknowledgments to **Thomas Hashka**, the creator of MNHN-Tree-Tools, the heart of my program, for providing explanations of the mathematical concepts in use in the project.

And finally I want to thank the two neighbor students, **Marine** and **Anaïs**, who welcomed me and helped me with practical aspects of the life in the Museum.

Summary:

Alpha-satellites are tandemly repeated DNA sequences located in the centromeric region of chromosomes. They have particular evolution mechanisms that led to the existence of numerous sizes and distribution patterns among species. In primates, their length is about 170 b.p. and they are organised in two distinct ways: monomeric and high-order-repeats (HOR). The monomeric organisation is more particular to the pericentromere and the HOR are mostly located in the centromere. Although these sequences are homologous, they show a diversity of sequences that can be classified in families. They are extensively studied in Great Apes including humans, but work still needs to be done about them in Old World Monkeys. A PhD student of the National Museum of Natural history took interest in the Cercopithecini clade in particular for its diversity and characterised several alpha satellites families. Several clustering techniques were tried, including one of a computational tool named MNHN-Tree-Tools, written by another member of the same structure. It helped going further in the classification and the characterisation of the alpha-satellites was improved. Yet groups of sequences seemingly separated by a PCA projection were not made distinct by this tool. This justified the need for a different approach. The new tool, the Recursive Binary Divisor, is based on MNHN-Tree-Tools and dbscan. The underlying idea is that a recursive separation and projection of the data would allow to vary the relative importance of the dimensions. Such variation reflects the fact that different kmers differentiates groups from each others. RBD was benchmarked on its parameters and two trends were observed. Most parameters would decrease the running time as they increase due to finding less clusters, while the kmer length and number of sequences show steep increasing exponential curves. The use of RBD on *C. pogonias* permit the obtention of results that overlap the ones of previous studies: a C1-like and C2-like families were found, as well as the C6-like, C5T-like and C5G-like. This confirms the validity of the process as it shows the capacity to retrieve a classification confirmed by FISH experiments. Moreover, new clusters were distinguished, some of the very early in the clustering steps. It demonstrate the potential of the Recursive Binary Divisor to further classify alpha-satellites sequences of *C. pogonias*. More refinement is needed in order to fully exploit it, and new features can be developed already to offer more possibilities to the user of RBD.

Table of Contents

Table of Figures

Abbreviations:

RBD:	Recursive Binary Divisor
PCA:	Principal Component Analysis
bp:	base pairs
HOR:	High Order Repeats

Softwares and tools:

Python 3.4.2 https://www.python.org/
Numpy 1.14.5 https://numpy.org/
Subprocess https://docs.python.org/3/library/subprocess.html
Argparse https://docs.python.org/3/howto/argparse.html
Regex 2017.4.29 https://docs.python.org/3.5/library/re.html
GCC 4.9.2 https://gcc.gnu.org/
MNHN-Tree-Tools 1.0 http://treetools.haschka.net/
Hyperfine 1.11.0 https://github.com/sharkdp/hyperfine
R 3.6.3 https://www.r-project.org/
Rstudio 1.2.5033 https://www.rstudio.com/
Ggplot2 3.3.3 https://cran.r-project.org/web/packages/ggplot2
Cowplot 1.1.1. https://CRAN.R-project.org/package=cowplot
Seqinr 4.2-8 https://CRAN.R-project.org/package=seqinr

1. Introduction:

1.1. Alpha-satellite DNA

The centromeric region is a primary part of chromosomes construction (Steiner and Henikoff, 2015) and is characterized by a particular organization, structure and function that discriminates it from the rest of the genome. Its localization and its range are defined by the presence of the histone variant cenH3 (Schueler and Sullivan, 2006), and it is divided in two sub-regions, the centromere and pericentromere, which are distinguished by histone marks (Gopalakrishnan et al., 2009). The region is the target of mitotic spindle attachment during anaphase. Its composition is characterized by repeated sequences of DNA organised in tandem repeats named satellite DNA (Steiner and Henikoff, 2015). From a species to the other, the primary sequence, the nucleotide content and length of the tandem repeats differ: the basic repetition unit can vary from 7 pb in the red-necked wallaby up to 3,2 kb in the cow (Giannuzzi et al., 2012). Evolution of these tandem repeated sequences gave rise to plentiful monomer families with specific organizational arrangement. Nevertheless, the associated proteins and the function of the centromeric region is conserved. This is known as the centromere paradox (Malik and Henikoff, 2002). In primates in particular, satellite DNA sequences are named alpha-satellites. These alpha-satellites have a length of around 170bp (Tyler-Smith and Brown, 1987). They are spread through the entire genome but differ in organization types between the centromere and pericentromere. The monomeric organization is prevalent in the pericentromere, the high-order repeat (HORs) organization dominates in the centromere (Figure 1). The monomeric organisation is formed by a succession of repeated monomeres of the same family, whereas the HOR organisation is formed by the cyclic repetition of a succession of monomeres, of several families.

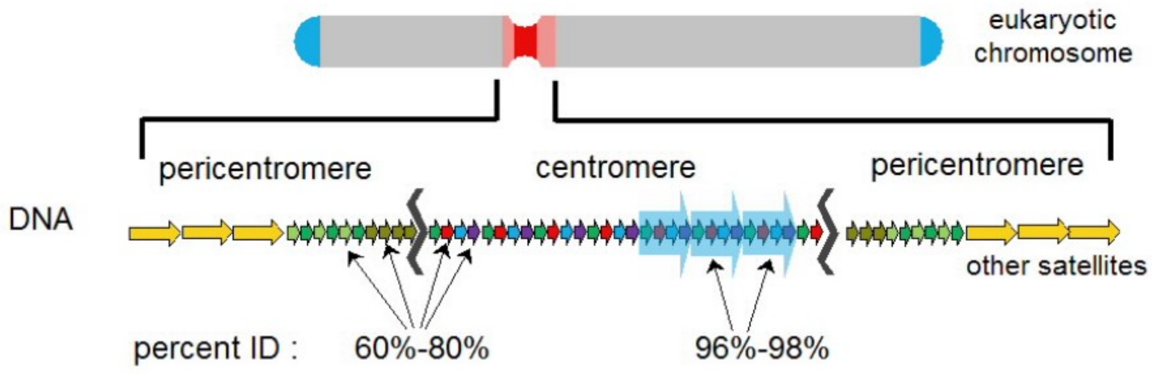


Figure 1: *Figure 1: alpha satellite DNA in human genome.*
Same colored arrow represents alpha-satellite from different families. Transparent-blue arrows shows HOR organisation in the centromere. I.e.: Repeats formed of successibe alpha-satellite belonging to different families. Figure from Christophe Escude.

Alpha-satellites have their own peculiar evolution mechanisms (Plohl et al. 2008) like rolling-circles amplification events. Successive occurrences of this type of event partially explain the repetitive structures observed in monomeric or HOR structure. The evolutionary dynamics of alpha satellite DNA have been partially uncovered by several studies, most of them compare human DNA sequences with those of great apes (Schueler and Sullivan 2006; Cellamare et al. 2009; Shepelev et al. 2009; Catacchio et al. 2015; Chiatante et al. 2017). An age gradient observed from centromere toward chromosome arms led to suggest that, on a single chromosome, alpha-satellite sequence families emerge and expand at the centromere core, hence splitting and pushing older families toward the distal arm of the chromosome, in the so-called pericentromeric regions (Schueler et al. 2005). HOR are therefore more similar between each others than with sequences of the monomeric organisation. Additionally, a certain amount of evidence point toward the existence of transfer of alpha satellite DNA between chromosomes. If alpha-satellites diversity and evolution is extensively studied in Great Apes and Humans, it is poorly known in Old World Monkeys. A way to better understand these subjects is to identify the sequence families through clustering, and construct phylogenetic trees from their relations.

1.2. Clustering and phylogeny of alpha-satellites

A recent study (Cacheux and *al*, 2018) took interest in the alpha-satellites DNA in Old Word Monkeys, and in the Cercopithecini clade in particular. Cercopithecini represent 35 species of Old World that have diverged over the last 10 Myr (Tosi 2008; Guschanski et al. 2013). It has been shown that numerous chromosomal rearrangements associated with centromere repositioning or emergence of new centromeres occurred in this clade (Dutrillaux et al. 1980; Moulin et al. 2008). This feature make them interesting for alpha-satellites DNA studies. Deep sequencing (Ion Torrent) of monomers and dimers (ie, two successive monomers) of the 60 chromosomes of *Cercopithecus solatus* (Dutrillaux et al. 1980; Moulin et al. 2008) provided evidence for the existence of 4 previously undiscribed families named C1 to C4 (Cacheux et al. 2016). The same deep sequencing technology applied to the 72 chromosomes of *Cercopithecus pogonias* revealed similarities in the presence of families C1 and C2 but differences in the sub-composition of C1 (Cacheux and *al*, 2018).

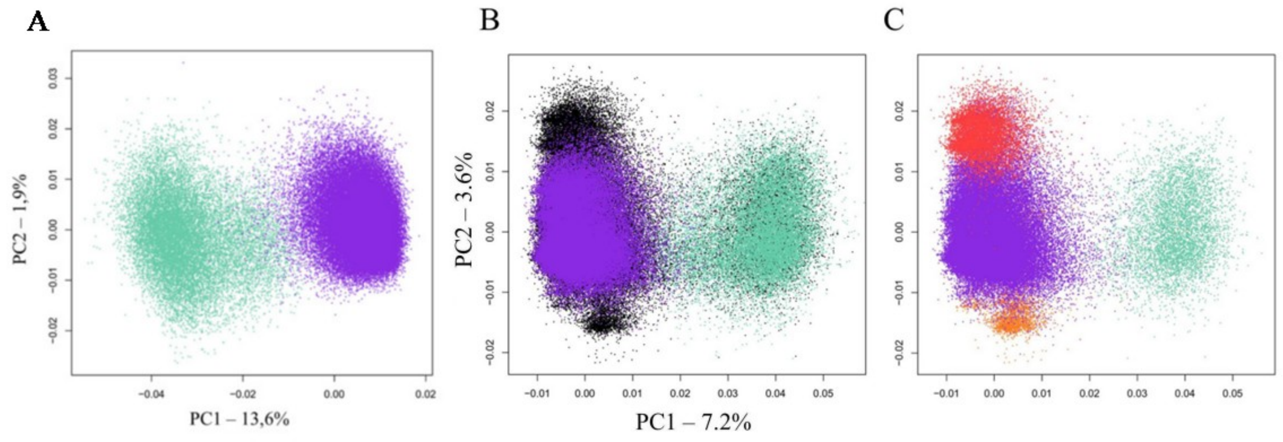


Figure 2: *Figure 2: Characterization of α-satellite DNA diversity in C.solatus and C. pogonias monomer dataset (Cacheux et al., 2016).*
(C) PCA projection of the normalized 5-mer frequency vectors for C. pogonias monomer dataset. Each family is represented respectively by a color : C1 (purple), C2 (green), C5 (red), C6 (orange)

Further work on *C. pogonias* revealed similar projections of the sequences in the predominant C1 and C2 families. Hints of the existence of numerous sub-families were gathered, but no way to clearly differentiate them was found.

1.3. In silico processing of repeated sequences

The repetitive organisation, and the relative sequence similarity and the large number of repetitions of alpha-satellites sequences make them a challenge for current bioinformatics tools. First, sufficient length and quality of sequencing must be reached in order to assemble genome scaffolds. Then, the most commonly used methods are alignment tools or database tools. The first cannot align such sequences because they are so similar it ends up with multiple alignement. The second requires building reference databases, and might not fit for the target organism. An emerging practice is the kmer-count method, which is an alignment and database free, that allows faster computation. This is the reason to be of MNHN-Tree-Tools (Haschka et *al*, 2021). This tool is build on the dbscan algorithm (Ester et *al*, 1996). Dbscan clusters points scattered in n-dimensions based on density. Such process is independent of cluster shape, and requires little user input. Iterating through densities, clusters are found and form layers, dense clusters are fused in lesser dense ones. From this a hierarchy is drawn, and a phylogeny can be inferred. However, MNHN-Tree-Tools does not separate visually distinct groups that are observed on projections of the Cercopithecini alpha-satellites data (the two higher comet-like structures on figure 3).

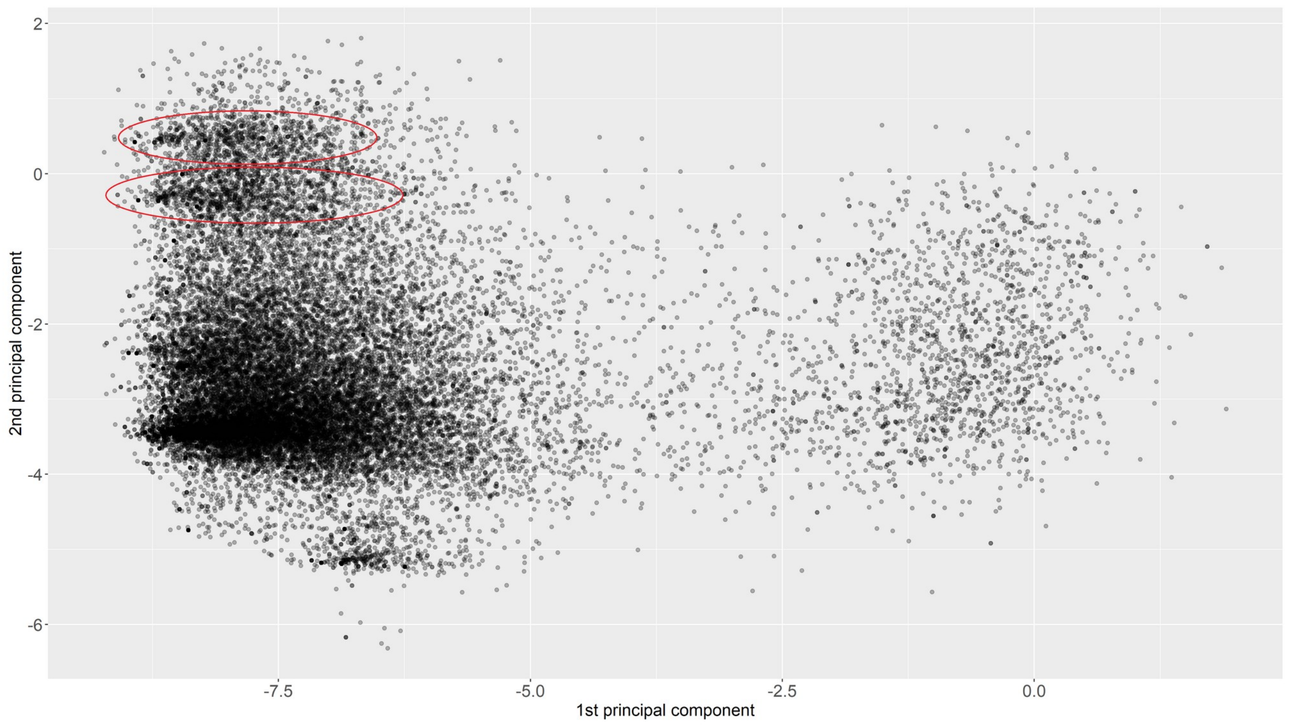


Figure 3*: PCA projection of the sequences. Red ellipses show comet-like structures.*

It has been shown that iterative binary divisions on different PCA projections yields good results on tandem repeats (Human Satellites, HSat2 and 3) sequences families identifications (Altemose et *al*, 2014).

During my traineeship, I developed a new algorithm in order to try to separate these groups in a classification. It uses a recursive binary classification as described in Altemose et *al*, (2014), and the clustering method of MNHN-Tree-Tools to identify cluster pairs. The new tool is named Recursive Binary Divisor; its algorithm, its benchmark, and a result of its use on *C. pogonias* data are presented in this report.

2. Material and methods:

2.1. Sequence data

The cells came from the Collection of cryopreserved living tissues and cells of vertebrates (Cell collection, Muséum national d'Histoire naturelle, Paris). The cells used for DNA extraction were fibroblastic cell. The extraction was performed and the gDNA of *C.pogonias* has been treated by XmnI or HindIII endonucleases activities. The gDNA was then transferred on a gel and gDNA migrated by electrophoresis. The gel was imaged by UV. Monomers, dimers and trimers constituting the libraries were extracted. The Pogo-M-XH library was created by mixing the *C.pogonias* monomers cut by HindIII and XmnI.
Samples were loaded onto the Flow Cell and sequencing was performed with Illumina technology. The Reagent Kit was adapted for having a paired-end sequencing of 250 pb and until 15 million of reads per flow cell. After sequencing, the adapters were removed from the sequences.
Before filtering, reverse and forward reads for all libraries were merged using FLASH (Magoc et al., 2011) with default parameters. Then, sequences with an average Phred sequence score above 25 were kept. The next step of filtering was only applied to monomer libraries. α-satellite sequences were identified with a BLAST search using a reference α satellite sequence of *Chlorocebus aethiops* (AM23721). Using default BLAST parameters, a homology filter as applied: all sequences exhibiting a hit with the reference sequence were considered as α-satellite. Sequences without the XmnI or HindIII digested sites at the extremities (respectively 5' – NNTTC … GAANN – 3' and 5'-ATGC … ATGC - 3') and a length outside 162 – 182 pb or 166 – 186, respectively for the XmnI or the HindIII monomer libraries, were not considered for further analysis. An iterative blast was performed to the unselected sequences by the first BLAST, using all first sequence selection as references for each library. Iterative BLASTs were performed until no sequence was recovered from the non-homologous dataset. All the recovered sequences from the iterative BLAST were filtered by the same extremities and length filters as previously. Both set of sequences were merged. All sequences were then reoriented if necessary to have the same orientation than the reference sequence. The orientation information was preserved for investigating possible sequencing biases during reading. Finally, the last four nucleotides added to the HindIII library sequences during the library preparation blunting step (3' ATCG) were discarded. The result is a library of phased sequences of similar length, for different species. This work has been accomplished by Noémie Chabot, and presented in her internship report.

2.2. The classification method: dbscan

Dbscan is an algorithm developed to find clusters in a large amount of data based on density information/criteria. It depends on a density calculated from two parameters, a radius epsilon and a minimal number of points minpoints. The formula defining the density threshold is the following:

$$\rho_{\text{clust}} > \rho_{\text{limit}}(\epsilon, \text{minpts}) = \frac{\text{minpts}}{V(\epsilon)} \tag{1}$$

where ρ is the density. It is calculated from a number of points (minpts) in a volume V(ϵ). This volume is the one of an n-dimensional sphere and is calculated from a radius ϵ. No other user input is necessary. Any group of points with a density superior to the threshold (Equation 1) is considered as a cluster (see Figure 4). The fact that the algorithm does not need a number of points to start with (e.g. k-means clustering) or recompute distance each step (e.g. hierarchical clustering) nor it forces every point in a cluster makes it reproducible, and able to detect a cluster of any shape in n-dimensions.

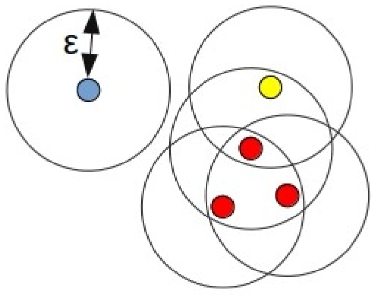


Figure 4: *Figure 4: Illustration of dbscan cluster analysis (minPts=3).*
Red points within an epsilon radius from each others: they are core points. They form a cluster. The yellow point is a border point: it is included in the cluster because it reaches at least one core point. The blue point does not reach minPts or a core point.

2.3 MNHN-Tree-Tools

The dbscan clustering method is used by of the core feature of MNHN-Tree-Tools. This tool is constituted by a series of functions that allows the user to go from a fasta file to layers of a. The first function counts the kmers. The resulting matrix is then passed onto the second function that performs a PCA (Principal Components Analysis) on this matrix; it yields an eigen value for each kmer and their corresponding eigen vectors. A number p of these are used to project the sequences as points in a p-dimensional space, where p is a parameter chosen by the user. Finally, the dbscan algorithm is used to determine the number of clusters found with the epsilon radisus and minimum of points (minpts) parameters. The output is composed of binary files containing the sequences of each layers, and a file retaining information of each epsilon for each layer.

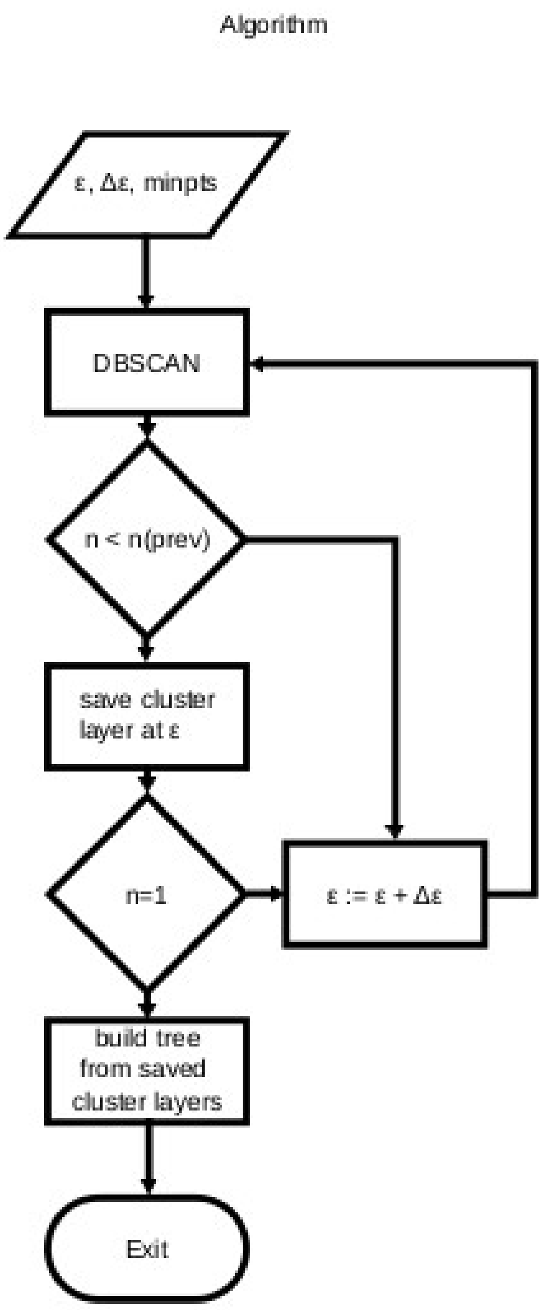


Figure 5: *Figure 5: programmation organigram of MNHN-Tree-Tools.*
Figure from MNHN-Tree-Tools: A toolbox for tree inference using multi-scale clustering of a set of sequences. (Haschka et al, 2021)

The epsilon value is iteratively increased by the user-defined parameter delta-epsilon (adaptative clustering). Thus, the density threshold required to find clusters is further lowered. Larger clusters are found, that includes the previous smaller clusters (figure 6).
A hierarchy is drawn, and a classification tree inferred, since the more different sequences are, the more they have diverged, the further they are projected on the PCA. If it makes sense to compare the tree to an evolutionary tree is an opened question and it will probably depend of the analysed sequences.

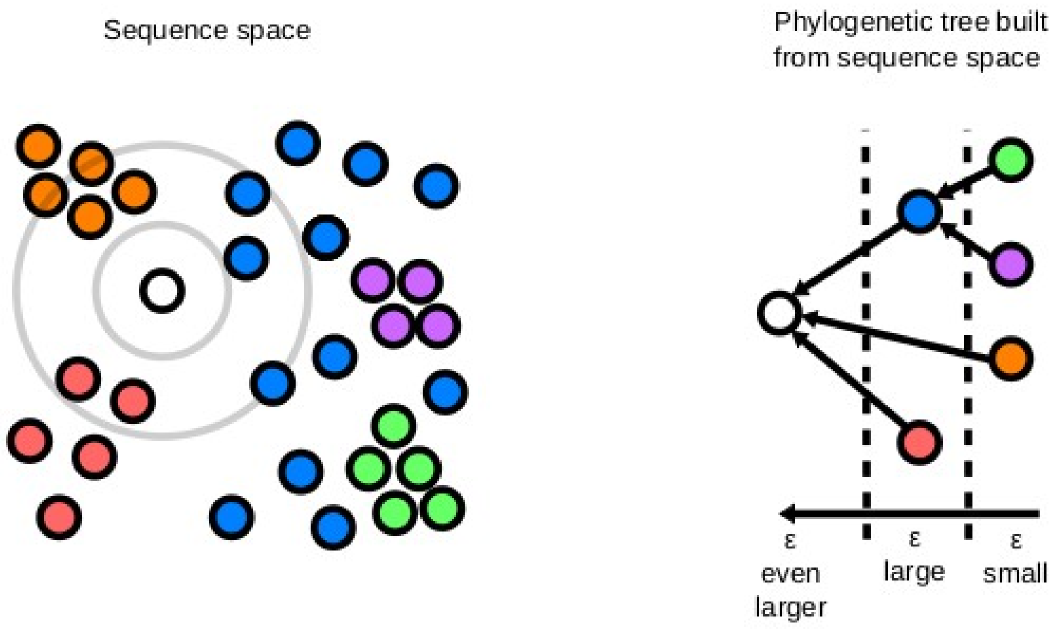


***Figure 6:** The adaptive clustering algorithm finds dense (new) clusters in less dense (old) more dispersed clusters.*
Finally the MNHN-Tree-Tools collection allows us to build a tree like the one on the right, from a sequence space as shown on the left (Haschka et al, 2021).

2.4. A new algorithm: Recursive Binary Divisor

The Recursive Binary Divisor (RBD) algorithm is based on MNHN-Tree-Tools. It is written in python 3.4, using libraries such as numpy, argparse, subprocess and regex. It has been tested with python 3.8.5 and the corresponding versions of the libraries, and is functional with these. RBD requires to have MNHN-Tree-Tools installed and its binaries in the path. Both repositories are available on GitHub (https://github.com/Louis-MG/RecursiveBinaryDivisor).
To conduct the benchmark, the tool hyperfine was used. The default command line for the runs used test.fst (a 1000 sequences (~170bp) fasta file), a kmer length of 5, the seventh first dimensions of the pca, an epsilon of 0.8 and a delta epsilon of 0.001, a minpoints of 3, a minsize of 10, 32 threads

and the verbosity option at level one. The parameters were replaced one by one by a list of values, each value has been ran 5 times. The statistics are collected by hyperfine and were used to analyse CPU time usage.
The machine was the cluster of the National Museum of National History.

3. Results:

3.1. RBD algorithm

A. Algorithm

The Recursive Binary Divisor algorithm is constituted of a main loop. That loop iterates through the clusters found. For each cluster, kmer-counts and a PCA projection are generated. Then, dbscan operates a clustering with an epsilon radius and minpts parameters on the points in the PCA projection. While more than two clusters are found, the delta epsilon parameter is used to increase epsilon. By increasing epsilon, smaller clusters will fuse to become a smaller number of bigger clusters. This is repeated until either the cluster is splited in two new clusters, either the number of clusters found becomes smaller than 2. In the first case, clusters are saved and will go through the same process. If 1 cluster is found, it is saved but will not be visited. If 0 cluster is found, the loop goes to the next cluster. See figure 7 for the programmation organigram.

RBD has a global complexity of O(4^k) where 4^k is the number of words of length k with the DNA alphabet (AGCT). If k is adjustable by the user, a good compromise between information and calculation time is found for k = 5. Above all, this value yielded good results in previous studies (Cacheux et *al*, 2018, Haschka et *al*, 2021). When k >= 6, the initiation step (kmer counts) bottlenecks the computation time. In other configurations, the clustering steps with epsilon and delta epsilon will take more time. The complexity of this part of the algorithm is O(n), where n is the number of cluster found. This number is unknown prior to run.
We can also underline that dbscan does not force every point to be in a cluster: then at each step some of the points will not be in a cluster (if clusters are found). These points are then called “orphans”.

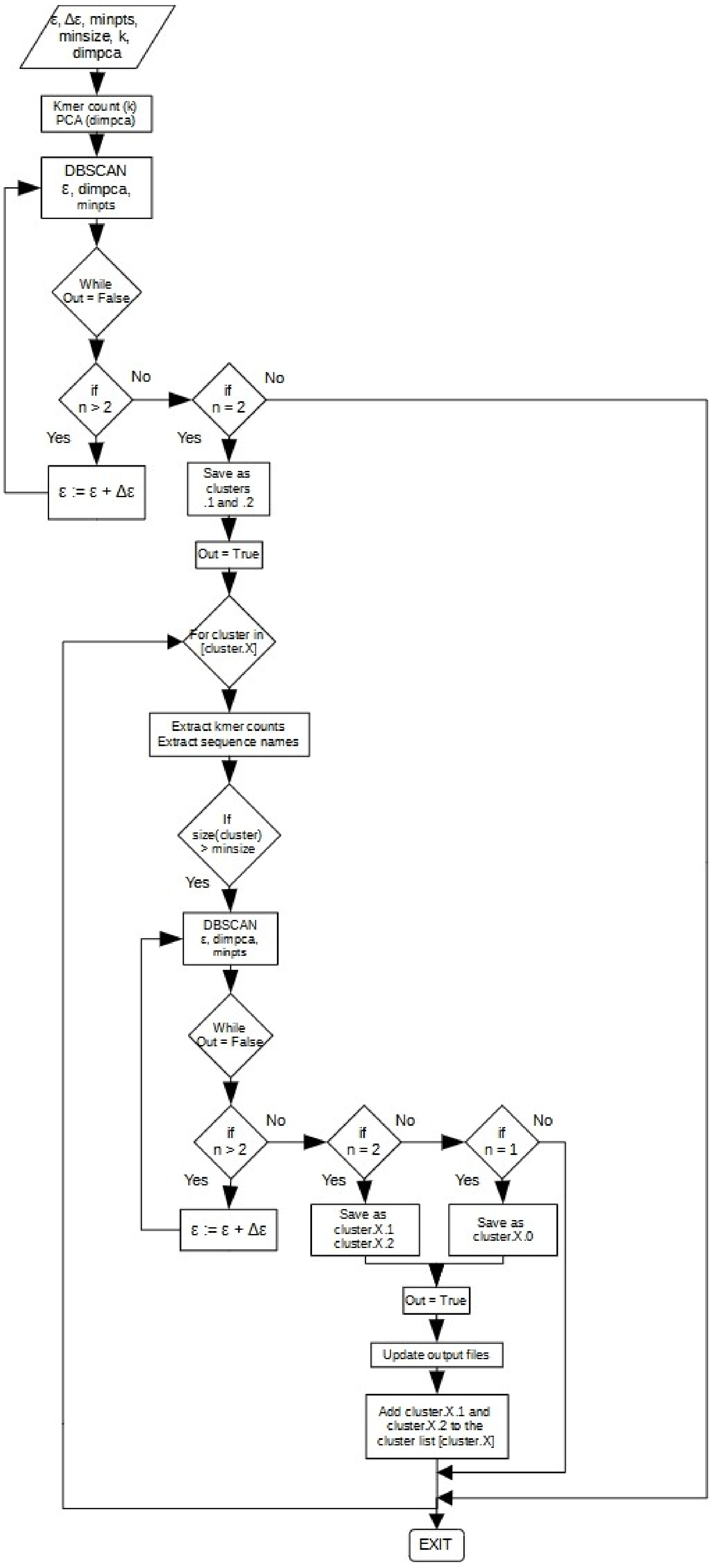
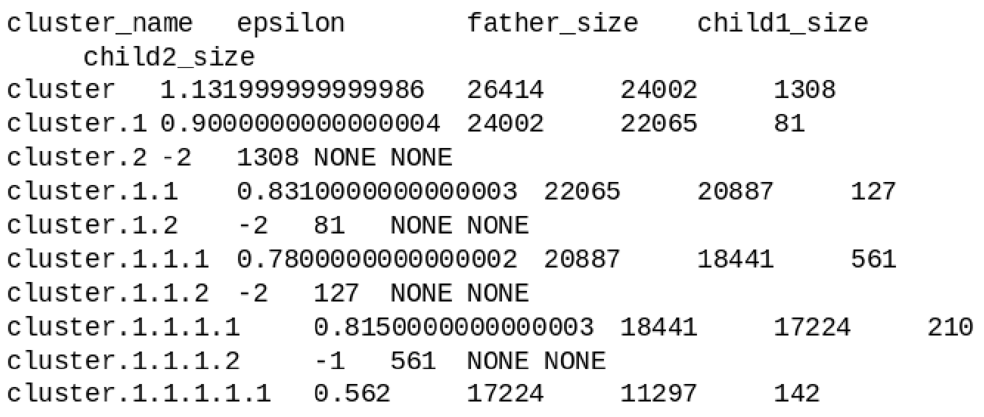

ε, Δε, minpts, minsize, k, dimpca
Kmer count (k) PCA (dimpca)
DBSCAN ε, dimpca, minpts
While Out = False
if n > 2
No
Yes
if n = 2
No
Yes
ε := ε + Δε
Save as clusters .1 and .2
Out = True
For cluster in [cluster.X]
Extract kmer counts Extract sequence names
If size(cluster) > minsize
Yes
DBSCAN ε, dimpca, minpts
While Out = False
if n > 2
No
Yes
if n = 2
No
Yes
if n = 1
No
Yes
ε := ε + Δε
Save as cluster.X.1 cluster.X.2
Save as cluster.X.0
Out = True
Update output files
Add cluster.X.1 and cluster.X.2 to the cluster list [cluster.X]
EXIT

Figure 7: *programmation organigram of RBD tool.*

B. Implementation

The tool is used through the CLI (Command Line Interface), with flags to specify required and optional arguments. Required arguments are the name of the fasta file, the epsilon value, and the output folder name. The optional ones are kmer length (default at 5), the delta epsilon (default at 0.001), the number of dimension from the pca (default at 7), the minium size of the clusters to try to split (default at 50), the number of threads (default 4) and the verbosity level (default at 0, up to 2).
The output is composed of cluster's folders and 3 tabulated text files: “cluster_parameters.txt”, “sequence_parameters.txt”, “sequence_summary.txt”, where “parameters” is a string of parameters used for the analysis: kmer, epsilon, delta epsilon, minpoints and pca dimensions put together and separated by an underscore. The file “cluster_parameters.txt” contains 5 headed columns: the cluster name, the epsilon code, the cluster size, and the respective sizes of its two children one and two. This file only contains information about clusters that were splited. If a cluster has a size inferior to the argument minsize, it will not be shown here. Figure 8 shows an exemple.



```
cluster_name   epsilon        father_size    child1_size
     child2_size
cluster   1.131999999999986   26414     24002     1308
cluster.1 0.9000000000000004  24002     22065     81
cluster.2 -2   1308 NONE NONE
cluster.1.1    0.8310000000000003  22065     20887     127
cluster.1.2    -2   81   NONE NONE
cluster.1.1.1  0.7800000000000002  20887     18441     561
cluster.1.1.2  -2   127  NONE NONE
cluster.1.1.1.1     0.8150000000000003  18441     17224     210
cluster.1.1.1.2     -1   561  NONE NONE
cluster.1.1.1.1.1   0.562     17224     11297     142
```

Figure 8: *Output "cluster_parameters" exemple of RBD.*

The cluster name keeps track of its past, from the right (most recent) to the left (older). E.G.: cluster.1.1.2 is the second child issued from the binary division of cluster.1.1, itself the first child coming from cluster.1, itself the first child of the first binary division of the original file. A “.0” indicates that the binary division of the older cluster yielded only one child. Knowing it could have a biological meaning, the information is tracked in the files. Some clusters will be “branches” of the tree, meaning they will be further divided, and others will be “leafs” of the tree, meaning they will not be further divided. The epsilon column shows various numbers, either strictly positive floats either negative integers. A strictly positive float indicates the epsilon value for which the binary division occurred. A negative integers indicates that the binary division did not occurred for various reasons. -1 means only one cluster was found (cluster.X.0). The code -2 signifies that no cluster was found. In both cases children sizes are NONE.
“sequence_parameters.txt” contains 2 columns: the name of the sequences, and their respective last cluster. “sequence_summary.txt” contains 2 columns as well: positive integers and cluster names. The positive integers is the number of sequences that have the cluster name as the last cluster they

belonged to in “sequence_parameters.txt”. Which means this is the size of the clusters that are leafs, but the number of remaining sequences in the cluster if it has been further divided. These sequences are the “orphans” of the division.
The cluster’s folders are named after the cluster names (e.g. cluster.1.1). They each contain the fasta file with the sequences in the cluster, the kmer counts file (counts.kmer) for these sequences, the pca file (counts.pca) computed from the kmer counts.

3.2. CPU benchmark

The algorithm has several parameters that affects computation steps, hence might affect the running time of the program. A CPU time benchmark has been ran with Hyperfine on the following: kmer length, the number of pca dimensions used by dbscan, minpoints, delta epsilon, the size of the input fasta file, the number of threads given. The results are plotted in the figure 9.

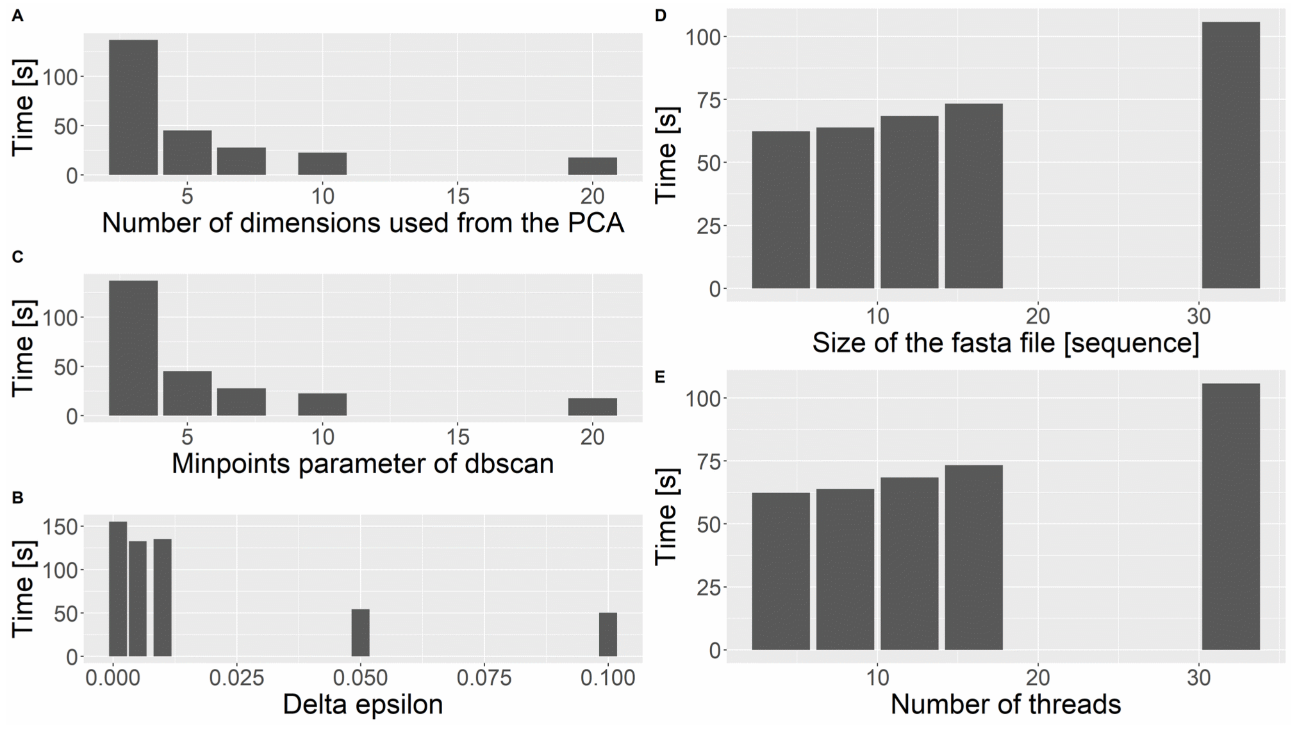


Figure 9: *graphics of CPU running time of RBD as a function of different parameters.*
(A) Time ine function of dimpca values. (B) Time as a function of delta epsilon values. (C) Time as a function of minpoints values. (D) Time as a function of the number of sequences. (E) Time as a function of the number of threads used.

The graphics of running time as a function of the size of the fasta file, delta, minpoints, kmer length and dimpca show an exponential curve. Only the fasta file and kmer length (figure 10) show an increasing exponential curve, the other parameters have the opposite effect: they decrease the running time as they increase. The number of threads barely changes the running time.

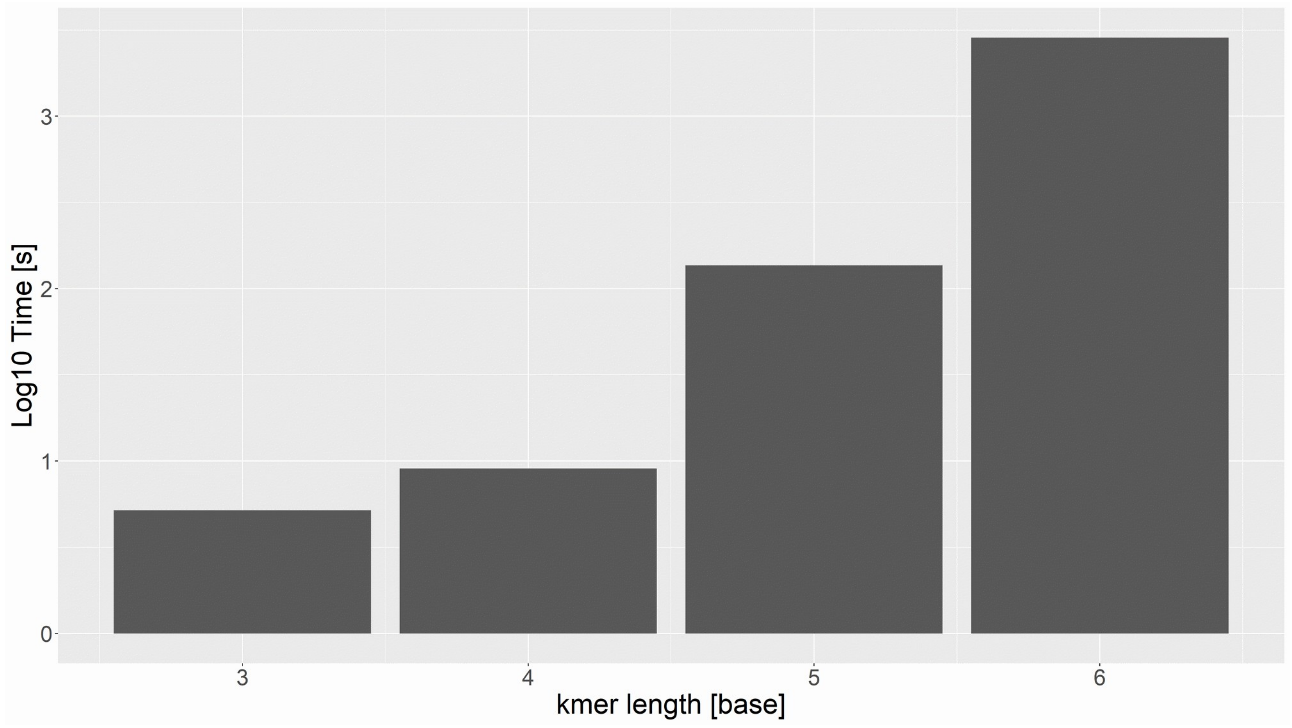


Figure 10: *Running time as a function of kmer length.*

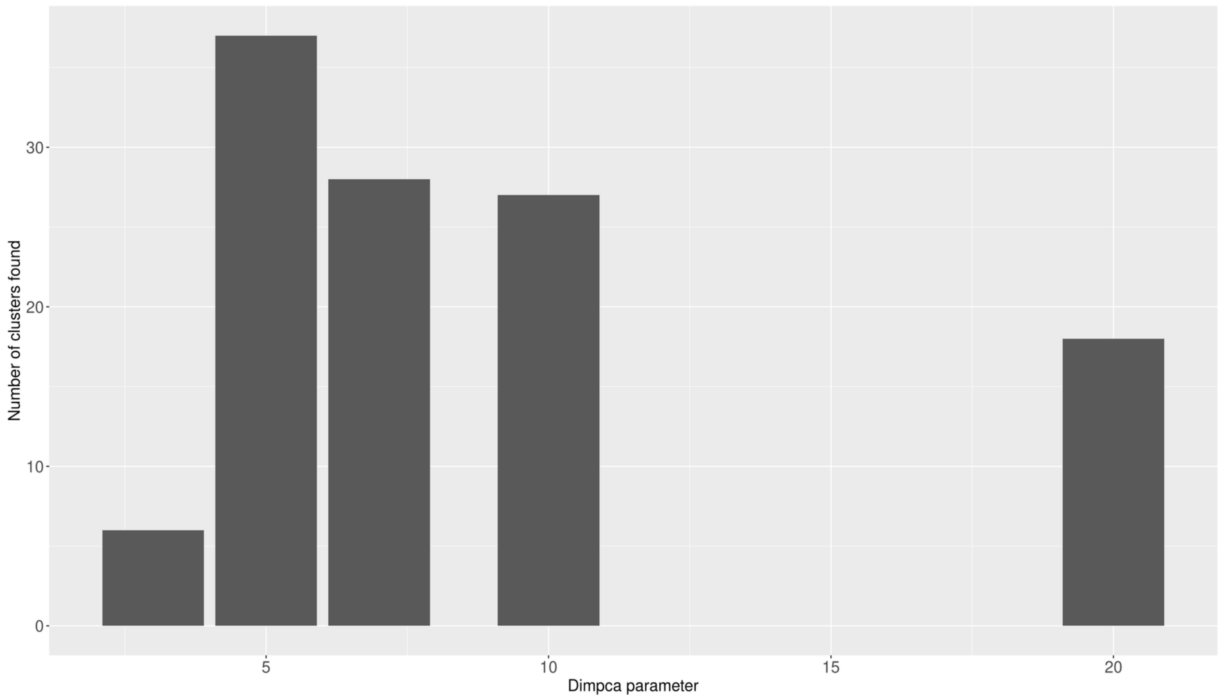


Figure 11: *number of clusters found as a function of the dimpca parameter*

3.3. Wall-clock benchmark

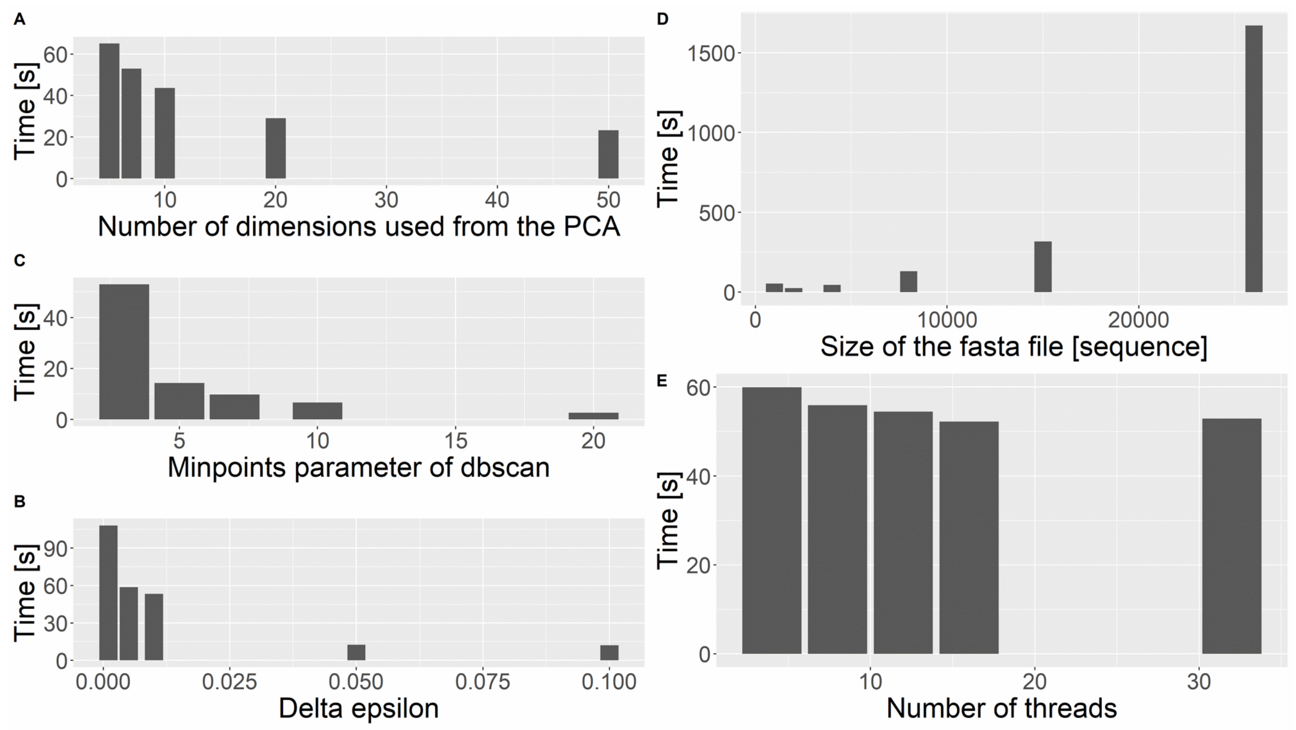
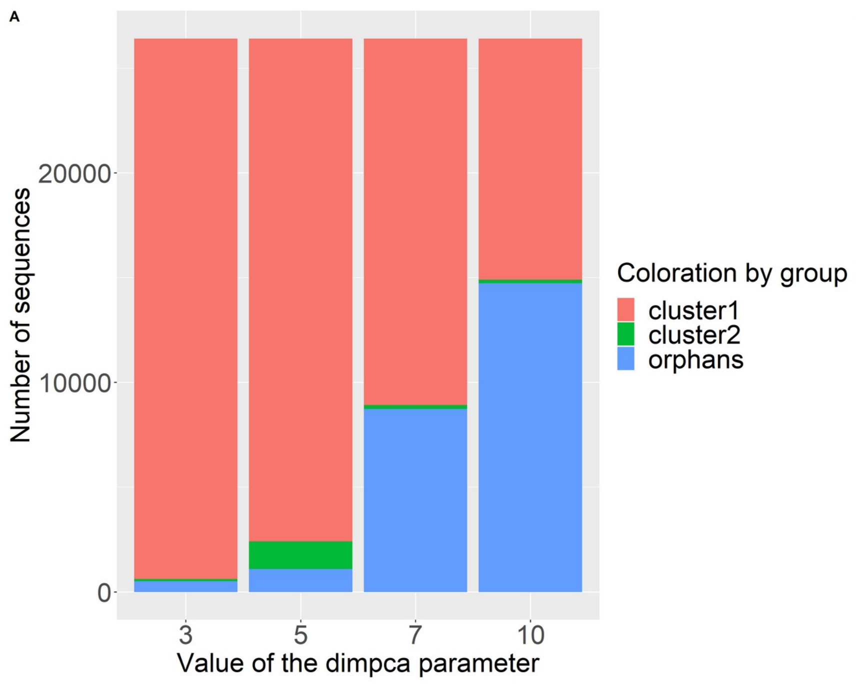


Figure 12: *graphics of wall-clock running time of RBD as a function of different parameters. (A) Time ine function of dimpca values. (B) Time as a function of delta epsilon values. (C) Time as a function of minpoints values. (D) Time as a function of the number of sequences. (E) Time as a function of the number of threads used.*

For more insight from a user point of vue (as the programm is intened to be used by biologists), the benchmark was ran on for wall-clock time measurement on the same machine (figure 12). We observe the same tendancies except for the number of threads.

3.4. Results on reference sequences

Several runs has been completed in order to find parameters fitting what was previously known on the reference sequences. As Chacheux et *al* showed the existence in *C. pogonias* of two large families C1 and C2, separated by the first axis of the PCA (based of a kmer length equals to 5), their distinction in particular was one of the primary features wanted in the results and the easiest to see. The run with a kmer length of 5, dimpca of 5, epsilon of 0.5, delta epsilon at 0.001, minpoints equals 50, yielded such results (cluster.1 contained around 90.8% of the dataset sequences and cluster.2 contained around 4.95% of the dataset sequences). Other values were tested, in particular dimpca. The different results given by different values of this parameter are shown in figure 13.
44 clusters were found in the dataset, including the single child clusters (marked with a .0). The 3 first PCA dimensions were used to visualise the sequences on the first PCA projection.



B

dimpca	cluster1	cluster2	orphans
3	25794	103	517
5	24002	1308	1104
7	17495	190	8729
10	11512	168	14734

Figure 13: *Size of cluster.1, cluster.2 and orphans of the first clustering step in function of different values for dimpca parameter.*
(A) Stacked bar chart. (B) Table with raw counts.

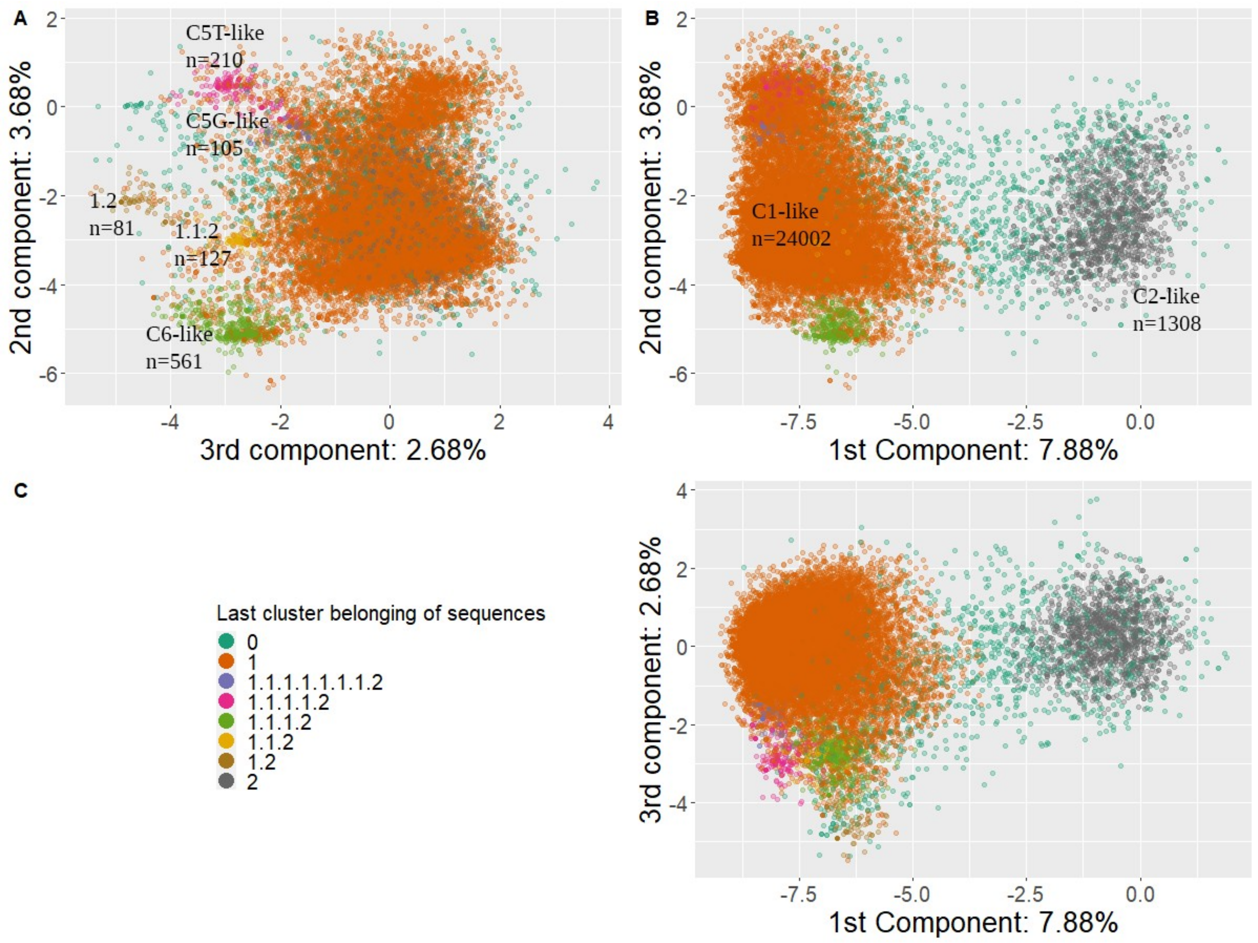


Figure 14: *PCA projection of the sequences, colored by cluster.*
Only representative clusters are shown. (A) Plot using 2nd and 3rd principal components. (B) Plot using 1st and 2nd principal components. (C) Plot using 1st and 3rd principal components.

Figure 14 displays the results obtained with a coloration of specific clusters. These were chosen based on their visibility, their correspondence to known families. Clusters 1 and 2 were chosen for corresponding to families C1 (ochre, 90.8% of the dataset) and C2 (grey, 4,9% of the dataset). Orphans from this binary division appear in dark turquoise. Cluster 1.1.1.2 (light green, 2.1% of the dataset) appears to correspond to the C6 family. The C5 family seems to be partially recovered (cluster 1.1.1.1.2, 0.79% of the data set, in pink, cluster 1.1.1.1.1.1.1.2, 0.39% of the dataset in purple). Other clusters are colored in brown (cluster 1.2, 0.30% of the dataset), and in gold (cluster 1.1.2, 0.48% of the dataset).

4. Discussion:

4.1. Benchmark

The results of the benchmark show a few surprises. First, the growing number of PCA dimensions used by dbscan decreases the running time. Considering more dimensions means more computations to get the distances between projections, one might think that it implies longer running time. But the simple explanation is that the algorithm finds less clusters. With dimpca at 7, the tool finds 28 clusters. At a value of dimpca equals to 20, the tool finds 18 clusters (figure 11).
The distances between points increases but the starting epsilon was not changed, hence less clusters with a density above the threshold were found. An other unexpected result is the running time as a function of the number of threads. The parameter n = 16 yields an equally fast computation time as with n = 32. To check whether or not this result was due to the small sample size of the test file, the same command was ran on a fasta file of size s = 4000 sequences. Running time were again very close.
The running time as a function of the kmer length shows a very steep exponential. The plots shows well how big is the difference between each run, and the benchmark for user-CPU time could not finish in time to get the value for k = 7. To investigate whether or not this was due to the algorithm finding a lot more clusters, there were counted with the result of the run for wall-clock time as it had the same parameters. An increase of 25% was found. For reference, with k equals to 5, 35 clusters were found whereas a kmer length of 7 gave rise to 44 clusters. The change might or might not be worth the time difference as the number of cluster does not guarantee their biological meaningfulness. This is still to be looked into.
The only difference observed for the wall-clock time benchmark was for the number of threads. Instead of a shorter time, we observe a higher time as the number of threads increases. This is due to the fact that when using mutilple threads, the computation load is divided between threads. So the addition of the time for each thread gives rise to higher measurement than the one experienced by the user which is the elapsed time. Otherwise, the curves follow the same tendancies.
MNHN-Tree-Tools also has a use of GPUs through the OpenCL; but the hardware to test running times of RBD under such conditions were not available and this parameter was not tested. It is predicted that it would indeed be faster but that the python part, not using the GPU accelerated

calculus, would bottleneck the program at some point. A potential improvement of the tool resides in threading and parallelizing. This could be done with the built-in library threading, or external modules such as Ray and Dask.
Another interesting thing to look at would be RAM usage. But no adapted tools were found, as they are generally oriented toward hardware benchmarking for games, thus prevalently developed for the Windows platform.

4.2 Known dataset

The Recursive Binary Divisor clustering allowed to recover several previously known families. It did separate well C1 and C2 families. It should be noted that a fair amount (about a thousand) of sequences were not distributed in the two clusters at this first step. This might reflect the divergence of certain sequences from an old amplification event. Mutations along with no other amplification event for these sequences might have led to a single sequence with much different nucleotide composition. This divergence is reflected in the kmer counts and thus in the PCA projection. The C6 family is also found in the first clustering steps. The C5 family is divided in 2 comet-like structures, more visible in figure 3 with the first and second principal components.
One is C5T and the other C5G. By looking at it with the second and third principal components, one could see the comet like structure split in two distant groups. Here, one group for each comet-like is clustered by RBD. These are visible in pink and purple..
Apart from finding already documented families, RBD highlighted the potential existence of two new families. One is particularly visible with the brown colored cluster 1.2 (figure 12, panel A and C). This hypothesis of its existence is made even more interesting considering that this cluster is found at the second clustering step. Discovering a cluster so early might have a biological meaning to look into. The other one, in gold, is more visible on the panel A of figure 9. Again, it is separated quite early in the clustering steps and seems worth investigating.
A lot of other clusters were found and are not displayed. The reasons are multiple. Firstly, they are located in the C1-like cluster which is very dense. Hence, they are hardly visible when plotting all the points. Secondly, the exploration of this type of data takes a lot of time and will be completed in a close future. New results will then be produced will more refined parameters.

5. Conclusion:

RBD is a new tool using an adaptive clustering technique with Dbscan. Its is based on a recursive binary clustering, done a PCA projection in n-dimensions of the previously made clusters. It is used through the command line, with flagged parameters make it an easy-to-use and flexible tool. Certain parameters such as the kmer length, minpoints have a heavy impact on the running time, although the critical values are at least not necessary to yield exploitable results on real data. The Recursive Binary Divisor did find C1, C2, and C6 like families, previously characterized families of alpha-satellites in *C. pogonias*. It even displayed the potential to distinguish groups of sequences that were not separated by the tool it is based on, MNHN-Tree-Tools. While new results still need confirmation by molecular experiments, the method seems adequate for already known families.

New clusters were also formed by RBD, opening new perspectives for characterisation of alpha-satellites in Old World Monkeys. Exploration of the first results is still going on, and more fine-tuning of the parameters might give a better insight on the data. There is still plenty of room for RBD improvement. Its process could be parallelized; a new potential parameter would be the directionality of epsilon. In the actual state of the program, epsilon grows, and so the 2 clusters saved (when it happens) are the smaller ones. Changing the growth for a diminution might allow to find the 2 biggest clusters for which the split happens (although it could also result in a bigger and a smaller cluster due to the order of discovery of dbscan). Another way to explorate the data would be to change certain parameters during the run: but which one, why and how remains to be decided. Overall, RBD shows a lot of potential, and will definitely be improved in the future.

6. Bibliography

Altemose N, Miga KH, Maggioni M, Willard HF (2014) Genomic Characterization of Large Heterochromatic Gaps in the Human Genome Assembly. PLOS Computational Biology 10(5): e1003628. https://doi.org/10.1371/journal.pcbi.1003628

Cacheux, L., Ponger, L., Gerbault-Seureau, M., Richard, F.A., and Escudé, C. (2016). Diversity and distribution of alpha satellite DNA in the genome of an Old World monkey: Cercopithecus solatus. BMC Genomics *17*.

Cacheux L., Ponger L., Gerbault-Seureau M., Loll F., Gey D., Anne Richard F., Escudé C., The Targeted Sequencing of Alpha Satellite DNA in Cercopithecus pogonias Provides New Insight Into the Diversity and Dynamics of Centromeric Repeats in Old World Monkeys, Genome Biology and Evolution, Volume 10, Issue 7, July 2018, Pages 1837–1851, https://doi.org/10.1093/gbe/evy109

Catacchio CR, Ragone R, Chiatante G, Ventura M. 2015. Organization and evolution of Gorilla centromeric DNA from old strategies to new approaches. Sci Rep. 5(1):14189.

Cellamare A, et al. 2009. New insights into centromere organization and evolution from the white cheeked Gibbon and marmoset. Mol Biol Evol. 26(8):1889–1900.

Chiatante G, Giannuzzi G, Calabrese FM, Eichler EE, Ventura M. 2017. Centromere destiny in dicentric chromosomes: new insights from the evolution of human chromosome 2 ancestral centromeric region. Mol Biol Evol. 34(7):1669–1681.

Claus O. Wilke (2020). cowplot: Streamlined Plot Theme and Plot Annotations for 'ggplot2'. R package

Dutrillaux B, Couturier J, Chauvier G. 1980. Chromosomal evolution of 19 species of sub-species of Cercopithecinae. Ann Genet. 23(3):133–143.

Flemming, W. (1882). Zellsubstanz, kern und zelltheilung.. (Leipzig, F. C. W. Vogel).
Giannuzzi, G., Catacchio, C., and Ventura, M. (2012). Centromere Evolution: Digging into Mammalian Primary Constriction. p.

Gopalakrishnan, S., Van Emburgh, B.O., Shan, J., Su, Z., Fields, C.R., Vieweg, J., Hamazaki, T., Schwartz, P.H., Terada, N., and Robertson, K.D. (2009). A novel DNMT3B splice variant expressed in tumor and pluripotent cells modulates genomic DNA methylation patterns and displays altered DNA binding. Mol Cancer Res 7, 1622–1634.

Guschanski K, et al. 2013. Next-generation museomics disentangles one of the largest primate radiations. Syst Biol. 62(4):539–554.

H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.

Malik, H.S., and Henikoff, S. (2002). Conflict begets complexity: the evolution of centromeres. Current Opinion in Genetics & Development 12, 711–718.

M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in Proceedings of the 2nd International Conference on Knowledge Discovery and Data mining, 1996, pp. 226–231.

Moulin S, Gerbault-Seureau M, Dutrillaux B, Richard FA. 2008. Phylogenomics of African guenons. Chromosom Res. 16(5):783–799.

Plohl M, Luchetti A, Mestrović N, Mantovani B. Satellite DNAs between selfishness and functionality: structure, genomics and evolution of tandem repeats in centromeric (hetero)chromatin. Gene. 2008 Feb 15;409(1-2):72-82. doi: 10.1016/j.gene.2007.11.013. Epub 2007 Dec 4. PMID: 18182173.

Robert C. Edgar, MUSCLE: multiple sequence alignment with high accuracy and high throughput, *Nucleic Acids Research*, Volume 32, Issue 5, 1 March 2004, Pages 1792–1797, https://doi.org/10.1093/nar/gkh340

Schueler, M.G., and Sullivan, B.A. (2006). Structural and Functional Dynamics of Human Centromeric Chromatin. Annual Review of Genomics and Human Genetics 7, 301–313.

Schueler MG, et al. 2005. Progressive proximal expansion of the primate X chromosome centromere. Proc Natl Acad Sci U S A.102(30):10563–10568.

Shepelev VA, Alexandrov AA, Yurov YB, Alexandrov IA. 2009. The evolutionary origin of man can be traced in the layers of defunct ancestral alpha satellites flanking the active centromeres of human chromosomes. PLoS Genet. 5(9):e1000641.

Smit,AFA & Green,P RepeatMasker at http://www.repeatmasker.org

Steiner FA, Henikoff S. Diversity in the organization of centromeric chromatin. Curr Opin Genet Dev. 2015 Apr;31:28-35. doi: 10.1016/j.gde.2015.03.010. Epub 2015 May 16. PMID: 25956076.

Thomas Haschka, Loic Ponger, Christophe Escudé and Julien Mozziconacci. MNHN-Tree-Tools: A toolbox for tree inference using multi-scale clustering of a set of sequences **[preprint]** Bioinformatics.

Thompson, J. D., Higgins, D. G., & Gibson, T. J. (1994). CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic acids research*, *22*(22), 4673–4680. https://doi.org/10.1093/nar/22.22.4673

Tosi AJ. 2008. Forest monkeys and Pleistocene refugia: a phylogeographic window onto the disjunct distribution of the Chlorocebus lhoesti species group. Zool J Linn Soc. 154(2):408–418.

Tyler-Smith, Chris; Brown, William R. A. (1987). "Structure of the major block of alphoid satellite DNA on the human Y chromosome". Journal of Molecular Biology. 195 (3): 457–470. doi:10.1016/0022-2836(87)90175-6. PMID 2821279