Dziadkiewicz and Dojer *BMC Genomics* 2020, **21**(Suppl 2):274
https://doi.org/10.1186/s12864-020-6610-4

BMC Genomics

RESEARCH Open Access

Getting insight into the pan-genome structure with PangTree

Paulina Dziadkiewicz[1,2] and Norbert Dojer[1*]

From 17th RECOMB Satellite Conference on Comparative Genomics, RECOMB-CG 2019
Montpellier, France. 1–4 October 2019

Abstract

Background: The term *pan-genome* was proposed to denominate collections of genomic sequences jointly analyzed or used as a reference. The constant growth of genomic data intensifies development of data structures and algorithms to investigate pan-genomes efficiently.

Results: This work focuses on providing a tool for discovering and visualizing the relationships between the sequences constituting a pan-genome. A new structure to represent such relationships – called *affinity tree* – is proposed. Each node of this tree has assigned a subset of genomes, as well as their homogeneity level and averaged *consensus sequence*. Moreover, subsets assigned to sibling nodes form a partition of the genomes assigned to their parent.

Conclusions: Functionality of affinity tree is demonstrated on simulated data and on the *Ebola virus* pan-genome. Furthermore, two software packages are provided: *PangTreeBuild* constructs affinity tree, while *PangTreeVis* presents its result.

Keywords: Pan-genome, Multiple genome alignment, Affinity tree

Background

Pan-genomes

The amount of genomic data is enormous and its growth rate is increasing constantly. As an example, bacteria genomes availability has risen over 10,000-fold over last 25 years [1]. One of the reasons for this phenomenon is that DNA sequencing technologies are more and more cost and time efficient. Since sequencing cost is decreasing faster than storing and processing costs, optimization of these operational expenditures has become a substantial issue [2]. On the other hand, access to numerous sequenced genomes enhances the potential of comparative genomics.

*Correspondence: dojer@mimuw.edu.pl
[1]Faculty of Mathematics, Informatics and Mechanics, University of Warsaw, Banacha 2, 02-097 Warsaw, Poland
Full list of author information is available at the end of the article

Consequently, the idea of a *pan-genome* has emerged. Initially proposed as a single data structure for joint analysis of a group of bacterial genes [3], in the presence of a variety of whole genome sequences available this term has evolved. Currently, it refers to a model of joint analysis of genomes of different organisms. Related data structure is expected to support various operations, including construction, comparison, visualization, annotation, read mapping etc. [4].

Simultaneous access to many genomes makes the data structure information-rich. Pan-genome can serve as a demonstration of single organism's genome compared to its relatives, but also as a complex scheme exposing the variety of the component sequences. The first perspective is particularly important for applications in personalized medicine, because it makes possible to highlight one's individual properties which can be crucial in the mean of its phenotype. The other one can be useful when a reference genome for a population is needed.

Modeling pan-genomes

Several pan-genome models were proposed, ranging from collections of unaligned sequences to sophisticated (e.g. HMM-based) models that require complex preprocessing of sequence data [4]. Compromise solutions include multiple sequence alignment (MSA) and various sequence graphs.

The classical MSA model has a form of a matrix with rows obtained from input sequences by inserting gaps in appropriate positions. This model has several advantages, e.g. matrix columns naturally define a joint coordinate system for all input sequences. Additionally, most common sequence evolutionary events – insertions, deletions and substitutions – are well highlighted. However, such representation cannot handle rearrangements that violate colinearity of input sequences, e.g. duplications or inversions. In order to properly represent such rearrangements, the whole alignment is usually split into *blocks*, which represent aligned fragments of input sequences as classical colinear MSA. A small example of such *block alignment* scheme is presented in Fig. 1.

Sequence graphs naturally represent rearrangements of any kind. Moreover, in these models identical fragments of different input sequences may be combined into one, reducing the MSA redundancy in sequence representation. Such reduction is particularly important when a pan-genome is used for read mapping. However, it comes with the cost of loosing information regarding the input sequences, so additional graph annotation is required to provide a common coordinate system.

Graph-based MSA models were also proposed. One of the first such solutions was probably *Partial Order Alignment* (POA) [5]. In this approach graph nodes represent residues of input sequences and directed edges represent consecutive residues. Moreover, undirected edges join aligned residues with different labels. As illustrated in Fig. 2, this graph can be constructed from an alignment matrix straightforwardly: in each column identical letters are collapsed into one node, undirected edges link nodes obtained from the same column and directed edges join nodes representing consecutive residues from at least one of the sequences.

Several graph-based alignment models were applied to the problem of whole genome alignment, e.g. A-Bruijn graph [6], Enredo graph [7], Cactus graph [8] (see [9] for review). There were also proposed alignment-free sequence graph models (e.g. colored de Bruijn graphs [10]), having meaningful advantages in terms of scalability and performance.

Pan-genome structures

All multiple sequence alignment representations are complex, even classical colinear one. Some MSA applications require previous extraction of selected properties that could be represented in a more compact way. For example, *sequence profile* is a vector of symbol frequency distributions in particular MSA columns, *consensus sequence* is composed of symbols dominating consecutive columns, while *information content vector* quantifies the levels of that domination.

The above concepts could be straightforwardly extended to most graph-based alignment models, but we anticipate possible results unsatisfactory for at least two reasons. Firstly, the result would be still a sequence graph with the same or similarly complex structure, and this structure is what makes the graph alignment hard to use. Secondly, in all the above objects MSA columns are treated independently, while graphs encode some dependency between residues (e.g. two nodes or paths can represent alternative variants of the same locus). We find this property an important advantage of graph-based

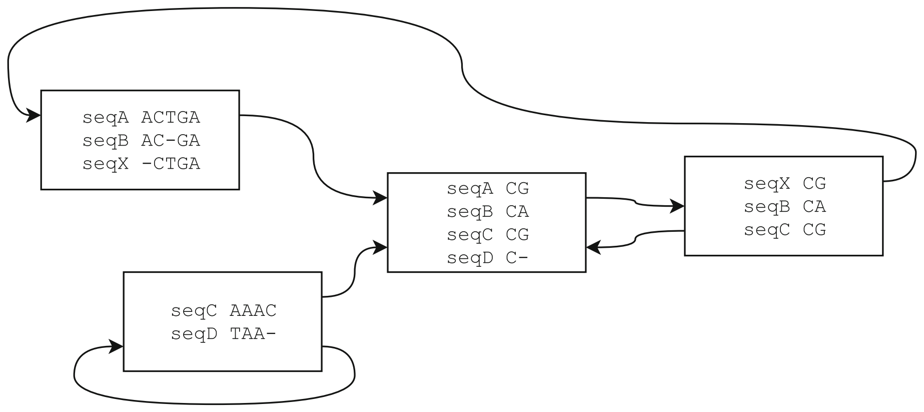


Fig. 1 Block alignment example

```
seq1   A G T C G T C G C C G C G T A C G C T - - - -
seq2   C G T - A G T G C C G G C T A C G T T - - - -
seq3   - - - - T T T A C A C C G T A C T C G T A C C
```

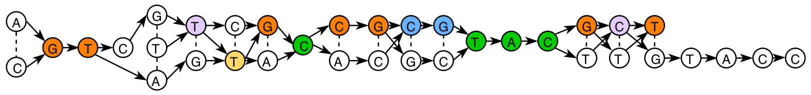


Fig. 2 PO-MSA construction from multialignment. For each alignment column, identical nucleotides are merged into one node. Undirected edges connect aligned nodes, while the directed ones reflect consecutive nucleotides

alignment models and expect it to be reflected in the result.

One of the possible solutions was proposed in [11]: instead of a single sequence profile or consensus there should be computed several consensus sequences, which in common would represent the whole (potentially heterogeneous) set of aligned sequences. The paper presents an algorithm identifying homogeneous sequence subgroups and computing for each subgroup a consensus sequence. Input alignment for the algorithm is represented as a POA and each consensus sequence forms a path in a POA graph.

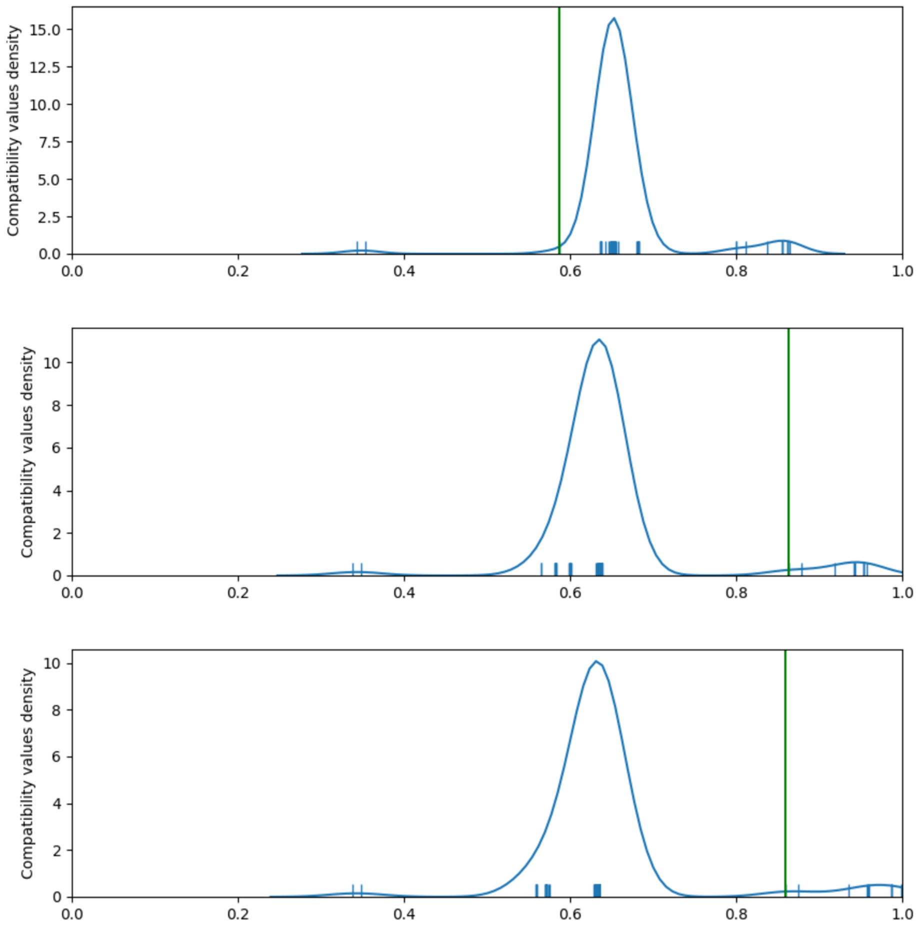


Fig. 3 Distributions of the value of sequence compatibility with consensuses C_1 (top), C_2 (middle) and C_3 (bottom), computed in consecutive iterations. Vertical green lines indicate resulting compatibility thresholds, i.e. right ends of the largest gaps between individual compatibility values (indicated by short vertical blue lines). C_1 is calculated from all input sequences, while C_2 and C_3 – only from sequences having in the previous iteration compatibility above the threshold. The set of sequences exceeding the compatibility threshold has changed in the second iteration, since a more homogeneous subgroup was recognized with C_2. The third iteration doesn't change the respective set and the procedure converges. However, consensus C_3 represents selected sequences better than C_2, which is reflected by their larger average compatibility

Our goals

The concept of POA consensus sequences has found several applications. Examples include paralog separation in EST alignments [11], gene isoform identification in de-novo assembly of RNA-seq data [12] and assembly of third generation sequencing reads [13]. However, to our best knowledge, it has never been applied to pan-genomic research.

The aim of this article is to fill this gap. For this purpose we extend the above concept, based on the following principles:

- pan-genome may be split into different genome subsets, depending on the assumed subset homogeneity level,
- subsets of a wide range of levels can contribute to the description of the relationships between genomes; the same applies to their consensus sequences,
- the image of the pan-genome structure should be complemented by subset hierarchy.

In the following sections we introduce the notion of *affinity tree* that meets the above requirements. Furthermore, we propose an algorithm computing affinity tree for given MSA, implemented in the package *PangTreeBuild*. Functionality of our approach is evaluated on simulated data and on the MSA of *Ebola virus* genomes. Finally, we present a tool for affinity tree visualization, called *PangTreeVis*.

Methods

POA and POA consensus sequences

Partial Order Alignment (POA) [5] was probably one of the first graph-based MSA models. In this approach graph nodes represent residues of input sequences (aligned residues are combined into one if they are labeled alike) and directed edges join residues that are neighboring in at least one of the input sequences. Moreover, nodes representing aligned residues with different labels are connected with undirected edges.

Lee [11] presented a tool for calculating POA consensus sequences. In this method each edge is assigned a weight corresponding to the number of covering input sequences. Then *HeaviestBundle* algorithm searches the heaviest path in the graph using dynamic programming. Next, the algorithm calculates the *compatibility* of each sequence with the consensus, defined as

$$C_{ij} = \frac{N_{ij}}{N_i}$$

where N_i is the length of seuquence i and N_{ij} is the number of POA nodes belonging to both the sequence i and the consensus j. The sequence is *assigned* to the consensus, if the result is larger than the assumed threshold. Finally, edge weights are recalculated and the whole procedure is repeated for remaining sequences until all sequences are assigned or none of them exceeds a compatibility threshold. This forms a partition of the input sequences into subsets with assigned consensus paths.

Affinity tree

As opposite to the flat division produced by the algorithm of Lee [11], *affinity tree* allows to represent the hierarchy of sequence subsets. Each node of this tree has the following attributes assigned:

- a subset of input sequences,
- a consensus sequence being their common representation,
- minimum of the sequence-consensus compatibilities (*minComp*).

The value of the *minComp* attribute reflects node's homogeneity level. The root node has all input sequences

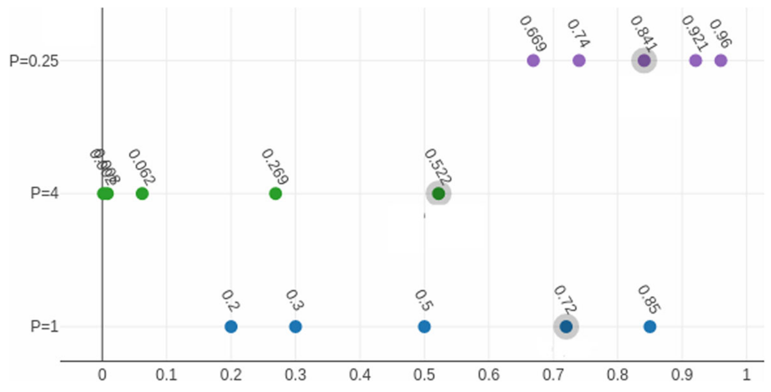


Fig. 4 Compatibility threshold selection using three different P values. For $P = 0.25$ the biggest gap is between values 0.74 and 0.841 which originate from 0.3 and 0.5. For $P = 4$ the treshold is in 0.522 which comes from value 0.85. The largest gap among original values of compatibilities ($P = 1$) is between 0.5 and 0.72 so 0.72 is the boundary chosen as treshold in this case

assigned and each non-leaf node has at least two children nodes that form a partition of the sequences assigned to their parent into more homogeneous subsets. Subsets with homogeneity level bigger than the assumed parameter `stop` form leafs.

Affinity tree building algorithm

The input of the algorithm consists of a set of aligned sequences, represented with the POA model. Affinity tree is created in BFS ordering: starting from the root, sequences assigned to consecutive nodes are split into smaller, more homogeneous subsets until the `stop` homogeneity level is achieved. The pseudocode of the algorithm is shown in Algorithm 1.

The procedure of splitting sequences into subsets is similar to that of Lee [11], but we introduced some important modifications. The first one regards the choice of subset consensus sequences. In Lee's algorithm they are computed for the whole current set of sequences and then sequence subsets are selected based on the compatibility with the consensus. The artifact of this approach is that fragments of the consensus sequence may be representative to sequences not assigned to it rather than assigned ones. In the case of longer (e.g. genomic) sequences this behavior occurs more commonly than in the case of short ones which Lee's algorithm was designed for. In order to avoid this artifact we recalculate consensuses based on selected sequences only and repeat this procedure until convergence. We observed that very seldom more than two iterations are needed, typical case is presented on Fig. 3.

Algorithm 1 Affinity Tree generation algorithm

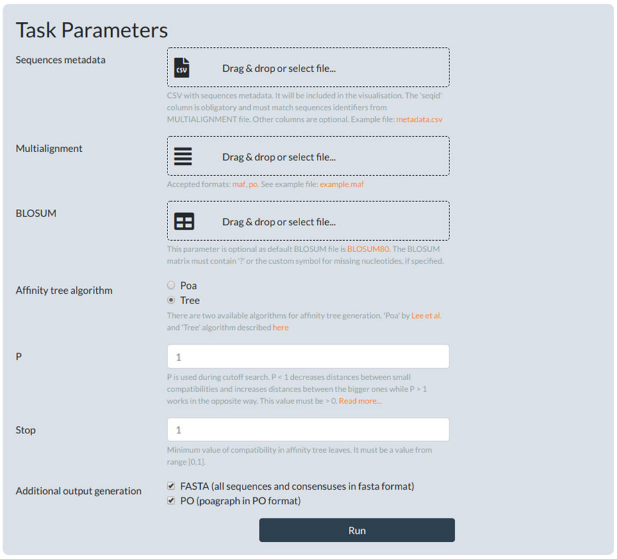
```
procedure AFFINITYTREE(sequences, stop)
    consensus = HeaviestBundle(sequences)
    comps = CalcCompatibilities(sequences, consensus)
    minComp = min(comps)
    root = MakeNode(sequences, consensus, minComp)
    tree = MakeTree(root)
    leftNodes = EmptyQueue()
    if minComp $<$ stop then
        leftNodes.Append(root)
    end if
    while leftNodes $\neq \emptyset$ do
        node = leftNodes.Pop()
        leftSeqs = node.Sequences
        minComp = node.MinCompatibility
        siblingComps = $\emptyset$
        while leftSeqs $\neq \emptyset$ do
            child = GetChild(leftSeqs, minComp, siblingComps)
            tree.AddChildToNode(child, node)
            leftSeqs -= child.Sequences
            siblingComps.Add(child.MinCompatibility)
            if child.MinCompatibility $<$ stop then
                leftNodes.Append(child)
            end if
        end while
    end while
    return tree
end procedure
```

Another problem with a direct application of Lee's algorithm is that the nodes of the tree have various homogeneity levels, which are not known a priori. In order to avoid using an arbitrarily chosen threshold of Lee's approach, we invented a data-driven compatibility threshold selection method. The idea is that the compatibilities between sequences and a consensus should form clusters corresponding to homogeneous subsets. In particular, good candidates for consensus-assigned sequences should be separated from poor ones by a large difference in compatibility. Therefore, as the cutoff threshold we choose the right boundary of largest gap in sorted compatibility values.

This rule works well in the case of the first child of a node. However, in the other case this procedure would force splitting remaining sequences into smaller and smaller subsets with inadequately high homogeneity levels. Therefore, after creating the first child of a node, for the purpose of threshold selection we introduce two modifications to the list of sequence compatibilities:

- parent *minComp* value is added,
- all but the smallest compatibilities larger than all *minComp* values of already created siblings are removed.

Since the parent *minComp* value is smaller than all considered compatibilities, the first modification allows (and defines criteria for) creating a node covering all remaining sequences. The second modification restricts searching to compatibility gaps overlapping the interval covering the *minComp* values of a parent and already created siblings (however, the right end of the gap still can be larger).


Task Parameters

Sequences metadata: Drag & drop or select file...
CSV with sequences metadata. It will be included in the visualisation. The 'seqid' column is obligatory and must match sequences identifiers from MULTIALIGNMENT file. Other columns are optional. Example file: metadata.csv

Multialignment: Drag & drop or select file...
Accepted formats: maf, po. See example file: example.maf

BLOSUM: Drag & drop or select file...
This parameter is optional as default BLOSUM file is BLOSUM80. The BLOSUM matrix must contain '?' or the custom symbol for missing nucleotides, if specified.

Affinity tree algorithm: Poa [✓ Tree]
There are two available algorithms for affinity tree generation. 'Poa' by Lee et al. and 'Tree' algorithm described here

P: 1
P is used during cutoff search. P < 1 decreases distances between small compatibilities and increases distances between the bigger ones while P > 1 works in the opposite way. This value must be > 0. Read more...

Stop: 1
Minimum value of compatibility in affinity tree leaves. It must be a value from range [0,1].

Additional output generation: [✓ FASTA (all sequences and consensuses in fasta format)] [✓ PO (poagraph in PO format)]

Fig. 5 Using PangTreeBuild via PangTreeVis interface

Although the restriction could potentially cause choosing an accidental threshold, the interplay between data-driven threshold selection and iterative consensus recalculation (see Algorithm 2 for details) minimizes this effect.

Algorithm 2 Get child algorithm

```
procedure GetChild(sequences, minComp, siblingComps)
    sequences = leftSequences
    consensus = HeaviestBundle(sequences)
    consensusNotSet = True
    while consensusNotSet do
        comps = CalcCompatibilities(sequences, consensus)
        if siblingComps $\neq \emptyset$ then
            comps.Add(minComp)
            comps.RemoveLargerThan(max(siblingComps))
        end if
        threshold = FindLargestInterval(comps)
        qualSequences = sequences[comps > threshold]
        qualConsensus = HeaviestBundle(qualSequences)
        if qualConsensus == consensus then
            consensusNotSet = False
        end if
        consensus = qualConsensus
    end while
    return MakeNode(qualSequences, consensus, threshold)
end procedure
```

Finally, we introduced an additional parameter P that controls the granularity level of the resulting tree. For the purpose of calculation of the largest gap compatibilities and *minComp* values are transformed according to the formula

$$t_p(c) = c^p$$

For $P > 1$ the distance between smaller compatibilities decreases and the distance between higher compatibilities increases. Consequently, the sequence set assigned to a particular node is split into smaller and more homogeneous subsets and hence more children are created for the node. The opposite happens for $P < 1$. The effect of this parameter is presented in Fig. 4.

Implementation

The algorithm was implemented in software called PangTreeBuild, which is available as a Python 3.6 package or a console application for Linux operating system. Moreover, it could be executed from the PangTreeVis web browser interface (see Fig. 5).

As an input PangTreeBuild requires a block alignment in MAF format, which is widely used to represent whole genome alignments and can be interpreted as a pan-genome model. PangTreeBuild internal data model is POA so alignment blocks are converted to it as shown in Fig. 2. Subsequently, edges between blocks are filled in and the connections that create cycles are identified and removed using the *Mafgraph* tool [14]. Next, unaligned fragments of input sequences, which are usually absent in MAF files, are complemented from NCBI database or provided FASTA files.

Given such a pan-genome model and parameters P and `stop`, AffinityTree procedure is run. The constructed data model together with the resulting affinity tree are saved as a JSON file, which can be visualized using PangTreeVis tool.

PangTreeVis is an interactive browser application. The visualization has two parts (see Fig. 6). The first one – pan-genome graph – consists of two views: general (for navigating purposes) and detailed (single residue resolution). The second part is affinity tree visualization and contains tree diagram featured by its detailed characteristics and metadata that could be provided in advance for PangTreeBuild.

Results

The PangTreeBuild algorithm was applied to two types of MSA datasets: simulated (yielded from genome sequence evolution simulations) and real-life (computed for *Ebolavirus* genomes).

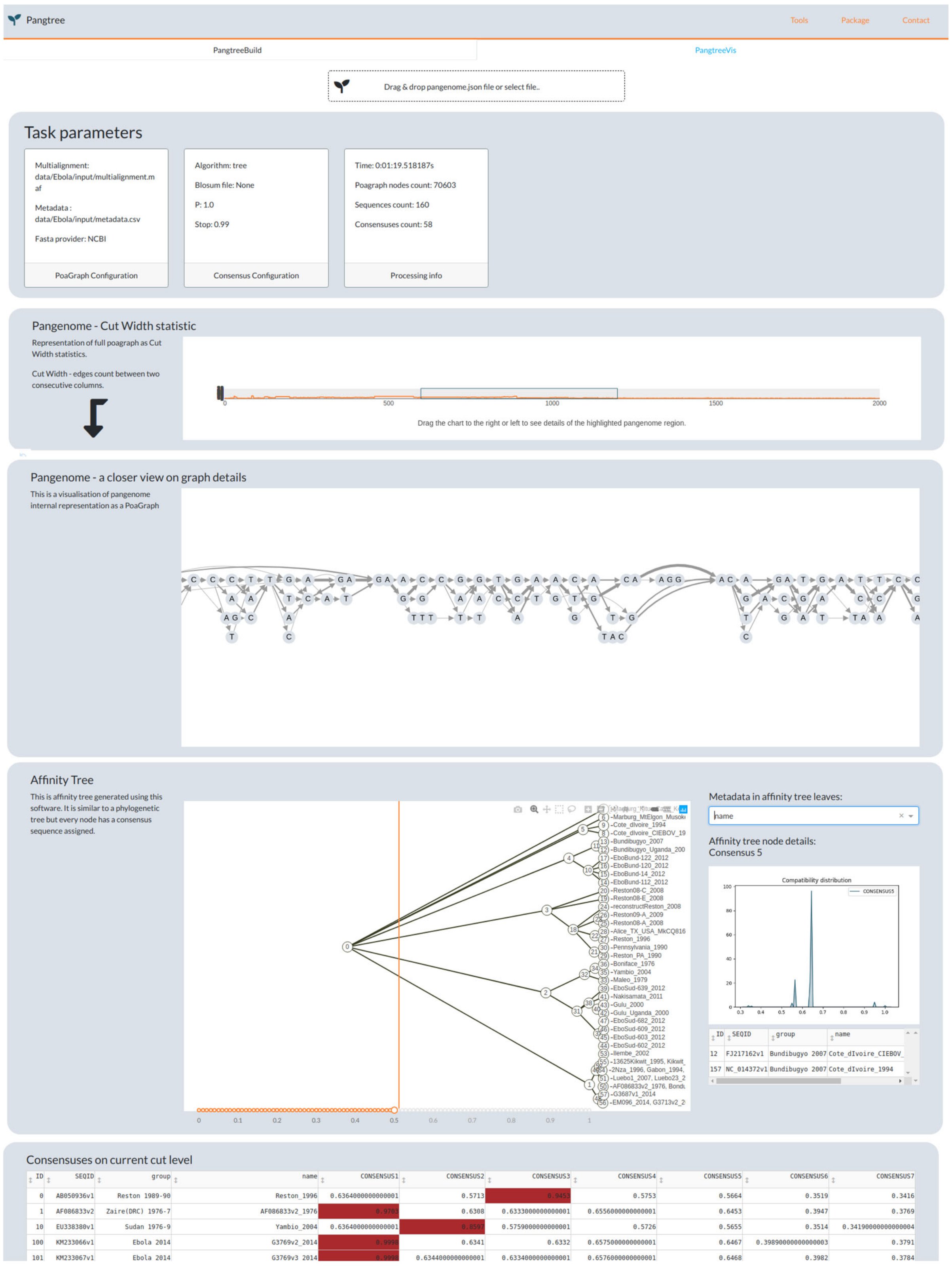
Simulated data

To test whether affinity tree correctly discovers evolutionary patterns, we generated MSA following sequence evolution simulation. We applied the scheme used in the data simulated for the purpose of Alignathon assessment [15]: MSA was generated using genome simulation software *Evolver* [16] together with *evolverSimControl* [17] – a wrapper for running Evolver's simulations guided by phylogenetic trees.

The root sequence consisted of two 100*kbp*-long fragments of human chromosomes 20 and 21. Two evolutionary parameter sets were prepared, both based on the example provided in the *evolverSimControl* repository. In the first parameter set, possible evolutionary events were restricted to substitutions, deletions and non-duplicative insertions, resulting in colinear MSAs (i.e. without rearrangements). In the second one, genome rearrangement events were added: intra-chromosomal inversions, tandem repeats, transpositions and duplications, as well as inter-chromosomal transpositions and duplications.

The simulations were guided by a phylogenetic tree of 138 yeast strains (reconstructed in [18] and deposited in TreeBASE repository under accession no. S12670). From each simulation, the MSA reflecting simulated evolutionary events was created (see [17] for details) and restricted to chromosome 20 sequences of leaf genomes. In this way 10 sequence sets and their MSAs were generated for each simulation parameter set. All input datasets, parameters and simulation scripts are provided as example data in the PangTreeBuild repository.

For each MSA affinity trees were generated by PangTreeBuild with parameters `stop`= 0.999 and $P \in \{0.25, 1, 4\}$. Similarly, Lee's algorithm was applied to each



Task parameters

PoaGraph Configuration

Multialignment: data/Ebola/input/multialignment.maf

Metadata : data/Ebola/input/metadata.csv

Fasta provider: NCBI

Consensus Configuration

Algorithm: tree

Blosum file: None

P: 1.0

Stop: 0.99

Processing info

Time: 0:01:19.518187s

Poagraph nodes count: 70603

Sequences count: 160

Consensuses count: 58

Pangenome - Cut Width statistic

Representation of full poagraph as Cut Width statistics.

Cut Width - edges count between two consecutive columns.

Drag the chart to the right or left to see details of the highlighted pangenome region.

Pangenome - a closer view on graph details

This is a visualisation of pangenome internal representation as a PoaGraph

Affinity Tree

This is affinity tree generated using this software. It is similar to a phylogenetic tree but every node has a consensus sequence assigned.

Affinity tree node details:
Consensus 5

ID	SEQID	group	name
12	FJ217162v1	Bundibugyo 2007	Cote_dIvoire_CIEBOV_
157	NC_014372v1	Bundibugyo 2007	Cote_dIvoire_1994

Consensuses on current cut level

ID	SEQID	group	name	CONSENSUS1	CONSENSUS2	CONSENSUS3	CONSENSUS4	CONSENSUS5	CONSENSUS6	CONSENSUS7
0	AB050936v1	Reston 1989-90	Reston_1996	0.6364000000000001	0.5713	0.9453	0.5753	0.5664	0.3519	0.3416
1	AF086833v2	Zaire(DRC) 1976-7	AF086833v2_1976	0.9703	0.6308	0.6333000000000001	0.6556000000000001	0.6453	0.3947	0.3769
10	EU338380v1	Sudan 1976-9	Yambio_2004	0.6364000000000001	0.8597	0.5759000000000001	0.5726	0.5655	0.3514	0.34190000000000004
100	KM233066v1	Ebola 2014	G3769v2_2014	0.9998	0.6341	0.6332	0.6575000000000001	0.6467	0.39890000000000003	0.3791
101	KM233067v1	Ebola 2014	G3769v3 2014	0.9998	0.6344000000000001	0.6334000000000001	0.6576000000000001	0.6468	0.3982	0.3784

Fig. 6 Pan-genome visualization in PangTreeVis. Consecutive panels present: input data and parameters, general view on the pan-genome graph, detailed view on the fragment selected on the general view panel, affinity tree visualization and details of the genome set division yielded by the cut selected on the affinity tree panel

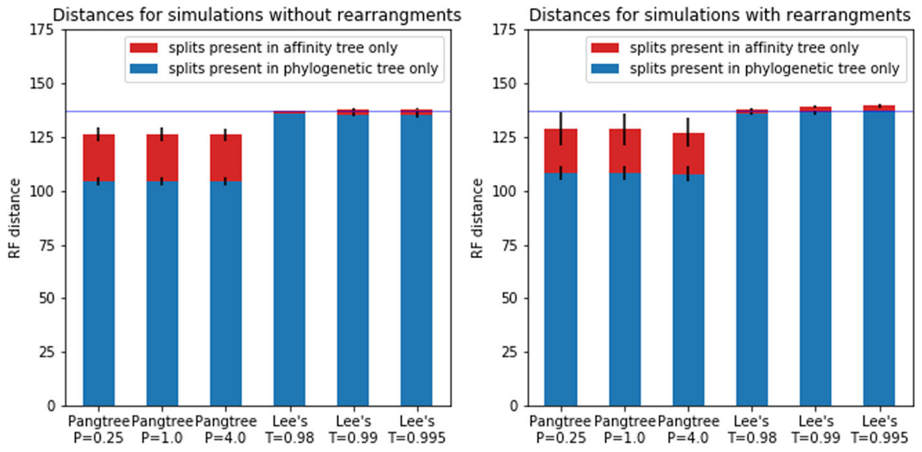


Fig. 7 Robinson-Foulds distances between phylogenetic tree and affinity trees. Each bar represents RF distance averaged over replicates of simulated data and algorithm parameters (as described under bar). The horizontal blue line represent the number of non-trivial splits in the phylogenetic tree (i.e. maximal possible height of the blue bars)

MSA with three compatibility thresholds: 0.98, 0.99 and 0.995. Resulting partitions were converted to affinity trees in the following way: each leaf was a child of the root and represented either a subset of Lee's partition or an original sequence not assigned to any subset. For the purpose of comparison with the original phylogenetic tree using the Robinson-Foulds (RF) metric, we added to each leaf children, representing original sequences assigned to a corresponding subset. Results are presented on Fig. 7.

RF distance is defined as the number of splits that are induced by one of the trees but not by the other. The number of splits induced by a single tree equals the number of its nodes. For a fixed number of leafs, the largest possible number of splits is obtained by bifurcating trees. In our case phylogenetic tree is bifurcating, while affinity trees are highly multifurcating, since they are intended to express only the most evident partitions. Consequently, the part of the RF distance that is due to splits induced by the phylogenetic tree only reflects the granularity of affinity trees rather than the inconsistency between trees. As is shown on Fig. 7, such splits contribute to $>$ 80% of the distances for PangTree affinity trees and $>$ 95% of the distances for Lee's affinity trees. However, in the last case this is very close to the total number of non-trivial splits in the phylogenetic tree (i.e. 137), which means that only very few of them are shared by Lee's affinity trees.

The above observation is confirmed by Fig. 8, which presents more detailed analysis of the reconstruction of splits induced by the phylogenetic tree. PangTree algorithm reconstructs splits induced by most of longer edges, especially in the case of datasets without rearrangements, while Lee's algorithm performs very poorly.

The red bars on Fig. 7 represent the splits that can be considered as false positives (i.e. supported by affinity trees, but absent in the original phylogenetic tree). Absolute numbers of such splits are visibly higher for the PangTree algorithm than for Lee's algorithm, which is due to the difference in the total number of splits yielded by both methods. The proportion of false positive splits (i.e. false discovery rate) for the PangTree algorithm is near the half of the one for the Lee's algorithm.

In order to evaluate the behavior of *minComp* as a subset homogeneity measure, we compared it against the corresponding measure in the phylogenetic tree – maximal distance from an internal node to its leafs. Results are presented on Fig. 9.

Ebola dataset

Ebola outbreak in 2014 intensified studies concerning this dangerous virus. As it is crucial to find cure and vaccine for the illness it causes, a large set of sequenced samples was prepared. The data and associated studies are collected in UCSC Ebola Portal [19]. The dataset include MSA of 158 genomes of *ebolavirus* and 2 genomes of *marburgvirus* (the closest Ebola relative). Every sequence from this alignment is assigned to one of the seven groups: Ebola 2014, Bundibugyo 2007, DRC 2007, Reston 1989-90, Sudan 1976, Zaire(DRC) 1967-7 and Marburg 1987. This division slightly differs from official taxonomy [20], the comparison is presented in Table 1.

The length of a single genome of Ebola or Marburg virus is $\sim 19kbp$. Due to high sequence similarity POA for all 160 genomes contains only 70603 nodes. Affinity tree computed with the parameters $P = 0.25$, $\texttt{stop}= 0.99$ is presented in Fig. 10.

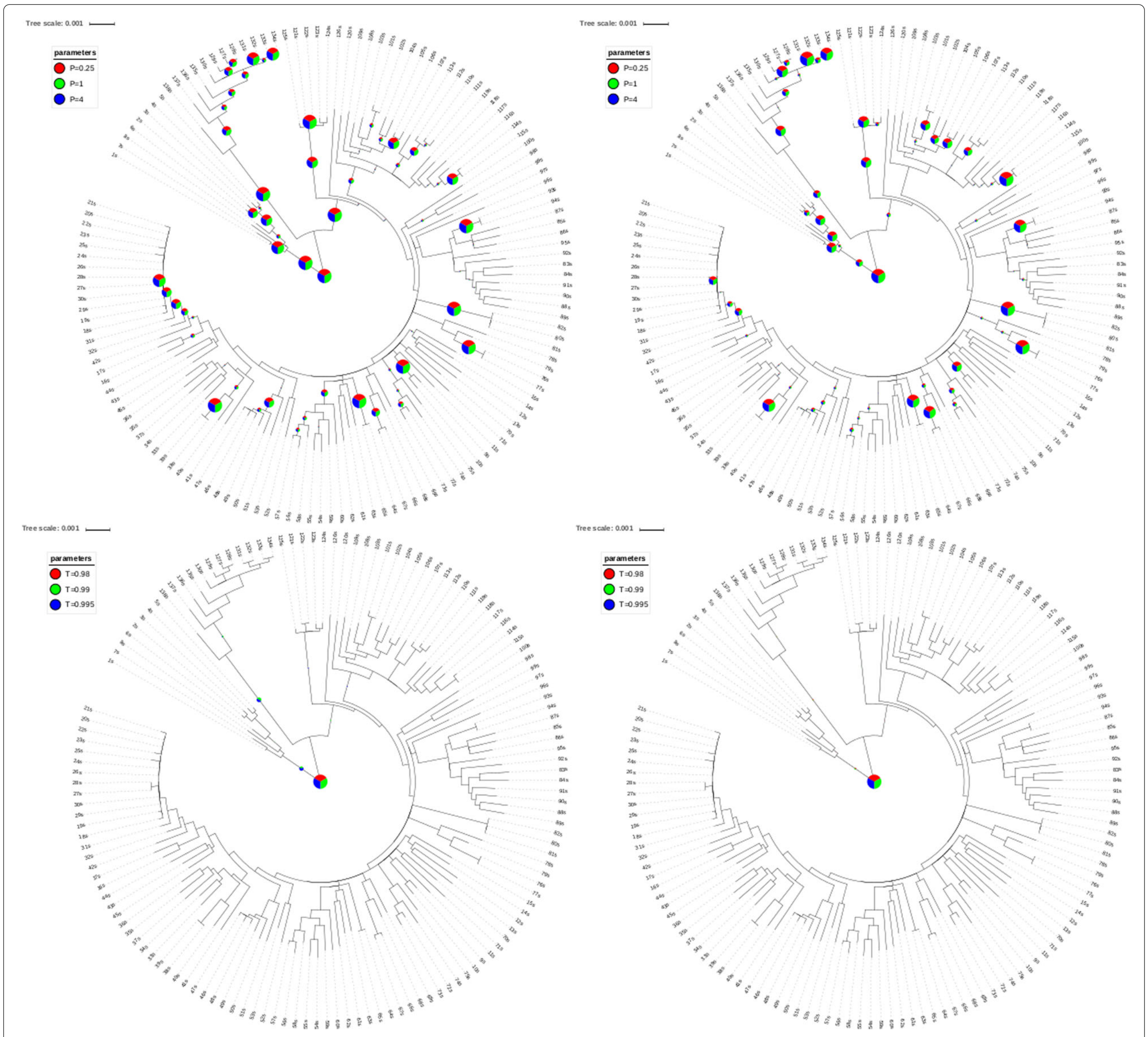


Fig. 8 Phylogenetic tree and the affinity trees' support of its nodes. The size of circles on the branches represent the number of affinity trees inducing identical leaf split (in particular, root splits are supported by all trees, so they have maximal possible size). Pie-charts inside the circles represent proportions of contributing affinity trees that were calculated with respective parameters. Top row: PangTree algorithm, bottom row: Lee's algorithm. Left column: simulations without rearrangements, right column: simulations with rearrangements

As we can see, the only non-leaf child of the root separates all *ebolavirus* genomes from *marburgvirus* genomes. *Ebolavirus* genomes are later split into 5 subsets, corresponding to 5 *ebolavirus* species represented in the dataset: Tai Forest, Bundibugyo, Reston, Sudan and Zaire. The subsets differ in homogeneity: *minComp* ranges from $\sim$ 0.96 for *Sudan* and *Reston ebolavirus* to $\sim$ 0.99 for *Zaire ebolavirus*.

In order to look into details of the relationships between species, we calculated compatibilities between their consensus sequences. Results are presented in Table 2. All the compatibilities are around 0.6, but slightly higher are those between *Zaire ebolavirus* and other species, as well as those between *Tai Forest* and *Bundibugyo ebolaviruses*.

The compatibility measure describes similarity averaged over the whole sequence. In order to analyze its variation over the genome sequence, we calculated *local compatibilities* between consensuses of the *ebolavirus* species. Results presented on the Fig. 11 show that local compatibilities for all species pairs share similar patterns. Firstly, compatibility significantly decreases in non-coding areas. Secondly, there are few regions of lower similarity

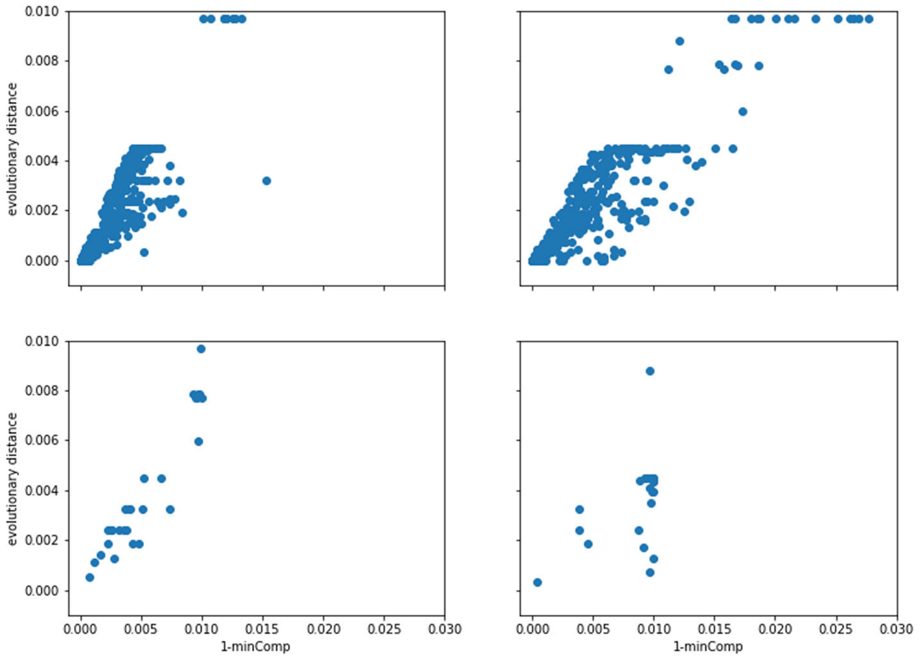


Fig. 9 Homogeneity of sequence groups measured in affinity trees and in the phylogenetic tree. Each inner node of an affinity tree is represented by a single dot. Homogeneity of the corresponding sequence group is measured in the affinity tree by 1 — *minComp* (X-axis) and in the phylogenetic tree by the maximum distance from leafs representing these sequences to their lowest common ancestor (Y-axis). Top row: PangTree algorithm, bottom row: Lee's algorithm. Left column: simulations without rearrangements, right column: simulations with rearrangements

in coding areas. One of them, near the 3' end of gene L (around position 16.5*kbp*), is known to exhibit some interspecies variation [21].

Performance

Computational requirements of affinity tree calculations for datasets used in this study are compared in Table 3. All the computations were performed on a laptop with a 1.8GHz Intel Core i7-8565U CPU and 16GB RAM. The requirements seem to scale linearly with respect to the size of input data, so PangTreeBuild probably may be applied to hundreds of virus genomes or dozens of bacterial ones. However, the impact of MSA complexity could be crucial and is hard to estimate.

Table 1 Ebola and Marburg viruses taxonomy compared to genome classification in UCSC Ebola Portal

ebola-/marburgvirus species	Matching groups in Ebola Portal
	Ebola 2014
Zaire	DRC 2007
	Zaire(DRC) 1967-7
Bundibugyo	Bundibugyo 2007
Tai Forest	
Reston	Reston 1989-90
Sudan	Sudan 1976
Bombali	-
Marburg	Marburg 1987

Discussion

The relationships between homologous sequences are usually represented using two complementary structures – phylogenetic tree and multiple sequence alignment. In the current work we proposed a novel structure, called affinity tree, which joins both perspectives. As opposite to phylogenetic trees, affinity trees are not intended to be a detailed reconstruction of evolutionary history. Instead, they provide:

- hierarchy of most evident homogeneous sequence subgroups,
- subgroup reference (or consensus) sequences,
- graph-based multiple sequence alignment, joining input and reference sequences,
- local and global sequence homogeneity measures.

Consequently, affinity tree can serve as a pan-genome model, supporting interesting features for comparative

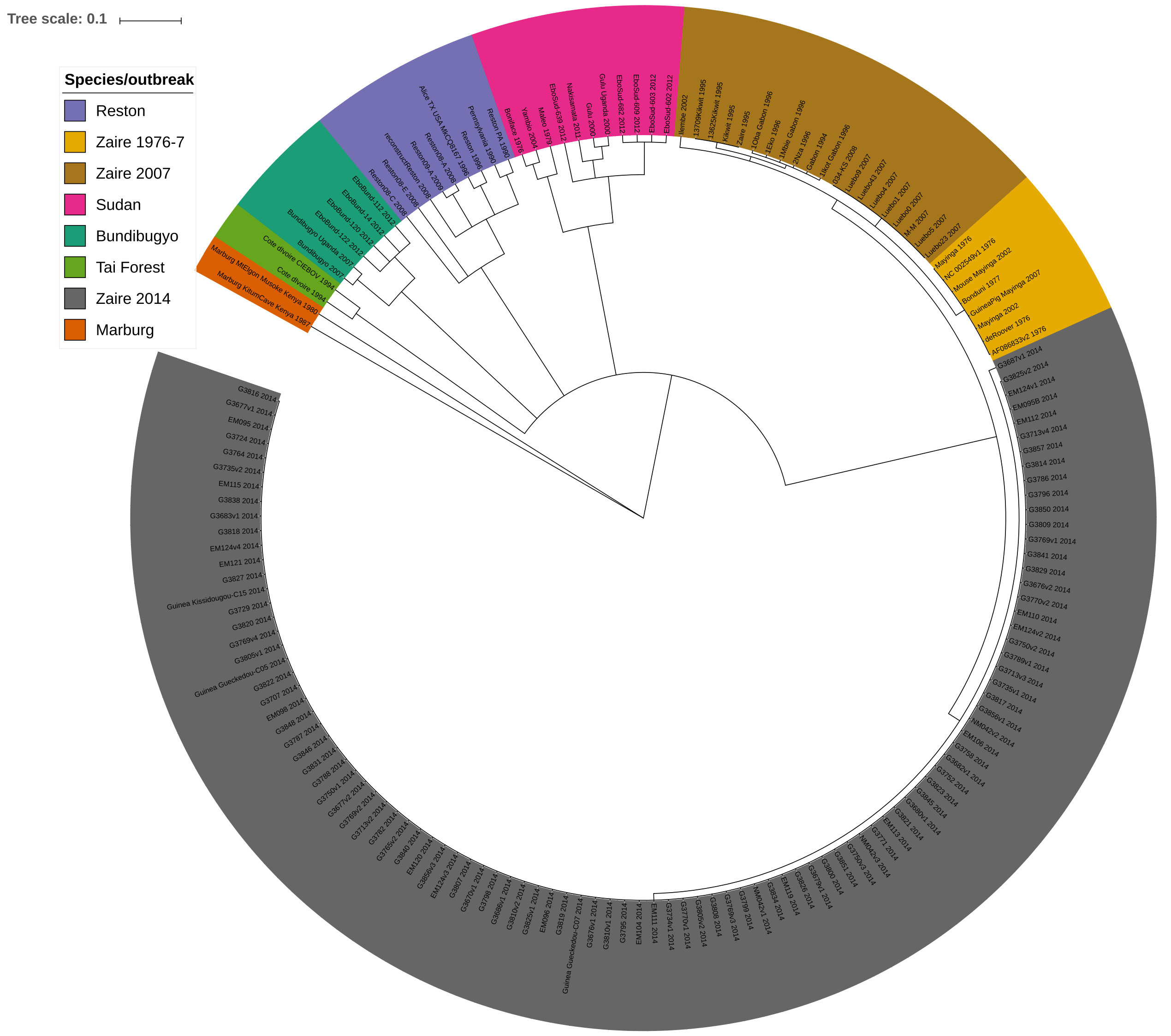


Fig. 10 Affinity tree for Ebola dataset

Table 2 Compatibilities between consensus sequences for Ebola species

species	*Zaire*	*Sudan*	*Reston*	*Bundibugyo*	*Tai Forest*
Zaire	1.000	0.635	0.634	0.658	0.647
Sudan	0.639	1.000	0.575	0.573	0.563
Reston	0.639	0.576	1.000	0.580	0.570
Bundibugyo	0.659	0.570	0.576	1.000	0.637
Tai Forest	0.648	0.560	0.566	0.637	1.000

Each entry contains the compatibility of the consensus sequence for the species from the first column with the consensus sequence for the species from the header (note the asymmetry in the compatibility definition)

genomics studies. Given a whole-genome alignment, affinity tree specifies homogeneous subgroups of contributing genomes. Each subgroup is characterized in terms of its consensus sequence and subgroup genetic diversity. Graph-based alignment induced on consensus sequences represents spatial distribution of similarity between subgroups, while alignment between sequences constituting a subgroup and their consensus – of similarity within the subgroup. Furthermore, spatial distribution of compatibility of mixed individual's genome with consensus sequences of various subgroups can delineate the mosaic structure of that genome. In summary, graph-based alignment of consensus sequences can serve as a diversified population reference genome.

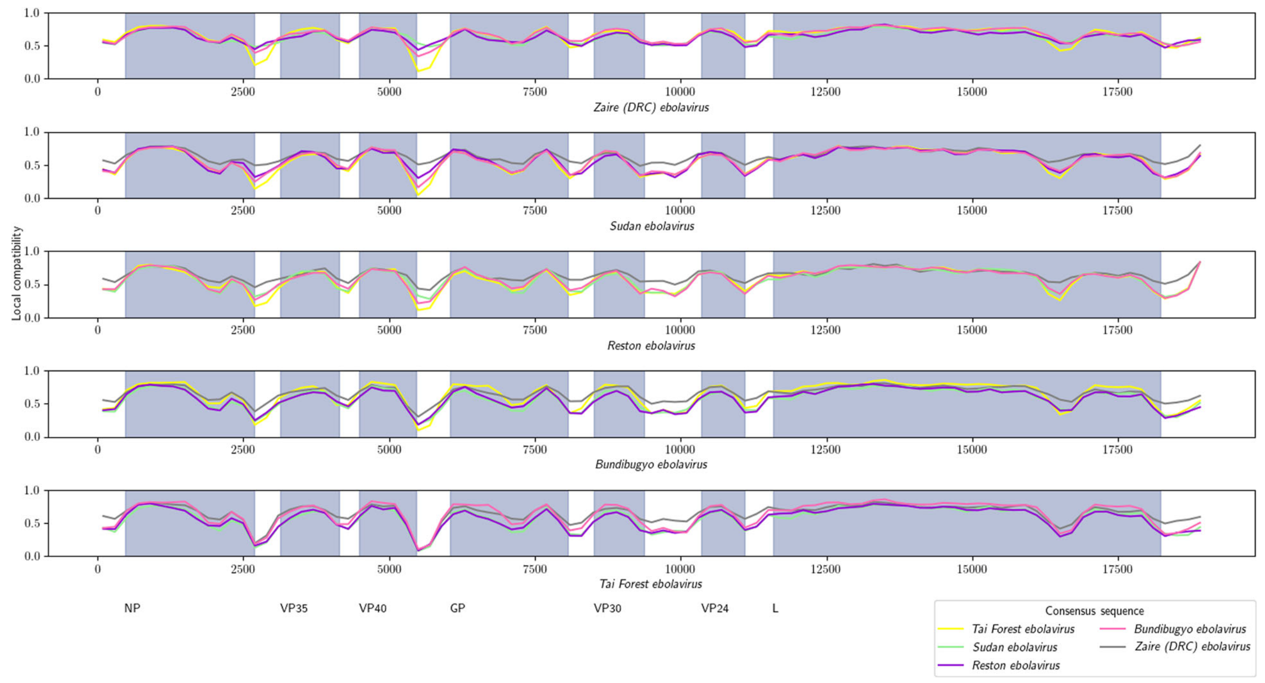


Fig. 11 Local compatibilities between consensus sequences of *ebolavirus* species. Each plot shows local compatibilities (i.e. calculated within 400*bp* windows) with the consensus sequence of the species from the caption. Compatibilities of the same sequences are drawn in the same color on all plots, as denoted in the legend. Dark background indicates coding sequences, respective genes are listed below

Several practical issues should be considered when using affinity trees in comparative genomics studies. First, joint visualization of the features supplied by the affinity tree model is challenging. PangTreeVis provides the visualization of the alignment graph (on both genome-scale and nucleotide level) and of the tree structure, enables interactive subgroup selection and viewing genome-subgroup relationships. Although the basic requirements are satisfied, the advantages of the affinity tree model could be highlighted better if some extensions were provided, e.g. graph visualization on intermediate scale levels, support for local genetic diversity layer and additional layers (annotation, experimental etc.).

Second, the quality of the affinity tree hardly depends on the provided whole-genome alignment. PangTreeBuild algorithm was designed to pan-genomes with limited number of structural variants. It performs well on datasets examined in the current study, but our experiments show that large-scale rearrangements influence both the algorithm efficiency and the quality of resulting affinity tree. Consequently, complex alignment graphs may require an adjusted building algorithm.

Conclusions

Affinity tree and its generation algorithm support the pan-genomic research field. Apart from hierarchical division of aligned genomes, affinity tree describes homogeneity of resulting subgroups and provides subgroup reference sequences. The introduced method gives new insight into multiple sequence alignment analysis – its result can serve as both taxonomic study and a population reference pan-genome. Two complementary software packages: PangTreeBuild and PangTreeVis enable affinity tree construction and visualization.

Table 3 Performance of the PangTreeBuild algorithm

Dataset	Number of genomes	Total sequence length	MSA file size	Computation time	Memory peak
Ebola pan-genome Simulated	160	3Mbp	4.9MB	2m44s	280MB
- without rearrangements	138	13.8Mbp	24MB	20m25s	1.18GB
- with rearrangements	138	13.8Mbp	27MB	24m46s	1.38GB

Abbreviations

BFS: Breadth first search; CPU: Central processing unit; DNA: DeoxyriboNucleic acid; EST: Expressed sequence tag; HMM: Hidden Markov model; JSON: JavaScript object notation; MAF: Multiple alignment format; MSA: Multiple sequence alignment; NCBI: National center for biotechnology information; POA: Partial order alignment; RAM: Random-access memory; RF: Robinson-Foulds; RNA: RiboNucleic acid; UCSC: University of California, Santa Cruz

Acknowledgements

We thank Anna Lisiecka for implementing the *Mafgraph* tool.

About this supplement

This article has been published as part of *BMC Genomics Volume 21 Supplement 2, 2020: Proceedings of the 17th Annual Research in Computational Molecular Biology (RECOMB) Comparative Genomics Satellite Workshop: genomics*. The full contents of the supplement are available online at https://bmcgenomics.biomedcentral.com/articles/supplements/volume-21-supplement-2.

Authors' contributions

ND designed the study and prepared simulation data. PD implemented the PangTree software and executed experiments. ND and PD analyzed results and wrote the paper. All authors read and approved the final manuscript.

Funding

This work was supported and publication was funded by the National Science Centre, Poland, under grant number 2016/21/B/ST6/01471.

Availability of data and materials

Both the dataset supporting the conclusions of this article and the PangTree 1.0 software are available in the GitHub repositories: https://github.com/pangtree/pangtreebuild (PangTreeBuild and datasets) and https://github.com/pangtree/pangtreegui (visualization tool). The software is licensed under the terms of the MIT license.

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Author details

[1]Faculty of Mathematics, Informatics and Mechanics, University of Warsaw, Banacha 2, 02-097 Warsaw, Poland. [2]Faculty of Mathematics and Information Science, Warsaw University of Technology, Koszykowa 75, 02-097 Warsaw, Poland.

Published: 16 April 2020

References

1. Insights from 20 years of bacterial genome sequencing. Funct Integr Genomics. 2015;15(2):141–61. https://doi.org/10.1007/s10142-015-0433-4.
2. Stein LD. The case for cloud computing in genome informatics. Genome Biol. 2010;11:207.
3. Genome analysis of multiple pathogenic isolates of streptococcus agalactiae: implications for the microbial "pan-genome". Proc Natl Acad Sci U S A. 2005;102(45):16530.
4. Computational pan-genomics: status, promises and challenges. Brief Bioinforma. 2018;19(1):118–35. Computational Pan-Genomics Consortium.
5. Christopher Lee MFS, Catherine Grasso. Multiple sequence alignment using partial order graphs. Bioinformatics. 2002;18(3):452–64.
6. Pevzner PA, Tang H, Tesler G. De novo repeat classification and fragment assembly. Genome Res. 2004;14(9):1786–96. https://doi.org/10.1101/gr.2395204. http://genome.cshlp.org/content/14/9/1786.full.pdf+html.
7. Paten B, Herrero J, Beal K, Fitzgerald S, Birney E. Enredo and pecan: Genome-wide mammalian consistency-based multiple alignment with paralogs. Genome Res. 2008;18(11):1814–28. https://doi.org/10.1101/gr.076554.108. http://genome.cshlp.org/content/18/11/1814.full.pdf+html.
8. Paten B, Diekhans M, Earl D, John JS, Ma J, Suh B, Haussler D. Cactus graphs for genome comparisons. J Comput Biol. 2011;18(3):469–81. https://doi.org/10.1089/cmb.2010.0252. PMID: 21385048. https://doi.org/10.1089/cmb.2010.0252.
9. Kehr B, Trappe K, Holtgrewe M, Reinert K. Genome alignment with graph data structures: a comparison. BMC Bioinformatics. 2014;15(1):1. https://doi.org/10.1186/1471-2105-15-99.
10. Holley G, Wittler R, Stoye J. Bloom filter trie–a data structure for pan-genome storage. In: International Workshop on Algorithms in Bioinformatics. Springer; 2015. p. 217–30. https://doi.org/10.1007/978-3-662-48221-6_16.
11. Lee C. Generating consensus sequences from partial order multiple sequence alignment graphs. Bioinformatics. 2003;19(8):999–1008.
12. Schulz MH, Zerbino DR, Vingron M, Birney E. Oases: robust de novo rna-seq assembly across the dynamic range of expression levels. Bioinformatics. 2012;28(8):1086–92.
13. Vaser R, Sović I, Nagarajan N, Šikić M. Fast and accurate de novo genome assembly from long uncorrected reads. Genome Res. 2017;27(5):737–46. https://doi.org/10.1101/gr.214270.116. http://genome.cshlp.org/content/27/5/737.full.pdf+html.
14. Lisiecka A. Mafgraph. https://github.com/anialisiecka/Mafgraph. Accessed 12 Sep 2019.
15. Earl D, Nguyen N, Hickey G, Harris RS, Fitzgerald S, Beal K, Seledtsov I, Molodtsov V, Raney BJ, Clawson H, Kim J, Kemena C, Chang J-M, Erb I, Poliakov A, Hou M, Herrero J, Kent WJ, Solovyev V, Darling AE, Ma J, Notredame C, Brudno M, Dubchak I, Haussler D, Paten B. Alignathon: a competitive assessment of whole-genome alignment methods. Genome Res. 2014;24(12):2077–89. https://doi.org/10.1101/gr.174920.114. http://genome.cshlp.org/content/24/12/2077.full.pdf+html.
16. Edgar RC, Asimenos G, Batzoglou S, Sidow A. Evolver. http://www.drive5.com/evolver. Accessed 12 Sep 2019.
17. Earl D, Paten B, Diekhans M. evolverSimControl. https://github.com/dentearl/evolverSimControl. Accessed 12 Sep 2019.
18. Wang Q-M, Liu W-Q, Liti G, Wang S-A, Bai F-Y. Surprisingly diverged populations of s accharomyces cerevisiae in natural environments remote from human activity. Mol Ecol. 2012;21(22):5404–17.
19. UCSC Ebola Portal. https://genome.ucsc.edu/ebolaPortal/. Accessed 12 Sep 2019.
20. Proposal for a revised taxonomy of the family filoviridae: classification, names of taxa and viruses, and virus abbreviations arch virol. Arch Virol. 2010;155(12):2083–103.
21. Emanuel J, Marzi A, Feldmann H. Filoviruses: ecology, molecular biology, and evolution. In: Advances in Virus Research, vol. 100. Elsevier; 2018. p. 189–221. https://doi.org/10.1016/bs.aivir.2017.12.002.

Publisher's Note