BMC Bioinformatics

RESEARCH Open Access

Reconstructing B cell lineage trees with minimum spanning tree and genotype abundances

Nika Abdollahi[1,3], Lucile Jeusset[1,2], Anne de Septenville[2], Frederic Davi[2] and Juliana Silva Bernardes[1*]

*Correspondence: juliana.silva_bernardes@sorbonne-universite.fr

[1] UMR 7238, Laboratoire de Biologie Computationnelle et Quantitative, Sorbonne University, Paris, France
[2] AP-HP, Hôpital Pitié-Salpêtrière, Department of Biological Hematology, Sorbonne University, Paris, France
[3] IMGT®, The International ImMunoGeneTics Information System, CNRS, Institute of Human Genetics, Montpellier University, Montpellier, France

Abstract

B cell receptor (BCR) genes exposed to an antigen undergo somatic hypermutations and Darwinian antigen selection, generating a large BCR-antibody diversity. This process, known as B cell affinity maturation, increases antibody affinity, forming a specific B cell lineage that includes the unmutated ancestor and mutated variants. In a B cell lineage, cells with a higher antigen affinity will undergo clonal expansion, while those with a lower affinity will not proliferate and probably be eliminated. Therefore, cellular (genotype) abundance provides a valuable perspective on the ongoing evolutionary process. Phylogenetic tree inference is often used to reconstruct B cell lineage trees and represents the evolutionary dynamic of BCR affinity maturation. However, such methods should process B-cell population data derived from experimental sampling that might contain different cellular abundances. There are a few phylogenetic methods for tracing the evolutionary events occurring in B cell lineages; best-performing solutions are time-demanding and restricted to analysing a reduced number of sequences, while time-efficient methods do not consider cellular abundances. We propose ClonalTree, a low-complexity and accurate approach to construct B-cell lineage trees that incorporates genotype abundances into minimum spanning tree (MST) algorithms. Using both simulated and experimental data, we demonstrate that ClonalTree outperforms MST-based algorithms and achieves a comparable performance to a method that explores tree-generating space exhaustively. Furthermore, ClonalTree has a lower running time, being more convenient for building B-cell lineage trees from high-throughput BCR sequencing data, mainly in biomedical applications, where a lower computational time is appreciable. It is hundreds to thousands of times faster than exhaustive approaches, enabling the analysis of a large set of sequences within minutes or seconds and without loss of accuracy. The source code is freely available at github.com/julibinho/ClonalTree.

Keywords: B cell receptor repertoire, Lineage Tree, phylogenetics

Background

B cells are essential components of the adaptive immune system. They express a cell surface receptor, the B cell receptor (BCR), recognising a vast array of antigens. The main components of the BCR are immunoglobulins (IG), which can be secreted in a soluble form as antibodies. IGs are heterodimers composed of two identical heavy chains (IGH) and two identical light chains (IGL); each chain possesses a variable and a constant region. The variable regions are responsible for antigen-binding specificities, whereas the constant regions are associated with cell-signalling components of the BCR and facilitate interactions with other immune-system molecules. The variable regions are encoded by Variable (V), Diversity (D) in the case of heavy chains, and Joining (J) genes, which are rearranged during lymphopoiesis by a complex genetic mechanism known as V(D)J recombination [1]. During this process, the combinatorial diversity is generated by connecting these genes into an exon, further enhanced by random deletions and insertion at their joining sites. These mechanisms allow newly-formed naive B cells to express a vast repertoire of distinct BCRs ($> 10^{13}$) [2].

When exposed to an antigen, naive B cells' genes encoding the IG undergo multiple rounds of mutations, e.g., Somatic HyperMutations (SHM) and Darwinian antigen selection. This stage of the B cell development is known as affinity maturation since it leads to a progressive increase of the IG's affinity for their cognate antigens and occurs in specialised structures of secondary lymphoid organs, the germinal centres. During affinity maturation, B cells encounter numerous biological processes such as antigen presentation, proliferation, and differentiation. Natural selection selects B-cells with higher IG-antigen affinities, which will proliferate and undergo clonal expansion, while those with lower affinity will be eliminated [3]. Affinity maturation produces a functionally heterogeneous population with different B cell lineages, each formed by the naive/unmutated B cell and its variants. Thus, the number of unique variant sequences and their respective abundances (genotypes) provide an important perspective on the ongoing evolutionary process and help elucidate clonal selection.

Understanding BCR repertoire evolution is necessary to answer fundamental biological questions such as clonal selection during antigen challenges, immune senescence, development of efficient vaccines, therapeutic monoclonal antibodies, or further understanding of B cell tumour developments. Recently, an evolutionary approach was used to quantify dissimilarity between BCR repertoires of young and aged individuals after influenza vaccination [4]. Antibody evolutionary studies have also been used to guide the clonal reconstruction of BCR repertoires [5, 6].

Several studies have analysed B cell lineage trees to understand the evolutionary mechanisms involved in several diseases [7, 8]. In this later contribution, the authors showed that lineage trees are largely shaped by antigen-driven selection occurring during an immune response. Genetic evolution, such as observed during affinity maturation, is often modelled through phylogenetic inference. This well-known methodology describes the evolution of related DNA or protein sequences in various species. Theoretically, phylogenetic inference methods could be used to build B cell lineage trees by replacing species with BCR IGH sequences (with different mutations). However, in a phylogenetic tree, the root is usually unknown, the observed sequences are usually represented only in the leaves, and the inner nodes represent the relationships among sequences. Conversely, in a B cell lineage tree, the root, the BCR IGH sequence of the unmutated B cell giving rise to the lineage, can be deduced by aligning the IG variable region sequences with sequences stored in a reference database [9]. This alignment allows identifying the germline V, (D) and J genes in the ancestor cell from which the B-cell lineage derives [10]. Another distinct point is that B cells with different BCR mutations can coexist; therefore, the observed BCR IGH sequences can be leaves or internal nodes in the tree [11, 12]. Due to simultaneous divergence, multifurcations are also common [13]. IG sequences are under intense selective pressure, and the neutral evolution assumption is invalid. Moreover, the context dependence of SHM violates the assumption that sites evolve independently and identically. Under these circumstances, conventional phylogenetic tree algorithms seem unsuitable for reconstructing B cell lineage trees. The performance of such methods varies substantially in terms of the tree topology and the ancestral sequence, as shown previously [13, 14].

Some computational tools have been explicitly designed to build B-cell lineage trees. Igtree [15] employs the maximum parsimony criterion to find the minimal set of events that could justify the observed sequences. It first constructs a preliminary tree, which only contains observed sequences, then uses a combined score based mainly on sequence mutations to gradually add internal nodes (unobserved sequences). IgPhyML combines the maximum likelihood approach [16] with a codon substitution model that uses a Markov process to describe substitutions between codons [17]. IgPhyML has modified the codon substitution model to incorporate hot/cold-spot biases observed in BCR IGH sequences [18]. GCtree [19] employs the maximum parsimony principle and incorporates the cellular abundance of a given genotype in phylogenetic inference. This information is used for ranking parsimonious trees, obtained by dnapars [20] with the assumption that more abundant parents are more likely to generate mutant descendants. GCtree uses a likelihood function based on the Galton-Watson Branching process [21]. It is an accurate method, but its computational complexity is high, especially for a high number of BCR IGH sequences. GLaMST [22] is another method for creating B cell lineage trees. It is based on a minimum spanning tree (MST) algorithm and it iteratively builds the lineage tree from the root to leaves by adding minimal edge costs. GLaMST is more time-efficient than GCtree, but it ignores genotype abundance information.

Here we propose ClonalTree, a method to build B cell lineage trees, combining MST and genotype abundances. ClonalTree is a multi-objective-based approach that first minimises edge costs and then maximises genotype abundances to infer maximum parsimony trees. Using simulated and experimental data, we demonstrate that ClonalTree outperforms GLaMST and achieves a comparable performance to GCtree. ClonalTree has however a lower time complexity and great potential for many applications, particularly in clinical settings where time constraint is essential.

Methods

In this section, we describe our approach, the data sets used in our experiments, and the metrics used to evaluate the performance of inferred trees.

Approach

ClonalTree reconstructs B cell lineage trees based on the minimum spanning tree (MST) and cellular (genotypes) abundances. We start with a formal description of the B-cell lineage tree reconstruction problem and minimum spanning tree algorithms. Next, we describe how we model the problem as a multi-objective optimisation by modifying the

Prim's algorithm [26] to incorporate genotype abundance information. Ultimately, we discuss how the BCR IGH sequence distance was computed, and show how trees can be improved by creating intermediate nodes that describe non-observed sequences or by performing editing operations.

Problem statement

Given a set of observed BCR IGH sequences with the same V(D)J rearrangement event and an inferred naive/unmutated BCR IGH sequence, we look for a minimum-sized directed tree structure, the B-cell lineage tree, which might represent the affinity maturation process. Vertices (nodes) represent BCR IGH sequences, and the weight of edges connecting vertices represents the distance between sequences in terms of mutation, insertion, and deletion operations. All observed sequences are reachable from the root (the inferred naive/unmutated BCR IGH sequence).

Minimum spanning tree

Given a connected undirected graph (V, E), where V is a set of vertices, and E is the weight edges, its minimum spanning tree (MST) is a subset of vertices and edges that form a tree (a connected graph without cycles/loops) so that the sum of the weights of all the edges (the cost) is at minimum. For a given connected undirected graph, several MSTs can exist. All trees have similar costs, but their topologies are different. Most MST construction algorithms are greedy approaches [26, 27], where edges are sorted according to their weights and selected according to some criteria. In each step, greedy algorithms make a locally-optimal choice hoping that this choice will lead to a globally optimal solution.

Multi-objective optimisation

To find the most parsimonious B-cell lineage tree, we model the problem as a multi-objective optimisation problem. Thus, we have two objective functions: the first one minimises the sum of edge weights, while the second function maximises the genotype abundance. There are many methods to solve a multi-objective problem [28]; we used the hierarchical optimisation criteria [29], in which two or more objective functions are ranked from the most to the least important, and are optimised in this pre-established order. The first function can be modelled by some minimum spanning tree algorithm, such as Prim's [26] or Kruskal's [27]. Both are greedy approaches and present low time complexity. However, Prim's algorithm runs faster than Kruskal's in dense graphs [30]. Therefore, we modified Prim's algorithm to incorporate the second objective function. We start at the root and add all its neighbours with minimum edge weight to a priority queue. We then iteratively extract from the priority queue the node with the lowest edge weight (the first objective function) and highest genotype abundance (the second objective function). If no cycle is formed, the node and the edge are added to the tree. For each added node, all its neighbours with minimum edge weights are included in the priority queue. We keep on adding nodes and edges until we cover all nodes. To decrease the time complexity of the algorithm, we add each node only once to the priority queue.

We highlight the fact that the original Prim's algorithm has only one objective function, which minimises the sum of edge weights (cost). Here we included a second objective

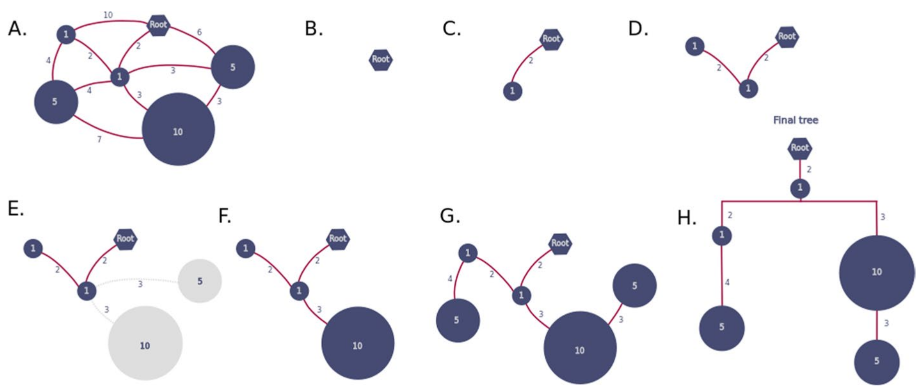

Fig. 1 ClonalTree construction example. We start with a connected weighted graph (**a**) where nodes represent BCR genotype sequences, edge weights their distances, and node weights their abundances. The graph can be fully connected, or one can disable edges whose weight is lower than a threshold δ. Then, we first place the inferred ancestral sequence (the root) (**b**) and iteratively add nodes to the tree with the lowest edge weight and highest genotype abundance (**c, d**); when edges have the same weight (**e**), we choose that connected to the node with the highest abundance (**f**), we repeat until all nodes were added to the tree (**g**), the final tree is shown in (**h**)

function to maximise genotype abundance. If a set of edges have the same weight, we will choose the one that connects nodes with high abundance. Prim's algorithm has a time complexity of $O(|V|^2)$ in the worst case but can be improved up to $O(|E| + \log |V|)$ when using data structures based on Fibonacci heaps [31]. Figure 1 shows a simple example of the tree construction process.

Measuring genotype abundances

A B cell lineage tree might represent the relationships between the unmutated ancestral BCR IGH sequence and its mutated descendants. Multiple copies of a given variant could indicate its importance in affinity maturation and clonal expansion processes. Thus, we used variant abundances (here called genotype abundance) to guide the B cell lineage tree reconstruction. A genotype can be defined differently and can represent a set of identical or similar sequences (some mutations are allowed). A common way of defining a BCR genotype is to group sequences with the same IGHV and IGHJ genes and identical complementarity determining region 3 (CDR3, the most variable part of the variable region) amino acid sequences. In both cases, the genotype abundance accounts for the number of such elements within a population of BCR IGH sequences.

Computing BCR IGH sequence distances

The reconstruction of B cell lineage trees requires a distance measurement between clonally-related sequences. This distance is usually used to weigh edges in the graph. Several works have used pairwise edit distances, such as Levenshtein [32]. However, its computation usually requires dynamic programming algorithms that become computationally expensive in time and memory when the number of sequences per lineage tree is large. Another weak point of pairwise distances is that they provide a local landscape without considering mutated regions across all the sequences. ClonalTree first performs a multiple sequence alignment to consider evolutionary events such as mutations, insertions, and deletions over all clonally-related sequences. Next, it computes a normalised hamming distance [33] between each pair of genotype sequences and then uses it as the weight edge connecting two genotypes. In order to avoid a fully connected graph, we can apply a threshold δ, that will disable edges whose weight is lower than δ.

Editing the reconstructed lineage tree

A greedy algorithm makes the optimal choice at each step, attempting to find the optimal way to solve the optimisation problem. It never reconsiders its choices, while optimal algorithms always find the best solution. A way to review decisions and improve the ClonalTree algorithm is to edit the obtained lineage tree. Thus, we implemented two strategies: adding unobserved intermediate nodes to the tree and removing/restoring sub-trees.

Unobserved internal nodes might represent unobserved sequences that were not sampled or disappeared during affinity maturation. In those cases, the evolutionary relationships were also lost. One way to recover them is to analyse the reconstructed tree to identify common ancestors not yet represented. This process is similar to building a phylogenetic tree among species, where unobserved internal nodes represent common ancestors of descendants. However, only leaves' nodes are observed in a classical phylogenetic tree, and all internal nodes are unobserved. In contract, internal nodes can be observed or unobserved in a B cell lineage tree. When we detect a missing common ancestral in the tree, we add an unobserved internal node. It generally happens when we observe a distance between sister nodes that is smaller or equal to the distance for their parent, see nodes {d, e} in Additional file 1: Fig S3-A. To add unobserved internal nodes, we traverse the tree in a pre-order manner, and for each pair of sister nodes, we verify if a common ancestor is missing. If it is the case, we add an unobserved internal node in the tree, connect it to the observed nodes by direct edges Additional file 1: Fig S3-B, and update edge weights Additional file 1: Fig S3-C. For this later, let d_{pm} and d_{pn} be the edge weights connecting a pair of sister nodes m and n to their parent p, respectively. Let d_{mn} be the distance between the sister nodes and i an unobserved internal node added to the tree. The edge weights are updated as follows:

$$\begin{aligned} d_{pi} &= max(d_{pm} - d_{mn}, d_{pn} - d_{mn}, 1) \\ d_{im} &= d_{pm} - d_{pi} \\ d_{in} &= d_{pn} - d_{pi} \end{aligned} \tag{1}$$

We can detach a sub-tree from an internal node by removing its edge and reattach it to another internal node or leaf. We perform this editing operation to reduce the depth of the lineage tree by keeping the overall cost. Here, the motivation is to try to find the most parsimonious tree. We consider all branching nodes (i.e., nodes with more than one descendant) for this edition operation. Then, we try to detach and reattach each node (under edition condition) to any other node in the lineage tree. If this operation reduces the tree depth, we accept it and examine the resulting lineage tree again for additional edition operations that may further reduce the tree depth. We repeat this process until no editing operation can reduce the depth of the tree (see an illustration in Additional file 1: Fig S4).

Data sets

We used two types of data sets to measure the ClonalTree performance and compare it with existing algorithms: simulated and experimental. GCtree simulator produced simulated data [19], while one of two experimental data sets was created during routine diagnostic procedures of a patient with CLL at the Pitié Salpêtrière hospital (Paris-France), and the second one is a public data set [11]. In the former case, informed consent for use of the diagnostic sample for research purposes was obtained.

Simulated lineage trees

To create simulated lineage trees, we used the B cell lineage simulator provided by GCtree. The simulator produces a B cell lineage by randomly selecting IGHV, IGHD, and IGHJ germline genes from the IMGT database [9]. Then, nucleotide(s) can be added to or removed from the junction region: IGHV-IGHD and IGHD-IGHJ. Next, it performs a branching process, and point mutations can be included in the descendants. Somatic hypermutations are simulated by a sequence-dependent process, where mutations are preferentially introduced within specific hot-spot motifs [34]. We kept the simulator default parameters and generated 92 artificial lineage trees. The sizes of simulated trees ranged from 6 to 99 nodes, the number of observed sequences between 20 and 200, the degree of root nodes varied from 1 to 42, and depth trees from 2 to 7, see Table 1.

Experimental data

We used a public data set from a previously reported experiment [11], where the authors combined multiphoton microscopy and single-cell mRNA sequencing to obtain IGH sequences extracted from germinal B cells of a lineage sorted from the mouse germinal centre. The data set, labelled as TAS-42, contains 65 IGHV sequences and 42 genotypes. For this data set, a genotype is a group of identical sequences. The ancestor IGHV gene was inferred with Partis [35], and the final data set contains 66 sequences: 65 from the original data set plus one representing the reconstructed germline sequence.

The second experimental data set was generated by sampling sequences from the most abundant clone of a BCR repertoire associated with a CLL patient. The data set contains 3406 sequences, annotated with IGHV4-34*01/IGHD3-3*01/IGHJ4*02 genes using IMGT/HighV-QUEST software [24]. For this data set, a genotype groups identical BCR IGH sequences; we obtained 20 genotypes with different abundances. Thus, the data set was labelled as CLL-20. To predict the hypothetical unmutated ancestral sequence (the root of this lineage tree), we considered the germline sequences of the corresponding IGHV and IGHJ genes provided by IMGT/HighV-QUEST. For the junction, we took it from the sequence with the lowest number of mutations on the IGHV gene compared to the germline determined by IMGT/HighV-Quest. Eventually, we concatenated germline IGHV, the junction sequence, and the IGHJ sequences to obtain the hypothetical unmutated sequence.

Tree comparison and evaluation

To evaluate the performance of B cell lineage reconstruction algorithms, we used two types of metrics to compare tree topologies (graph editing distances [23]) and ancestral sequence inferences (MRCA [13] and COAR [19]), explained in more details below.

Graph editing distance

Let G_1 and G_2 be two graphs; the Graph Editing Distance (GED) finds the minimum set of graph transformations able to transform G_1 into G_2 through edit operations on G_1. A graph transformation is any operation that modifies the graph: insertion, deletion, and substitutions of vertices or edges. GED is similar to string edit distances such as Levenshtein distance [32] when we replace strings by connected directed acyclic graphs of maximum degree one. We used two versions of GED, one applied to the whole tree (GED tree-based) and another to each branch separately (GED path-based). The latter version is more stringent than the first one since any difference in the path from each leaf to the root is considered a transformation.

The problem of computing graph edit distance is NP-complete [36], and there is no optimal solution in a reasonable time. This problem is difficult to approximate, and most approximation algorithms have cubic computational time [37, 38]. Here we could use an optimal algorithm implementation since the number of nodes of evaluated trees was small. Nevertheless, we favoured a grid cluster to compute GED for trees with more than 50 nodes.

Ancestral reconstruction distances

We also compared trees by measuring their ancestral sequence reconstruction agreement. For that, we used two measures previously defined: The Most Recent Common Ancestor (MRCA) [13], and The Correctness Of Ancestral Reconstruction (COAR) [19]. The MRCA metric focuses on the most recent common ancestral, while COAR considers the entire evolutionary pathway. Both measures emphasise the importance of a correct ancestral reconstruction and do not penalise minor differences in the tree topologies whether the ancestral reconstruction is accurate.

The classical MRCA distance is calculated by iterating through all pairs of leaves. It compares the most recent ancestral sequence of each pair of leaves presenting into two trees simultaneously - for instance, the ground truth (T_1) and the inferred tree (T_2). Since internal nodes in the B cell lineage trees can also represent observed genotypes, we modified MRCA to consider all node pairs instead of leaves only. For a given pair of observed nodes i and j, where $(i,j) \in T_1$ and $(i,j) \in T_2$, the MRCA(i,j) is the normalised hamming distance between the most recent ancestral sequences in T_1 and T_2 trees, given by:

$$\mathrm{MRCA}(i,j) = \frac{\mathrm{Hamming}(S_i, S_j)}{\max(|S_i|, |S_j|)} \tag{2}$$

where S_i and S_j are the nucleotide sequences associated to nodes i and j, and Hamming(S_i, S_j) computes the hamming distance between S_i and S_j. Then, the MRCA metric is obtained by:

$$\text{MRCA}(T_1, T_2) = \frac{\sum_{i,j} \text{MRCA}(i,j)}{C_2^n} \tag{3}$$

where n is the number of observable nodes in T_1, or T_2, and C_2^n gives the total number of node pairs; see an example in Additional file 1: Fig S5.

COAR [13] is another measure to evaluate the reconstruction of ancestral sequences. It compares evolutionary paths in the trees from the root (the unmmutated sequence) to any leaf. To compute it, we consider each leaf $i \in T_1$, find the path p_i from i until the root, and compare it to all paths $p_j \in T_2$ that contain i. To compare paths and obtain COAR(i), we used the Needleman–Wunsch alignment algorithm [39] with a scoring matrix based on negative hamming distance and gap penalties. The COAR metric is the average of the COAR(i) of all leaves in T_1:

$$\text{COAR}(T_1, T_2) = \frac{\sum_{i \in T_1} \text{COAR}(i)}{N_L} \tag{4}$$

where N_L is the number of leaves in T_1, and COAR(i) is computed by the Additional file 2: Algorithm 1. See also, an example in Additional file 1: Fig S6.

The numeric values of MRCA and COAR range in the interval [0, 1], where 0 represents a perfect ancestral sequence reconstruction, and 1 is the worst case. To see more details of these distance calculations, please refer to [13] and [19].

Computational tools used for comparisons

There are only a few B cell lineage reconstruction softwares; some are not available for download [15], others are specific for Windows operating system [40], and others are very time consuming. Among the available methods, we selected two state-of-art tools to compare with ClonalTree: GCtree [19] and GLaMST [22]. GCtree is an exhaustive solution; it uses dnapars [20] to find a parsimony forest and then ranks parsimonious trees according to genotype abundance information. On the other hand, GLaMST [22] is a time-efficient algorithm that uses MST but does not consider genotype quantities. The authors have shown a higher performance of these tools in the respective publications, with GCTree outperforming IgPhyML [13], and GLaMST surpassing IgTree. Thus, we decided to keep only GCTree and GLaMST for evaluation and comparison.

Results

To reconstruct B-cell lineage trees, ClonalTree starts placing the root (the inferred ancestral sequence) and iteratively adds nodes presenting minimal edge cost and maximum genotype abundance; therefore, it optimises two multi-objective functions rather than a single one based only on edge costs as implemented in other approaches. We first validated our method with several artificial lineage trees that simulate the affinity maturation process. Then, we used two data sets containing experimental B cell lineages for biological validations.

Table 1 Statistics of simulated lineage trees

	Min	Mean	Max	Std
Tree depth	2	4	7	1
Degree of Root	1	11	42	9
Number of nodes	6	34	99	21
Number of leaves	5	25	82	16
Number of observed sequences	30	81	200	48

Table 2 Parameter setting evaluation

Parameters			**Performance metrics**		
a	**r**	**t**	**Median**	**Median***	**Correct paths**
×	×	×	3	2	41
×	×		4	4	39
×		×	4	4	35
×			4	4	34
	×	×	4	4	41
		×	4	4	41
	×		4	4	35
			4	4	35

'×' indicates that the parameter was turned on, while an empty box indicates that it was turned off

Median* is the median without considering the outliers. 'Correct paths' reports the number of correct reconstructed paths (from leaf to root) in the tree, that is, when the GED path-based distance was zero

Reconstructing B cell lineage trees from simulated data

To evaluate ClonalTree performance and compare it to GCtree and GLaMST, two state-of-art tools (see "Computational tools used for comparisons" section), we generated simulated data sets using 92 different settings, varying the root gene sequence and the relative probabilities of mutation, insertion, and deletion (see "Simulated lineage trees" section, and Table 1). The simulated lineage trees served as ground truth that we would like to recover using B cell lineage tree algorithms. Thus, the performance measures how close reconstructed trees are to simulated ground truths. To quantify this resemblance, we computed different types of distances based on graph editing distances (GED) that measure the dissimilarity between two graphs/trees [23], and two previously defined distances related to the correctness of common ancestral inferences [13, 19]. We computed two types of GED distances, based on the entire tree (GED tree-based) and separated paths (GED path-based), see "Graph editing distance" section, and used MRCA [13] and COAR [19], to measure the correctness of ancestral reconstruction, see "Ancestral reconstruction distances" section.

ClonalTree parameter settings

We first varied ClonalTree parameters to determine the best configuration. ClonalTree has three Boolean parameters, 'a' considers genotype abundances, 'r' adds unobserved internal nodes when necessary, and 't' tries to reduce the tree depth by performing attach/detach operations, see "Approach" section. Table 2 shows all possible variations

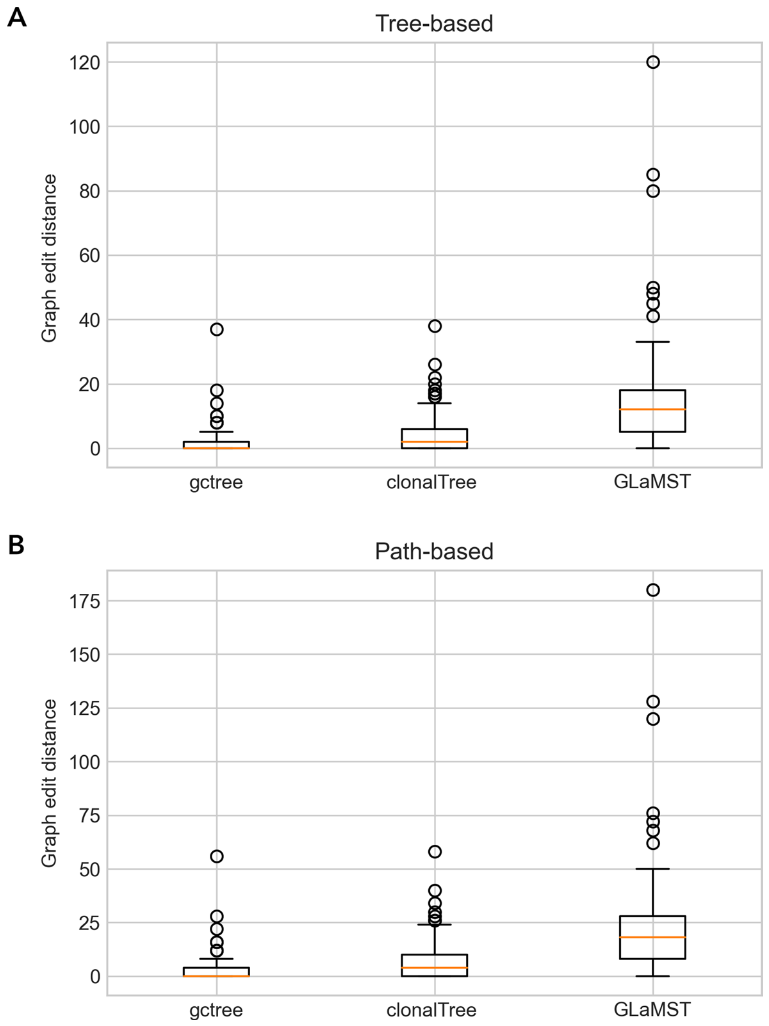

Fig. 2 Performance comparison between GCtree, ClonalTree, and GLaMST using GED distances. Box-plots in **a** present GED tree-based distances, while **b** display GED path-based ones

for these parameters, where an 'x' indicates that the parameter was turned on, and an empty box indicates that it was turned off. To measure the performance of each configuration, we computed the GED path-based distance between inferred and ground truth trees. GED path-based compares tree paths, from leaves to the root, being more sensitive to topology changes and resulting in higher distance values than the GED tree-based ones. Thus, it is more appropriate to perform configuration comparisons. We measured the performance by the overall median, the median without outliers, and the number of correct paths (GED path-based equal to zero). ClonalTree produced the best results when all three parameters were turned on. We observed that when 'a' was turned on, 't' played an important role; when it was turned off, the number of correct paths decreased. It is also interesting to note that when only the parameter 'a' was turned on, ClonalTree produced fewer outliers and the maximum GED path-based distance was 20; however, this configuration produced higher median values. Note that the original Prim's algorithm corresponds to the last row in Table 2 and the last box-plot in Additional file 1: Fig

S1, when all parameters were turned off. This configuration achieved fewer outliers but produced higher medians and inferred fewer correct paths. Thus, we kept all parameters turned on for the rest of the computational experiments.

Comparison with state-of-art methods

We compared ClonalTree to two state-of-art tools, GCtree [19], based on an exhaustive approach that uses genotype abundance information to find the most parsimonious tree, and GLaMST [22], which uses a MST-based method that does not explore genotype quantities, see "Computational tools used for comparisons" section. Figure 2 shows boxplots of GED distances for each compared method on the 92 simulated lineages trees. GCtree and ClonalTree had comparable performances, but ClonalTree outperformed GLaMST. Reconstructed B cell lineage trees of GCtree and ClonalTree displayed similar topologies, while trees produced by GLaMST were different. For GED tree-based distances (Fig. 2A), median values were 0, 2, and 12 and the highest distances 37, 38, and 120 for GCtree, ClonalTree and GLaMST, respectively. GLaMST presented the highest median value and the highest distance. ClonalTree produced 39 correct trees (GED tree-based distance equal to zero), while GLaMST produced only two.

Figure 2B, which displays GED path-based distances, confirms that GCtree and ClonalTree reconstructed B cell lineage trees with similar paths. Median values were 0, 3, and 18, with the highest distances 56, 58, and 180 for GCtree, ClonalTree, and GLaMST, respectively. As observed for GED tree-based, GLaMST presented the highest median value and distance. ClonalTree produced 41 correct paths (GED path-based distance equal to zero), while GLaMST produced only one path. In order to better evaluate the performance, we split the trees into three categories according to their number of sequences: small (between 30 and 50), medium (between 60 and 80), and large (having more than 90 sequences), see Additional file 1: Fig S2. We observed a slight difference between GCtree and ClonalTree, mainly in the small and large groups, with worse results obtained in the medium group. On the other hand, we observed that GLaMST had difficulties in all groups; GED distances increased as the number of sequences grew.

The GED metric depends on the tree topologies, mainly the GED-based path that penalises each path difference. Therefore, it is also essential to estimate the accuracy of ancestral reconstructions without penalising minor differences in the tree topologies. For that, we used two metrics: the MRCA and the COAR, see "Ancestral reconstruction distances" section. MRCA distance focuses on the most recent common ancestor and does not consider the entire evolutionary lineage. On the other hand, COAR measures the correctness of ancestral reconstruction from the root to any leaf. We first compared ClonalTree and GLaMST to ground truth trees, and then both tools with GCtree. The latter comparison is important since it gives us a basis for evaluating these methods on experimental data sets, where the true lineage evolution is unknown. Figure 3 shows MRCA distance distributions for ClonalTree and GLaMST when compared to ground truth trees (A) or GCtree (B). For both plots, we observed better performance for ClonalTree, which could reconstruct recent ancestral relationships more accurately. Figure 4 shows COAR distance distributions when comparing ClonalTree and GLaMST with ground truth trees (A) or with GCtree ones (B). Similarly, we observed that the trees produced by our approach are closer to both ground truth and GCtree trees.

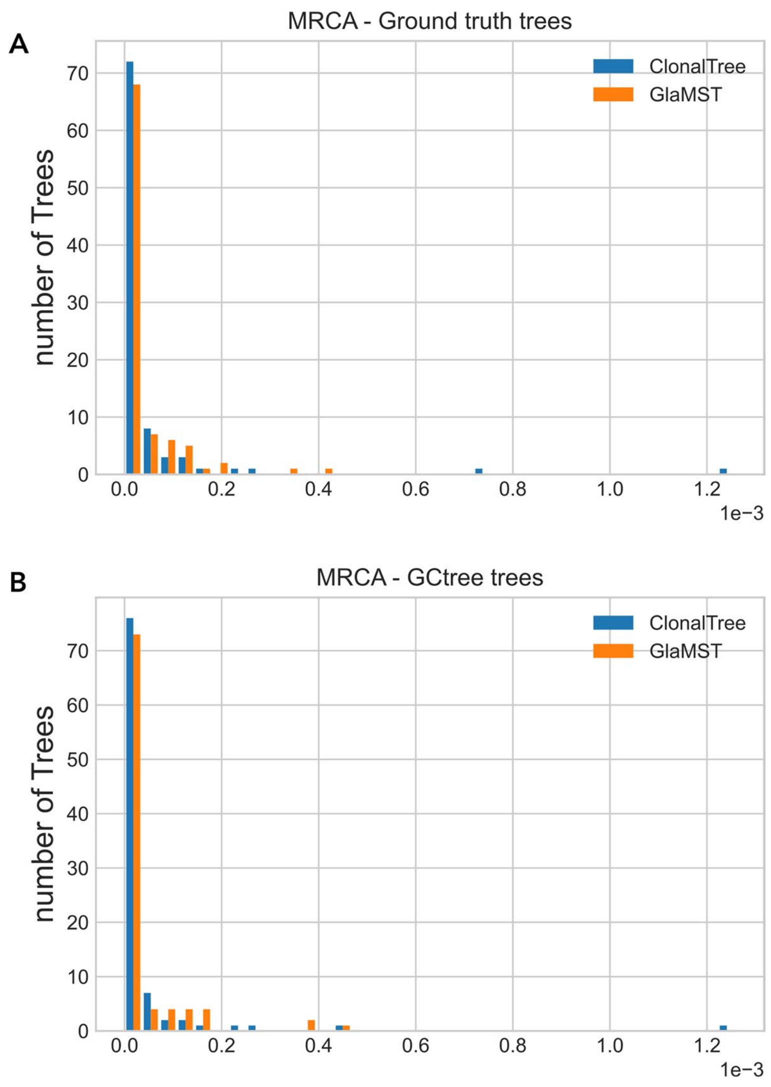

Fig. 3 Most Recent Common Ancestor (MRCA) distance distributions. **a** compares ClonalTree, and GLaMST with ground truth trees, and **b** with trees generated by GCtree

Biological validation using BCR sequencing data

We performed a biological validation on two experimental data sets: TAS-42 and CLL-20 (see "Data sets" section). Since ground truth trees are unavailable for these samples, we compared the inferred trees of ClonalTree and GLaMST with the trees inferred by GCtree. We considered trees generated with GCtree as references for the experimental validation since such a tool achieved the best performance on simulated data sets.

The TAS-42 is a public data set generated by lineage tracing and single-cell germinal centre BCR sequencing [11]. TAS-42 contains 66 sequences, used as input for GCtree, GLaMST, and ClonalTree. The 2^{nd} and 3^{rd} columns of Table 3 show tree distances for ClonalTree and GLaMST when compared to GCtree, respectively. The GED tree-based distance was smaller for ClonalTree, producing a tree with ten differences from GCtree. GLaMST produced a more disparate tree than ClonalTree; Its GED tree-based distance was 47. GED path-based distances also showed that ClonalTree produced more similar

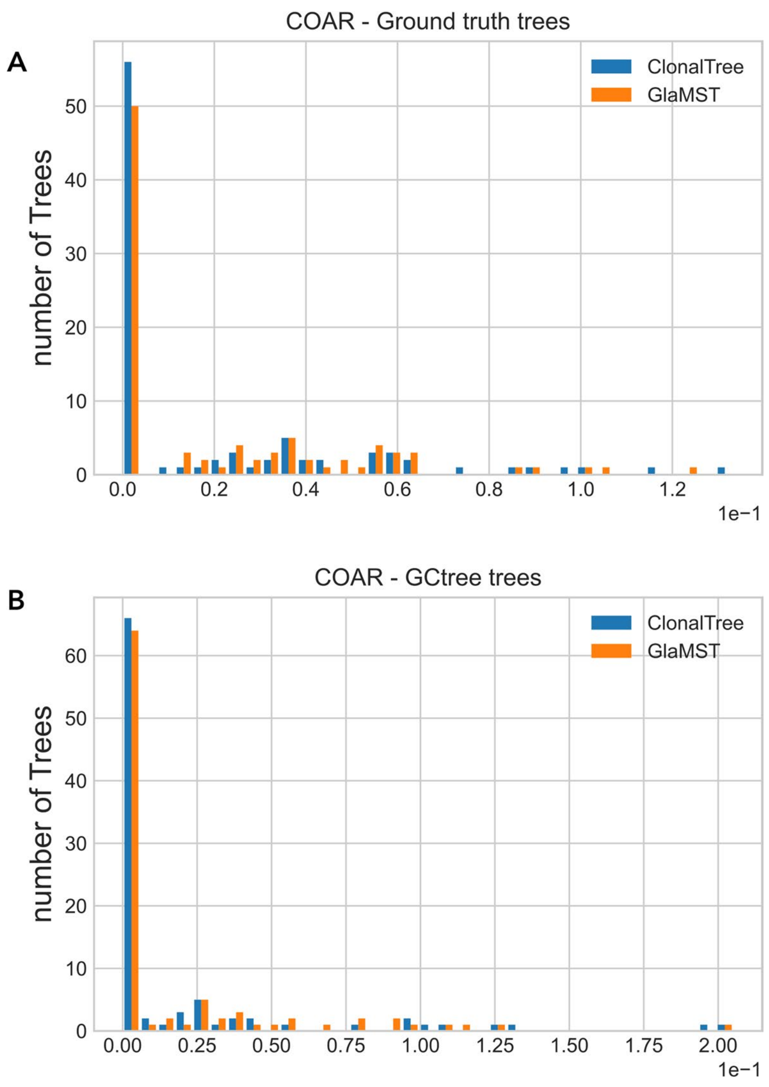

Fig. 4 Correctness Of Ancestral Reconstruction (COAR) distance distributions. **a** compares ClonalTree, and GLaMST with ground truth trees, while **b** with trees generated by GCtree

evolutionary paths to GCtree than GLaMST. Only 22 evolutionary paths differ from GCtree against 230 for GLaMST. Most GLaMST paths contained various unobserved nodes, producing many mismatches compared to GCtree. We temporarily removed them to evaluate the tool's performance. In the absence of unobserved nodes, ClonalTree presented just one mismatched path, while GLaMST produced seven. We also evaluated the agreement of ancestral sequence reconstructions. MRCA values were low for ClonalTree and GLaMST; both tools detected recent ancestral accurately for most node pairs. We did not observe a significant difference between MRCA values presented in Table 3. However, COAR distance values indicated that ClonalTree had better reconstructed entire evolutionary paths; we observed significantly lower COAR values for ClonalTree than GLaMST. Since ClonalTree achieved a lower GED path-based distance, it was expected to present lower COAR distances. COAR also considers sequence

Table 3 Performance evaluation of ClonalTree, and GLaMST on experimental BCR repertoire data sets.*GED path-based when removing unobserved nodes

Metric	TAS-42		CLL-20	
	ClonalTree	GLaMST	ClonalTree	GLaMST
GED tree-based	10	47	3	36
GED path-based	22	230	18	1616
GED path-based *	1	7	2	13
MRCA $*10^{-4}$	5.16	5.7	6.4	7.5
COAR	0.02	0.54	0.11	0.78

dissimilarities as GED path-based distance rather than only differences in the paths. Both measures confirmed that ClonalTree produces trees that are closer to the GCtree ones.

The CLL-20 data set was obtained from the blood samples of a patient with CLL (Chronic lymphocytic leukaemia). We only analysed the most abundant clone, which contains 3406 sequences and 20 different genotypes. For these sequences, IMGT/HighV-QUEST [24] inferred ancestor IGHV, IGHD, and IGHJ genes using germline sequences from the IMGT [9] database. The 4th and 5th columns of Table 3 show tree distances for ClonalTree and GLaMST, respectively. ClonalTree GED tree-based distance was the smallest among experimental data sets, showing that ClonalTree generated a tree with a very similar topology to the one generated with GCtree. Likewise, GED path-based distance was smaller for ClonalTree. GLaMST presented an extremely high GED path-based distance; it added many unobserved nodes, causing several mismatches when comparing tree paths. We also evaluated tree path agreement without considering unobserved nodes. ClonalTree presented only two different paths against 13 of GLaMST. As observed in the data set TAS-42, we noted a slight difference between ClonalTree and GLaMST MRCA values, indicating that both tools reconstructed the most recent ancestral properly. However, a notable difference was again observed in COAR, with the ClonalTree distance being smaller than the GLaMST one. This result is coherent with a better GED path-based distance since COAR accounts for the entire path from a leaf to the root.

These experimental results show that the trees generated by ClonalTree were closer to those obtained by GCtree; when compared to GLaMST, all distance metrics were smaller for ClonalTree, especially GED path-based and COAR distances. In summary, ClonalTree reconstructed whole evolutionary lineages more accurately than GLaMST.

Time complexity and running time

The time complexity of a phylogenetic reconstruction algorithm is a function that represents the computing time required to analyse an instance of the problem. Typically, it depends on the number of treated sequences n and possibly other parameters. The computational cost of algorithms that exhaustively explore the tree-generating space, producing eventually optimal solutions, increases considerably with input size. On the other hand, time-efficient approaches solve problems faster but can sacrifice accuracy, precision, or completeness. GCtree is an exhaustive approach that explores the entire tree-generating space to find the most parsimonious tree. It uses dnapars with $O(n^3)$ complexity to produce a forest of equally parsimonious trees. Then GCtree ranks the equally parsimonious trees through the abundance of genotypes. Although ranking parsimony trees requires just a polynomial increase in the dnapars runtime, finding the parsimony forest is computationally demanding and restricted to analysing small-sized problems. MST-based algorithms are time-efficient; they are faster than GCtree because they construct a single tree instead of a forest. Their time complexity is determined by the MST algorithm's complexity and the pairwise distance calculation. GLaMST and ClonalTree use Prim's algorithm to grow the tree from the root node and interactively add edges and nodes (possibly unobserved). The time complexity of Prim's algorithm is $O(|E| + \log |V|)$, where $|V| = n$ and E is the number of edges or connections among nodes. However, GLaMST computes pairwise edit distances using a dynamic programming algorithm with time complexity around $O(n^2)$. Contrarily to GCtree and GLaMST, ClonalTree uses hamming distance since sequences are previously aligned with Clustal Omega [25]. Both alternatives decrease the ClonalTree time complexity substantially. Hamming distance is computed in $O(n)$, while multiple sequence alignment is in $O(n * (\log(n))^2)$.

In order to provide an estimation of running times of different tools, we choose three data sets with different properties: number of sequences and genotypes. Table 4 shows the execution times of each compared tool. Data set D_1 corresponds to a simple simulation, containing 30 sequences and six genotypes. D_1 is the simplest case, and all tools took less than 1 second. D_2 is a data set provided by GLaMST, containing 684 observed sequences and 541 different genotypes. GCtree took 118 hours, GLaMST spent 55 minutes, and ClonalTree took less than 15 minutes to reconstruct a lineage tree for this data set. D_3 is the CLL-20 data set, containing 3406 sequences but just 20 genotypes. ClonalTree achieved the lowest execution time (less than 1 second), while GLaMST took 1.5 seconds. GCtree needed 118 hours to produce a tree for the D_3 data set. Although D_3 had more sequences than D_2, it contained fewer genotypes, which explains why execution times were lower for D_3, compared to D_2.

Table 4 ClonalTree, GLaMST and GCtree running times on three data sets

Data set	Number of sequences	Number of genotypes	Tool	Running time
D_1	30	6	ClonalTree	0.006 (s)
			GLaMST	0.035 (s)
			GCtree	0.6 (s)
D_2	685	541	ClonalTree	15 (m)
			GLaMST	55 (m)
			GCtree	100 (h)
D_3	3406	20	ClonalTree	0.032 (s)
			GLaMST	1.5 (s)
			GCtree	118 (h)

Times were measured in (s)econds, (m)inutes, and (h)ours. Note that we used a pre-compiled version of GLaMST since we did not have Matlab software to recompile the source code

Discussion

We have explored 92 simulated B cell lineage trees, ranging in trees' depths, degrees of roots, number of nodes and leaves, and number of observed sequences, among other tree properties. Our simulations showed that ClonalTree reconstructed accurate trees, preserving the correctness of ancestral reconstruction, compared to ground truth trees. Our approach outperformed GLaMST, a method based only on the minimum spanning tree; ClonalTree systematically produced more maximum parsimonious trees than GLaMST. Compared to GCtree, an exhaustive sequence-based phylogenetic method, ClonalTree presented a comparable performance in most cases, and we observed few differences in the obtained trees. We also evaluated ClonalTree on experimental BCR sequencing data. As observed on simulated data sets, ClonalTree outperformed GLaMST, producing trees closer to GCtree. Our approach obtained similar tree topologies for both experimental data sets, with minor errors in ancestral reconstructions.

ClonalTree is time efficient, showing the lowest time complexity among the compared tools. Its computational efficiency is directly linked to the algorithms choices and programming language. ClonalTree employs multiple sequence alignments, which are faster to compute and preserve important evolutionary properties that would otherwise be lost when using pairwise sequence comparison. Moreover, it is coded in python and uses freely available libraries/software, without any operating system requirement, and is available for scientific community usage. On the other hand, GLaMST is coded in Matlab, a proprietary programming language, and only an executable for Windows is available for users that do not possess a Matlab licence, limiting its usage by the scientific community. Due to its ability to process a high number of sequences within a reasonable amount of time, it is particularly adapted to large datasets obtained by high-throughput repertoire sequencing. ClonalTree may have a great potential for studying the dynamics of immune responses in various disease conditions, such as chronic infectious diseases, autoimmune diseases and lymphoid malignancies. It can compare trees generated in these situations to those generated in normal immune response dynamics. Moreover, it can help understand B-cell clonal lineage evolution by comparing trees of a specific B-cell lineage at different time points, for example, before and after vaccination or treatment.

However, ClonalTree has some limitations; for instance, it does not consider hot spot mutated positions to define a distance between sequences; in the current version, all positions have the same weight. Moreover, it considers only nucleotide sequences as input and does not provide tree visualisations. Future work will address these limitations to provide a more flexible and user-friendly tool.

Conclusion

We addressed the computational problem of reconstructing lineage trees from high-throughput BCR sequencing data. It is challenging since this data often contains a subset of BCR IGH sequences, partially representing the dynamic process of BCR affinity maturation. Therefore, efficient algorithms must reconstruct the entire evolutionary events from partially observed sequences. Here we propos ClonalTree, a fast method that combines minimum spanning trees with genotype abundance information to reconstruct accurate evolutionary trees. ClonalTree outperformed MST-based methods on simulated and experimental data, and presented similar performances, mainly on experimental data, compared to exhaustive and time-consuming methods such as GCtree. However, ClonalTree is computationally more efficient, presenting a lower time complexity. It was hundreds to thousands of times faster than GCtree, allowing the analysis of large data sets within minutes or seconds and with minor loss of accuracy. ClonalTree's high and fast performance allows the users to consider all the available genotypes when reconstructing lineage trees. It can help researchers understand B cell receptor affinity maturation, mainly when data from a dense quantitative sampling of diversifying loci are available. Integrating ClonalTree into existing BCR sequencing analysis frameworks could speed up lineage tree reconstructions without compromising the quality of evolutionary trees.

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s12859-022-05112-z.

Additional file 1: Fig. S1. ClonalTree configuration performance; **Fig. S2.** Performance comparison among GCtree, ClonalTree, and GLaMST using GED distances on three categories of trees; **Fig. S3.** Editing the reconstructed B cell lineage tree by adding unobserved internal nodes; **Fig. S4.** Editing the reconstructed BCR lineage tree to reduce the depth of the tree while keeping its overall cost; **Fig. S5.** An example of MRCA calculation; **Fig. S6.** An example of COAR calculation for a given leaf.

Additional file 2. COAR algorithm.

Acknowledgements
The authors are grateful to Thibaud Verny for his insightful comments and fruitful discussion. We also acknowledge the anonymous reviewers that contributed to improving the manuscript and the method.

Author contributions
JSB and FD designed the project. NA and JSB developed the tool. NA and ADS collected data, and produced simulated benchmarks. JSB, LJ and ADS tested and evaluated the method. JSB wrote the manuscript. All authors read and approved the final manuscript.

Funding
This work has been supported by "2016 Programme Doctoral de Cancérologie" from Sorbonne University, and SIRIC CURAMUS (Cancer United Research Associating Medicine, University and Society, INCA-DGOS-Inserm_12560) and by an ERA-NET TRANSCAN-2/French National Cancer Institute grant (NOVEL consortium).

Availability of data and materials
The data supporting this work's findings can be found at https://github.com/matsengrp/GCtree/tree/master/example and https://github.com/julibinho/ClonalTree/tree/main/Data. The source code and install instructions are available at https://github.com/julibinho/ClonalTree/.

Declarations

Ethics approval and consent to participate
Informed consent for use of the patients samples for research purposes was obtained.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 5 October 2022 Accepted: 13 December 2022
Published online: 27 February 2023

References
1. Bassing CH, Swat W, Alt FW. The mechanism and regulation of chromosomal v (d) j recombination. Cell. 2002;109(2):45–55.
2. Tonegawa S. Somatic generation of antibody diversity. Nature. 1983;302(5909):575–81.
3. Victora GD, Nussenzweig MC. Germinal centers. Annu Rev Immunol. 2012;30:429–57.
4. de Bourcy CF, Angel CJL, Vollmers C, Dekker CL, Davis MM, Quake SR. Phylogenetic analysis of the human antibody repertoire reveals quantitative signatures of immune senescence and aging. Proc Natl Acad Sci. 2017;114(5):1105–10.
5. Safonova Y, Pevzner PA. Igevolution: clonal analysis of antibody repertoires. bioRxiv, 2019;725424

6. Lee DW, Khavrutskii IV, Wallqvist A, Bavari S, Cooper CL, Chaudhury S. Brilia: integrated tool for high-throughput annotation and lineage tree assembly of b-cell repertoires. Front Immunol. 2017;7:681.
7. Greaves M, Maley CC. Clonal evolution in cancer. Nature. 2012;481(7381):306–13.
8. Hoehn KB, Fowler A, Lunter G, Pybus OG. The diversity and molecular evolution of b-cell receptors during infection. Mol Biol Evol. 2016;33(5):1147–57.
9. Giudicelli V, Chaume D, Lefranc M-P. IMGT/GENE-DB: a comprehensive database for human and mouse immunoglobulin and T cell receptor genes. Nucleic Acids Res. 2004;33(Database issue):256–61.
10. Alamyar E, Duroux P, Lefranc M-P, Giudicelli V. Imgt® tools for the nucleotide analysis of immunoglobulin (ig) and t cell receptor (tr) v-(d)-j repertoires, polymorphisms, and ig mutations: Imgt/v-quest and imgt/highv-quest for ngs. In: Immunogenetics. Springer, p. 569–604 (2012)
11. Tas JM, Mesin L, Pasqual G, Targ S, Jacobsen JT, Mano YM, Chen CS, Weill J-C, Reynaud C-A, Browne EP. Visualizing antibody affinity maturation in germinal centers. Science. 2016;351(6277):1048–54.
12. Kuraoka M, Schmidt AG, Nojima T, Feng F, Watanabe A, Kitamura D, Harrison SC, Kepler TB, Kelsoe G. Complex antigens drive permissive clonal selection in germinal centers. Immunity. 2016;44(3):542–52.
13. Davidsen K, Matsen FA IV. Benchmarking tree and ancestral sequence inference for b cell receptor sequences. Front Immunol. 2018;9:2451.
14. Yermanos A, Greiff V, Krautler NJ, Menzel U, Dounas A, Miho E, Oxenius A, Stadler T, Reddy ST. Comparison of methods for phylogenetic b-cell lineage inference using time-resolved antibody repertoire simulations (absim). Bioinformatics. 2017;33(24):3938–46.
15. Barak M, Zuckerman NS, Edelman H, Unger R, Mehr R. Igtree: creating immunoglobulin variable region gene lineage trees. J Immunol Methods. 2008;338(1–2):67–74.
16. Felsenstein J. Evolutionary trees from dna sequences: a maximum likelihood approach. J Mol Evol. 1981;17(6):368–76.
17. Liò P, Goldman N. Models of molecular evolution and phylogeny. Genome Res. 1998;8(12):1233–44.
18. Hoehn KB, Lunter G, Pybus OG. A phylogenetic codon substitution model for antibody lineages. Genetics. 2017;206(1):417–27.
19. DeWitt WS III, Mesin L, Victora GD, Minin VN, Matsen FA IV. Using genotype abundance to improve phylogenetic inference. Mol Biol Evol. 2018;35(5):1253–65.
20. Felsenstein J. PHYLIP (phylogeny Inference Package), Version 3.5 C. Joseph Felsenstein (1993)
21. Harris TE. The theory of branching process (1964)
22. Yang X, Tipton CM, Woodruff MC, Zhou E, Lee FE-H, Sanz I, Qiu P. Glamst: Grow lineages along minimum spanning tree for b cell receptor sequencing data. BMC Genomics. 2020;21(9):1–11.
23. Sanfeliu A, Sanfeliu A, Fu KS. A distance measure between attributed relational graphs for pattern recognition. IEEE Trans Syst Man Cybern. 1983;SMC–13(3):353–62.
24. Lefranc M-P, Duroux P, Li S, Giudicelli V, Alamyar E. IMGT/highv-quest: the IMGT web portal for immunoglobulin (ig) or antibody and t cell receptor (tr) analysis from ngs high throughput and deep sequencing. Immunome Res 2012;**08**(01).
25. Sievers F, Higgins DG. Clustal omega, accurate alignment of very large numbers of sequences. In: Multiple Sequence Alignment Methods. Springer, p. 105–116 (2014)
26. Prim R. Shortest connection networks and some generalizations. Bell Syst Tech J. 1957;36(6):1389–401.
27. Kruskal JB. On the shortest spanning subtree of a graph and the traveling salesman problem. Proc Am Math Soc. 1956;7(1):48–50.
28. Marler RT, Arora JS. Survey of multi-objective optimization methods for engineering. Struct Multidiscipl Optim. 2004;26(6):369–95.
29. Waltz F. An engineering approach: hierarchical optimization criteria. IEEE Trans Autom Control. 1967;12(2):179–80.
30. Huang F, Gao P, Wang Y. Comparison of prim and Kruskal on shanghai and Shenzhen 300 index hierarchical structure tree. In: 2009 International Conference on Web Information Systems and Mining (2009). p. 237–241.
31. Fredman ML. Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms. Technical Report 3.
32. Levenshtein VI. Binary codes capable of correcting deletions, insertions, and reversals. Sov Phys Doklady. 1966;10:707–10.
33. Hamming RW. Error detecting and error correcting codes. Bell Syst Tech J. 1950;29(2):147–60.
34. Yaari G, Vander Heiden J, Uduman M, Gadala-Maria D, Gupta N, Stern JN, O'Connor K, Hafler D, Laserson U, Vigneault F. Models of somatic hypermutation targeting and substitution based on synonymous mutations from high-throughput immunoglobulin sequencing data. Front Immunol. 2013;4:358.
35. Ralph DK, Matsen FA IV. Likelihood-based inference of b cell clonal families. PLoS Comput Biol. 2016;12(10):1005086.
36. Garey MR, Johnson DS. Computers and intractability: a guide to the theory of NP-completeness. Mathematical Sciences Series. W.H. Freeman (1979)
37. Neuhaus M, Bunke H. Bridging the gap between graph edit distance and kernel machines. Series in machine perception and artificial intelligence. Singapore: World Scientific; 2007.
38. Riesen K. Structural pattern recognition with graph edit distance: approximation algorithms and applications. Advances in Computer Vision and Pattern Recognition. Springer (2016)
39. Needleman SB, Wunsch CD. A general method applicable to the search for similarities in the amino acid sequence of two proteins. J Mol Biol. 1970;48(3):443–53.
40. Kepler TB. Reconstructing a b-cell clonal lineage. i. statistical inference of unobserved ancestors. F1000Research 2013;2.