Phylogenetics

Rapid alignment-free phylogenetic identification of metagenomic sequences

Benjamin Linard [1,2,3*], Krister Swenson [1,4], Fabio Pardi [1,4]

1. LIRMM, Univ Montpellier, CNRS, Montpellier, France
2. ISEM, Univ Montpellier, CNRS, IRD, EPHE, CIRAD, INRAP, Montpellier, France
3. Montpellier SupAgro, Univ Montpellier, Montpellier, France
4. Institut de Biologie Computationnelle, Montpellier, France

*To whom correspondence should be addressed.

Abstract

Motivation: Taxonomic classification is at the core of environmental DNA analysis. When a phylogenetic tree can be built as a prior hypothesis to such classification, “Phylogenetic placement” (PP) provides the most informative type of classification because each query sequence is assigned to its putative origin in the tree. This is useful whenever precision is sought (e.g. in diagnostics). However, likelihood-based PP algorithms struggle to scale with the ever increasing throughputs of DNA sequencing.
Results: We have developed RAPPAS (Rapid Alignment-free Phylogenetic Placement via Ancestral Sequences) which uses an alignment-free approach removing the hurdle of query sequence alignment as a preliminary step to PP. Our approach relies on the precomputation of a database of k-mers present with non-negligible probability in the relatives of the reference sequences. The placement is performed by inspecting the stored phylogenetic origins of the k-mers in the query, and their probabilities. The database can be reused for the analysis of several different metagenomes. Experiments show that the first implementation of RAPPAS is already faster than the previous likelihood-based PP algorithms, while keeping similar accuracy for short reads. RAPPAS scales PP to the era of routine metagenomic diagnostics.
Availability: Program and sources freely available for download at gite.lirmm.fr/linard/RAPPAS.
Contact: benjamin.linard@lirmm.fr

Keywords

Phylogenetic placement ; metagenomic ; alignment-free ; species identification ; ancestral sequence reconstruction

1. Introduction

Metagenomic sequencing aims to study bulk DNA extractions which represent complex biological communities and have become the norm to analyze a large spectrum of environmental samples. Applications range from health care to agro-environmental projects with important implications in large-scale DNA-based biomonitoring (Baird and Hajibabaei, 2012). Furthermore, with DNA sequencing miniaturization DNA-based diagnostics will soon be pushed to every layer of society (Gilbert, 2017; Zaaijer *et al.*, 2016). The bioinformatic analysis of metagenomes is challenging because most sampled organisms are still poorly known and identifying the species at the origin of metagenomic sequences (MSs) remains a bottleneck for diagnoses based on environmental DNA.

Several classes of bioinformatic/algorithmic approaches are possible for taxonomic classification of MSs. The choice will generally depend on the availability of prior knowledge (a reference database of known sequences and their relationships) and on which speed/precision trade-off is acceptable to answer the biological question. For instance, without priors the exploration of relatively unknown clades can rely on unsupervised clustering (Sedlar *et al.*, 2017; Ondov *et al.*, 2016). The precision of this approach is limited and clusters are generally assigned to taxonomic levels via posterior analyses involving pairwise alignments to reference databases. An alternative is to contextualise pairwise alignments with taxonomic metadata. For instance, MEGAN (Huson *et al.*, 2007, 2016) assigns a MS to the “Last-Common-Ancestor” (LCA) of all sequences to which it could be aligned with a statistically-significant similarity score (using BLAST or other similarity search tools). A faster alternative is to build a database of taxonomically-informed k-mers which are pre-computed from a set of reference sequences and stored in a database. Classification algorithms are then assigning MSs to the LCA sharing the highest number of those k-mers (Liu *et al.*, 2018; Müller *et al.*, 2017; Wood and Salzberg, 2014). These approaches trade-off high memory consumption and lower sensitivity for much faster analysis of bacterial communities (Liu *et al.*, 2018). All of these approaches use simple sequence similarity metrics and limited modelisation of the evolutionary phenomena linking both MSs and reference sequences. Still, particularly in the case of k-mer-based approaches, this choice

allows the analysis of millions of MSs in a timescale compatible with large-scale monitoring and everyday diagnostics.

Nevertheless, in many cases highly precise species identification is desired. This is particularly true for medical diagnostics where adapted treatments will be administered only after robust identification of the pathogen(s) (Butel, 2014) . Especially in the case of viruses (poorly known and characterized by rapid evolution) only advanced evolutionary modelling is sufficiently resolutive to elucidate a detailed classification. A possible answer in this direction may appear to be the use of phylogenetic inference methods. However, the standard tools for phylogenetic inference are not applicable to metagenomic datasets, not least because of the sheer number of sequences to analyze which can not be handled by current implementations (Izquierdo-Carrasco *et al.*, 2011).

A solution to this bottleneck is phylogenetic placement (PP) (Matsen *et al.*, 2010; Berger *et al.*, 2011). The central idea of this technique is to ignore the evolutionary relationships between the metagenomic sequences and to focus only on elucidating the relationships between the metagenomic sequence and the known reference genomes. In more detail, a number of reference genomes (or genomic markers) of well-known species are aligned together and a “reference” phylogenetic tree is inferred from the resulting alignment. This reference is the backbone onto which the reads matching the reference genomes will be “placed”, i.e. assigned to specific edges. When a “query MS” (an individual sequence among the MSs) is placed on an edge, this is taken to represent the evolutionary point where it diverged from the phylogeny. Note that when the placement edge of a query MS is not close to a tip of the tree, the interpretation is that the MS does not belong to any reference genome but to a yet unknown/unsampled genome for which the divergence point has been clearly identified. This is different from any LCA-based approaches where a MS assigned to an internal node usually means that it cannot be confidently assigned to any species of the corresponding subtree. Currently, only two PP methods are available, PPlacer (Matsen *et al.*, 2010; McCoy and Matsen, 2013) and EPA (until recently a component of RAxML) (Berger *et al.*, 2011). They are both likelihood-based, i.e. they seek to place each MS in the position that maximises the likelihood of the tree that results from the addition of the MS at that position. Alternative algorithms were also described, but never publicly released (Filipski *et al.*, 2015; Brown and Truszkowski, 2013). Despite their robustness in identifying communities (Matsen *et al.*, 2010), these methods show some limitations: (i) likelihood-based methods require that each query MS is aligned to the reference genomes prior to the placement itself, a complex step that may dominate the computation time for longer references and (ii) despite being much faster than a full phylogenetic reconstruction, their implementations do not scale with current sequencing throughputs. Recently, the

development of EPA-ng (Barbera *et al.*, 2018) provided an answer to the second point via the choice of advanced optimization and efficient parallel computing. But no fundamental algorithmic changes were introduced and the method still relies on a preliminary alignment.

To avoid these bottlenecks, we have developed the phylogenetic placement program RAPPAS (Rapid Alignment-free Phylogenetic Placement via Ancestral Sequences). Its innovative algorithm relies on two key ideas: i) “ancestral k-mers” can be built from the references tree and reference alignment to which future MSs will be placed and stored in a database ii) later, new query MSs can be placed on the tree by searching for matches between their k-mer composition and the ancestral k-mers stored in the database. For a specific group of organisms, the maintenance of the the RAPPAS database can be seen as a periodical “build-and-update” cycle, while the phylogenetic placement of new metagenomes becomes a scalable, alignment-free, process compatible with day-to-day routine diagnostics. This radical algorithmic evolution bypasses the alignment of MSs to the reference, a requirement for previous algorithms. Moreover, for short length reads RAPPAS produces placements that are as accurate as previous ML-based methods. Despite being in its initial implementation, RAPPAS is already faster than the most optimized versions of competing tools.

2. Materials and Methods

2.1 Ghost nodes and Ancestral k-mers

The preliminary step to RAPPAS placements is to build a database of k-mers (which we call “phyo-kmers”), that may be present with non-negligible probability in a MS related to the reference sequences. The input for this phase are a reference tree (noted “RefT”) and a reference alignment (noted “RefA”) on which was built the RefT. The refT is assumed to be rooted. This process is divided in three steps (Figure 1): A) the injection of “ghost nodes” on every edge of the RefT, B) a process of ancestral state reconstruction based on this extended tree and C) the generation of the phylo-kmers and their storage in a data structure.

The first step attempts to simulate evolution that might have occurred and resulted in sequences not present in the reference alignment. For each edge of the refT, a new edge branching off at its midpoint *M* is created (Figure 1, step A). The length b_{new} of the new edge is set to the average length over all paths from *M* to its descendant leaves. The new leaves of these new edges, which we refer to as “ghost nodes” (e.g. G in Fig. 1) represent unsampled sequences that may have split off the reference tree at some point in the past.

Newly sampled MSs which are only distantly related to the reference sequences should match the "sequence states" that will be inferred at the ghost nodes.

The second step computes the state probabilities associated to all ghost nodes via standard phylogenetic methods for ancestral sequence reconstruction (Figure 1, step B). Briefly, given a RefA of m sites over S possible states (4 for DNA, 20 for amino acids), for each individual ghost node G we compute a $S \times m$ table $(p_{s,i})$ of marginal probabilities (Yang *et al.*, 1995), where $p_{s,i}$ represents the probability of state s at site i. From this table, we compute the probabilities of any given k-mer at any ghost node G. Let k be an empirically chosen k-mer size and let $n \in [1, m-(k-1)]$ be a particular site of the RefA. Assuming independence among sites (see, e.g., (Penny, 2004)), the probability P of k-mer $s_1 s_2 \ldots s_k$ at sites $[n, n+k-1]$ is given by:

$$(1) \qquad P = \prod_{i=1}^{k} p_{s_i, n+i-1}$$

Each k-mer with a probability P larger than a (parameterizable) threshold (by default $1/S^k$) is then registered together with a) the edge corresponding to the ghost node of origin and b) the probability P associated to this k-mer at the ghost node of origin (Figure 1, step C). In the end, all the k-mers reconstructed above are stored in a data structure linking each k-mer with the list of edges for which it was reconstructed, and the probabilities associated to each of these edges. These "phylogenetically-informed" k-mers will be referred to as "phylo-kmers" and this structure will be referred to as the "phylo-kmers database" (pkDB). When the same k-mer is generated multiple times for a same edge, at different positions n of the RefA, only the highest probability is retained in the pkDB. For a specific reference (RefT + RefA), the user needs to generate the corresponding pkDB as a preliminary step to all subsequent phylogenetic placements (Figure 2-B). The same pkDB can be reused for multiple different datasets of MSs.

2.2 Placement algorithm

RAPPAS places each query sequence on the sole basis of the matches that can be found between k-mers extracted from a query MS and k-mers stored in the pkDB. An algorithm akin to a weighted vote is employed: each k-mer casts multiple votes – one for each edge associated to the k-mer in the pkDB – and each vote is weighted proportionally to the (logarithm of the) k-mer probability at the voted edge. The edge receiving the largest total weighted vote is the one considered as the best placement. This algorithm is repeated for each query MS. Its detailed description and complexity analysis can be found in Suppl. File S1. Unlike for likelihood-based alignment + placement algorithms, the running time of

RAPPAS does not depend on the length of the reference sequences, and rather than scaling linearly in the size of the reference tree, it only depends on the number of edges that are actually associated to some k-mer in the query MS.

To follow the standards imposed by previous PP methods, the placements are reported in a JSON file following the “jplace” specification (Matsen *et al.*, 2012). In these files, the “likelihood weight ratio”, “pendant branch length” and “distal branch length” reported by RAPPAS are equivalent to those reported by EPA and PPLacer algorithms. Elseways, the “likelihood” field contains the sum of the votes received by each placement (see Suppl. File S1 for more details).

2.3 Methods tested

The performance of RAPPAS (v1.00), both in terms of accuracy and computational efficiency, was compared against that of EPA (from RAxML v8.2.11) (Berger and Stamatakis, 2011), PPlacer (v1.1alpha19) (Matsen *et al.*, 2010) and EPA-ng (v0.2.0-beta, with the agreement of its authors) (Barbera *et al.*, 2018).

2.4 Accuracy experiments

The accuracy of the placements produced by RAPPAS were evaluated following established simulation protocols (Matsen *et al.*, 2010; Berger *et al.*, 2011). Briefly, these protocols rely on independent pruning experiments, in which sequences are removed from the tree and placed back to the pruned tree. The pruned sequences are expected to be placed on the branch from which they were pruned. If the observed placement disagrees with the expected one, a topological metric quantifies the distance between them by counting the number of nodes separating the two edges.To make the placement procedure more challenging, we do not only prune randomly chosen tree leaves, but whole subtrees identified by their parent node. We consider this approach more realistic in the light of current metagenomic challenges because it is indeed common that large sections of the Tree Of Life are absent from the reference datasets. We also decided to simulate not only one query per pruned leaf but as many non-overlapping reads as permitted by the RefA length (e.g., a pruned sequence of 500bp results to 3 non-overlapping simulated MSs of 150 bps each).

2.7 Runtime and memory experiments

CPU times were measured for each software with the Unix command ‘time’, using a single CPU core and on the same desktop PC (64Gb of RAM, Intel i7 3.6GHz, SSD storage). The whole process time was measured: for EPA, EPA-ng and PPlacer this includes the

compulsory alignment of query reads to the reference alignment. In the case of RAPPAS this includes the overhead induced by the initial loading of the pkDB into memory. The program fast aligner 'hmmalign' from the HMMER package (Eddy, 2011) was used to build the profile alignments, as in previous placement studies (Matsen *et al.*, 2010). Placement runtimes were measured for two datasets, D652 which corresponds to a short reference alignment of 2,2kb and D155 which corresponds to a 4-fold longer 8,8kb alignment. Each software was launched independently 3 times and only the shortest of measured CPU time was retained. For EPA, EPA-ng and PPlacer, model parameters were manually defined to avoid any optimization phase prior to the placement. For EPA-RAxML and EPA-ng, a single thread was explicitly assigned (option -T 1). PPlacer implementation only uses a single thread and required no intervention. RAPPAS is based on Java 8+ which uses a minimum of 2 threads. We ensured fair comparison by using the unix controller 'cpulimit', which aligns CPU consumption to the equivalent of a single thread (options --pid process_id --limit 100). Defaults values were used for all other parameters.

2.8 Datasets

Datasets submitted to the procedure evaluating accuracy were aggregated from (Matsen *et al.*, 2010; Berger *et al.*, 2011) to the exception of D155 which was built for this study. Each reference dataset is composed by a refA + RefT built from the RefA. The RefT is assumed to be correct (the tree is well supported). Each dataset is identified by the the number of taxa present in the tree (D150 representing a tree 150 taxa). Basic descriptive metadata describing the nature of the locus, the length of the alignment and the gap content are present in figure 3 (left headers) and Suppl. File S2. Globally, the datasets represent a panel of loci commonly used in metabarcoding and metagenomic studies.

The MS submitted to the runtime experiments are based on two sources. Runtimes measures on the D652 dataset uses real world 16S rRNA bacterial amplicons retrieved from the Earth Microbiome Project (Thompson *et al.*, 2017) via the European Nucleotide Archive (Silvester *et al.*, 2018) and the help of custom scripts from github.com/biocore/emp/ . Only reads with length superior to 100bp were retained (max length 150bp). For the runtime experiments based on the D155 dataset, 10^7 Illumina short reads of 150bp were simulated from the 155 viral genomes composing this dataset using the Mason read simulator (Holtgrewe, 2010) . A SNP rate of 0.1% and very low indel rate of 0.0001% was chosen to avoid generating longer profile alignments which would have impacted negatively the runtime of EPA, EPA-ng and PPlacer.

3. Results

3.1 RAPPAS process

RAPPAS operates PP in a two steps process which can be seen as a “build database once / analyze many dataset” approach (Figure 2-B). Before any phylogenetic placement, one must define a reference dataset which defines the phylogenetic context in which will be placed future query sequences. Typically, in a context of metagenomic analysis and large-scale species identification, these are trees of well-established marker genes (*16S rRNA, cox1* …) or a selection short genomes (organelle genomes, viruses…). The reference tree and alignment are used to build the pkDB of RAPPAS (Figure 2-B), this phase concentrating the heaviest computations. Then, query sequence can be routinely matched to the DB to produce phylogenetic placement reports without the need for pre-alignment of the reads to the reference alignment, as required by previous PP software (Figure 2-A). The placement analysis ends with the production of a standard “jplace” file (Matsen *et al.*, 2012) which contains phylogenetic placement results and allows the exploitation of RAPPAS results in all compatible such as iTol (visualization) Guppy (diversity analyses) or BoSSA (visualisation and diversity analysis) (Letunic and Bork, 2016; Matsen and Evans, 2013; Lefeuvre, 2018).

3.2 Accuracy

The simulation procedure (see methods) was repeated for 10 different datasets representing genes commonly used for species identification. Figure 3 resumes the results of each of these simulations and detailed mean values are reported in Suppl. File S3. The closer to zero are the node distances (ND) measures in the placements experiment, the more accurate is the method. Horizontal lines linking mean ND facilitates the comparison of the method. For all datasets EPA, EPA-ng and PPlacer show very similar performance with an average node distance (ND) below 5 in most cases, which confirms the similarity of their algorithms and demonstrates their high accuracy. A second trend comes from the tested read lengths, the longer are the placed reads the more accurate is the placement (Figure 3, left to right boxes). The worst performances were observed for the D628, D855 datasets which agrees with previous studies (Matsen *et al.*, 2010; Berger *et al.*, 2011).

RAPPAS was submitted to the same tests but using fours k-mer sizes. Globally, similar results are observed: the longer are the placed reads the more accurate is the placement. The corresponding results can nevertheless be classified in two groups. First, are the datasets for which using different values of *k* has a limited impact (Figure 3, D140, D150, D500, D714, D855, D1604). In those, RAPPAS shows globally an accuracy which is

comparable to previous software and shows mean ND values below 5 in most cases. Note that the corresponding accuracy results are less balanced in RAPPAS than with other software (in both a positive and negative way). In simulations based on the shortest read lengths (Figure 3, leftmost boxes) similar or even slightly better accuracies (leftmost boxes of D150, D152, D714) measured. At the opposite, some of the longest read lengths resulted to slightly lower accuracy than in other methods (rightmost boxes of D150, D500).

The second group contains datasets where different values of *k* resulted to variable accuracy (Figure 3, D218, D628, D652, D155). 3 out of these 4 datasets are also characterized by a high proportion of gaps in their reference alignments (Figure 2, left headers). They all show a multimodal distribution of the measured ND, which is particularly visible in the rightmost boxes of D218 and D628. Finally, D155 which corresponds the longest reference alignment (8,8kb) is the only dataset where only the longer k-mer sizes improved RAPPAS accuracy and where RAPPAS fails to provide an accuracy comparable to EPA and PPlacer when considering the longer read lengths.

3.3 Speed

Figure 4 resumes runtimes measured for the placement of 10^3 to 10^7 reads on two different datasets. To consider the actual time needed for placing a set of metagenomic sequences, tEPA, EPA-ng and PPlacer runtime includes both read alignment and placement itself while in RAPPAS runtime includes the overhead related to the pkDB loading. EPA, PPlacer and EPA-ng show a runtime linear to the number of placed reads with both datasets. With the D155 dataset, EPA and PPlacer failed to place a set of 10^7 reads (too high memory requirements) and PPlacer showed placement runtimes up to 10x faster than EPA while EPA-ng was up to 64x faster than PPlacer and 252x faster than EPA. It is important to note that none of these approaches can produce runtimes faster than the profile alignment itself which is reported via the blue line of Figure 4.

Contrary to previous software, the RAPPAS time of RAPPAS becomes linear only for the largest read datasets. When few ($<=10^4$) reads are placed, the overhead imposed by the pkDB loading has an impact on runtime. At the opposite, for larger read sets ($>10^4$) this overhead becomes negligible and RAPPAS shows a placement runtime which is faster than every other methods. With D652, which is a short RefA (Figure 4, rightmost box), RAPPAS was up to 5x faster than EPA-ng (5x for k=12 and 3x for k=8) and up to 35x faster than PPlacer (35x for k=12 and 20x for k=8). With D155, which is a longer reference alignment (Figure 4, leftmost box), greater speed increases are observed: RAPPAS was 43x faster than EPA-ng (39x for k=8 or 43x for k=12) and 135x times faster than PPlacer. More

concretely, using a single CPU and a common desktop computer, RAPPAS placed one million 100-150 bp reads in 30 minutes on D652 and 11 minutes on D155.

3.4 Implementation

RAPPAS is distributed as a standalone command-line application. It is compatible for any system and infrastructure supporting Java 1.8+. Ancestral sequence reconstruction is performed via an external program of which path is set at RAPPAS execution. Currently Phyml 3.3+ and PAML 4.9+ binaries are fully supported by RAPPAS. The most important requirement for RAPPAS is to access enough memory to build the full pkDB (by analogy, to load the pkDB when operating later placements). Figure 5 reports the memory peaks experienced for three of the tested datasets and different values of k. D150 (few leaves, short alignment), D155 (few leaves, long alignment) and D1604 (many leaves, short alignment) show memory peaks below 16Gb, which confirms that RAPPAS can be launched on most standard desktop computer. Nevertheless, for larger *k* values (>=10) we strongly recommend allocating more memory to avoid unnecessary reallocations and accelerate the process of building the pkDB. GIT repository, instructions to compile RAPPAS and tutorials are available at https://gite.lirmm.fr/linard/RAPPAS.

4. Discussion

Advantages of Alignment-free placements

The innovative algorithm introduced in RAPPAS removes two of the major limitations imposed by previous PP algorithms: i) their reliance on a pre-computed multiple alignment merging both reference alignment and query sequences and ii) the fact that this pre-process has to be repeated for each new set of query sequences (Figure 2-A). The idea of attaching k-mers to pre-computed metadata to accelerate sequence classification was already explored by tools combining taxonomically-informed k-mers and LCA-based classification (Liu *et al.*, 2018; Müller *et al.*, 2017; Wood and Salzberg, 2014; Ounit *et al.*, 2015). But none of these methods were targeting highly accurate classifications such as the one based on phylogenetic inference. By generating sets of phylogenetically-informed k-mers, RAPPAS brings an analog idea to the methodological family of phylogenetic placements and bases the resulting classifications on a sequence space which is statistically-supported by advanced models of evolution. When compared to previous PP methods RAPPAS provides two critical algorithmic properties. First, placement complexity do not depends on the length of the RefA because any alignment can be resumed by its phylo-kmers composition, with an upper bound of $(\#states)^k$possible phylo-kmers. Second, placement runtime only depends

on the number of branches associated to the phylo-kmers queried in the pkDB (not the tree size). This second property has limited impact for small RefT holding very similar taxa or small values of k (k-mers will be assigned to many tree branches) but is highly favorable for large RefT and large values of k (k-mers will be assigned to a low proportion of the branches) . As a consequence, the initial implementation of RAPPAS already shows placements that are faster than previous PP algorithms. The speed gain provided by our alignment-free approach appears to be even more preponderant when longer reference alignments are considered (Figure 4). Finally, while the present implementation is based on a single thread, most steps of RAPPAS algorithm are atomic and can be parallelisable. This property offers many options for reaching even faster placements in the future.

Gaps and akm generation

The presence of dense gappy regions in the RefA have an impact on accuracy, a phenomenon accentuated by longer read lengths. D218, D628 and to a lower extant D652 are the most gappy of the tested datasets and show increasingly worse accuracy (compared to EPA/PPlacer) when *k* increases from 8 to 12 (Figure 3). This limitation may be related to several sources. First, the standard evolutionary models used during ancestral state reconstruction do not account for gaps (standard models in PhyML or PAML don't). Second, when generating phylo-kmers from a highly gappy RefA column a decision between the two following choices needs to be made: i) this position is still taken into account, because posterior probabilities can be computed even if they are based on a limited amount of terminals with an actually known state ii) this column is ignored to build an phylo-kmer which overcomes this likely less informative column. The first choice may generate k-mers that are unrelated to most regions of the RefT but may be necessary to discriminate placements in clades where sequence signal is actually present and informative. The second choice may produce k-mers which are more representative of the reference dataset but may lead to accuracy loss in the said clades. Currently, RAPPAS makes the arbitrary choice of ignoring columns with more than 99.9% of gaps and generate phylo-kmers which "jump" over gappy columns when the input RefA is composed of more than 30% of gaps (these values can be parameterized). In the future, a more controlled approach could be based on the consideration of inexact matches between the query and the pkDB. This solution were already explored in several k-mer based sequence classification tools (Müller *et al.*, 2017; Wood and Salzberg, 2014) via the use of "spaced seeds" techniques, which allow probabilistic matches between k-mers following a particular match template. Alternatively inexact k-mer matches could be statistically evaluated via substitution matrices (at least in

gappy regions) (Horwege *et al.*, 2014). Solutions to adapts these techniques to RAPPAS algorithm should be explored in the future to reduce its sensitivity to gaps.

Accuracy improvement

RAPPAS placements are accurate with state-of-the-art read lengths (150-300bp are common in Illumina kits). However, it tends to produce slightly less accurate placements for longer read lengths (Figure 3). Currently, during DB construction and for a single branch each phylo-kmer can be associated to only one posterior probability (figure 1) which corresponds to the RefA position for which the highest probability was computed (see methods). While this choice was made to reduce memory footprint, longer RefA will likely hold the same k-mers in different positions, a phenomenon which is accentuated for lower *k*s. Currently, this process ignores the fact that different evolutionary constraints can be observed in different regions of the RefA (for instance a conserved and a variable regions) and only one region (the one generating the highest probabilities) will account for the placement score of this query sequence. This design choice may be the reason for the poor accuracy observed in the D155 dataset, which corresponds to a complete viral genome and not a single genomic marker (Figure 3, rightmost box). Considering that long reads sequencing is developing rapidly, future versions of RAPPAS will have to consider the RefA positions from which originated the phylo-kmers. Adding this information will produce larger pkDBs and will require research on its data structure to limit memory footprint while keeping fast runtimes. Advanced solutions for the indexation of taxonomically-informed k-mers were already explored by some LCA-based approaches such as MetaCache (Müller *et al.*, 2017) or MetaOthello (Liu *et al.*, 2018). In particular, the use of k-mer minimizers, which aim to both efficient k-mer storage and very fast database access by considering the redundancy imposed by overlapping k-mers is an improvement that could be explored in future versions of RAPPAS implementation. An other answer could be to limit the database to the most discriminant phylo-kmers, as several k-mers of high frequency in the genomes or marker loci may become a source of noise rather than a source of discriminative phylogenetic signal. Such approaches were already explored by the read classifier CLARCK (Ounit *et al.*, 2015) but so far appears to be less accurate than previous solutions (at least for bacterial genomes, see (Müller *et al.*, 2017)).

Yet, while RAPPAS algorithm opens to many possible optimisations, it is already faster than the most optimized methods of phylogenetic placement and is compatible with the routine analysis of large amounts of metagenomic reads while resulting to placement which are as accurate as previous methods when short reads (corresponding to current Illumina

standards) are considered. The principle behind RAPPAS algorithm is also particularly adapted to the more recent long read sequencing technologies (Lu *et al.*, 2016). They are portable and allow field sequencing, but also virtually provide real-time sequencing data with very short delays between the loading of the sample and the acquisition of the data itself. As a preparatory step, one could easily build a set of akmDBs from reference datasets maintained in standard molecular markers databases such as EukRef (/eukref.org), SILVA (Quast *et al.*, 2013), RDP (Cole *et al.*, 2014), PhytoRef (Decelle *et al.*, 2015) or more specialized reference datasets maintained locally. These databases can then be shared on the sequencing devices allowing RAPPAS to place millions of base pairs in the order of the minute and with limited resources. It needs to be determined how RAPPAS algorithm should evolve for a better handling of longer (and potentially low-quality) query sequences, but its versatile algorithm finally opens the possibility of considering phylogenetic placements as a mean to robust and real-time species identification.

Acknowledgments

We would like to thank Olivier Gascuel for its support, as well as Anna-mieke Vandamme, Krystof Theyis, Pieter Libin and Frédéric Mahé for sharing data and fruitful discussions. BL is also grateful to Vincent Lefort for its everyday advices. BL, KS and FP conceived the study; BL and FP designed the algorithm. BL coded the software ; BL, KS and FP wrote the manuscript.

Funding

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 634650 (Virogenesis.eu) and from the Univ. of Montpellier under the InterLabex NUMEV Cemeb SupArgo (post-doctoral fellowship to BL).

References

Baird,D.J. and Hajibabaei,M. (2012) Biomonitoring 2.0: a new paradigm in ecosystem assessment made possible by next-generation DNA sequencing. *Mol. Ecol.*, **21**, 2039–2044.

Barbera,P. *et al.* (2018) EPA-ng: Massively Parallel Evolutionary Placement of Genetic Sequences.

Berger,S.A. *et al.* (2011) Performance, accuracy, and Web server for evolutionary placement of short sequence reads under maximum likelihood. *Syst. Biol.*, **60**, 291–302.
Berger,S.A. and Stamatakis,A. (2011) Aligning short reads to reference alignments and trees. *Bioinformatics*, **27**, 2068–2075.
Brown,D.G. and Truszkowski,J. (2013) LSHPlace: fast phylogenetic placement using locality-sensitive hashing. *Pac. Symp. Biocomput.*, 310–319.
Butel,M.-J. (2014) Probiotics, gut microbiota and health. *Médecine et Maladies Infectieuses*, **44**, 1–8.
Cole,J.R. *et al.* (2014) Ribosomal Database Project: data and tools for high throughput rRNA analysis. *Nucleic Acids Res.*, **42**, D633–42.
Decelle,J. *et al.* (2015) PhytoREF: a reference database of the plastidial 16S rRNA gene of photosynthetic eukaryotes with curated taxonomy. *Mol. Ecol. Resour.*, **15**, 1435–1445.
Eddy,S.R. (2011) Accelerated Profile HMM Searches. *PLoS Comput. Biol.*, **7**, e1002195.
Filipski,A. *et al.* (2015) Phylogenetic placement of metagenomic reads using the minimum evolution principle. *BMC Genomics*, **16 Suppl 1**, S13.
Gilbert,M.T.P. (2017) Documenting DNA in the dust. *Mol. Ecol.*, **26**, 969–971.
Holtgrewe,M. (2010) Mason – A Read Simulator for Second Generation Sequencing Data.
Horwege,S. *et al.* (2014) Spaced words and kmacs: fast alignment-free sequence comparison based on inexact word matches. *Nucleic Acids Res.*, **42**, W7–11.
Huson,D.H. *et al.* (2007) MEGAN analysis of metagenomic data. *Genome Res.*, **17**, 377–386.
Huson,D.H. *et al.* (2016) MEGAN Community Edition - Interactive Exploration and Analysis of Large-Scale Microbiome Sequencing Data. *PLoS Comput. Biol.*, **12**, e1004957.
Izquierdo-Carrasco,F. *et al.* (2011) Algorithms, data structures, and numerics for likelihood-based phylogenetic inference of huge trees. *BMC Bioinformatics*, **12**, 470.
Lefeuvre,P. (2018) BoSSA: A Bunch of Structure and Sequence Analysis.
Letunic,I. and Bork,P. (2016) Interactive tree of life (iTOL) v3: an online tool for the display and annotation of phylogenetic and other trees. *Nucleic Acids Res.*, **44**, W242–5.
Liu,X. *et al.* (2018) A novel data structure to support ultra-fast taxonomic classification of metagenomic sequences with k-mer signatures. *Bioinformatics*, **34**, 171–178.
Lu,H. *et al.* (2016) Oxford Nanopore MinION Sequencing and Genome Assembly. *Genomics Proteomics Bioinformatics*, **14**, 265–279.
Matsen,F.A. *et al.* (2012) A format for phylogenetic placements. *PLoS One*, **7**, e31009.
Matsen,F.A. *et al.* (2010) pplacer: linear time maximum-likelihood and Bayesian phylogenetic placement of sequences onto a fixed reference tree. *BMC Bioinformatics*, **11**, 538.
Matsen,F.A.,4th and Evans,S.N. (2013) Edge principal components and squash clustering: using the special structure of phylogenetic placement data for sample comparison. *PLoS One*, **8**, e56859.
McCoy,C.O. and Matsen,F.A.,4th (2013) Abundance-weighted phylogenetic diversity measures distinguish microbial community states and are robust to sampling depth. *PeerJ*, **1**, e157.
Müller,A. *et al.* (2017) MetaCache: context-aware classification of metagenomic reads using minhashing. *Bioinformatics*, **33**, 3740–3748.
Ondov,B.D. *et al.* (2016) Mash: fast genome and metagenome distance estimation using MinHash. *Genome Biol.*, **17**, 132.
Ounit,R. *et al.* (2015) CLARK: fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers. *BMC Genomics*, **16**, 236.
Penny,D. (2004) Inferring Phylogenies.—Joseph Felsenstein. 2003. Sinauer Associates, Sunderland, Massachusetts. *Syst. Biol.*, **53**, 669–670.
Quast,C. *et al.* (2013) The SILVA ribosomal RNA gene database project: improved data processing and web-based tools. *Nucleic Acids Res.*, **41**, D590–6.
Sedlar,K. *et al.* (2017) Bioinformatics strategies for taxonomy independent binning and visualization of sequences in shotgun metagenomics. *Comput. Struct. Biotechnol. J.*, **15**, 48–55.

Silvester,N. *et al.* (2018) The European Nucleotide Archive in 2017. *Nucleic Acids Res.*, **46**, D36–D40.
Thompson,L.R. *et al.* (2017) A communal catalogue reveals Earth's multiscale microbial diversity. *Nature*, **551**, 457–463.
Wood,D.E. and Salzberg,S.L. (2014) Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biol.*, **15**, R46.
Yang,Z. *et al.* (1995) A new method of inference of ancestral nucleotide and amino acid sequences. *Genetics*, **141**, 1641–1650.
Zaaijer,S. *et al.* (2016) Using mobile sequencers in an academic classroom. *Elife*, **5**.

Figure legends

Figure 1: Generation of ancestral k-mers

The same procedure is repeated on each branch of the the reference tree. A. One ghost branch ended by a terminal ghost node G is attached to the middle of the original branch b. B. A technique of ancestral state reconstruction will generate a probability table for the sequences states expected at node G. C. Ancestral k-mers are generated from this table and assigned to the branch b.

Figure 2: Comparison between previous PP software and RAPPAS

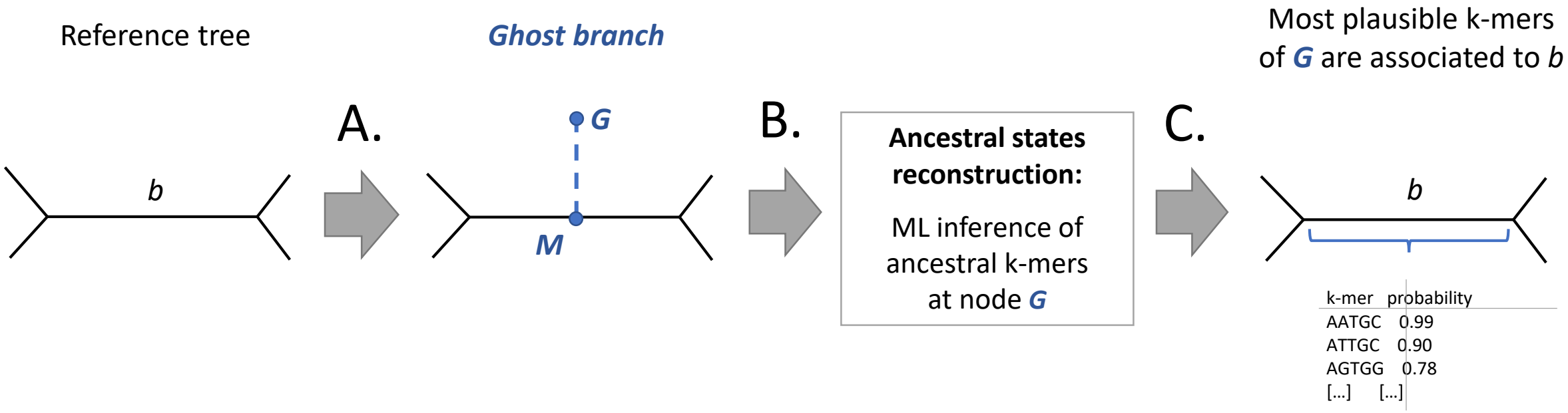
The pipeline from query sequence datasets to placement results (jplace files) is described. A. Previous software required the preliminary step of aligning a dataset to the reference alignment via an external tool (red box) and repeated this step for each new dataset. This extended alignment is the input for the phylogenetic placement itself (blue box). B. In a preliminary step, RAPPAS builds a database of ancestral k-mers once. Then, many datasets can be placed without alignment by matching their k-mer content to this database. Both operations are run in RAPPAS (blue boxes).

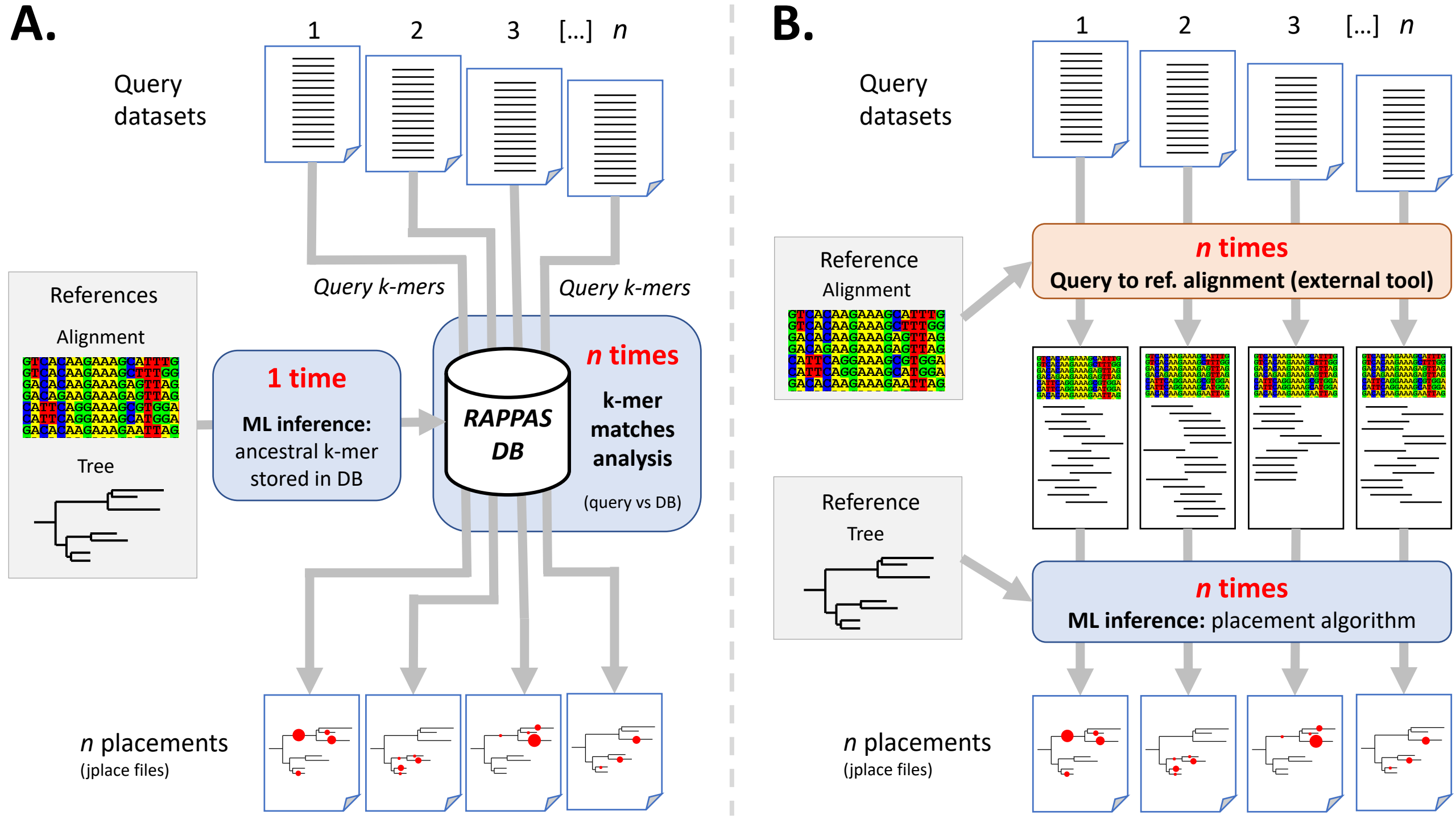
Figure 3: Placement accuracy

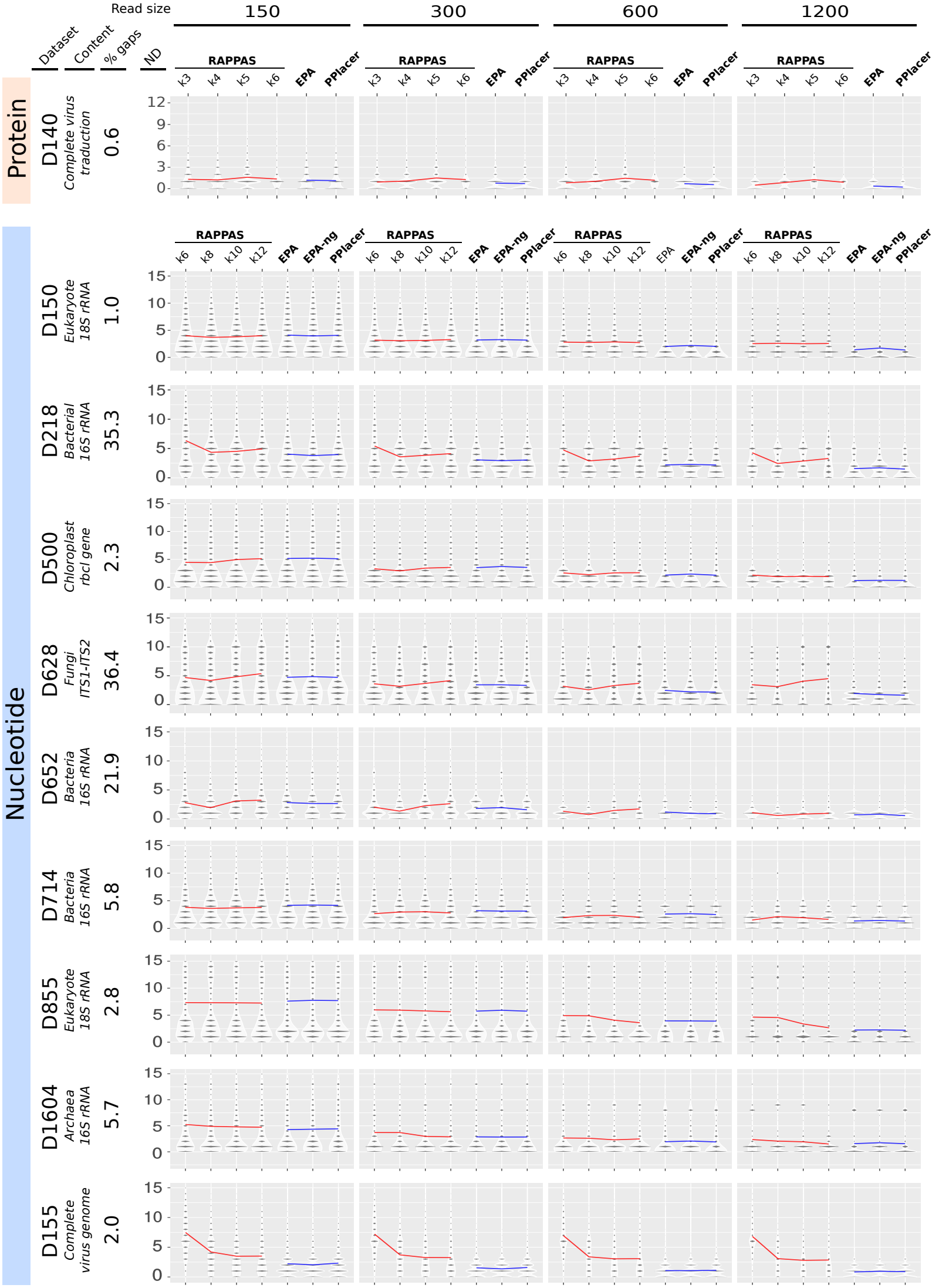
Different dataset corresponding to different marker genes or small genomes (left headers) were submitted to a procedure testing placement accuracy for different read sizes (top header). Each grey box corresponds to a couple of dataset and read length. In each box, the X axis represents the tested software (RAPPAS was tested for several k-mer sizes), while the Y axis corresponds to the measured “node distance” (ND) (a metric of placement accuracy, the closer to 0, the more accurate the placement). White violins and included horizontal grey lines show the distribution of the ND measured for each software. Distribution means are linked by a red (RAPPAS at different ks) or a blue (other software) continuous line.

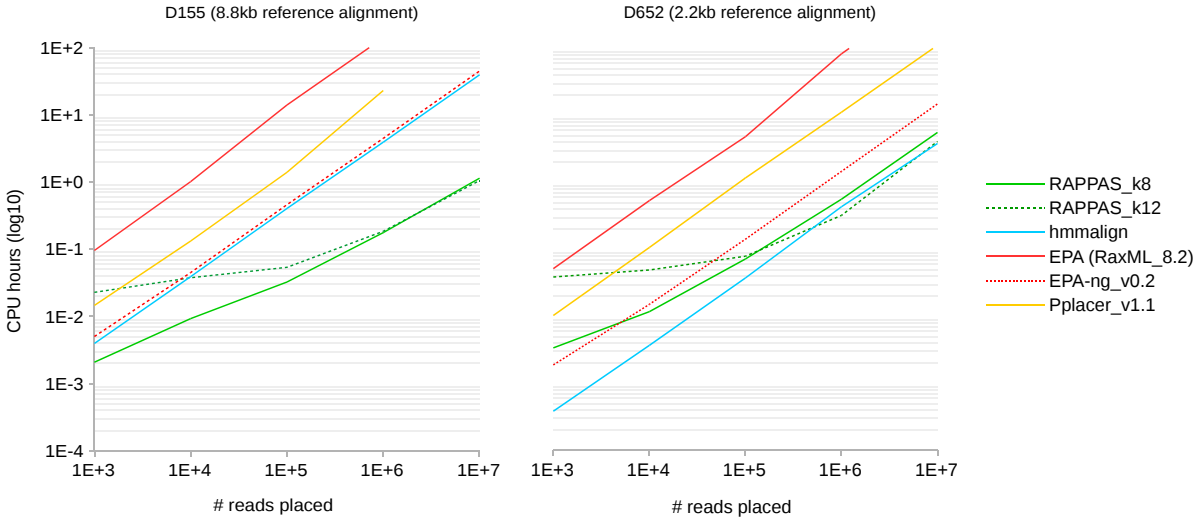
Figure 4: Placement runtime

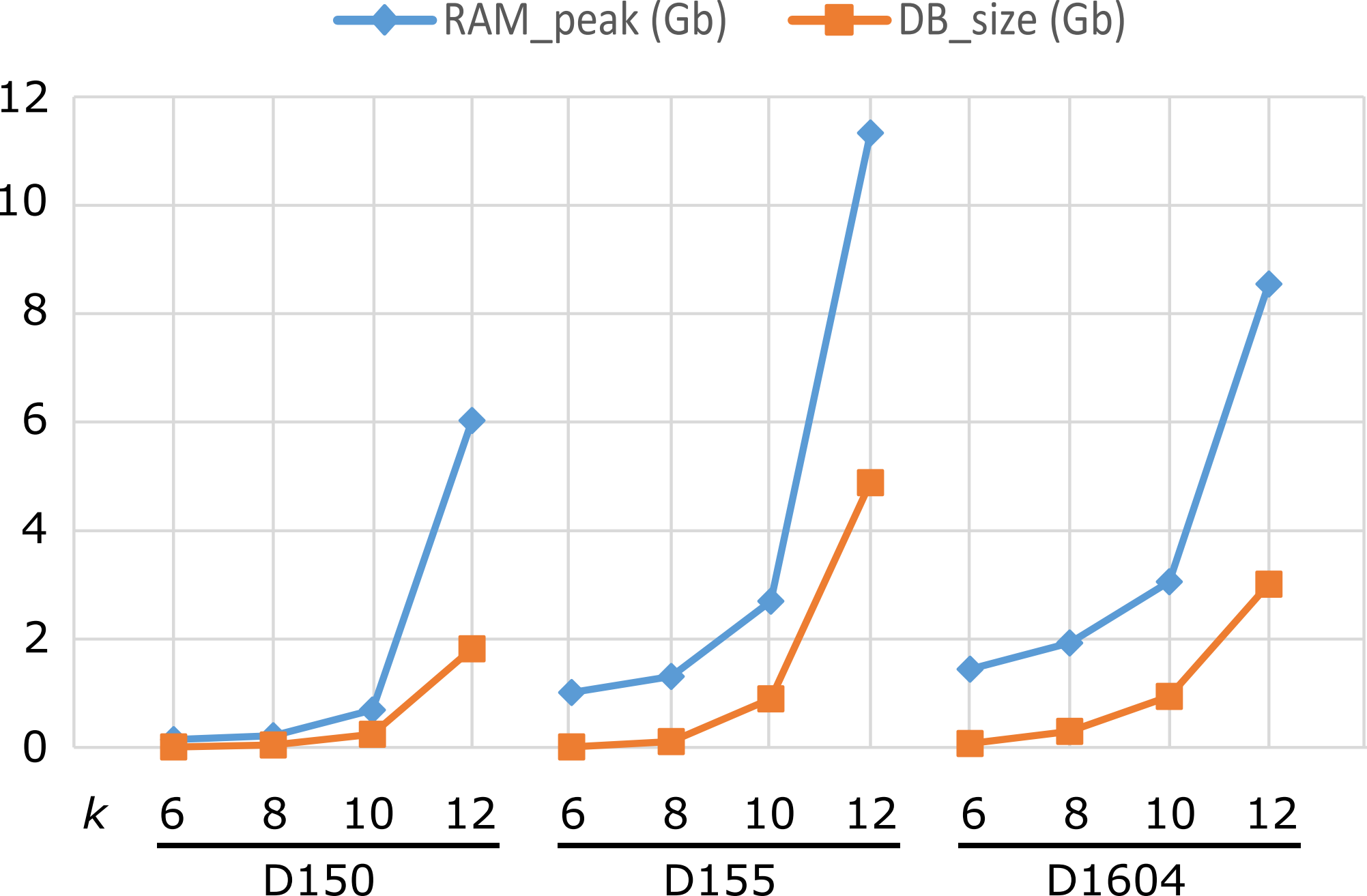
Figure 5: Memory requirement related to akmDB construction


Reference tree
Ghost branch
Most plausible k-mers of G are associated to b
A.
B.
C.
b
G
M
Ancestral states reconstruction:
ML inference of ancestral k-mers at node G
b
k-mer probability
AATGC 0.99
ATTGC 0.90
AGTGG 0.78
[...] [...]


A.
1
2
3
[...]
n
Query datasets
References
Alignment
Tree
1 time
ML inference:
ancestral k-mer stored in DB
Query k-mers
Query k-mers
RAPPAS DB
n times
k-mer matches analysis
(query vs DB)
n placements
(jplace files)
B.
1
2
3
[...]
n
Query datasets
Reference
Alignment
n times
Query to ref. alignment (external tool)
Reference
Tree
n times
ML inference: placement algorithm
n placements
(jplace files)


Read size
150
300
600
1200
Dataset
Content
% gaps
ND
Protein
Nucleotide
RAPPAS
EPA
EPA-ng
PPlacer
k3
k4
k5
k6
k8
k10
k12
D140
Complete virus traduction
0.6
D150
Eukaryote 18S rRNA
1.0
D218
Bacterial 16S rRNA
35.3
D500
Chloroplast rbcl gene
2.3
D628
Fungi ITS1-ITS2
36.4
D652
Bacteria 16S rRNA
21.9
D714
Bacteria 16S rRNA
5.8
D855
Eukaryote 18S rRNA
2.8
D1604
Archaea 16S rRNA
5.7
D155
Complete virus genome
2.0


D155 (8.8kb reference alignment)
D652 (2.2kb reference alignment)
CPU hours (log10)
1E+2
1E+1
1E+0
1E-1
1E-2
1E-3
1E-4
1E+3
1E+4
1E+5
1E+6
1E+7
reads placed
RAPPAS_k8
RAPPAS_k12
hmmalign
EPA (RaxML_8.2)
EPA-ng_v0.2
Pplacer_v1.1


RAM_peak (Gb)
DB_size (Gb)
12
10
8
6
4
2
0
k
6
8
10
12
6
8
10
12
6
8
10
12
D150
D155
D1604