Ultra-large alignments using families of Hidden Markov Models

Nam-phuong Nguyen[1], Siavash Mirarab[1], Keerthana Kumar[1], Sheng Guo[2], Li-San Wang[3], Junhyong Kim[4], and Tandy Warnow[1*]

[1]Department of Computer Science, University of Texas at Austin,
[2]Genomics and Computational Biology Graduate Group, University of Pennsylvania,
[3]Department of Pathology and Laboratory Medicine, University of Pennsylvania,
[4]Department of Biology, University of Pennsylvania,
[*]To whom correspondence should be addressed; E-mail: tandywarnow@gmail.com

Many biological questions, including the estimation of deep evolutionary histories, the resolution of rapid radiations (e.g., the avian and mammalian trees), and the detection of remote homology between protein sequences, rely upon multiple sequence alignments (MSA) and trees of large datasets. However, the challenges in aligning large datasets can discourage biologists from utilizing the full range of biological data. We present UPP, a statistical MSA method that utilizes a new machine learning technique – the Family of Hidden Markov Models. UPP is highly accurate even on ultra-large datasets containing fragmentary sequences. UPP is also very fast, and produced a highly accurate alignment of a 1,000,000-sequence dataset in about two days, using a small number of processors. Thus, UPP enables highly accurate alignments, without the need for supercomputers.

Introduction

Multiple sequence alignment (MSA), gene tree estimation (which depends on an MSA), and species tree estimation (which can depend on the estimation of multiple gene trees (*1, 2*)) are initial steps in many bioinformatics pipelines, with applications to orthology inference (*3*), biomolecular sequence structure and function prediction (*4*), drug target identification (*5*), and the inference and quantification of selection (*6*). Because of the impact of alignment estimation error on these inferences (*7–9*), many methods have been developed to estimate alignments (*10, 11*) and trees (*12*). Multiple sequence alignment of large datasets, containing several thousand to many tens of thousands of sequences, is sometimes necessary; examples include gene family tree estimation for multi-copy genes (e.g., the p450 or 16S genes), viral evolution, remote homology detection, and the inference of deep evolution (*13*); however, current MSA methods have poor accuracy on large datasets, especially when they evolved under high rates of evolution (*14*). These limitations can discourage biologists from utilizing the full range of biological data, and affect downstream inferences. We present UPP (Ultra-large alignments using Phylogeny-aware Profiles), a statistical MSA method that utilizes a new machine learning technique – the Family of Hidden Markov Models – to address these limitations. UPP is designed for multiple sequence alignment estimation for single genes (including multi-copy genes that can appear many times inside a single organism), and produces more accurate alignments on datasets that are large, that span large evolutionary distances, or that contain fragmentary sequences. UPP combines techniques designed for structural alignment prediction and techniques designed for positional homology prediction (two different but related problems (*15, 16*)), and provides excellent accuracy on both phylogenetic and structural benchmarks. Furthermore, UPP is fast and very scalable, producing alignments on 10,000 sequences in under an hour, and on one million sequences in just over two days, using a small number of processors.

UPP addresses the needs of current projects that aim to construct alignments and trees on large datasets, containing several hundred sequences or more. Some of these projects are attempting to assemble ultra-large trees, with many tens of thousands of sequences. For example, iPTOL (*17*) (the iPLANT Tree of Life project) plans to construct a tree on 500,000 plant species, and the Thousand Transcriptome Project (*18*) is constructing gene family trees with more than 100,000 sequences for approximately 1000 species. Large-scale phylogenomic projects like these are enabled by next generation sequencing (NGS) technologies, which have made the generation of sequence data much more affordable (*19*). Upcoming sequencing technologies (*20, 21*) will enable even larger datasets containing sequences from throughout the genomes of many organisms. Ambitious projects, such as the Genome 10K Project (*22*), that plan to estimate species trees with thousands to tens of thousands of organisms, will be able to take advantage of these new data, provided that computational methods are available and able to provide sufficient accuracy on ultra-large datasets.

Efficient maximum likelihood (ML) gene tree estimation for datasets containing thousands (*23*) to tens of thousands (*24*) of sequences is now feasible, but the accuracy of ML trees depends on having accurate multiple sequence alignments (*25*), and estimating highly accurate large-scale alignments is extremely challenging; indeed, some datasets with only 1,000 sequences can be difficult to align well (*26, 27*). This is particularly true for non-coding data, which can evolve under higher rates of evolution than coding data, making alignment estimation difficult (*28*). However, non-coding data can be essential for species tree estimations of rapid radiations; for example, the avian phylogenomics project observed that intron alignments provided substantially higher levels of phylogenetic signal than exon alignments for estimating the avian phylogeny (*29, 30*).

MSA methods have also been evaluated with respect to accuracy defined by shared structural features in proteins, and one approach for estimating alignments uses “templates”, which

are statistical models of the molecule (often Hidden Markov Models (*31*)). Although template-based MSA methods (e.g., (*32–38*)) often perform well under structural alignment criteria, it is not clear how well they will perform when phylogenetic estimation is the objective (*15, 39*). On the other hand, multiple sequence alignment methods designed for phylogenetic estimation (e.g., BAli-Phy (*40, 41*)) tend to be limited to datasets that are small and have low rates of evolution. More generally, studies show that phylogenetic estimation accuracy is typically reduced when alignments are estimated on large datasets that evolve under high rates of evolution (*14, 26*). SATé-I (*26*) and SATé-II (*27*) co-estimate gene sequence alignments and trees, and have better accuracy and scalability than standard multiple sequence alignment methods for large datasets; however, even SATé-II (which improves on SATé-I) is limited to datasets with at most 50,000 sequences. PASTA (*42*) is the most recent addition to the SATé suite of methods, and can analyze datasets with up to 200,000 sequences. However, SATé-I, SATé-II, and PASTA have not been evaluated for structural alignment purposes, nor as protein alignment methods.

Furthermore, many biological datasets contain substantial numbers of fragmentary sequences (SOM (*43*) Section S2.1), resulting in part from incomplete assembly or insufficient transcript sampling. Although some methods (e.g., HMMER (*31, 44*) and MAFFT-Profile (*45*)) can add individual sequences (even short fragments) into existing alignments, MSA methods are not designed to analyze datasets containing a mixture of fragmentary and full-length sequences, and have not been tested under these conditions. Thus, little is known about the accuracy of alignments on datasets of any size that contain fragmentary sequences, nor about the accuracy of trees estimated on such alignments.

Thus, large-scale multiple sequence alignment estimation is a basic step in many problems, including gene tree estimation and protein structure and function prediction, but existing methods have not been shown to provide sufficient accuracy on datasets that are large, that evolve under high rates of evolution, or that contain fragmentary data. The new technique we present,

UPP, addresses these challenges.

UPP: Ultra-large alignment using Phylogeny-aware Profiles. UPP begins with unaligned sequences, and selects a subset (called the "backbone dataset") of the sequences, and the remaining sequences are the "query sequences". UPP preferentially selects the backbone sequences from those that are considered to be "full-length", in order to provide robustness to fragmentary data. UPP uses PASTA to compute an alignment and a tree on the backbone sequences (called the "backbone alignment" and "backbone tree", respectively). The backbone tree is then used to create a collection of subsets of the sequence dataset, with subsets ranging in size from at most ten to up to the full dataset, and all but the smallest subsets contain other subsets (see Fig. 1, Methods, and Section S1.2.4 for additional details). UPP adds each query sequence into the backbone alignment using methods (HMMBUILD, HMMSEARCH, and HMMALIGN) within the HMMER suite of tools, as follows. First, UPP uses HMMBUILD to build profile HMMs, one for each subset alignment (defined to be the backbone alignment restricted to the subset sequences); this is called the "HMM Family". The fit of each query sequence to each HMM is then assessed using HMMSEARCH; this returns a "bit score", which is a measure of the quality of the match between the query sequence and the HMM. The subset HMM with the best bit score is selected, and the sequence is added to the subset alignment using HMMALIGN. By transitivity, this process defines how the query sequence should be added into the backbone alignment. Once all the query sequences are added into the backbone alignment, transitivity defines the final output multiple sequence alignment.

UPP can also be used iteratively. In the first iteration, the UPP alignment is computed, and a maximum likelihood tree is estimated using FastTree. This ML tree is then used to select the backbone dataset for the next iteration, thus ensuring appropriate phylogenetic diversity in the backbone. While this resampling technique is generally beneficial, it is particularly helpful

where there is highly uneven taxon sampling (e.g., a densely sampled in-group and very sparsely sampled distant outgroup), when fragmentary sequences are unevenly distributed throughout the phylogeny, or when sequence lengths changed substantially over evolutionary history. In each of these cases, the technique used to select backbone sequences could lead to backbones that fail to have adequate representation in all the major clades – thus reducing the accuracy of the resultant alignment.

Performance Study

We demonstrate UPP's accuracy on a collection of biological and simulated datasets, in comparison to leading multiple sequence alignments. We compare estimated alignments to reference (true or curated) alignments, and maximum likelihood trees on these alignments to reference trees, and record alignment error and tree error.

Datasets. Because structural alignment and phylogenetic alignment have different purposes and potentially different criteria (*15, 16*), we use both simulated and biological datasets (with structurally-based alignments) to evaluate UPP in comparison to other MSA methods.

The simulated datasets include 1000-sequence nucleotide datasets with average length 1000-1023 from (*26*) that were generated using ROSE (*46*), and used to evaluate SATé in comparison to other MSA methods on large datasets; 10,000-sequence datasets we generated using Indelible (*47*) with average sequence length 1000; and subsets of the million-sequence RNASim (*48*) dataset with average sequence length 1554.5. RNASim is a simulator for RNA sequence evolution that we present here, and that was designed to simulate a complex molecular evolution process using a non-parametric population genetic model that generates long-range statistical dependence and heterogeneous rates. The simulated AA datasets include the 5000-sequence datasets used in (*24*), with average sequence lengths varying from 179.4 to 346.9, and that were

based on proteins from the COG database (*49*).

The biological datasets include the three largest datasets from the Comparative Ribosomal Website (CRW) (*50*), each a set of 16S sequences. We include the 16S.3 dataset (6,323 sequences of average length 1557, spanning three phylogenetic domains), the 16S.T dataset (7,350 sequences of average length 1492, spanning three phylogenetic domains), and the 16S.B.ALL dataset (27,643 sequences of average length 1371.9, spanning the bacteria domain). The CRW datasets have highly reliable, curated alignments inferred from secondary and tertiary structures. We include ten large amino acid datasets (10 AA) with curated multiple sequence alignments (the eight largest BAliBASE datasets (*51*) and IGADBL_100 and coli_epi_100 from (*52*)); these range in size from 320 to 807 sequences and have average sequence lengths that range from 56.7 to 886.3. We also used HomFam datasets, which are amino acid sequence datasets ranging in size from 10,099 to 93,681 sequences, and having average sequence lengths ranging from 29.1 to 469.8; these datasets were used in (*53*) to evaluate protein multiple sequence alignment methods on large datasets, and have Homstrad (*54*) reference alignments on very small subsets of their sequences.

We generated fragmentary datasets by selecting a random subset of sequences and a random substring (of a desired length) for each selected sequence (see Section S1.1.8 for full details). Empirical statistics (number of sequences, number of sites in the reference alignment, average and maximum p-distances, average gap length, and percent of the matrix that is gapped) for each dataset and model condition can be found in Tables S4 and S5 in the SOM.

Methods. We use Clustal-Omega (*53*) version 2.1, MAFFT (*55, 56*) version 6.956b, Muscle (*57, 58*) version 3.8.31, Opal (*59*), PASTA (*42*), SATé-II (*27*), and UPP to compute multiple sequence alignments. we show results for only iteration of UPP in the main paper; see Section **??** for results using more than one iteration. We use FastTree (*24*) and RAxML (*60*) to compute

maximum likelihood trees on estimated and reference alignments.

Performance Metrics. We compare estimated alignments and their maximum likelihood (ML) trees to reference alignments and trees. We use FastSP (*61*) to compute alignment error, recording the sum-of-pairs false negative (SPFN) rate (which is the percentage of the homologous pairs in the reference alignment that are not in the estimated alignment) and the sum-of-pairs false positive (SPFP) rate (which is the percentage of homologous pairs in the estimated alignment that are not present in the reference alignment). SPFN and SPFP rates are given in the SOM, and the means of these two alignment error rates are given in the main paper. We report tree error using the false negative (FN) rate (also known as the missing branch rate), which is the percentage of internal edges in the reference tree that are missing in the estimated tree. We also report ΔFN, the difference between the FN rate of the estimated tree versus the FN rate of the tree estimated on the true alignment, to evaluate the impact of alignment estimation error on phylogenetic analysis. Most typically, $\Delta(FN) > 0$, indicating that the estimated tree has higher error than the ML tree on the true alignment.

Computational resources. The majority of experiments were run on the homogeneous Lonestar cluster at the Texas Advanced Computing Center (TACC). Because of limitations imposed by TACC, these analyses are limited to 24 hours, using 12 cores with 24 GB of memory; methods that failed to complete within 24 hours or terminated with an insufficient memory error message were marked as failures. For experiments on the million-sequence RNASim dataset, we ran the methods on a dedicated machine with 250 GB of main memory and 12 cores and ran until an alignment was generated or the method failed. We also performed a limited number of experiments on TACC with checkpointing, to explore performance when time is not limited.

Results

Initial experiments. We let UPP(Default) denote the default version of UPP in which we use backbones of 1000 sequences, use PASTA to compute the backbone alignment, and add sequences to the backbone alignment using the HMM Family technique. We also explore UPP(Fast), the variant where we use use backbones of 100 sequences but keep the other algorithmic parameters fixed, and "NoDecomp" versions of UPP(Fast) and UPP(Default) to indicate that we use just one HMM instead of a family of HMMs to represent the backbone alignment. We show results for one iteration of UPP.

Since UPP computes its backbone using PASTA, we compared UPP to PASTA, and included a comparison to SATé-II, focusing on the RNASim datasets (Section S2.7). As shown in (*42*), PASTA is more accurate than SATé-II, can analyze larger datasets, and is computationally more efficient than SATé-II. A comparison between UPP(Default), SATé-II, and PASTA shows that UPP(Default) typically had the lowest alignment error rates (Figures S33-S40) and was much more robust to fragmentation (Figures **??** and S41). UPP produced more accurate trees than SATé-II (Figure S34). PASTA had a small advantage over UPP with respect to tree estimation on datasets without fragments (Figure S34), but was less accurate than UPP on datasets with fragments (Figures **??** and S42). UPP(Fast) was also able to analyze larger datasets than PASTA and SATé-II: the million-sequence RNASim dataset was analyzed by UPP(Fast, NoDecomp) in 52 hours, but PASTA failed to complete on the dataset and SATé-II failed to complete on even the 100K RNASim dataset (Section S1.3). We focus the remainder of the discussion on Clustal-Omega, Muscle, MAFFT, and UPP; results for the full set of methods can be found in Section S2.9.

Phylogenetic alignment accuracy

We begin by evaluating UPP for phylogenetic estimation purposes. We use simulated datasets, since these provide true alignments and true trees, and thus allow us to exactly quantify error in identifying true positional homology (i.e., descent from a common ancestor).

Alignment estimation error on RNASim datasets. We examined performance on the RNASim datasets with up to 200,000 sequences, using UPP(Fast) to reduce running times (results obtained using backbones of 1000 sequences showed improved accuracy but took longer; see Table 1). We compare UPP(Fast) to MAFFT (default MAFFT on the 10K dataset, MAFFT-PartTree on larger datasets), Clustal-Omega, and Muscle (Fig. 2(a)). Default MAFFT produced less accurate alignments than MAFFT-PartTree on the RNASim 10K dataset (Fig. S31), and failed to complete on the 50K and larger datasets (Section S1.3).

UPP(Fast) succeeded in analyzing all the datasets within the 24 hour time limit, and MAFFT-PartTree succeeded in analyzing the datasets with up to 100K sequences; however, the other methods failed to align RNASim datasets with more than 10K sequences. Muscle failed because it required more than 24 GB of memory on these larger datasets, and Clustal-Omega failed to return an alignment but without giving an error message (see Section S1.3 for details). On the RNASim 10K dataset (Figure 2(a)), the error rates were 13.3% for UPP(Fast), 34.9% for Clustal-Omega, 39.0% for MAFFT-PartTree, and 64.6% for Muscle. UPP(Fast)'s alignment error rate was quite stable across all numbers of sequences (up to 200,000), varying between 12.5-13.3%.

We analyzed the million-sequence RNASim dataset using UPP(Fast), UPP(Fast,NoDecomp), and UPP(Default,NoDecomp), using a dedicated machine, allowing the analysis to exceed the 24 hour time limit in TACC. Both UPP(Fast, NoDecomp) and UPP(Default, NoDecomp) completed in less than three days and produced very accurate alignments (13.0% and 11.1% align-

ment error, respectively; see Table S3). UPP(Fast) took 12 days to align this dataset, and produced a slightly more accurate alignment than UPP(Fast, NoDecomp) (alignment error 12.8%).

Results on the Indelible NT simulated datasets. The 10,000-sequence Indelible simulated datasets evolved under low (10000M4), moderate (10000M3), or high (10000M2) rates of evolution. The difficulty in estimating alignments increased with the rate of evolution; therefore, we refer to these model conditions as easy (10000M4), medium (10000M3), and hard (10000M2). We ran UPP(Default), MAFFT-PartTree, Muscle, and Clustal-Omega on ten replicates for each model condition.

UPP had very low average alignment error across all three model conditions: 3.3%, 1.3%, and 0.1% on the hard, medium, and easy model conditions, respectively (Fig. 2(b)). The accuracy of the other methods, however, degraded rapidly with the increase in the rate of evolution. For example, under the hard model condition, Muscle had 99.5% average alignment error, MAFFT-PartTree had 98.3% error and Clustal-Omega failed to generate an alignment (Figure 2(b)). Under the medium model condition, MAFFT-PartTree had 24.4% error, Clustal-Omega had 65.6% error, and Muscle had 87.6% error (Figure 2(b)). Finally, under the easy model condition, MAFFT-PartTree, Muscle and UPP all had very low error (below 0.4%), and Clustal-Omega had 3.4% error (Figure 2(b)). Figures S50 and S51 provide additional results.

Results on simulated AA datasets with 5000 sequences. On the 5000-sequence simulated amino acid datasets, UPP had very low error (2.9%), MAFFT-Default had 4.9%, Muscle had 5.5%, and Clustal-Omega had 6.5% (Figure 2(b)). See Figures S58-S59 for full results.

Results on 1000-sequence simulated nucleotide datasets. The nine 1000-sequence model conditions studied in (*26, 27*) varied in gap length distribution and overall difficulty (as influenced by the relative frequency of insertions and deletions (indels) to substitutions, and rate

of evolution). Although there is sequence length heterogeneity in these datasets, all sequences fall within the range considered "full-length"; therefore, because these datasets have only 1000 sequences, UPP(Default) is identical to PASTA on these data. We were able to run Opal and MAFFT-L-INS-i (the most accurate version of MAFFT) in addition to Clustal-Omega, Muscle, and UPP(Default). Error rates varied across the model conditions, but the relative performance of methods was fairly stable: UPP(Default) and Opal had the lowest alignment error rates (with UPP(Default) more accurate than Opal under all models except those with the lowest rate of evolution), Muscle and MAFFT-L-INS-i were typically close in error (with about twice as much error as UPP(Default) and Opal), and Clustal-Omega had the highest error (Figure S47). As an example, on 1000M3, one of the easiest model conditions, UPP(Default) and Opal had the lowest alignment errors (5.6% and 5.4%, difference not statistically significant), followed by MAFFT-L-INS-i (14.1%), Muscle (15.1%), and finally Clustal-Omega (34.3%). On 1000M1, one of the hardest model conditions, UPP(Default) had 19.9% error, followed by Opal with 25.1%, MAFFT-L-INS-i with 52.2%, Muscle with 52.5%, and finally by Clustal-Omega with 91.8%. Figure S48 provides results for SPFN and SPFP errors on these data.

Impact of MSA estimation technique on phylogenetic tree estimation. We next evaluated the impact of MSA estimation on phylogenetic tree estimation. We show results for three of the 1000-sequence model conditions, each with medium gap lengths, and varying the rate of evolution. Under relatively low rates of evolution (1000M3), error rates were generally low, but under moderate (1000M2) to high (1000M1) rates of evolution, the tree error rates increased for most methods (Figure S49). For example, on the hardest model condition (1000M1), the ΔFN error rates were 4.2% for UPP(Default), 15.4% for MAFFT-L-INS-i, 20.5% for Opal, 26.5% for Muscle, and 52.0% for Clustal-Omega. At the other extreme, on a very easy model condition (1000M3), ΔFN error rates were generally good: 0.2% for UPP(Default), 1.7% for

MAFFT-L-INS-i, 1.8% for Opal, and 3.7% for Muscle; only Clustal-Omega did poorly under this model condition (11.4% ΔFN).

Most methods do well on the 5000-sequence simulated AA datasets, except for Opal, which failed to align the COG438 dataset (terminated early due to memory error) and had high ΔFN (12.2%) on the remaining datasets; comparing the other methods, UPP(Default) had the most accurate results (1.9% ΔFN), followed by Muscle with 3.1% and Clustal-Omega with 4.1% (Figure S60).

Performance on the Indelible and RNASim datasets with 10,000 sequences (Figure 5) show that UPP(Default) had very low FN error, within 0.7% tree error of ML on the true alignment, even on the hardest Indelible datasets. With the exception of the easiest Indelible model conditions where all methods perform equally well (within 0.4% tree error of ML on tree alignment), the other methods produced significantly less accurate trees. For example, on the Indelible 10000M2 model, UPP had 9.5% tree error while Muscle and MAFFT had 67.4% and 69.4% tree error, and Clustal-Omega failed to generate an alignment (see Section S1.3).

We computed maximum likelihood trees using FastTree on three UPP alignments (UPP(Fast), UPP(Fast,NoDecomp), and UPP(Default,NoDecomp)) of the million-sequence RNASim dataset and FN tree error was still very low: 8.4% for UPP(Fast,NoDecomp), 7.7% for UPP(Default,NoDecomp), and 7.5% for UPP(Fast), all coming within 2-2.8% of the tree error of FastTree on the true alignment. The phylogenetic accuracy of these trees is noteworthy, given that the sequences were not particularly long (1500 sites, on average), indicating not only the quality of the sequence alignment produced by UPP(Fast) and UPP(Fast,NoDecomop), but FastTree's ability to produce reasonable results on extremely large datasets.

We used the RNASim datasets to explore the impact of increased taxon sampling, which is expected to improve phylogenetic accuracy (*13*). As expected, tree error was reduced with increased taxon sampling when using true alignments: maximum likelihood trees had missing

branch rates of 10.6%, 8.1%, 6.9%, and 6.1% on the RNASim 10K, 50K, 100K, and 200K datasets, respectively. We then tested this on the UPP alignments, to see if the beneficial impact held for alignments estimated using UPP. Maximum likelihood trees computed on UPP(Fast) alignments also reduced in error with increasing numbers of taxa: UPP(Fast) trees had 11.8% FN error at 10K sequences, 9.4% at 50K sequences, 8.3% at 100K sequences, 7.6% at 200K sequences, and 7.5% at 1,000,000 sequences. Thus, UPP alignments are good enough to show the beneficial impact of increased taxon sampling on phylogenetic accuracy (similar patterns hold for other UPP variants, see Table 1).

Structural alignment accuracy

We used biological datasets with structurally-defined reference alignments to evaluate UPP with respect to structural alignment accuracy. On the ten amino-acid datasets (10 AA) with full alignments, Muscle had the highest average alignment error (30.2%) and the other methods (MAFFT-L-INS-i, Opal, and UPP) have very close error rates between 23.5% and 24.3% (Figure 2(c); see also Figures S55 and S56 for additional results). The 19 HomFam datasets and three CRW datasts are too large for MAFFT-L-INS-i or Opal, and so we use MAFFT-Default (or MAFFT-PartTree on CRW 16S.B.ALL) on these data. On the 19 HomFam datasets, Muscle failed to align two datasets, and had generally very high error on those it could align; the other methods succeeded in aligning all the datasets. Comparing methods on just the 17 datasets that Muscle succeeded in aligning, UPP had 22.5% alignment error, followed by MAFFT-Default with 25.3% error, Clustal-Omega with 27.7% error, and Muscle with 48.1% average error (Figure 2(c); see also Figures S53 and S54 for additional results). On the 16S.B.ALL CRW dataset, MAFFT-default failed to run (see Section S1.3), and so we report MAFFT-default only for 16S.3 and 16S.T, and use MAFFT-PartTree for 16S.B.ALL. UPP had the lowest average alignment error across these three datasets (16.3%), MAFFT had 28.8%, Muscle had 30.7%, and

Clustal-Omega had 43.3% (Figure 2(c)). Overall, UPP had the the best or close to the best results on these biological datasets, showing that UPP produced excellent alignments according to structural benchmarks on both nucleotides and amino acid sequences.

Results on fragmentary datasets.

Figure 4 shows results on fragmentary versions of the 1000M2 simulated datasets and the CRW 16S.T biological dataset, varying the percentage of fragmentary sequences from 0% to 50%, with average fragment length 500 (roughly half the length of the full-length 1000M2 sequences and one third the length of the full-length 16S sequence). UPP(Default) had substantially lower error than the other methods, at all levels of fragmentation. In most cases, alignment error increased as the amount of fragmentary data increases, but methods differed in their responses. Muscle was the most impacted by the amount of fragmentary data, with very large increases in alignment error as the amount of fragmentary data increases. MAFFT-default was also impacted, but not as severely as Muscle. UPP and Clustal-Omega were largely unaffected by fragmentation on these data (with error rates that change only in small ways); however, Clustal-Omega had poor accuracy consistently, while UPP had consistently good accuracy. Interestingly, the relative performance of methods changed with the amount of fragmentation; for example, Muscle was more accurate than Clustal-Omega on the 16S.T dataset before we introduce fragmentation, but less accurate when 12.5% of the sequences were fragmentary. Differences between methods were reduced on model conditions with lower rates of evolution, but UPP(Default) still demonstrated greater robustness to fragmentary data than the other methods (SOM S2.11).

Phylogenetic accuracy was also impacted by fragmentary data, but responses varied by the alignment method. Results on the RNASim 10K datasets (Figure 6) with fragmentation varying from 0% to 50%, and all fragments of length 500 (i.e., about one third of the average

length of the full-length sequences) show that UPP(Default) and MAFFT-default were both highly robust to fragmentary data (ΔFN error rates only changing by 3% for UPP and 2% for MAFFT-default). In contrast, tree errors for Clustal-Omega and Muscle were very impacted by fragmentation. Muscle had 7.3% ΔFN on full-length sequences, and then 35.6-49.0% ΔFN under all the fragmentary conditions. Clustal had 9.1% ΔFN on full-length sequences, and then 25.4%-25.8% on the fragmentary conditions. Furthermore, while both UPP(Default) and MAFFT-default are highly robust to fragmentary data, UPP(Default) has better accuracy under all levels of fragmentation on this model condition: 0.8% ΔFN on full-length sequences, and at most 3.9% ΔFN even when half the sequences are fragmentary. MAFFT-default had 5.9% ΔFN for full-length sequences, and ΔFN between 5.8% and 7.1% on the fragmentary sequences.

Figure S64 shows similar trends for the fragmentary 1000M2 and 1000M3 datasets, but MAFFT (run using the most accurate setting, L-INS-i) is not as robust to fragmentation. ΔFN on the fragmentary 1000M2 datasets ranged for UPP(Default) ranging from 2.5-4.7%, from 34.7-65.0% for Muscle, and from 55.8-70.8% for Clustal-Omega (Figure S64). Results on fragmentary versions of the 1000M3 model condition, which has a lower rate of evolution than 1000M2, show ΔFN for UPP(Default) ranging from 0.2-2.2%, while ΔFN ranges from 12.7-27.8% for Muscle and from 18.8-57.2% for Clustal-Omega. MAFFT-L-INS-i shows somewhat better tolerance to fragmentary data than Clustal-Omega or Muscle, but still has high error rates: 18.1-43.5% error on the 1000M2 model condition, and 4.0-14.1% for 1000M3.

Running time.

Running times for UPP(Fast) on the RNASim datsets with up to 200,000 sequences, using 12 processors, show a close to linear trend, so that UPP(Fast) completes on 10K sequences in 55 minutes, on 50K sequences in 4.2 hours, on 100K sequences in about 8.5 hours, and on 200K sequences in about 17.8 hours (Figure 7). Table 1 explores the trade-off between

running time and accuracy (both alignment and tree) of UPP variants. For example, using UPP(Fast) instead of UPP(Default) reduces the running time substantially (by a factor of 7 to 10) and only produces a small increase in tree error and alignment error. However, UPP is extremely parallelizable, and so speed-ups are easily achieved through increasing the number of processors.

Comparison to previous work.

MSA estimation for the purpose of structure and function prediction of protein sequences (and in some cases of RNA sequences) is often based on machine learning models such as templates or profile Hidden Markov Models (HMMs) (*31*). These models are used to represent a seed alignment (typically a curated structural alignment) for the molecule, which is then coupled with methods, such as HMMALIGN (*44*) and MAFFT-Profile (*45*), to add the input sequences into the seed alignment and produce a final multiple sequence alignment. Similarly, profile HMMs for different protein families and superfamilies are provided in databases such as Pfam (*62*) and PhyloFacts (*63*), and can be used to build multiple sequence alignments. However, these "template-based" methods (e.g., (*32–34*)) can only be used to align sequences that match the seed alignment, and so are not *de novo* MSA methods. PROMALS (*37*), SATCHMO (*35*), and SATCHMO-JS (*36*) are protein MSA methods that build HMMs during the course of the alignment estimation, and so can be used in *de novo* alignment; unfortunately, these methods are computationally intensive because they perform a sequence of HMM-HMM alignments to construct their final alignment, and so cannot be used on large datasets.

Phylogenetic placement is a related problem where profile HMMs have also been used: the input is a set of query sequences and a reference tree and alignment, and the objective is to find an edge in the reference tree for each query sequence. SEPP (*64*) is a method for phylogenetic placement that decomposes the reference tree into ten disjoint subsets of sequences

of approximately the same size, builds a profile HMM on each subset alignment, aligns each query sequence to the backbone alignment using the best scoring profile HMM, and then finds the best location in the reference tree using pplacer (*65*), a method that optimizes the position of the query sequence into the reference tree under the maximum likelihood criterion. We modified SEPP so that it could be used as a multiple sequence alignment method, using the backbone sequence alignment and tree computed by UPP as the reference alignment and tree. Alignment error was not substantially different for UPP and SEPP (Figures S28 and S26); however, trees computed using UPP on the 16S.3 and 16S.T CRW datasets were more accurate than trees computed using SEPP (Figure S29). For example, on 16S.3, SEPP(Default,10%) had 7.2% ΔFN and UPP(Default) had 4.8%. On 16S.T, SEPP(Default,10%) had 15.2% ΔFN and UPP(Default) had 6.5% ΔFN. UPP was also more robust to fragmentary data than SEPP (Figure S30). Thus, UPP and SEPP had similar accuracy on fragment-free datasets with at most moderate rates of evolution, but under other conditions UPP produced more accurate trees than SEPP.

Discussion

Although the relative performance of multiple sequence alignments varied by datasets, UPP in most cases showed improved alignment accuracy compared to PASTA, SATé-II, Clustal-Omega, Muscle, and MAFFT. By design, UPP(Default) is identical to PASTA on datasets without fragments and at most 1000 sequences, but UPP is highly robust to fragmentary data whereas PASTA is not. On larger datasets, UPP alignments tend to be more accurate than PASTA alignments, but trees based on PASTA alignments (for fragment-free datasets) are typically more accurate than trees based on UPP alignments. However, trees estimated on UPP alignments are typically more accurate than trees based on all other methods (including SATé-II). Moreover, for datasets with fragmentary sequences, UPP provided the best alignment and

tree accuracy of all the methods we tested. Finally, UPP was the only method we tested that was able to analyze the million sequence RNASim dataset.

UPP exhibits great scalability, both with respect to running time (which scales in a nearly linear manner) and parallelism, but also with respect to alignment accuracy. For example, our study showed that the alignment error on the backbone alignment is typically quite close to the alignment error on the alignment returned by UPP(Default). This close relationship between the accuracy of the backbone alignment and the final alignment is weaker for small backbones (but still present), and also weaker when we use a single HMM or MAFFT-Profile instead of an HMM Family to align query sequences. Thus, the HMM Family technique is a key algorithmic technique to providing scalability for alignment accuracy, so that large datasets can be aligned nearly as accurately as smaller datasets.

However, the other algorithmic techniques also contribute to UPP's improved accuracy. Restricting the backbone to full-length sequences improves the robustness to fragmentary sequences, and the re-sampling technique improves the close relationship between the backbone alignment accuracy and the final alignment accuracy. Using PASTA for the backbone alignment gives better results than using less accurate alignment methods, and because PASTA is computationally efficient it also makes it feasible to use large backbones. Thus, the different algorithmic steps work together to provide the improved accuracy, scalability, and robustness to fragmentary sequences. Furthermore, while good accuracy with respect to structural benchmarks was achieved using a simpler version of UPP (in particular, using MAFFT-L-INS-i instead of PASTA for the backbone alignment and a single HMM rather than the HMM Family Technique to align query sequences), the best accuracy was obtained using our default setting, which also gave better results on the phylogenetic benchmarks. Thus, UPP has excellent accuracy with respect to both phylogenetic and structural benchmarks, indicating that alignments produced by UPP are highly accurate with respect to positional homology and also structural

homology (see (*16*) for further discussion of these related but different concepts).

By design, UPP is a highly modular algorithm, and substitutions in its algorithmic steps could lead to additional improvements. Because our results show that using small backbones reduces accuracy only slightly, this opens the possibility of using sophisticated but computationally intensive multiple sequence alignment methods (for example, statistical methods based on stochastic models of sequence evolution involving indels (*40, 66*)) to produce the backbone alignment. The HMM Family technique is another part of this pipeline that could be improved, for example through using new techniques to compute HMMs (which might incorporate structural information) or to add query sequences to alignments (another active area of research). Thus, UPP is an algorithmic paradigm rather than a specific method, and future work will explore the design space enabled by this paradigm.

In summary, UPP enables highly accurate analyses of sequence datasets that have been considered too difficult to align, including datasets that evolved with high rates of evolution, that contain fragmentary sequences, or that are very large. While datasets like these are increasingly being generated in large-scale sequencing projects, the limited ability to analyze these datasets has discouraged biologists from using the full range of their data. Instead, large-scale transcriptomic and genomics projects often sub-sample from the available data (in terms of taxa, genomic regions, and sites within genes) in order to obtain datasets that are small enough, that evolve sufficiently slowly, and that do not contain fragmentary data, so that available MSA methods can be reliably run on these datasets.

UPP's robustness to fragmentary data, and its high accuracy even for ultra-large datasets with high rates of evolution, increases the range of genomic data that can be used in scientific studies. As a result, scientific questions that would be improved through larger sequence datasets might be able to be addressed with greater accuracy using UPP. A prime case of where UPP could be useful is for phylogeny estimation of rapid radiations or deep evolution, since

phylogeny estimation is often improved by dense taxon sampling (*13*). For example, the avian genome project is planning to eventually sequence all roughly 10,400 living bird species, and such efforts require scalable and accurate alignment techniques such as UPP. However, datasets on smaller numbers of taxa can also include extremely large multi-copy gene families (e.g., the 1KP gene sequence datasets for 1000 species and more than 100,000 sequences). Understanding the evolutionary history of these large gene families requires gene family trees and alignments, that can easily involve many tens of thousands of sequences. Thus, UPP is a tool for both current and future genomics and transcriptomics projects, that will enable biologists to utilize the full range of their data to address biological problems of broad interest.

Methods

Reference alignments and trees. For simulated datasets, the reference alignment is the true alignment (known because we simulate evolution and record the events); for biological datasets, the reference alignment is the curated structural alignment. Reference trees for the simulated datasts are the model trees that generate the data. For the biological datasets, we use RAxML with bootstrapping on the reference alignments to obtain ML trees with branch support, and then we collapse all branches with less than 75% support; this is the same technique used in (*27*) to produce the reference trees on the CRW datasets. The reference trees for the biological datasets are typically incompletely resolved. In this case, the recovery of low support branches in the biological datasets is largely influenced by chance, making the FN rate preferable to the standard bipartition error rate, also called the Robinson-Foulds (RF) (*67*) error rate. FN rates are identical to the RF error rates when estimated and reference trees are fully resolved, and so FN rates are also appropriate for the simulated dataset analyses; hence, we report FN rates for all analyses.

The UPP algorithm. We describe a single iteration of UPP run in default mode; see Section S1.2.4 for additional details. The input to UPP is a set of sequences. In the first step, UPP determines whether the dataset has fragmentary sequences, based on the estimated median length of "full-length" sequences. Any sequence that is shorter than 75% of this median length, or longer than 125% of the median length, is not considered to be full-length, and will not be included in the backbone dataset (except in a "directed sampling" step, as described earlier). Then, a random subset S_0 of 1000 full-length sequences is selected, and a PASTA alignment A and tree T are computed on the subset. (If the number of full-length sequences is less than 1000, then S_0 is the entire set of full-length sequences.) The set S_0 is called the "backbone dataset" and the tree and alignment computed using PASTA on S_0 are called the "backbone alignment" and "backbone tree".

The backbone tree is then used to produce a collection $\mathcal{C}$ of subsets of the backbone dataset, as follows. First, $\mathcal{C}$ is initialized to include S_0. Then an edge in the backbone tree T is found whose removal splits the tree into two subtrees of approximately equal size; this edge is called a "centroid edge", and the leaf sets of the two subtrees produced by removing the centroid edge are added to $\mathcal{C}$. At this point, $\mathcal{C}$ contains three sets: one containing the entire set of backbone sequences, and two others with roughly half the backbone sequences. This process is repeated on every subtree with more than ten leaves. Thus, the collection $\mathcal{C}$ contains a set of subsets of S_0, where the smallest subsets might contain fewer than ten leaves, and where every subset (except for the smallest subsets) contains two other subsets. For example, if the backbone tree contains 1000 leaves, then $\mathcal{C}$ would contain one set of 1000 sequences, two sets of approximately 500 sequences, four sets of approximately 250 sequences, etc., down to some number of sets with ten or fewer sequences.

We then compute the backbone alignment restricted to each subset of sequences in $\mathcal{C}$; these are called the subset alignments. We use HMMBUILD (from the HMMER3 suite of tools) to

build an HMM on each subset alignment, with a match state for each site that has at least one non-gap character (note that this is not the default way of running HMMBUILD). This produces a set $\mathcal{H}$ of profile HMMs, one for every subset alignment, with approximately the same number of states in each HMM (the only condition where different profile HMMs will have different numbers of states are when the subset alignments contain different numbers of all-gap sites).

Each sequence in $S - S_0$ is called a query sequence. We use HMMSEARCH (which takes alignment uncertainty into account) to compute the fit between each query sequence and each profile HMM in $\mathcal{H}$. The HMM with the best fit (defined by the best bit score returned by HMMSEARCH) is selected for the query sequence.

HMMALIGN is then used to add query sequence s to the subset alignment A_s associated to the HMM H_s selected by s. This produces a local alignment of s to A_s (and hence an alignment of $A_s \cup \{s\}$). By transitivity, this defines how to add s into the backbone alignment on S_0, which we call the "extended alignment for s". When the sequence s has a character (nucleotide or amino acid) that is not aligned to anything in the backbone alignment, the extended alignment will have an "insertion site".

Once all the extended alignments are computed, we can merge them all into a single multiple sequence alignment on S. This approach will tend to have potentially many insertion sites, which can be masked out during the tree estimation step to improve speed.

UPP can be used iteratively, but iteration only occurs if the distribution of backbone sequences in a tree estimated on the UPP alignment provides inadequate phylogenetic diversity. Thus, the first step is to determine if all the major clades in the estimated tree contain at least one tenth of the expected number of backbone sequences. If the estimated tree passes this test, no resampling is triggered. Otherwise, UPP uses the tree to select the backbone sequences, ensuring that every major clade contributes appropriately to the backbone sequence dataset. See Sections S1.2.1, S1.2.2, and S1.2.4 for additional details.

References and Notes

1. W. Maddison, *Syst. Bio.* **46**, 523 (1997).

2. S. V. Edwards, *Evolution* **63**, 1 (2009).

3. C. Afrasiabi, B. Samad, D. Dineen, C. Meacham, K. Sjolander, *Nucl. Acids Res.* **41**, W242 (2013).

4. J. A. Eisen, *Genome Research* **8**, 163 (1998).

5. A. Abadio, *et al.*, *BME Genomics* **12** (2011).

6. J. Eisen, C. Fraser, *Science* **300**, 1706 (2003).

7. K. M. Wong, M. A. Suchard, J. P. Huelsenbeck, *Science* **319**, 473 (2008).

8. W. Fletcher, Z. Yang, *Mol Biol Evolution* **27**, 2257 (2010).

9. G. Jordan, N. Goldman, *Mol Biol Evolution* **29**, 1125 (2012).

10. C. Do, K. Katoh, *Methods in Molecular Biology: Functional Proteomics, Methods and Protocols* (Humana Press, 2008), vol. 484, pp. 379–413.

11. D. J. Russell, ed., *Multiple Sequence Alignment Methods*, vol. 1079 of *Methods in Molecular Biology* (Springer, 2014).

12. J. Felsenstein, *Inferring Phylogenies* (Sinauer Associates, Sunderland, Massachusetts, 2003).

13. D. J. Zwickl, D. M. Hillis, *Syst. Biol.* **51**, 588 (2002).

14. K. Liu, C. R. Linder, T. Warnow, *PLoS Currents: Tree of Life* (2010).

15. S. Iantomo, K. Gori, N. Goldman, M. Gil, C. Dessimoz, *Multiple Sequence Alignment Methods* (Springer, 2014), vol. 1079 of *Methods in Molecular Biology*, pp. 59–73.

16. G. R. Reeck, *et al.*, *Cell* **50**, 667 (1987).

17. P. Soltis, D. Soltis, iPToL: the iPLANT Tree of Life Project. https://www.iplantcollaborative.org/challenge/iplant-tree-life.

18. G. K.-S. Wong, One thousand plant transcriptome project (1kp). https://sites.google.com/a/ualberta.ca/onekp/home.

19. D. C. Koboldt, K. M. Steinberg, D. E. Larson, R. K. Wilson, E. R. Mardis, *Cell* **155**, 27 (2013).

20. K.-O. Mutz, A. Heilkenbrinker, M. Lonne, J.-G. Walter, F. Stahl, *Current Opinion in Biotechnology* **24**, 22 (2013).

21. C.-S. Ku, D. H. Roukos, *Expert Rev. Med. Devices* **10**, 1 (2013).

22. D. Haussler, S. O'Brien, O. Ryder, Genome 10K. https://genome10k.soe.ucsc.edu.

23. A. Stamatakis, *Bioinformatics (Oxford, England)* **22**, 2688 (2006).

24. M. N. Price, P. S. Dehal, A. P. Arkin, *PloS One* **5**, e9490 (2010).

25. D. Morrison, *Australian Syst Bot* **19**, 479 (2006).

26. K. Liu, S. Raghavan, S. Nelesen, C. R. Linder, T. Warnow, *Science (New York, N.Y.)* **324**, 1561 (2009).

27. K. Liu, *et al.*, *Syst Biol* **61**, 90 (2012).

28. T. M. Prychitko, W. S. Moore, *Mol Biol Evol* **20**, 762 (2003).

29. S. Mirarab, M. S. Bayzid, B. Boussau, T. Warnow, *Science* (2014). Submitted, companion paper to the Avian Phylogenomics Project paper.

30. E. Jarvis, S. Mirarab, *et al.*, *Science* (2014). Submitted, main paper of the avian phylogenomics project.

31. S. R. Eddy, *Bioinformatics* **14**, 755 (1998).

32. A. Neuwald, *Bioinformatics* **25**, 1869 (2009).

33. E. P. Nawrocki, Structural RNA homology search and alignment using covariance models, Ph.D. dissertation, Washington University in St. Louis (2009).

34. L. Shang, *et al.*, *BMC Systems Biology* **7**, S13 (2013).

35. R. C. Edgar, K. Sjölander, *Bioinf* **19**, 1404 (2003).

36. R. Hagopian, J. Davidson, R. Datta, G. Jarvis, K. Sjölander, *Nucl Acids Res* **38 (Web Server Issue)**, W29 (2010). PMCID: PMC2896197.

37. J. Pei, N. Grishin, *Bioinformatics* **23**, 802 (2007).

38. F. Sievers, *et al.*, *Mol Syst Biol* **7** (2011).

39. D. A. Morrison, *Syst Biol* **58**, 150 (2009).

40. M. A. Suchard, B. D. Redelings, *Bioinformatics* **22**, 2047 (2006).

41. B. Redelings, M. Suchard, *BMC Evol Biol* **7**, 40 (2007).

42. S. Mirarab, N. Nguyen, T. Warnow, *Research in Computational Molecular Biology* (Lecture Notes in Computer Science, Springer, 2014), vol. 8394, pp. 177–191.

43. N. Nguyen, *et al.*, Information on materials and methods is available on Science online (2014).

44. S. Eddy, *Genome Inform* **23**, 205211 (2009).

45. K. Katoh, M. C. Frith, *Bioinformatics (Oxford, England)* **28**, 3144 (2012).

46. J. Stoye, D. Evers, F. Meyer, *Bioinf* **14**, 157 (1998).

47. W. Fletcher, Z. Yang, *Molecular Biology and Evolution* **26**, 1879 (2009).

48. S. Guo, L.-S. Wang, J. Kim, Large-scale simulation of RNA macroevolution by an energy-dependent fitness model (2009). http://kim.bio.upenn.edu/software/rnasim.shtml.

49. R. L. Tatusov, M. Y. Galperin, D. A. Natale, E. V. Koonin, *Nucleic Acids Research* **28**, 33 (2000).

50. J. J. Cannone, *et al.*, *BMC Bioinformatics* **3**, 2 (2002).

51. J. D. Thompson, B. Linard, O. Lecompte, O. Poch, *PloS One* **6**, e18093 (2011).

52. G. B. Gloor, L. C. Martin, L. M. Wahl, S. D. Dunn, *Biochemistry* **44**, 7156 (2005).

53. F. Sievers, *et al.*, *Molecular Systems Biology* **7** (2011).

54. K. Mizuguchi, C. Deane, T. Blundell, J. Overington, *Protein Sci* **7**, 24692471 (1998).

55. K. Katoh, K.-i. Kuma, H. Toh, T. Miyata, *Nucleic Acids Research* **33**, 511 (2005).

56. K. Katoh, H. Toh, *Bioinformatics* **23**, 372 (2007).

57. R. C. Edgar, *BMC Bioinformatics* **5**, 113 (2004).

58. R. C. Edgar, *Nucleic Acids Research* **32**, 1792 (2004).

59. T. J. Wheeler, J. D. Kececioglu, *Bioinformatics (Oxford, England)* **23**, i559 (2007).

60. A. Stamatakis, *Bioinformatics (Oxford, England)* pp. 1–2 (2014).

61. S. Mirarab, T. Warnow, *Bioinformatics* **27**, 3250 (2011).

62. M. Punta, *et al.*, *Nucleic Acids Research* **40**, D290 (2012).

63. N. Krishnamurthy, D. Brown, D. Kirshner, K. Sjolander, *Genome Biology* **7**, R83 (2006).

64. S. Mirarab, N. Nguyen, T. Warnow, *Proceedings of the Pacific Symposium on Biocomputing* pp. 247–58 (2012).

65. F. A. Matsen, R. B. Kodner, E. V. Armbrust, *BMC Bioinformatics* **11**, 538 (2010).

66. A. Bouchard-Côté, M. I. Jordan, *Proceedings of the National Academy of Sciences* **110**, 1160 (2013).

67. D. Robinson, L. Foulds, *Math. Biosci.* **53**, 131 (1981).

68. The authors acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC resources that have contributed to the research results reported within this paper. TW was supported by NSF grant 0733029; SM was supported by a graduate fellowship from the Howard Hughes Medical Institute; SN was supported by the University of Alberta through a grant to TW; LW and SG were supported by NSF grant 0331453; LW was supported in part by T32-HG000046 and NIA U24-AG041689. JK's participation in this project was funded, in part, by a Health Research Formula Fund from the Pennsylvania Department of Health, which disclaims responsibility for any analyses, interpretations or conclusions. The authors thank Erich Jarvis, Tom Gilbert, and Jim Leebens-Mack for their helpful critiques of early versions of the manuscript.

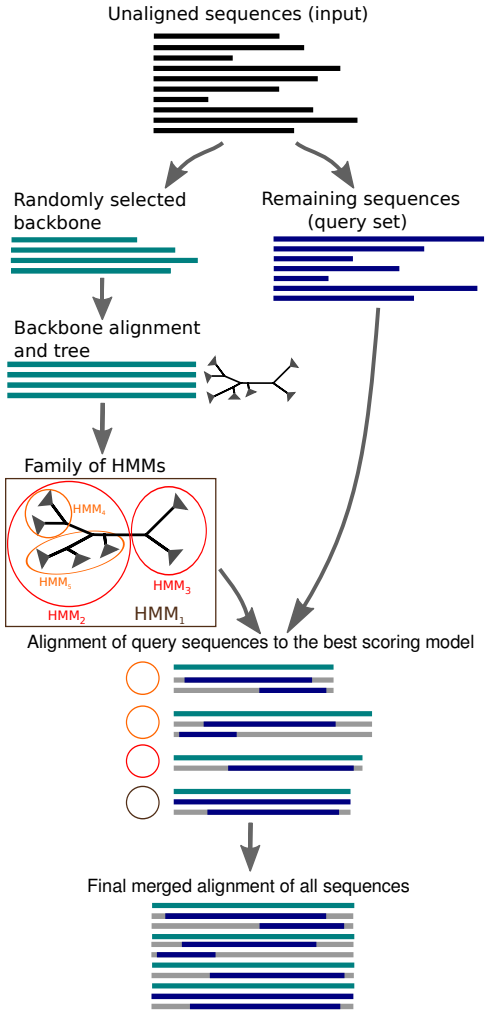
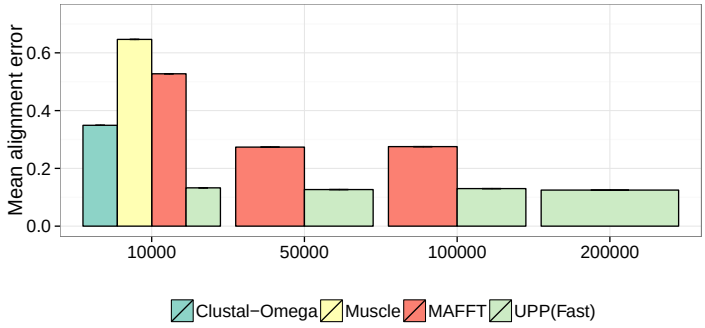
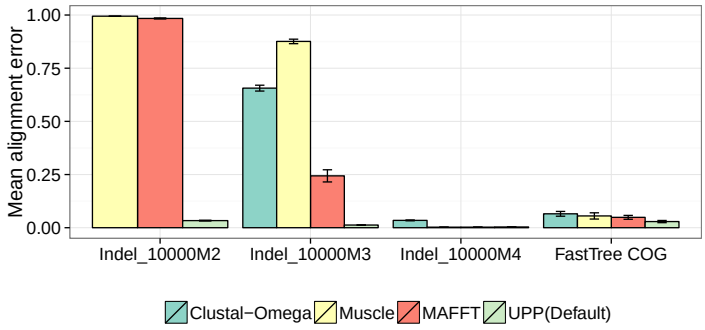


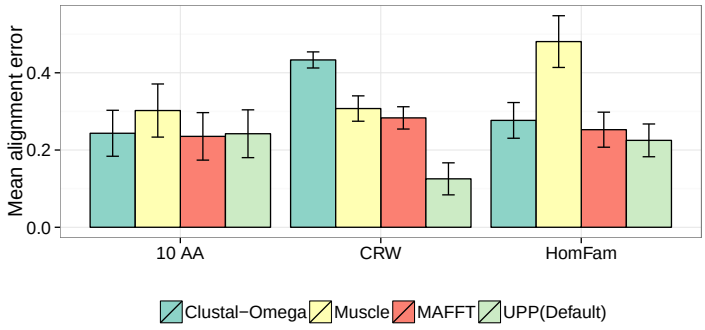
Figure 1: **The UPP algorithm and the HMM Family technique.** UPP uses a pipeline to compute an alignment on the dataset. In the first step, UPP splits the input sequence dataset into a backbone set and a query set. In the second step, an alignment and maximum likelihood tree (called the "backbone alignment and tree") are estimated on the backbone set using PASTA. In the third step, the remaining sequences are aligned using the backbone alignment and tree. UPP performs this step using the HMMER suite of tools to build a family of HMMs (called the "HMM Family") and align sequences using the HMM Family. UPP recursively decomposes the backbone tree into subtrees (using the centroid edge decomposition from SATé (*27*)), stopping the decomposition on any subtree with at most ten leaves. This results in a nested collection of subsets, each with an alignment defined by the backbone alignment restricted to the subset. Profile Hidden Markov Models (HMMs) are constructed on each subset alignment using HMMBUILD with match states for every site that has at least one residue or nucleotide), and each query sequence is aligned to each HMM using HMMSEARCH. UPP then uses HMMALIGN to add the query sequence into the subset alignment associated to the HMM with the largest bit score; this defines how the query sequence is aligned to the backbone alignment.



(a) Alignment error on RNASim datasets with 10K to 200K sequences



(b) Alignment error on other simulated datasets



(c) Alignment error on biological datasets

Figure 2: **Alignment error rates on simulated and biological datasets.** All methods were run with 24 GB of memory and 12 CPUs, and given 24 hours to complete. MAFFT is run using L-INS-i on the 10 large AA datasets, using default MAFFT on the FastTree COG datasets, HomFam datasets, and the CRW 16S.T and 16S.3 datasets, and using the PartTree command for the CRW 16S.B.ALL dataset (default MAFFT failed to align this dataset). Results not shown are due to methods failing to return an alignment within the 24 hour time period on TACC, using 12 processors.

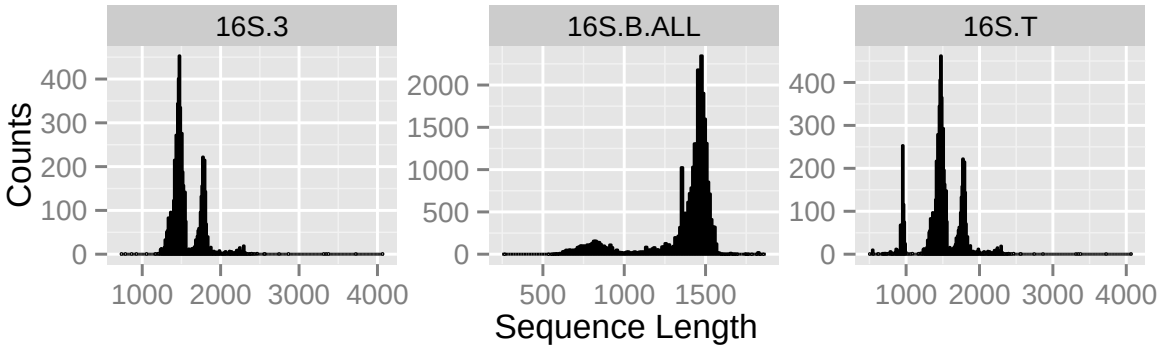
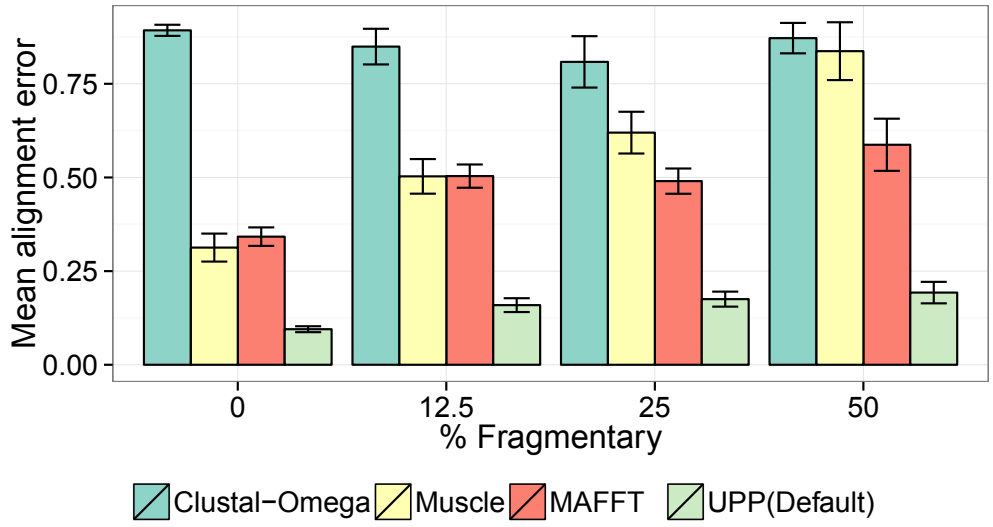
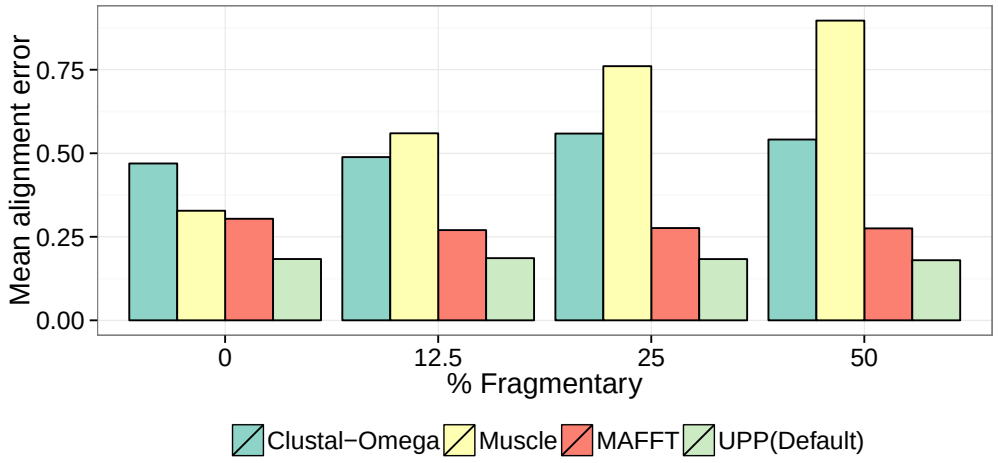


Figure 3: **Histogram of sequence lengths for the 16S Gutell CRW datasets.** The histogram of sequence lengths for the three CRW datasets demonstrates substantial sequence length heterogeneity, especially for 16S.T. The average length of the 16S sequence is approximately 1500.



(a) Fragmentary 1000M2 model conditions



(b) Fragmentary CRW 16S.T datasets

Figure 4: **Impact of fragmentary sequences on alignment error.** We show alignment error rates for different methods on the 1000-sequence 1000M2 datasets and the 7350-sequence CRW 16S.T dataset, but include results where a percentage of the sequences are made fragmentary, varying the percentage from 12.5% to to 50%. Fragmentary sequences have average length 500 (i.e., approximately half the average sequence length for 1000M2, and approximately one third the average sequence length for 16S.T). MAFFT is run using L-INS-i on the 1000M2 datasets and using MAFFT-Default on the 16S.T datasets.

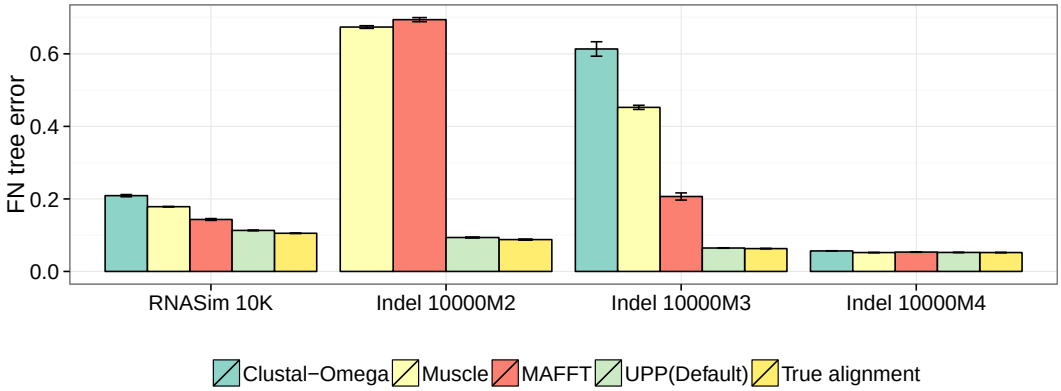


Figure 5: **Tree error on simulated datasets with 10,000 sequences.** We show FN tree error results on the RNASim 10K and Indelible datasets. ML trees were estimated using FastTree under the GTR model. All MSA methods were run with 24 GB of memory and 12 CPUs and given 24 hours to complete. MAFFT was run under the default setting on all datasets. Standard error bars are shown. Averages are computed over 10 replicates per dataset. Clustal-Omega terminated with an error message on the Indelible 10000M2 datasets and thus, results are not shown.

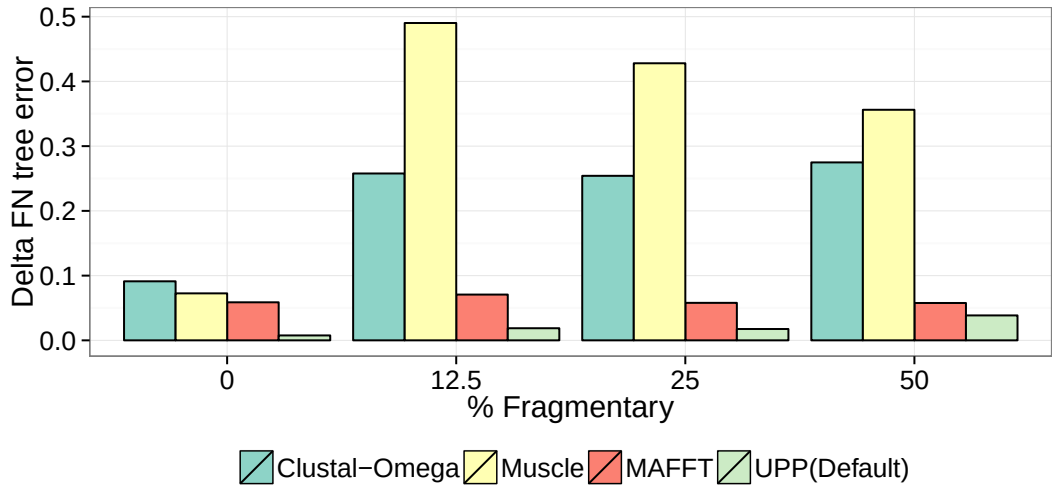


Figure 6: **Impact of fragmentation on tree error for the RNASim 10K datasets.** We show the ΔFN error rates of maximum likelihood trees computed using FastTree under the GTR model on alignments computed using Clustal-Omega, Muscle, MAFFT-Default, and UPP(Default), on the RNASim 10K dataset, including results on versions of the dataset where we make some of the sequences fragmentary. All fragments have average length 500, but we vary the percentage of the dataset that is fragmentary.

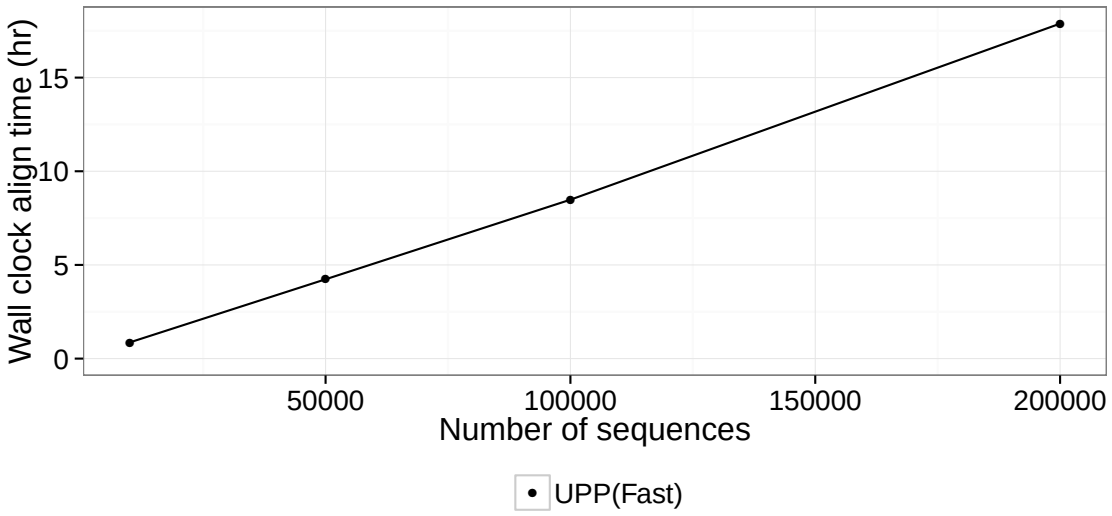


Figure 7: **Running time for UPP(Fast) on the RNASim datasets.** We show running time for UPP(Fast) on RNASim datasets with 10K, 50K, 100K, and 200K sequences. UPP(Fast) uses a backbone of size 100, computes the backbone alignment using PASTA, and then aligns the remaining sequences using the HMM Family technique. All analyses were run on TACC with 24 GB of memory and 12 CPUs.

Table 1: **Results for UPP variants on the RNASim datasets.** We show results for different variants of UPP on the RNASim datasets with 10,000 to 200,000 sequences. We report the average alignment error, ΔFN error (the difference between the error on the true alignment and on the estimated alignment), and running time (in CPU hours), using 12 processors with 24Gb of memory. The default setting for UPP is denoted UPP(Default); it uses a backbone of size 1000, uses PASTA to compute the backbone alignment, and the HMM Family technique; UPP(Fast) is obtained by using backbones of size 100 and keeping all other settings constant. The "NoDecomp" versions of these two methods replace the HMM Family technique with a single HMM. UPP(Default,MAFFT-Profile) uses MAFFT-Profile (with --add or with --addfragments) to add the query sequences into the backbone alignment, and otherwise is identical to UPP(Default); UPP(Fast,MAFFT-Profile) differs from this only by using a backbone of size 100.

Number seq.	Method	Align. error	FN	ΔFN	Time (hrs)
10,000	UPP(Fast,NoDecomp)	13.1%	14.2%	3.6%	0.1
10,000	UPP(Default,NoDecomp)	11.2%	13.6%	3.0%	0.2
10,000	UPP(Fast)	13.3%	11.8%	1.2%	0.9
10,000	UPP(Default)	10.3%	11.4%	0.8%	6.7
10,000	UPP(Fast, Mafft-Profile(add))	26.2%	18.0%	7.4%	0.2
10,000	UPP(Default,Mafft-Profile(add))	14.0%	14.8%	4.2%	0.3
10,000	UPP(Fast, Mafft-Profile(addfragments))	17.8%	15.5%	4.9%	1.0
10,000	UPP(Default, Mafft-Profile(addfragments))	12.7%	12.3%	1.7%	6.5
50,000	UPP(Fast,NoDecomp)	12.2%	10.7%	2.6%	0.4
50,000	UPP(Default,NoDecomp)	12.0%	10.5%	2.5%	0.9
50,000	UPP(Fast)	12.7%	9.4%	1.3%	4.2
50,000	UPP(Default)	11.2%	8.6%	0.5%	44.0
50,000	UPP(Fast, Mafft-Profile(add))	33.6%	13.8%	5.7%	2.1
50,000	UPP(Default, Mafft-Profile(add))	16.0%	10.1%	0.2%	3.5
100,000	UPP(Fast,NoDecomp)	13.5%	9.9%	3.3%	0.8
100,000	UPP(Default,NoDecomp)	11.2%	9.4%	2.8%	1.9
100,000	UPP(Fast)	13.0%	8.3%	1.4%	8.5
100,000	UPP(Default)	11.1%	7.6%	0.7%	82.3
100,000	UPP(Fast,Mafft-Profile(add))	40.2%	10.2%	3.3%	10.7
200,000	UPP(Fast,NoDecomp)	12.4%	8.5%	2.4%	1.9
200,000	UPP(Default,NoDecomp)	11.3%	8.6%	2.4%	6.1
200,000	UPP(Fast)	12.5%	7.6%	1.4%	17.9
200,000	UPP(Default)	10.6%	6.8%	0.7%	151.1