

Copyright
by
Nam-phuong Duc Nguyen
2014

The Dissertation Committee for Nam-phuong Duc Nguyen
certifies that this is the approved version of the following dissertation:

**Family of Hidden Markov Models and its applications
to phylogenetics and metagenomics**

Committee:

Tandy Warnow, Supervisor

Joydeep Ghosh

Raymond Mooney

Keshav Pingali

Mihai Pop

Shibu Yooseph

**Family of Hidden Markov Models and its applications
to phylogenetics and metagenomics**

by

Nam-phuong Duc Nguyen, B.S.;B.S. C.S.;M.S.

DISSERTATION

Presented to the Faculty of the Graduate School of
The University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2014

Dedicated to my parents who risked their lives so that their children may know freedom, to my siblings who helped make Utah a little more diverse, and to Lysa who has supported me every step of the way.

Acknowledgments

My parents left Vietnam so that my siblings and I could have an opportunity of a better life. It is due to their journey that I am able to complete my own journey towards my Ph.D. I am eternally grateful for their guidance in all aspects of life, from the mundane such as how to multiply numbers to the profound such as how to incorporate aspects of Vietnamese and American culture. My dad was originally a medical doctor in Vietnam, and his dream was for one of his children to pick up the mantle. While I may not be an M.D., I hope that earning a Ph.D. will suffice as a repayment for his sacrifices.

I thank my siblings Dan, Danchi, and An. Dan kept me current on sports and movies when I was too busy to go outside my room. Danchi kept me inspired with her stories about the joys of teaching. My little brother An even moved to Austin for a brief stint to keep me on my toes. I thank them for helping me get through the program.

My dissertation would not be possible without the guidance of my adviser, Tandy Warnow. The Ph.D. program can be a grueling experience, and at one of my most stressful moments, I was ready to give up. Tandy saw this and in response gave an inspirational speech to the lab about the nature of research and how frustrating it can be. However, she also encouraged us to not give up hope because through our frustrations, we develop insights into solving

the problem. Those insights lead to new experiments, and these experiments open the floodgate of research. It was after this speech that I began work on SEPP which lead to the main contributions of my dissertation. I thank Tandy for encouraging me to stay through the program.

I feel blessed to have been able to work with the many people in the Warnow lab: Bayzid for the discussions we have about our goals in life, Keerthana for helping me run experiments, Kevin and Shel for their advice on how to survive the program. I also thank Ruth for giving me much needed feedback on my dissertation. I look forward to working with her at UIUC. Finally, I thank Siavash for his keen insight on a multitude of topics. If there was a bug in the pipeline, I could give a brief description and Siavash would have an idea of the cause, and if not, a suggestion on a test to determine the cause. Collaborating with him has been a joy.

I thank my many other collaborators across various research projects. I thank Mihai Pop not only for his work on the development of TIPP, but also for his career advice during my post-doctoral job search. I thank Bo Liu for his help in designing simulated metagenomic experiments for TIPP. I also thank Jim Leebens-Mack, David Nelson, and Gane Ka-Shu Wong for all the knowledge I gained while working together on the 1KP project. I hope to continue these collaborations in the future.

I thank my Ph.D. committee members: Tandy Warnow, Joydeep Ghosh, Raymond Mooney, Keshav Pingali, Mihai Pop, and Shibu Yooseph. Their feedback during my dissertation proposal has been helpful to the devel-

opment of my final dissertation.

I am grateful for the support that Lydia Griffith and Laurie Alvarez have given me. I can't count the number of times that Lydia has helped me navigate the bureaucracy of the program. Without her, I would still be stuck registering for classes. I thank Laurie for being patient with me, even when I forgot to hand in my receipts for reimbursement.

The work in my dissertation covers several projects that were joint work with my collaborators, including Tandy, Siavash, Mihai, Bo, and Keerthana. Their contributions make this dissertation possible.

I thank my friends who helped me stay sane throughout this entire ordeal: Hieu for putting up with my late night guitar jams, Eric for flying out to Austin multiple times to contribute to the late night jams, and Jon for being online whenever I needed to talk. I thank Vy for cooking for me, even when I didn't eat her food. I thank Lisa for helping me proof my papers and providing stimulating discussions about Texas politics. I thank Tri Dang and Yuna Dang for keeping me from being homeless. Finally, I thank Alex, Annie, and Wendy for helping me stay young.

Finally, I thank Lysa for standing by my side throughout this process. I would write more words but as Lysa would say, words are impermanent so there's nothing left to say.

Family of Hidden Markov Models and its applications to phylogenetics and metagenomics

Publication No. _____

Nam-phuong Duc Nguyen, Ph.D.
The University of Texas at Austin, 2014

Supervisor: Tandy Warnow

A Profile Hidden Markov Model (HMM) is a statistical model for representing a multiple sequence alignment (MSA). Profile HMMs are important tools for sequence homology detection and have been used in wide a range of bioinformatics applications including protein structure prediction, remote homology detection, and sequence alignment.

Profile HMM methods result in accurate alignments on datasets with evolutionarily similar sequences; however, I will show that on datasets with evolutionarily divergent sequences, the accuracy of HMM-based methods degrade. My dissertation presents a new statistical model for representing an MSA by using a set of HMMs. The family of HMM (fHMM) approach uses multiple HMMs instead of a single HMM to represent an MSA. I present a new algorithm for sequence alignment using the fHMM technique. I show that

using the fHMM technique for sequence alignment results in more accurate alignments than the single HMM approach.

As sequence alignment is a fundamental step in many bioinformatics pipelines, improvements to sequence alignment result in improvements across many different fields. I show the applicability of fHMM to three specific problems: phylogenetic placement, taxonomic profiling and identification, and MSA estimation. In phylogenetic placement, the problem addressed is how to insert a query sequence into an existing tree. In taxonomic identification and profiling, the problems addressed are how to taxonomically classify a query sequence, and how to estimate a taxonomic profile on a set of sequences. Finally, both profile HMM and fHMM require a backbone MSA as input in order to align the query sequences. In MSA estimation, the problem addressed is how to estimate a “de novo” MSA without the use of an existing backbone alignment.

For each problem, I present a software pipeline that implements the fHMM specifically for that domain: SEPP for phylogenetic placement, TIPP for taxonomic profiling and identification, and UPP for MSA estimation. I show that SEPP has improved accuracy compared to the single HMM approach. I also show that SEPP results in more accurate phylogenetic placements compared to existing placement methods, and SEPP is more computationally efficient, both in peak memory usage and running time. I show that TIPP more accurately classifies novel sequences compared to the single HMM approach, and TIPP estimates more accurate taxonomic profiles than leading

methods on simulated metagenomic datasets. I show how UPP can estimate “de novo” alignments using fHMM. I present results that show UPP is more accurate and efficient than existing alignment methods, and estimates accurate alignments and trees on datasets containing both full-length and fragmentary sequences. Finally, I show that UPP can estimate a very accurate alignment on a dataset with 1,000,000 sequences in less than 2 days without the need of a supercomputer.

Table of Contents

Acknowledgments	v
Abstract	viii
List of Tables	xvi
List of Figures	xviii
Chapter 1. Introduction	1
Chapter 2. Background	4
2.1 Phylogenetics	4
2.2 Profile Hidden Markov Models	16
2.3 Applications of Profile HMMs	17
2.3.1 Phylogenetic Placement	17
2.3.2 Metagenomic Analyses	21
2.3.3 Multiple Sequence Alignment Estimation	26
Chapter 3. Family of Hidden Markov Models	30
3.1 Family of HMM	30
3.2 fHMM Alignment Algorithm	32
3.3 HMMER Commands	36
3.4 Comparison to existing algorithms and methods	37
Chapter 4. SEPP: SATé-enabled phylogenetic placement	40
4.1 SEPP Algorithm	41
4.2 Performance Evaluation	44
4.3 Results	49
4.3.1 Algorithm design experiments	49

4.3.2	Comparisons using the Default Setting for SEPP	53
4.3.3	Results on Simulated Datasets	53
4.3.4	Results on 16S.B.ALL	58
4.3.5	Comparing methods on query sequences of different levels of difficulty	59
4.3.6	Summary	59
4.4	Conclusion and future work	61
Chapter 5. TIPP: Taxonomic identification and phylogenetic profiling using families of Hidden Markov Models		63
5.1	Taxonomic Identification through Phylogenetic Placement . . .	64
5.2	TIPP Algorithm	68
5.2.1	Alignment Support Calculation	71
5.2.2	Abundance Profile Estimation	73
5.2.3	TIPP on Larger Markers	74
5.3	Performance Evaluation	74
5.4	Results	83
5.5	Conclusions and Future Work	92
Chapter 6. UPP: Ultra-large alignments using family of Hidden Markov Model		94
6.1	UPP: Ultra-large alignment using Phylogeny-aware Profiles . .	95
6.2	Performance Evaluation	100
6.3	Results	105
6.3.1	Phylogenetic Alignment Accuracy	106
6.3.2	Structural Alignment Accuracy	113
6.3.3	Results on Fragmentary Datasets.	114
6.3.4	Factors Influencing Accuracy	118
6.3.5	Running Time	119
6.4	Conclusion and Future Work	121
Chapter 7. Conclusion and future work		126
7.1	Conclusion	126
7.2	Future Work	127

Appendices	132
Appendix A. TIPP	133
A1 Precision and Recall Comparisons and Statistical Significance .	133
A1.1 HMMER+pplacer versus HMMER+EPA	133
A1.2 Experiment 1: TIPP variants	133
A1.2.1 HMMER+pplacer versus SEPP	133
A1.2.2 TIPP(0%,0%,100) versus TIPP(0%,95%,100) . .	136
A1.2.3 TIPP(0%,95%,100) versus TIPP(95%,95%,100)	136
A1.3 Leave-one-out experiments: TIPP versus MetaPhyler . .	139
A2 Leave-one-out Results in Tabular Format	146
A2.1 Experiment 1: TIPP Variants	146
A2.2 Leave-one-out experiments: TIPP versus MetaPhyler . .	146
A3 Results Omitted from Chapter 5	153
A3.1 Experiment 1: Leave-one-out TIPP Variants	153
A3.2 TIPP Boosting of EPA versus pplacer	157
A3.3 Non-leave-one-out Parameter Exploration Study	160
A3.4 ROC Curves	166
A3.5 Leave-one-out TIPP versus Metaphyler	169
A3.5.1 Leave-one-out 30 marker genes	169
A3.5.2 Leave-one-out 16S RNA gene	169
A3.6 16S RNA on archaea, leave-one-out experiments; effects of Halobacteria	174
A3.7 Experiment 2: Abundance profiling experiments	178
A3.8 Experiment 3: Exploring robustness to sequencing error on taxonomic identification experiments.	178
A4 Non-leave-one-out Running Time Study.	185
A5 Abundance profile calculation	186
A6 Dataset	187
A6.1 Marker Genes and Empirical Statistics	187
A6.2 Fragments	187
A7 Abundance Profile Datasets	190
A7.1 Metaphlan Simulated Dataset	190

A7.2	FACs HC	190
A7.3	FAMeS	190
A7.4	WebCarma dataset	191
A8	Methods	191
A8.1	EPA and pplacer: Likelihood-based phylogenetic placement	191
A8.2	Commands Used	193

Appendix B. UPP 197

B1	Materials and Methods	197
B1.1	Datasets	197
B1.1.1	CRW 16S biological datasets.	197
B1.1.2	FastTree COG simulated datasets.	198
B1.1.3	Large AA datasets with full reference alignments.	198
B1.1.4	HomFam datasets.	199
B1.1.5	1000-taxon simulated datasets.	200
B1.1.6	Indelible simulated datasets.	201
B1.2	Methods	201
B1.2.1	Basic alignment methods	201
B1.2.2	HMMER Commands	204
B1.2.3	Maximum Likelihood Tree Estimation	204
B1.2.4	UPP alignment method	205
B1.3	Early termination on large datasets	213
B2	Supplemental Figures and Tables	217
B2.1	Sequence length distribution	220
B2.2	Resampling on the CRW 16S.T dataset	222
B2.3	UPP pipeline exploration	225
B2.3.1	Backbone alignment method	226
B2.3.2	Backbone size	229
B2.3.3	Query sequence alignment method.	230
B2.3.4	Impact of using the HMM Family technique or a single HMM	232
B2.4	SEPP vs. UPP	239
B2.5	MAFFT variants	245

B2.6	PASTA on the ten large AA datasets	246
B2.6.1	PASTA commands	250
B2.6.2	Model selection for PASTA variants	252
B2.7	Comparisons between UPP, SATé-II, and PASTA	254
B2.8	Backbone and final alignment error.	265
B2.9	Results on full-length datasets	267
B2.10	TC Scores	286
B2.11	Results on fragmentary datasets	288
Bibliography		293

List of Tables

4.1	Dataset statistics for curated alignments.	48
4.2	Mean delta-error for all query sequences.	56
4.3	Mean delta-error for different categories of query sequences.	57
5.1	Summary of all simulated abundance datasets.	79
5.2	The normalized average <i>RMSE</i> for abundance profiling methods.	84
6.1	UPP variants on the RNASim datasets.	120
A1	Precision-Recall Differences between HMMER+pplacer and HMMER+EPA on the rpsB gene.	134
A2	Precision-Recall Differences between SEPP and HMMER+pplacer on the rpsB gene.	135
A3	Precision-Recall Differences between TIPP(0%,95%,100) versus TIPP(0%,0%,100) on the rpsB gene.	137
A4	Precision-Recall Differences between TIPP(0%,95%,100) and TIPP(95%,95%,100) on the rpsB gene.	138
A5	Precision-Recall Differences on 30 marker genes, Illumina.	140
A6	Precision-Recall Differences on 30 marker genes, 454.	141
A7	Precision-Recall Differences on 16S RNA gene on bacteria, Illumina.	142
A8	Precision-Recall Differences on the 16S RNA gene on bacteria, 454 error model.	143
A9	Precision-Recall Differences on 16S archaea gene, Illumina.	144
A10	Precision-Recall Differences on 16S archaea gene, 454.	145
A11	Leave-species-out results on TIPP variants for rpsB marker gene.	147
A12	Leave-genus-out results on TIPP variants for rpsB marker gene.	148
A13	Leave-family-out results on TIPP variants for rpsB marker gene.	149
A14	Leave-one-out results for 30 markers Illumina error model.	150
A15	Leave-one-out results for 30 markers 454 error model.	150

A16	Leave-one-out results for 16S bacteria 454 error model.	151
A17	Leave-one-out results for 16S bacteria Illumina error model. . .	151
A18	Leave-one-out results for 16S RNA gene on archaea 454 error model.	152
A19	Leave-one-out results for 16S archaea Illumina error model. . .	152
A20	Precision and recall of TIPP+EPA and TIPP+pplacer.	159
A21	Normalized <i>RMSE</i> for different methods on short fragment datasets.	181
A22	Normalized <i>RMSE</i> for different methods on long fragment datasets.	182
A23	Running time experiment.	186
A24	Marker gene statistics.	188
A25	Higher error fragment statistics.	189
B1	Performance of UPP variants on the million-sequence RNASim dataset.	217
B2	Empirical statistics for simulated datasets.	218
B3	Empirical statistics for the biological datasets.	219

List of Figures

2.1	Example of rooted and unrooted trees.	7
2.2	Example of an MSA.	8
2.3	Example of sequence evolution.	9
2.4	Example of scoring MSAs.	11
2.5	Computing error metrics of estimated tree.	15
2.6	Profile HMM representation of an MSA.	18
2.7	Computing Δ_{FN} error of query sequence placement.	22
2.8	Histogram of sequence lengths for the 16S Gutell CRW datasets.	28
3.1	Example of centroid decomposition.	32
3.2	Building an HMM.	33
3.3	Example of alignment using HMM families.	34
3.4	Example of alignment of query sequence through transitivity.	35
4.1	Example of SEPP pipeline.	44
4.2	Scatter plot of delta error versus time versus memory for the 16S.B.ALL dataset.	50
4.3	Scatter plot of delta error versus time versus memory for the M2 dataset.	51
4.4	Results on simulated datasets for model M2.	54
4.5	Results on 16S.B.ALL.	55
5.1	Taxonomic classification using phylogenetic placement	65
5.2	Comparing SEPP and HMMALIGN+pplacer for taxonomic identification	67
5.3	Non-leave-one-out experiments under 454-like errors.	88
6.1	Overview of the UPP algorithm.	97
6.2	Alignment error rates on different datasets.	107
6.3	Tree error on RNASim 10K and Indelible datasets	111

6.4	Impact of fragmentary sequences on alignment error.	115
6.5	Tree error on fragmentary RNASim 10K datasets	117
6.6	Running time for UPP(Fast) on the RNASim datasets.	119
A1	Leave-species-out experiment on the rpsB marker gene comparing TIPP variants.	154
A2	Leave-genus-out experiment on the rpsB marker gene comparing TIPP variants.	155
A3	Leave-family-out experiment on the rpsB marker gene comparing TIPP variants.	156
A4	Leave-one-out experiment for TIPP+EPA and TIPP+pplacer.	158
A5	Varying placement support.	161
A6	Varying alignment support with placement support of 50%.	163
A7	Varying alignment support with placement support of 95%.	164
A8	Varying alignment support and placement support.	165
A9	Varying decomposition size for TIPP(95%).	167
A10	ROC curve for TIPP on rpsB data.	168
A11	Leave-one-out experiment comparing MetaPhyler and TIPP-default on the 30 marker genes.	171
A12	Leave-one-out experiment comparing MetaPhyler and TIPP-default on the 16S bacteria marker gene.	172
A13	Leave-one-out experiment comparing MetaPhyler and TIPP-default on the 16S archaea marker gene.	173
A14	Taxonomy for Halobacteria class for the 16S RNA gene.	175
A16	Removing Halobacteria class from leave-family-out and leave-order-out experiments for Illumina fragments.	176
A18	Removing Halobacteria class from leave-family-out and leave-order-out experiments for 454 fragments.	177
A19	Abundance profiling results comparing different TIPP methods on short fragments.	179
A20	Abundance profiling results comparing different TIPP methods on long fragments.	180
A21	Non-leave-one-out experiments for fragments with Illumina-like and 454-like errors.	183
A22	Non-leave-one-out experiments results at phylum level.	184

B1	Distribution of the backbone sequences in the tree on the first iteration of UPP on 16S.T.	212
B2	Sequence length distributions of the HomFam datasets.	220
B3	Sequence length distributions of the ten large AA datasets.	221
B4	Alignment error rates for the first two iterations of UPP on the CRW 16S.T dataset.	223
B5	Tree error rates for first two iterations of UPP on the CRW 16S.T dataset.	224
B6	Alignment error for different UPP backbone alignments on the RNASim 10K dataset.	227
B7	Tree error rates for different UPP backbone alignments on the RNASim 10K dataset.	228
B8	Alignment and tree error of UPP variants on the RNASim datasets.	231
B9	Alignment error of UPP variants on the RNASim datasets.	233
B10	Tree error of UPP variants for RNASim datasets.	234
B11	Wall clock alignment time (hrs) of UPP variants on the RNASim datasets.	235
B12	Alignment error of UPP variants on the CRW 16S datasets.	236
B13	Tree error of UPP variants on the CRW 16S datasets.	237
B14	Wall clock alignment time (hrs) of UPP variants on the RNASim datasets.	238
B15	Alignment error of UPP and SEPP on the RNASim datasets.	240
B16	Tree error of UPP and SEPP on the RNASim datasets.	241
B17	Alignment error of UPP and SEPP on the CRW 16S datasets.	242
B18	Tree error of UPP and SEPP on the CRW 16S datasets.	243
B19	Alignment and tree error for SEPP and UPP on fragmentary CRW 16S.T datasets.	244
B20	Results of default MAFFT and MAFFT-PartTree on the 16S.T, 16S.3, and RNASim 10K datasets.	245
B21	Alignment and tree error of PASTA variants on the ten large AA datasets with full reference alignments, using substitution models selected by PROTEST.	248
B22	Alignment and tree error of different methods on the ten large AA datasets with full reference alignments, using substitution models selected by PROTEST.	249

B23	Alignment error of PASTA, SATé-II, and UPP for RNASim datasets.	255
B24	Tree error of PASTA, SATé-II, and UPP for RNASim datasets.	256
B25	Alignment error of PASTA, SATé-II, and UPP on the CRW datasets.	257
B26	Tree error rates for UPP, SATé-II, and PASTA on the CRW datasets.	258
B27	Alignment error of PASTA and UPP on the FastTree COG datasets.	259
B28	Tree error of UPP and PASTA on the FastTree COG datasets.	260
B29	Average alignment error of PASTA and UPP on HomFam datasets.	261
B30	SPFN and SPFP alignment error of PASTA and UPP on HomFam datasets.	262
B31	Alignment error of PASTA and UPP on the fragmentary 1000M2 datasets.	263
B32	Delta FN tree error of UPP and PASTA on the fragmentary 1000M2 datasets.	264
B33	Comparison of initial backbone alignment error and final UPP alignment error.	266
B34	Alignment error on the RNASim datasets.	268
B35	Tree error on the RNASim datasets.	269
B36	Wall clock alignment time (hrs) on the RNASim datasets. . .	270
B37	Average alignment error of different methods on the hardest 1000-taxon datasets.	271
B38	SPFN and SPFP alignment error of different methods on the hardest 1000-taxon datasets.	272
B39	Tree error of different methods on the hardest 1000-taxon datasets.	273
B40	Average alignment error on the Indelible datasets.	274
B41	SPFN and SPFP alignment error on the Indelible datasets. . .	275
B42	Tree error rates on the Indelible datasets.	276
B43	Average alignment error on the HomFam datasets.	277
B44	Alignment SPFN and SPFP errors on the HomFam datasets. .	278
B45	Average alignment error on the ten large protein datasets with full alignments.	279

B46	SPFN and SPFP alignment error rates on the ten large protein datasets with full alignments.	280
B47	RAxML tree error rates on the ten large protein datasets with full alignments, under the JTT model.	281
B48	Average alignment error on the FastTree COG datasets.	282
B49	SPFN and SPFP alignment error on the FastTree COG datasets.	283
B50	Tree error rates on the FastTree COG datasets.	284
B51	Alignment error rates on the CRW datasets.	285
B52	TC scores on the biological AA datasets	287
B53	Alignment error rates on the fragmentary 1000-taxon datasets.	289
B54	Tree error rates on the fragmentary 1000-taxon datasets.	290
B55	Alignment error rates on the fragmentary RNASim 10K datasets.	291
B56	Tree error rates on the fragmentary RNASim 10K datasets.	292

Chapter 1

Introduction

Nothing in biology makes sense
except in the light of evolution

Christian Theodosius
Dobzhansky

The theory of evolution is the cornerstone of modern biology. Under the principles of evolution, we have gained insights into hominid and human origins [51, 86], vaccine development [21, 94], and even environmental bioremediation [48]. Crucial to understanding many of these topics is the ability to estimate the evolutionary relationship between different biomolecular sequences.

A multiple sequence alignment (MSA) is a hypothesis of the evolutionary relationships between different characters in a set of biomolecular sequences. MSAs have been used in many bioinformatics analyses including phylogeny estimation [28], protein folding prediction [34], and functional annotation of proteins [20]. However, MSA estimation is computationally challenging, as many optimization algorithms for standard objective functions are NP-hard [8, 91], and most heuristic methods for MSA estimation do not grow linearly with the number of sequences [62].

One approach to address this problem is through the use of profile Hidden Markov Models (HMM) [15]. Profile HMMs are statistical models for representing an MSA alignment. They can be used to independently align new sequences to an existing MSA [15], and thus exhibit linear scaling in running time with respect to the number of new sequences to insert. However, profile HMMs are used for more than just MSA estimation; other uses include remote homology detection [20], sequence database searching [67], and classification of short environmental reads [24].

The ability of profile HMMs to accurately insert sequences into an existing MSA degrades, however, on datasets containing evolutionary divergent sequences [20, 59]. My investigation into this problem led to the development of a new statistical model which I call the *family of Hidden Markov Models* (fHMM). The fHMM is a statistical model for representing an MSA by using multiple HMMs. I show how fHMM can be used for accurate alignment of a sequence to an existing MSA. As sequence alignment is a vital step in many bioinformatics analyses, the fHMM can be used across a wide range of problems such as inserting sequences into a tree, taxonomically classifying short fragments, and aligning ultra-large datasets.

In Chapter 2, I formally introduce key concepts in phylogenetics such as MSA estimation and tree estimation. I also introduce profile HMMs and how they can be used for aligning query sequences to an MSA. Furthermore, I introduce three problems that will be addressed using fHMM: phylogenetic placement, taxonomic profiling and taxonomic identification, and MSA esti-

mation.

In Chapter 3, I describe the fHMM technique and show how fHMM can be used in sequence alignment. In Chapter 4, I present SEPP [55], a method for phylogenetic placement using fHMM. I present the results simulation study comparing SEPP and other placement methods. I show that SEPP results in more accurate placements than the single HMM approach, and that SEPP can accurately place sequences that are very evolutionarily divergent.

In Chapter 5, I introduce TIPP, a method for taxonomic identification and profiling using fHMM and statistic support measures. By incorporating statistical support within the fHMM alignment technique, the precision in taxonomically classifying novel sequences is greatly improved. In addition, I show that fHMM results in better estimation of the species abundance profile of simulated microbial communities.

In Chapter 6, I present UPP, a “de novo” MSA estimation technique using fHMM. I show how to use fHMM to align ultra-large datasets (large in the number of sequences) without the need of an initial backbone alignment and tree. I show how UPP can align datasets containing both short and full-length sequences. I show that this new technique can accurately align a dataset of 1,000,000 sequences in less than 2 days without the need of a supercomputer. Finally, in Chapter 7, I summarize the contributions of this dissertation and discuss future work.

Chapter 2

Background

In Section 2.1, I give a brief introduction to phylogenies and alignments and define concepts that will be used throughout my dissertation. In Section 2.2, I describe profile Hidden Markov Models (HMM) and their use in alignment estimation. Finally, in Section 2.3, I describe applications of profile HMMs in the realm of phylogenetic placement, metagenomic analyses, and ultra-large alignment estimation.

2.1 Phylogenetics

Phylogenetics is the study of the evolutionary relationships between different organisms. A typical molecular phylogenetic study begins by collecting biomolecular sequences (DNA, RNA, or amino acid sequences) from the species of interest. The evolutionary relationships between the different characters in the sequence are inferred through an alignment. From the alignment, a tree representing the evolutionary history between the different species is estimated. The steps of estimating an alignment and estimating a tree are core concepts used throughout my dissertation. I now provide more details on the alignment and the tree, and on how one might estimate an alignment and a

tree.

Tree: graphical model of evolution A *phylogeny* is a graphical model that represents the evolutionary relationships between different species. One of the most common representations is a *rooted tree* - a directed acyclic graph. Each leaf in the tree represents a species, and each internal node in the tree represents a *speciation event*. Speciation events occurs when one species give rise to new lineages of species. The root of the tree represents the most recent common ancestor of all the species. Throughout my dissertation, I will refer to the leaves of the tree as species, taxa, or sequences, interchangeably. Similarly, I refer to phylogenies as trees, though a phylogeny does not necessarily have to be tree-like, and more complicated representations such as phylogenetic networks do exist.

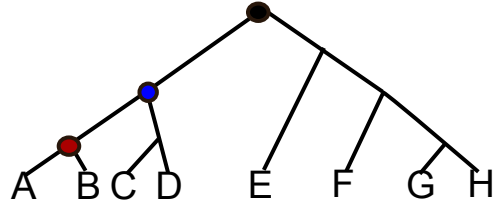
Figure 2.1(a) shows an example of a rooted phylogenetic tree. The relationship between the different species can be inferred from the tree. For example, species *A* and *B* are more closely related than species *A* and *C* because *A* and *B* share a more recent common ancestor (red node) than *A* and *C* (blue node). The given example is a rooted rooted, i.e., the direction of evolution is known. The root of the tree represents the most recent common ancestor (MRCA) of all the species (black node). In general, estimating the root of the tree is very difficult as most common models used in phylogeny estimation assume time-reversibility, and under these models, it is not possible to determine which node is the ancestor and which node is the descendant.

Thus, when I discuss phylogenies, I refer to unrooted trees.

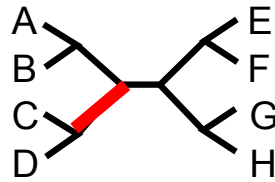
Figure 2.1(b) shows an unrooted version of Figure 2.1(a). An unrooted tree is a *binary* tree if all inner nodes have a degree of 3. If an inner node has degree greater than 3, it is called a *polytomy*. Polytomies represent evolutionary relationships that cannot be resolved. Figure 2.1(c) shows an example of a tree containing a polytomy.

Multiple Sequence Alignment. Biomolecular sequences are represented as character strings over an n -letter alphabet. The most common alphabets are the 4-letter alphabets for nucleotides ($\{A, T, C, G\}$ for DNA and $\{A, U, C, G\}$ for RNA) and the 20-letter alphabet for amino acid sequences. Because DNA is inherited from parent to child, biomolecular sequences are often used to reconstruct the evolutionary history of present day organisms.

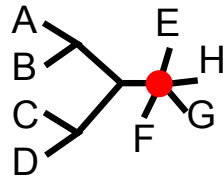
A fundamental step in understanding the relationship between the different sequences is to estimate an alignment on the sequences. A *multiple sequence alignment* (MSA) is a data structure that represents the evolutionary relationships between the individual characters in a set of sequences. An MSA on a set of sequences is defined by a matrix with a row for each sequence, and columns representing a site of common evolutionary origin. The sequences in the MSA are interspersed with gap characters (represented by “-”). Gap characters represent historical insertion and deletion events (called “indel” events). If a pair of characters descended from the same ancestral character, then they are called *homologous* and will be in the same column in



(a) A rooted phylogenetic tree.



(b) An unrooted phylogenetic tree.



(c) A polytomy.

Figure 2.1: Example of a) a rooted phylogenetic tree, b) the unrooted version of the same tree, and c) an unrooted tree with a polytomy. In the rooted tree, the red node is the MRCA of species A and B , and the blue node is the MRCA of species A and C . The black node is the MRCA of all the species in the tree. In b), the red edge represents the bipartition $\{CD|ABEFGH\}$. Finally, in c) the red node represents a polytomy.

the MSA. Homology is a transitive property, so if a nucleotide A is homologous to nucleotide B and C , then nucleotides B and C are also homologous to each other. Figure 2.1 shows an example of an MSA. The goal of an MSA is to infer sites of shared homologies between the different sequences.

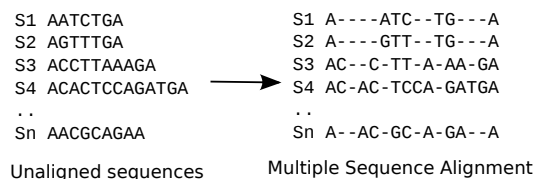


Figure 2.2: Example of an MSA estimated on the sequence set $\{S_1, \dots, S_n\}$. Originally the sequences are unaligned. The sequences are aligned by inserting gaps into the sequences such that homologous characters line up in the same site.

Simulation study. There are many different methods for estimating an MSA and for estimating a phylogenetic tree. As it is impossible to know the true history for a set of biological sequences, simulation studies are performed to test the performance of different alignment and phylogeny estimation techniques.

A typical molecular simulation study begins by generating a rooted model tree that represents the true evolutionary history of the set of sequences. The model tree can be generated by using a phylogenetic tree from a previous study, or it can be generated by simulating a tree under a stochastic model of speciation events (see [2] for a review of tree models). Once a model tree has

been generated, a stochastic model of sequence evolution is selected, as well as a model of indel events. These models include parameters for the rate of substitutions, insertion, and deletion events, as well as a model for the indel length distribution. Once all the parameters have been selected, a random sequence is generated at the root, and it is simulated down the model tree with substitution and indel events (see Fig. 2.3). The true sequence is known at each internal node, as well as the history of the mutation patterns. Thus, at the very end of the simulation, the true MSA and true phylogeny of the sequences are known and can be used to compare the accuracy of MSA and phylogeny estimation methods.

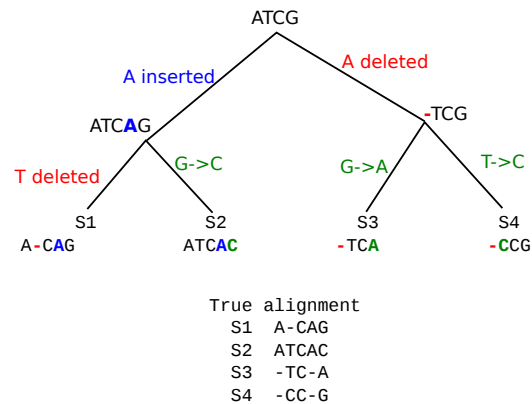


Figure 2.3: Example of sequence evolution down a model tree. The original sequence at the root is “ATCG”. Through a series of insertions (colored blue), deletions (colored red), and substitutions (colored green), the root sequence evolves to 4 new sequences. The goal of a molecular phylogenetic study is to infer from the unaligned sequences the true alignment and phylogeny.

MSA Estimation Computing an MSA can be formulated as an optimization problem of minimizing the differences between the sequences across the sites in the alignment. One example is given a set of sequences, we find the MSA that minimizes the sum-of-pairs (SP) error. The SP error is computed by summing the total number of mismatches (pairs of aligned “non-gap” characters that do not match) and indels (any “non-gap” characters aligned to a “gap” character) over all pairs of sequences in the MSA. Figure 2.4(a) shows the computation of the SP error for a pair of sequences. Figure 2.4(b) shows the SP error for two MSAs estimated on the same set of sequences. In this example, the bottom MSA has a lower SP error and would be considered more accurate under the SP error optimization criterion.

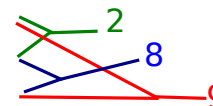
Computing an MSA that optimizes SP score (and many other similar metrics) is NP-complete [8, 91] and thus, finding an exact solution is computationally intractable for large datasets.

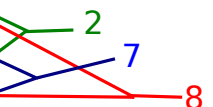
Alternative heuristics have been developed for MSA estimation (see [62, 89] for survey and comparison of current methods), including progressive methods which build an MSA by progressively aligning pairs of sequences and then merging the alignments using an estimated tree, and iterative methods which combine progressive methods and iteration so that the estimated MSA from the progress step is reused to estimate a better MSA. However, these methods do not scale linearly with the number of sequences to be aligned [62] and may have poor performance on large datasets or evolutionary divergent datasets.

Another class of MSA estimation methods includes methods that use

S1 A--ATC-TG--A
S3 ACC-TTA-AAGA
 Σ 011101111110=9

(a) Computing SP error for 2 sequences.

S1 A--ATC-TG--A  **2**
S2 A--GTT-TG--A **8**
S3 ACC-TTA-AAGA **9**
SP score = 2+8+9 = 19

S1 A-ATC-TG--A  **2**
S2 A-GTT-TG--A **7**
S3 ACCTTA-AAGA **8**
SP score = 2+7+8 = 17

(b) Computing SP error for an MSA.

Figure 2.4: Example of a) computing the SP error for a pair of sequences, and b) the SP scores for two different MSAs of the same set of sequences. In a), mismatches are highlighted red, and indels are highlighted blue. The SP error is the sum of the total number of mismatches and indels. In b) the SP scores for each pair of sequences are shown. The total SP error for an MSA is the sum of all the pairwise SP scores. In this example, the lower MSA has lower SP error and is more desirable under the SP error optimization criterion.

profile Hidden Markov Models (HMM). A profile HMM is a statistical representation of an MSA (see Section 2.2 for a more in-depth overview). HMM methods take an existing MSA and compute a profile HMM from that MSA. Sequences are then independently aligned to the profile. Thus, profile HMM methods scale linearly with the number of sequences to align to an existing MSA. However, the accuracy of HMM methods is impacted by the rate of evolution. On datasets containing evolutionarily divergent sequences, the accuracy for detecting homologies degrades [20, 59].

Quantifying error in alignments. If a true alignment is known via a simulation study, or a high quality curated alignment has been estimated, one can compare the error of an estimated alignment by examining the percentage of shared and missing homologies in the estimated alignment with respect to the reference alignment.

Three common metrics for quantifying the error of an estimated alignment are:

- *sum-of-pairs false positive* (SPFP) rate - the total number of homologies in the estimated alignment that are not found in the true alignment, normalized by the total homologies in the estimated alignment,
- *sum-of-pairs false negative* (SPFN) rate - the total number of homologies in the true alignment that are not found in the estimated alignment, normalized by the total homologies in the true alignment, and

- *total column* (TC) error rate - the number of columns in the true alignment that are not exactly recovered in the estimated alignment, normalized by the total columns in the true alignment.

In my dissertation, I report SPFN and SPFP rates, as well as the arithmetic mean of the two rates. In addition, I also report the TC error rate on protein datasets. This metric is of interest when the goal is to examine how well alignment methods recover conserved domains in the protein alignment.

Phylogeny estimation Many different phylogenetic methods exist for estimating a phylogenetic tree from an MSA such as distance-based methods, parsimony-based methods, Bayesian methods, and Maximum Likelihood methods (ML) (see [28, 29] for overview of current methods). ML methods give better accuracy than distance-based and parsimony-based methods [42, 92], and unlike Bayesian methods, can be accurately run on large datasets [47, 65, 78]. Thus I focus on ML-based methods for tree estimation for my dissertation.

Quantifying error in trees. There are many different metrics for quantifying tree error. My dissertation focuses on the topological differences between the estimated tree and true tree, measured in edges. Each edge in the tree defines a bipartition. For example, in Figure 2.1(b), the red edge represents the bipartition $\{CD|ABEFGH\}$ (note that $\{CD|ABEFGH\}$ is identical to $\{ABEFGH|CD\}$). Removal of this edge separates CD from $ABEFGH$.

Trees on the same leaf set can be compared by examining the bipartitions that they share in common and the bipartitions that are unique to each tree.

Three common metrics for quantifying the topological error of an estimated tree are:

- *Robinson–Foulds* (RF) rate [72] - the total number of bipartitions that are unique to the reference tree and estimated tree, normalized by the total bipartitions in both trees,
- *false positive* (FP) rate - the total number of bipartitions in the estimated tree that are not in found in the reference tree, normalized by the total bipartitions in the estimated tree, and
- *false negative* (FN) rate - the total number of bipartitions in the reference tree that are not in found in the estimated tree, normalized by the total bipartitions in the estimated tree. The FN rate is also known as the missing branch rate.

For binary trees, $FN = FP = RF$. My dissertation primarily reports the missing branch rate as the error metric for comparing trees. For simulated datasets, both the model trees and estimated trees are binary trees, thus reporting missing branch rate is identical to reporting the FP and RF rates. For biological datasets, the reference trees are ML trees estimated on curated alignments, with only highly support edges retained. Thus, the reference trees on biological datasets are non-binary, and the FP and FN rates will differ. It

is extremely easy for a method to estimate a tree with low FP rate by random chance; the method could estimate a unresolved tree. It is much more difficult for a method to estimate a tree with low FN rate by random chance. Thus, I focus on the FN rate throughout my dissertation.

Figure 2.5 shows an example of a true tree and an estimated tree. Each tree contains 5 bipartitions. The bipartitions $\{AB|CDEFGH\}$ and $\{CD|ABEFGH\}$ are found in the true tree, but not present in the estimated tree. Thus, the missing branch rate is 20%. Similarly, the bipartitions $\{AC|BDEFGH\}$ and $\{BD|ACEFGH\}$ are found in the estimated tree, but are not present in the true tree, yielding an FP rate of 20%.

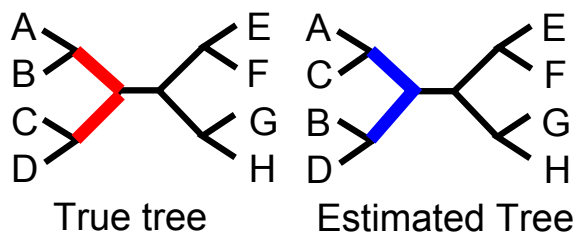


Figure 2.5: An example of the true tree and the estimated tree. The estimated tree has an FN rate of $\frac{2}{5}$ (two bipartitions colored red in the true tree are not found in the estimated tree; five bipartitions in true tree) and has an FP rate of $\frac{2}{5}$ (two bipartitions colored blue in the estimated tree are not found in the true tree; five bipartitions in estimated tree).

2.2 Profile Hidden Markov Models

A profile Hidden Markov Model (HMM) is a probabilistic model for representing an MSA. A profile HMM can be represented by a finite state machine (FSM, see Fig. 2.6). By transitioning through the FSM, a sequence can be generated from the HMM. Similarly for a given sequence, an alignment of the sequence to the HMM can be computed by finding the most probable path through the model for generating the sequence. I now describe the FSM in more detail.

The FSM for a profile HMM consists of a start state S and an end state E , a set of *match states* $M = \{M_1, \dots, M_n\}$, a set of *insertion states* $I = \{I_0, \dots, I_n\}$, a set of *deletion states* $D = \{D_1, \dots, D_n\}$, and directed *transition edges* $E = \{(M_i, M_{i+1}), (D_i, M_{i+1}), (I_i, M_{i+1}), (I_i, I_i)\}$ for $i \in \{1, \dots, n\}$. Each transition edge has an associated probability, and the sum of all transition edges leaving a state must sum up to 1. Each match and insertion state has an associated *emission probability vector* which is a probability that a character will exist at that state.

The match states represent contiguous columns (called “consensus columns”) in the MSA. In the simplest case, the match state M_i models the column c_i in the alignment. Insertion states represent insertion events in the sequence. Similarly, deletion states represent deletion events in a sequence. If a sequence contains no indels (i.e., it aligns perfectly to the MSA without the need of inserting any gaps), then the path through the model would proceed from match state to match state. However, if the sequence contains an in-

sersion event at the second character, then the path through the FSM would go from match state M_2 to insertion state I_2 , and would remain at insertion state I_2 until all the estimated insertion characters have been processed, at which it goes to the next match state. Finally, if the sequence seems to be missing a character relative to the MSA, then a deletion event has occurred. For example, if the first and second characters in the sequence are AA , but the first 3 columns in the MSA contain with ATA , then this suggests that the sequence had a deletion event in the second column. In this case, the path through the FSM would go from M_1 to D_2 to M_3 .

The process of aligning a sequence to a profile HMM is to find the most probable path through the FSM for generating the sequence. This can be solved via the Viterbi dynamic programming algorithm [90] in $O(L * |M|^2)$ time complexity, where L is the length of the sequence and $|M|$ is the total number of match states. Thus, alignment using a profile HMM grows linearly with the number of sequences to align.

2.3 Applications of Profile HMMs

2.3.1 Phylogenetic Placement

The first application of profile HMMs is in the problem of phylogenetic placement. As I briefly mentioned in Chapter 1, phylogenetic placement is a method for inserting query sequences into an existing phylogenetic tree. Phylogenetic placement is an alternative approach to phylogeny estimation for inferring the phylogenetic relationship between a set of query sequences and a

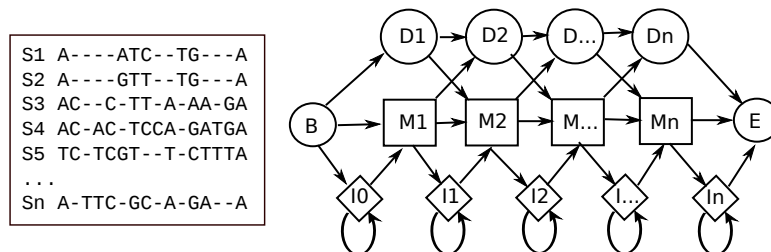


Figure 2.6: Profile HMM representation of an MSA using a finite state machine. Not shown are the emission probability vectors on the match states and insertion states.

set of full-length sequences in the existing tree. Rather than estimating a new phylogenetic tree on the entire set of full-length and query sequences, phylogenetic placement infers the relationships between the query sequence and the full-length sequences one at a time, making the computational complexity of placement grow linearly with the number of query sequences. In addition, if new query sequences are added, the placement algorithm needs to be run on just the new sequences. Phylogeny estimation, on the other hand, would have to be re-run on the entire set of sequences every time a new sequence is added.

Phylogenetic placement is extremely advantageous in the analyses of short DNA fragments taken from an environmental sample where there can be potentially millions of fragmentary reads. By placing a short read from an unknown species into a taxonomic tree, one can infer the taxonomic classification

of the read based on its placement within the taxonomic tree.

I now formally describe the phylogenetic placement problem as follows:

Phylogenetic Placement Problem.

- Input: the alignment A and tree T (called the *backbone* tree and *backbone* alignment) estimated on a set S of full-length sequences and query sequence s .
- Output: tree T' containing s obtained by adding s as a leaf to T .

Several methods have been developed for this problem using the following two steps:

- Step 1: align s to the backbone alignment A to produce the alignment A' , called the *extended alignment*
- Step 2: insert s into T using A' , optimizing some criterion

Methods for the first step include HMMALIGN [15] (a part of the HMMER software suite), PaPaRa [7], Mafft-profile [35], and PAGAN [50]. Methods for the second step include EPA (run within RAxML) [6] and pplacer [52], both of which seek to optimize maximum likelihood (pplacer also provides a Bayesian approach), and MLTreeMap [80], which can optimize either ML or Maximum Parsimony (MP). In [80], Stark and Berger found that optimizing ML resulted in overall better placements, albeit with an increase in running time.

Phylogenetic placement methods can be described by the methods used for the alignment and placement steps. Three such methods include PaPaRa+EPA [6], HMMALIGN+EPA [7], and HMMALIGN+pplacer [52]. As EPA and pplacer both optimize likelihood, they were found to have almost identical placement accuracy, but have somewhat different memory usage and algorithmic features [52].

The two techniques for computing the extended alignment, PaPaRa and HMMALIGN, are very different. HMMALIGN requires only a backbone alignment to align the query sequences. PaPaRa, on the other hand, is a phylogeny aware method and requires both a backbone alignment and backbone tree to align the query sequences. HMMALIGN computes a profile HMM to represent the MSA, and then aligns the query sequences to the HMM. In contrast, PaPaRa uses RAxML to estimate ancestral state vectors at all candidate insertion points on every edge of the tree, aligns the query sequence to every ancestral state vector, selects the alignment that had the best score, and uses it to extend alignment A to include s . Thus, PaPaRa is more computationally expensive as it depends on both the number of query sequences to align and on the size of the backbone tree.

In [7], Berger and Stamatakis reported that PaPaRa+EPA had better placement accuracy on large backbone trees or on short query sequences, and for small backbone trees or on longer query sequences, HMMALIGN+EPA had better placement accuracy. However, their study examined only a limited number of model conditions (7 datasets and at most 802 sequences in the

backbone set) and improvements in topological accuracy for PaPaRa+EPA over HMMALIGN+EPA were relatively small, with PaPaRa+EPA placing query sequences on average about one edge closer to the correct location, out of 799 edges. Thus under these datasets, PaPaRa+EPA and HMMALIGN+EPA had very similar placement accuracy, although substantially different running time (PaPaRa anywhere from 6 to 43 times slower).

Comparing placement accuracy. The metric used in my dissertation for measuring the accuracy of placement is the change in missing branch rate of the backbone tree before and after insertion of the query sequence (called Δ_{FN}). More formally, if FN is the number of missing branches in the backbone tree T , and FN' is the number of missing branches in T' , then $\Delta_{FN} = FN' - FN$. Note that unlike the FN rate used in reporting tree error for phylogeny estimation, Δ_{FN} is not normalized and thus is the actual change in the number of missing branches. Figure 2.7 shows an example of this computation. Let the initial backbone tree have 0 FN . After the insertion of the query sequence s into T , T' is missing bipartitions $\{As|BCDEFGH\}$ and $\{ABs|CDEFGH\}$ (bipartitions colored red in Fig. 2.7). The resulting Δ_{FN} is 2.

2.3.2 Metagenomic Analyses

The second application of profile HMMs is in the domain of taxonomic identification and profiling. Traditionally, unknown bacterial species of interest from an environmental sample was taxonomically identified by first culturing

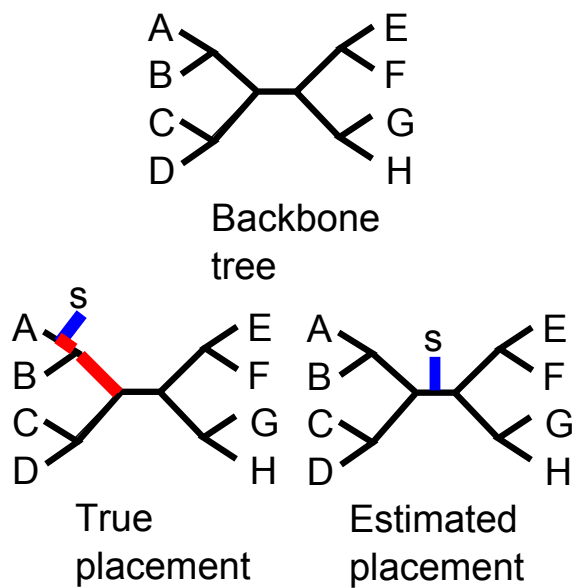


Figure 2.7: An example computing the Δ_{FN} error of query sequence placement. The backbone tree originally has 0 missing branches. After insertion of the query sequence s , the estimated tree T' is missing 2 bipartitions that are found in the true tree (missing edges colored red). Thus, the Δ_{FN} is 2.

a clonal colony of the unknown species in a laboratory environment, and then sequencing the genetic material directly from the colony. As the genetic material came from a single species, and the Sanger sequencing technology used resulted in read lengths of 800bps with roughly 20,000 to 200,000 reads per run [95], assembling the reads into longer contigs was computationally feasible on a desktop machine. From the contigs, the bacteria genome could be assembled and the species could be taxonomically identified. This pipeline allowed a window in understanding the microbial diversity in an environmental sample.

However, an estimated 99% of all microbial life cannot be cultured in a lab [4], and thus, the majority of bacterial life cannot be studied using this pipeline. *Metagenomics* is the study of analyzing genetic material taken directly from an environmental sample, and thus, bypasses the need for culturing microbes in the laboratory environment. Metagenomic analyses allows scientists to not only identify what species are present in an environmental sample, but to also estimate the relative abundances of the species present in the sample.

Metagenomic analyses are not without its difficulties. Unlike the traditional approach to taxonomic identification where reads are generated from a clonal colony, the reads generated from a metagenomic sample do not all come from a single species. In addition, the sequencing technology used typically generates much shorter reads than Sanger reads (80 to 100bps for Illumina reads), and there can be millions of sequences. Thus, a fundamental challenge in a metagenomic analyses is classifying the potentially millions of short reads

with taxonomic labels.

Methods for taxonomic identification depend on using previously sequenced genomes or genes as a reference database and extrapolating the knowledge in the reference dataset to classify the unknown reads. Simple similarity-based approaches (e.g., picking the best database hit as the best ‘guess’ at the taxonomic label) have been shown to be insufficiently accurate when the reference database does not contain species closely related to the query sequence [40], leading to the development of new and more sophisticated methods.

Classification methods fall into three types of categories: sequence homology methods, sequence composition-based methods, and phylogenetic methods (see [5] for survey of classification methods). Sequence homology methods tries to identify the reads by finding the closest related sequences in the reference database. Sequence composition-based methods use the DNA composition of the reads (typically using contiguous words of k -length known as *k-mers*) to identify the reads. Finally, phylogenetic methods attempt to best fit the query sequence into a phylogeny. An example is using phylogenetic placement to insert the metagenomic read into a taxonomic tree.

Sequence homology methods and sequence composition-based methods are typically designed to classify fragments from any part of the genome. Phylogenetic methods, however, are typically marker-based and have been designed to only classify reads that have been binned to a specific set of genes known as *marker genes*. Marker-based methods have better sensitivity in clas-

sifying reads binned to the markers, however, can only classify a subset of the sequences.

Abundance profiling, also called “phylogenetic profiling”, seeks to estimate the relative abundance of the species (or genera, or families, etc.) within a sequence dataset. While many methods produce these estimates by characterizing most (or all) of the sequences in the dataset, marker-based methods produce these estimates by characterizing only those sequences that match the marker genes they rely on. Since the marker genes are supposed to be single copy and universal, these estimations do not need to be corrected for the copy number in each genome, or for missing data. However, the restriction to sequences that match the marker genes has the potential to reduce accuracy since it means only a subset of the sequences are characterized.

Quantifying taxonomic identification error For the simulated taxonomic classification experiments, the true lineage of each fragment is known, so the metric for computing accuracy is given by the percentage of fragments classified correctly, incorrectly, and left unclassified at each taxonomic rank. Thus, a read may be unclassified at the species level, classified incorrectly at the genus level, and classified correctly at the remaining taxonomic levels.

Quantifying taxonomic profiling error For the simulated abundance profiling experiments, the true abundance of the metagenomes is known, so I compute the root-mean-squared error (*RMSE*) of the estimated taxonomic

profile.

Let C_l be the set of clades found in the true profile and all the estimated profiles for the taxonomic level l , R_x be the abundance of clade x for the reference profile, and E_x be the abundance of clade x for the estimated profile. Then $RMSE_l$ (root-mean-squared-error for a taxonomic level l) is:

$$RMSE_l = \sqrt{\sum_{x \in C_l} \frac{(R_x - E_x)^2}{|C_l|}} \quad (2.1)$$

2.3.3 Multiple Sequence Alignment Estimation

The third application of profile HMMs is in the domain of MSA estimation. I had previously described how MSAs can be used for phylogeny estimation, however, their utility also extends to many other bioinformatics pipelines, including orthology inference [1], biomolecular sequence structure and function prediction [18], and the inference and quantification of selection [17].

Because of the impact of alignment estimation error on these inferences [23, 33, 96], many methods have been developed to estimate alignments [13, 73] and estimate trees from the alignments [19]. 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 [100]; how-

ever, current MSA methods have poor accuracy on large datasets, especially when they evolved under high rates of evolution [45]. These limitations can discourage biologists from utilizing the full range of biological data, and affect downstream inferences.

Some of these projects are attempting to assemble ultra-large trees, with many tens of thousands of sequences. For example, iPTOL [76] (the iPLANT Tree of Life project) plans to construct a tree on 500,000 plant species, and the Thousand Transcriptome Project 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 [39]. Upcoming sequencing technologies [41, 61] will enable even larger datasets containing sequences from throughout the genomes of many organisms. Ambitious projects, such as the Genome 10K Project [27], 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.

Due to NGS sequencing technology, many biological datasets contain substantial numbers of fragmentary sequences (Fig. 2.8 and figs. B2 and B3 in the Appendix B), resulting in part from incomplete assembly or insufficient transcript sampling. Although some methods (e.g., HMMER [15] and MAFFT-Profile [35]) 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.

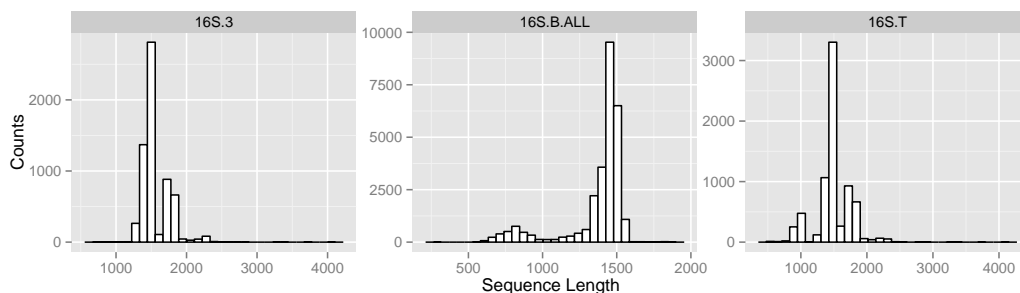


Figure 2.8: **Histogram of sequence lengths for the 16S Gutell CRW datasets.** The histogram of sequence lengths for the three CRW datasets demonstrates substantial sequence length heterogeneity, especially for 16S.T. The average length of the 16S sequence is approximately 1500.

Efficient maximum likelihood (ML) gene tree estimation for datasets containing thousands [77] to tens of thousands [65] of sequences is now feasible, but the accuracy of ML trees depends on having accurate multiple sequence alignments [60], and estimating highly accurate large-scale alignments is extremely challenging; indeed, some datasets with only 1,000 sequences can be difficult to align well [46, 47]. This is particularly true for non-coding data, which can evolve under higher rates of evolution than coding data, making alignment estimation difficult [66]. 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 [32, 54].

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.

Chapter 3

Family of Hidden Markov Models

In this chapter I present a new statistical model for representing an MSA called the Family of Hidden Markov Models (fHMM). This model was originally developed within SEPP [55] and was joint work between myself, Siavash Mirarab, and Tandy Warnow. In Section 3.1, I describe the fHMM and how to build the fHMM. In Section 3.2, I describe an algorithm for sequence alignment using the fHMM. Finally, in Section 3.3, I give the commands used to build fHMM and align sequences to the fHMM.

3.1 Family of HMM

The fHMM is a statistical model for representing an MSA using a collection of HMMs. The model was originally developed in SEPP [55] for the problem of phylogenetic placement. During our SEPP study, we realized that the utility of fHMM extends beyond phylogenetic placement. More specifically, the fHMM can be used as a replacement of an HMM for sequence alignment.

Building an fHMM. The basic outline for building an fHMM is to divide the input MSA into subsets of closely related sequences. HMMs are computed

on the individual subsets, and these HMMs make up the fHMM. I now provide more details on this process.

The necessary inputs for building the fHMM are an MSA (called the “backbone alignment”), a tree on the sequences in the MSA (called the “backbone tree”), and a maximum alignment decomposition size parameter m_a . The first step is to use the backbone tree to decompose the alignment into subsets of size at most m_a . We do this through a recursive decomposition technique called the “centroid edge decomposition” [47].

From the backbone tree, we select the centroid edge e (one whose removal separates the leaf set into approximately two equally sized subsets). We remove e from the backbone tree to produce two subtrees. For each subtree with more than m_a leaves, we recursively repeat this decomposition until all the subtrees produced by this decomposition have at most m_a leaves.

Figure 3.1 shows this process explicitly. The backbone tree in the figure has 11 taxa, and m_a is set to 3. In step 1, the centroid edge (colored red) is removed. This splits the tree into subtrees contains 5 taxa and 6 taxa. In step 2, each of these subtrees are further subdivided into trees of sizes 2, 2, 3, and 4. In step 3, one final centroid decomposition divides the subtree of size 4 into two subtrees of size 2 and 2. Step 4 shows the final result of this decomposition; the original backbone tree has now been decomposed into 5 subtrees, all with at most 3 leaves.

For a given subtree, we compute the alignment induced on the sequences

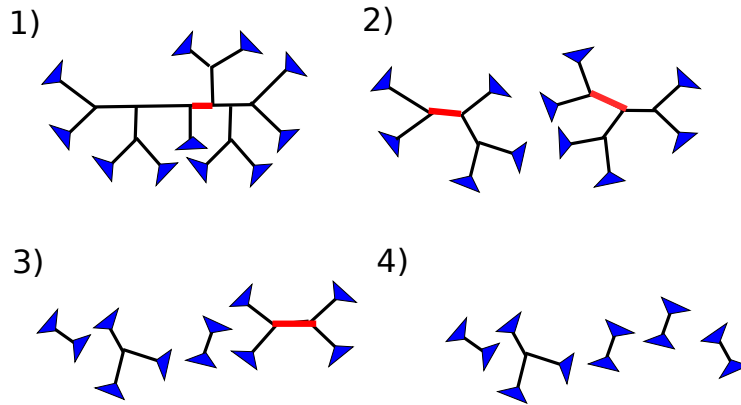


Figure 3.1: Example of centroid decomposition. The centroid edge (colored red) partitions the tree into roughly two equally sized subtrees. This edge is removed, and two subtrees are created. This process is recursively repeated on the subtrees until all subtrees contain at most as many sequences as the maximum decomposition size m_a . In this example, $m_a = 3$.

present in the subtree’s leaf set. This is done by selecting the alignment of the sequences from the backbone alignment. Note that the induced alignment may contain some sites that are fully gapped; these sites are removed from the subalignment. The HMM is then computed on the subalignment using HMMBUILD [14] (see Fig. 3.2).

3.2 fHMM Alignment Algorithm

For a given query sequence q , it is scored against each of the HMMs using HMMSEARCH [14], which reports a HMMER “bit score”, a measure of the quality of the match between the query sequence q and the HMM (see Fig. 3.3). The HMM that yields the best bit score is selected, and an extended subalignment is produced by inserting q into the subalignment using

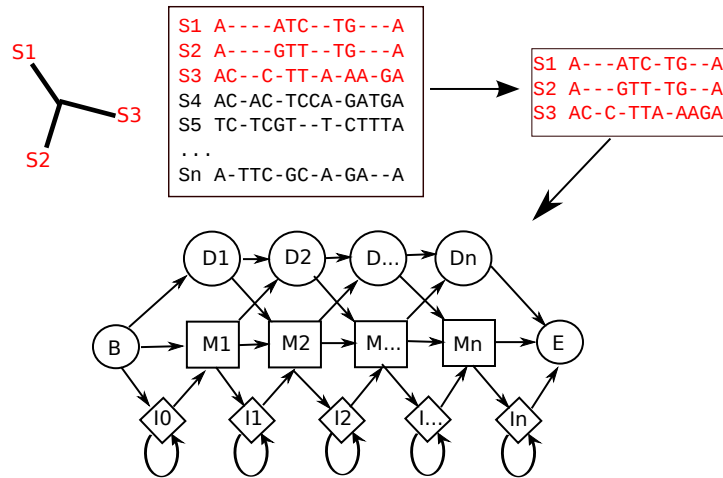


Figure 3.2: Example of building an HMM. Each of the final subtrees produced by the decomposition step defines a subalignment. For a given subtree, an induced alignment is created by taking the alignment of the sequences that are in the leaf set of the tree. Next, an HMM is computed on the induced alignment using HMMBUILD.

HMMALIGN.

Finally, we extend q 's alignment to the entire backbone alignment (see Fig. 3.4). This step is performed through transitivity. We can map the sites in the subalignment back to the original backbone alignment, and thus, any sequence that is aligned to the subalignment can easily be aligned to the backbone alignment by using this mapping. Note that insertion columns generated by the alignment of the query sequence result in insertion columns being added to the backbone alignment.

In the special case where a query sequence resulted in no scores against any of the HMMs (i.e., HMMSEARCH reports the sequence as non-

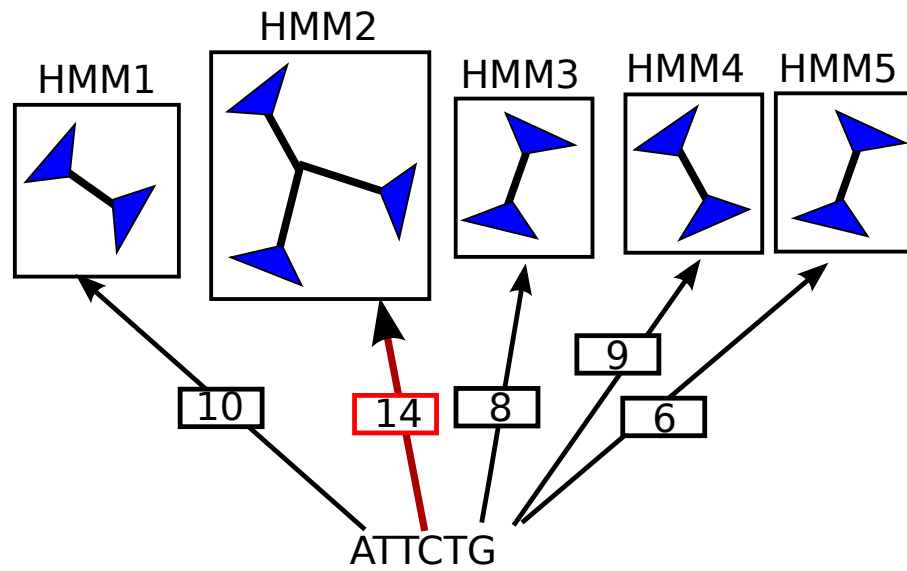


Figure 3.3: Example of aligning a query sequence using the families of HMMs. The query sequence is scored against a collection of HMMs. The HMM that yields the best bit score, HMM-2 in this case, is selected and the query sequence is aligned to that HMM.

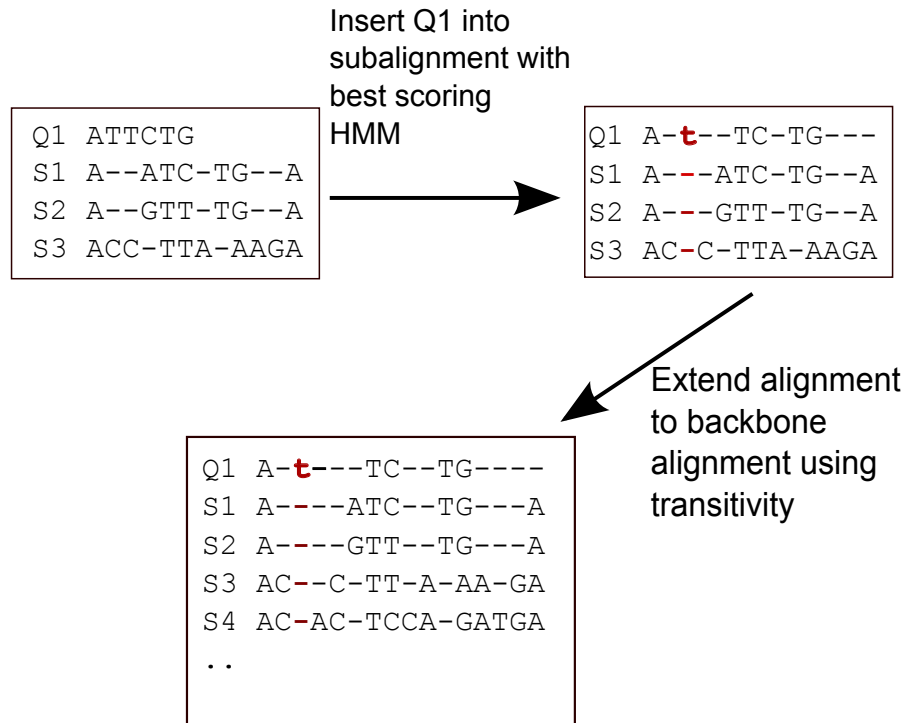


Figure 3.4: Example of extending the alignment of query sequence to the full backbone alignment through transitivity. We know how the sites of the subalignment map back to the original backbone alignment, so we use this information to map the alignment of the query sequence to the alignment on the entire backbone alignment. Note that insertion columns (shown in red) are also inserted into the backbone alignment.

homologous to all HMMs), the query sequence is omitted from the final alignment.

Over the next three chapters, I show how to apply the fHMM to phylogenetic placement, taxonomic profiling and identification, and ultra-large alignment estimation.

3.3 HMMER Commands

HMMER 3.0 [15] is used for building the fHMM, for searching for the best HMM for the alignment of a query sequence, and for inserting the query sequence into the alignment. I provide the HMMER commands used below.

- **HMMBUILD:**

```
hmmbuild --symfrac 0.0 --informat afa --<molecule_type>  
<output_profile> <backbone_alignment>
```

- **HMMSEARCH:**

```
hmmsearch --noali -o <output_file> --cpu 1 -E 99999999 --max  
<input_profile> <query_file>
```

- **HMMALIGN:**

```
hmmalign --allcol --dna <output_profile> <query_file>  
<output_alignment>
```

3.4 Comparison to existing algorithms and methods

The fHMM model has some similarities to two methods used in machine learning: ensemble learning (EL) [12] and mixture of experts (ME) [31]. EL obtains better classification accuracy by taking a consensus of multiple classification algorithms (see [98] for review of ensemble methods in bioinformatics). While both fHMM and EL methods use multiple classifiers or models to obtain better results, fHMM is only superficially similar to EL methods. First, EL methods are used for classification problems, whereas the fHMM is a general approach for representing an MSA. Thus, the utility of fHMM extends beyond classification. Second, EL methods work by obtaining a consensus of all the classification results. This is a very different approach compared to the fHMM algorithm for sequence alignment, as the goal is to find the HMM that best generates the query sequence, and use that HMM for sequence alignment.

Similar to EL methods, ME methods improve predictions (both classification predictions and regression analyses) by using multiple learners (see [99] for review of ME methods). Unlike EL methods, each learner is considered an “expert” in only a local portion of the input space, and thus, the amount to weight given to the output of a learner is dependent on the location of the input in the input space. A gating function is used to determine the domain of each learner in the input space. Thus, similar to the fHMM model, ME methods try to find the best learner or set of learners used to produce the output. Unlike the fHMM model, ME requires supervised training in learning domain for each learner. The fHMM, on the other hand, requires no training to select

which expert to use as it selects the HMM based upon a simple algorithm.

The idea of using multiple HMMs to represent an alignment is not novel. SCI-PHY [10] uses a family of HMMs (called subfamily of HMMs) for functional annotation. The first step in the SCI-PHY pipeline is the identification of the protein subfamilies. The input is a set of unaligned sequences, and the output is a hierarchical forest of HMMs. The process begins by treating each sequence as its own cluster. Next, SCI-PHY joins the two closest related clusters and aligns the clusters, and then computes an HMM on the alignment. The alignment step is performed using pairwise sequence alignment if it is aligning two sequences, and by using the HMM computed on the clusters if it is aligning an MSA to an HMM. SCI-PHY repeats this process until there is one cluster left, or the cost of joining two divergent clusters is too high. Thus, this results in a hierarchical forest of HMMs, each tree representing a protein subfamily.

Once the protein subfamilies have been identified, the subfamily of HMMs can be constructed from each protein subfamily. The subfamily HMM construction algorithm requires an MSA (found in the first step) and a decomposition of the MSA into subalignments. However, SCI-PHY uses a fundamentally different approach to construct the subfamily of HMMs. First, a master HMM is constructed from the input MSA. Each sub-HMM will have the same architecture as the master HMM (identical insertion, deletion, and match states and identical transition probabilities between the states). This allows for conserved columns in the original MSA to continue to be preserved

in subalignments, and the all-gapped columns in the subalignments to be able map to match states in the master HMM. The emission probabilities of the sub-HMM for the conserved columns and all-gapped columns are identical to the emission probabilities in the master HMM. For all other columns, the emission probabilities are estimated from the amino acid distribution of the columns in the subalignment.

There are several fundamental differences between fHMM and SCI-PHY. First, given a set of unaligned sequences, SCI-PHY builds a forest of trees to generate the protein subfamilies, whereas the fHMM approach uses a single tree estimated from the alignment on the sequences. Second, SCI-PHY requires, as input, the alignment decomposition of the subfamily, whereas fHMM uses a centroid edge decomposition to generate the alignment subsets. Finally, SCI-PHY requires that the architecture of the sub-HMMs matches the HMM computed on the MSA of the protein subfamily. The HMMs in fHMM are computed using only from the subalignments, and require no knowledge about the original MSA.

Chapter 4

SEPP: SATé-enabled phylogenetic placement

In this chapter, I show the application of the fHMM to the phylogenetic placement problem. As mentioned in Chapter 2, a phylogenetic placement method can be defined by the alignment method used to insert the query sequence into a backbone alignment, and the placement method used to insert the query sequence into the backbone tree. One application of profile HMMs is on the alignment step for the phylogenetic placement problem. For example HMMALIGN+pplacer computes a profile HMM on the backbone alignment, and the query sequences are inserted into the backbone alignment using the HMM.

I will show, however, that the accuracy of using a single HMM, such as in HMMALIGN+pplacer, degrades on evolutionarily divergence datasets. I present a new software called SATé-enabled phylogenetic placement [55] (SEPP) that uses the fHMM as a boosting technique for HMMALIGN and pplacer. Unlike HMMALIGN+pplacer which uses a single HMM, SEPP uses multiple HMMs to represent the backbone alignment. SEPP produces more accurate placements than HMMALIGN+pplacer and PaPaRa+pplacer on evolutionary divergent datasets, and is more computationally efficient, both in

terms of peak memory usage and running time when placing on very large backbone trees. I show that SEPP can be parametrized for speed or accuracy, depending on the application. These results show the advantages of a family of HMMs for representing a multiple sequence alignment, and form the basis of the remaining methods that I present in my dissertation.

In Section 4.1, I describe the SEPP algorithm. In Section 4.2, I describe the simulation study designed to evaluate the performance of SEPP. Section 4.3, I present results comparing SEPP with two different techniques for phylogenetic placement, which show that SEPP outperforms the other methods on hard datasets and is significantly faster and more computationally efficient on datasets with large backbone trees. Finally, in Section 4.4, I present possible ways of improving SEPP, as well as outlining extensions of SEPP toward taxonomic identification and profiling and ultra-large alignment estimation.

SEPP was developed together with Siavash Mirarab and Tandy Warnow, was presented at the Pacific Symposium on Biocomputing 2012, and was published in [55].

4.1 SEPP Algorithm

SEPP is a meta-method for existing methods for the two steps of phylogenetic placement (computing the extended alignment and placing the query sequence into a tree). SEPP has two stages of decomposition: a placement decomposition step, and an alignment decomposition step. The placement decomposition step decomposes the backbone sequence set into placement sub-

sets. The alignment decomposition step decomposes the placement subsets into fHMMs using the technique described in Chapter 3. To align and place a query sequence, the query sequence is scored against every HMM in each fHMM, and the HMM with the best bit score is selected. The query sequence is inserted into the subalignment that generated the best HMM using the base alignment method. Finally, the placement subset that generated the fHMM is selected, and the query sequence is inserted into the placement subtree using the base placement method. The placement location in the subtree is then used to find the placement location in the original backbone tree.

More formally, the input to SEPP consists of

- the backbone tree T and alignment A for the full-length sequences and a query sequence q , and
- positive integers a and p , with $p \geq a$,
- a base alignment method for aligning the query sequence to a multiple sequence alignment of full-length sequences, and
- a base placement method for inserting the query sequence into a tree, given the extended alignment that includes the query sequence.

The output of SEPP is the placement of q into the backbone tree T .

The default base methods for SEPP is HMMALIGN for producing the extended alignment and pplacer for inserting the query sequence into the backbone tree.

I now show how SEPP uses the parameters a and p to compute the extended alignment and placement of a query sequence into the tree (see Fig. 4.1 for example).

- Using the centroid decomposition for generating the fHMM, SEPP recursively divides the set of taxa in the tree T into disjoint subsets of size at most p . These subsets are called the “placement subsets.”
- SEPP computes an fHMM on each placement subset as described in Chapter 3 with the maximum decomposition size set to a , and the input alignment and tree set to the subalignment and subtree induced by the placement subset.
- SEPP uses the fHMM alignment algorithm to align the query sequence. The query sequence is scored against each the HMM in each fHMM to find the HMM that produces the highest bit score. Next, the query sequence is inserted into the subalignment that generated the HMM to produce an extended alignment. By default, the alignment method used is HMMALIGN.
- SEPP selects the placement subset that generated the fHMM that contains the best scoring HMM, and pplacer is used to insert the query sequence q into the subtree of the backbone tree induced by the placement subset using the extended alignment. Finally, the location of q in the subtree is used to insert q into the backbone tree T on the entire set of taxa.

Thus, the two parameters a and p control the behavior of SEPP.

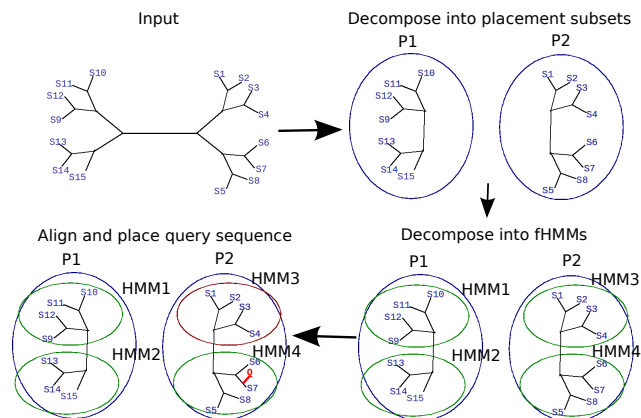


Figure 4.1: Example of the SEPP pipeline with $a=4$ and $p=8$. The input is a backbone alignment and tree on the set of full-length sequences and a query sequence. The first step is to decompose the backbone tree into placement subsets of at most 8 sequences using the centroid decomposition, producing 2 placement subsets in this example. The next step is to decompose each placement subtree into alignment subsets of at most 4 sequences, producing 4 HMMs in this example. The query sequence is aligned to the the HMM that produces the best match (HMM3 in this example), and is placed within the placement subset that contained the best scoring HMM (placement subset P2 in this example).

4.2 Performance Evaluation

In order to evaluate SEPP's performance, I compared SEPP versus HMMALIGN+pplacer and PaPaRa+pplacer on both empirical and simulated datasets.

I studied performance of these phylogenetic placement methods on 61

sequence datasets¹. I included 20 simulated 1000-taxon datasets that have evolved with substitutions and indels from each of three different model conditions (M2, M3, and M4), each with the “medium” gap length distribution (see Liu et al.[46] for these data). The three model conditions are chosen such that one dataset is hard, one is moderate, and one is easy. Because these are simulated datasets, the true alignment and true tree are known for each datasets.

I also used a large bacterial dataset, 16S.B.ALL, with 27,643 16S rRNA sequences, originally taken from the Gutell Comparative Ribosomal Website (CRW)[11], and also studied by Liu et al.[43]. This dataset has a curated alignment based upon confirmed secondary (and higher-order) structures, which are highly reliable. I use a ML bootstrap tree as the curated tree for this dataset, retaining only those branches with bootstrap support above 75%[43]. Thus, the 16S.B.ALL dataset has a curated tree and alignment as well.

Each dataset was randomly divided into two subsets of equal size, with one subset (S) used to define the backbone alignment and tree, and the other subset (R) used to produce the query sequences. These query sequences are created by taking substrings of normally-distributed lengths (from two distributions, described below), and with the start positions chosen uniformly at random.

Two categories of reads are generated for each sequence in the M2, M3,

¹All datasets used in this study are available at <http://www.cs.utexas.edu/~phylo/datasets>.

and M4 datasets: “long” reads, with a mean length of 250 and a standard deviation of 60, and “short” reads, with a mean length of 100 and a standard deviation of 20. A total of 10 fragmentary sequences are generated for each sequence, with half long and half short. Since these datasets each include 500 reference and 500 non-reference sequences, this process yields 2500 short and 2500 long reads per dataset. In summary, each M2, M3, and M4 dataset has a reference tree and alignment with 500 taxa and a total of 5000 fragmentary sequences, of which half are “short” and half are “long”.

For the 16S.B.ALL biological dataset, I create two categories of reads, with length distributions identical to those of simulated datasets. This dataset contains 27,643 taxa, of which I use 13,822 sequences for the backbone tree, leaving me with 13,820 sequences for creating fragmentary reads. For each of these 13,821 sequences, I generated one fragmentary sequence, randomly choosing between the long and short distributions. Thus, for this dataset the backbone tree and alignment has 13,822 taxa, and there are 13,821 fragmentary sequences.

The sequences in S are used to create two backbone alignments and trees, as follows. For sets S that are produced by simulating sequence evolution, I have the true alignment and the true tree. I restrict each of these (which have 1000 taxa) to the subset of 500 full-length sequences, and then run RAxML on the resultant tree/alignment pair in order to optimize the branch lengths and GTR+Gamma parameters. This produces the first alignment/tree backbone. The second backbone alignment/tree pair is produced by running

SATé on the set of full-length sequences.

For the 16S.B.ALL dataset, I use the curated alignment for the dataset and run RAxML on the alignment to produce a binary tree. I then restrict the tree to the subset of 13,822 sequences, and optimize the branch lengths and GTR+Gamma parameters on the tree using RAxML. This produces the first backbone alignment/tree pair. I use SATé on the subset of 13,822 full-length sequences to produce the second.

I used SATé to produce these estimated alignment/tree pairs because SATé produces more accurate alignments and trees than most two-phase method (where an alignment is first estimated and then a tree computed on that alignment) for these datasets[43]. I used SATé-2, the new algorithm design for SATé, for these analyses; this produces an alignment and an ML tree on the alignment estimated using RAxML. For the 16S.B.ALL dataset, I used FastTree[65] within SATé-2 in each iteration, and finished with RAxML in order to produce optimized GTR+Gamma parameters on the final tree.

I classify each query sequence for its likely difficulty in phylogenetic placement as follows. I use HMMER to produce a profile HMM on the reference alignment, and then to classify the query sequences with respect to the profile HMM using HMMSEARCH. The fragmentary reads are classified as easy to align (“easy”) if the obtained E-value is less than 10^{-5} , and as “hard” otherwise. Among the hard reads, there are some reads for which HMMER does not report any E-value due to default filtering settings of HMMER. I classify such reads as “very hard” reads. In earlier phylogenetic placement

studies, the hard fragments are excluded [52]; however, my study does not automatically eliminate hard fragments. Many very hard reads are able to be placed by SEPP because the reads will receive E-values with respect to one of the HMMs in the fHMMs. Those that fail to be placed at all by SEPP are removed from the experimental study; this process removes 9 from all the simulated datasets together and 5 from the biological dataset.

Table 4.1: Dataset statistics: I present statistics for the true alignments for the simulated datasets (M2, M3, M4) and statistics for the curated alignment on the biological dataset, 16S.B.ALL. However, a small number of query sequences are deleted from some of the runs.

Dataset	Type	Size backbone	Num generated query seqs	Avg p-dist	Max p-dist	% gap
M2	sim	500	5000	0.68	0.76	67
M3	sim	500	5000	0.66	0.74	53
M4	sim	500	5000	0.50	0.60	51
16S.B.ALL	emp	13,822	13,820	0.21	0.52	74

Table 4.1 shows various statistics for the true or curated alignment of the datasets included in our study. The p-distance is the fraction of sites within an alignment in which two sequences are different and “% gaps” is the percentage of gaps within the alignment. The empirical statistics show that the datasets vary substantially in terms of evolutionary distances, with datasets from model M2 having the largest evolutionary distances and 16S.B.ALL having the smallest.

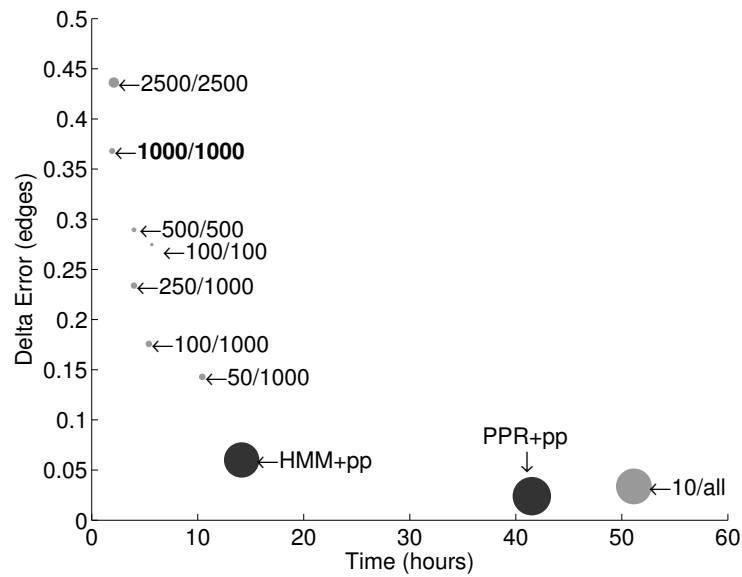
Measurements. I measure placement accuracy (averaged over all the query sequences), running time, and peak memory usage, for each method on each dataset. For the simulated datasets I report averages for these measurements over the 20 replicates in each model condition.

Computational aspects. I report the running times and peak memory usage, each measured separately for the computation of extended alignments and placement of query sequences. These reported values are for alignment and placement of all query sequences in each set. Since some query sequences are deleted from the study as they cannot be placed, the total number of query sequences is slightly smaller than the number generated. Thus, results for each simulated model condition are for 99997-100000 query sequences (20 replicas, each with 5000 query sequences), results for 16S.B.ALL are for 13,819-13,821 query sequences. Due to memory requirements of PaPaRa and pplacer, 16S.B.ALL experiments are run on a Linux machine with 16 cores and 256GB of main memory. The results for simulated datasets are obtained on a heterogeneous condor cluster.

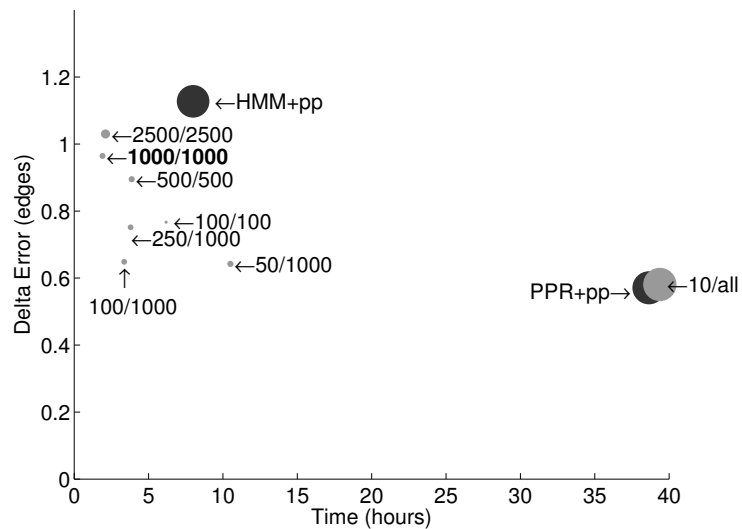
4.3 Results

4.3.1 Algorithm design experiments

Figures 4.2 and 4.3 show results where I vary the two algorithmic parameters a and p . Note that decreasing a to 50 (and sometimes to 10) and increasing p tends to improve the placement accuracy, but at a running time

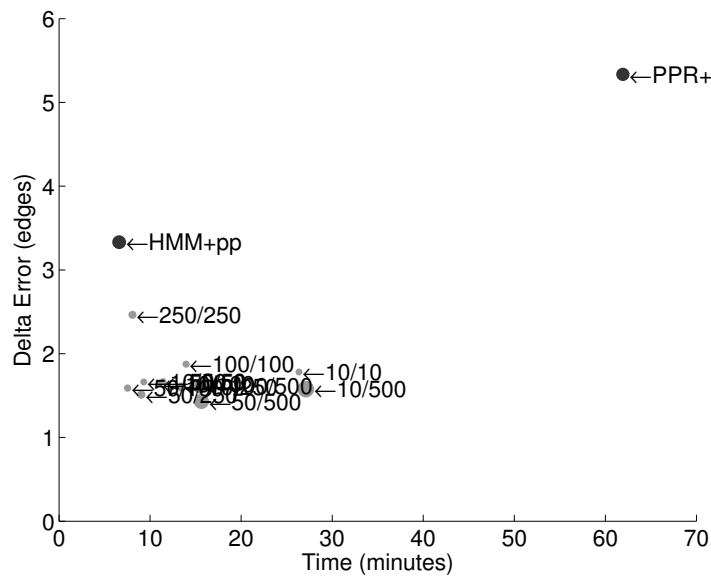


(a) 16S.B.ALL, Curated backbone

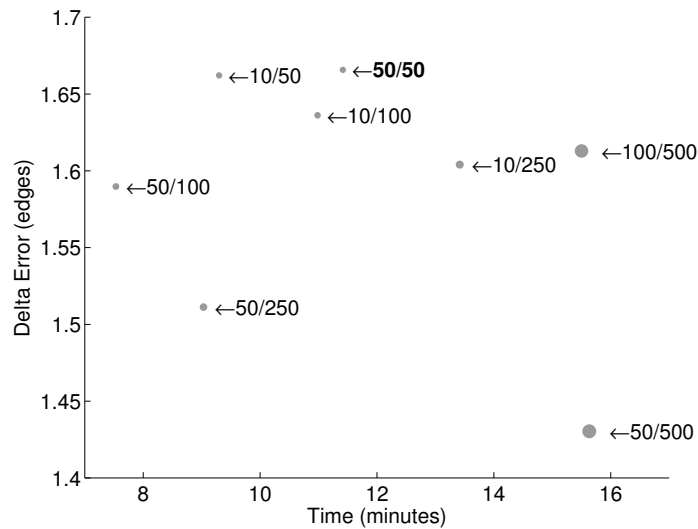


(b) 16S.B.ALL, SATé backbone

Figure 4.2: Scatter plot of delta error (x) versus time (y) versus memory (circle diameters). The symbol “ a/p ” refers to SEPP(a,p), where a is the alignment subset size and p is the placement subset size. The default setting is 1000/1000 for 16S.B.ALL; these points are bold-faced. HMM+pp and PPR+pp are HMMER+ppplacer and PaPaRa+pp. Note that the default setting for SEPP is far from optimal, with other settings providing better accuracy (and in some cases also better speed).



(a) M2, SATé backbone



(b) M2, SATé backbone - most accurate settings

Figure 4.3: Scatter plot of delta error (x) versus time (y) versus memory (circle diameters). The symbol “ a/p ” refers to $SEPP(a,p)$, where a is the alignment subset size and p is the placement subset size. The default setting is 50/50 for M2; these points are bold-faced. HMM+pp and PPR+pp are HMMER+pplacer and PaPaRa+pp. Note that the default setting for SEPP is far from optimal, with other settings providing better accuracy (and in some cases also better speed).

cost. Also, bigger improvements in accuracy are obtained by decreasing a than by increasing p . However, for most conditions, there is a wide range of parameter settings in which the differences in placement error are quite small (often less than half an edge), and within this collection there can be significant differences in running time.

The general principles are that smaller alignment subset sizes typically results in better placement accuracy, at the cost of increased running time, and that larger placement subset sizes results in better placement accuracy, at the cost of increased running time and peak memory usage. Using smaller alignment subset sizes improves accuracy as the small subsets would be less likely to contain many highly evolutionary divergent sequences. However, every time a subset is divided in half, there is twice as much work in finding the best alignment subset. Using larger placement subset sizes improves accuracy as the true placement is likely to be in the placement subset, however, requires more time and memory to check all possible placement locations.

To set the default parameters, I sought a setting that worked reasonably well with respect to both running time and placement accuracy. Setting $a = p = 1000$ for the 16S.B.ALL datasets and $a = p = 50$ for the simulated datasets produced good results. These settings correspond to setting the subset sizes to about 10% of the number of taxa in the backbone tree. Note, however, that setting $a=p=50$ is by no means optimal for the M2 model condition (four other settings, with a at most 50, have less error and complete faster). Similarly, setting $a=p=1000$ is the fastest for the 16S.B.ALL datasets,

but more accurate results can be obtained with other settings (each with a below 1000) for a running time cost. Note that while setting $a=p$ implies that only a single HMM is used to represent each placement subset, multiple HMMs are still being used to represent the backbone alignment.

4.3.2 Comparisons using the Default Setting for SEPP

I present results for PaPaRa+pplacer, HMMALIGN+pplacer, and the default setting for SEPP where we set $a=p$ to approximately 10% of the number of taxa in the backbone tree. This yields parameters 50/50 for the simulated datasets (backbone trees have 500 taxa) and 1000/1000 for the 16S.B.ALL dataset (backbone trees have 13,822 taxa).

4.3.3 Results on Simulated Datasets

The simulated datasets have backbone trees with 500 sequences and fairly high rates of evolution, with M2 having the highest rate and M4 having the lowest rate (Table 4.1). Placement error rates were impacted by the model, so that the missing branch rate for all methods is higher on model M2 than on model M3, and higher on model M3 than on model M4 (Table 4.2). Not surprisingly, absolute error rates are lower with the true alignment and tree than with the SATé alignment and tree. These trends also held for PaPaRa and SEPP.

Figure 4.4 and Table 4.2 show results for PaPaRa+pplacer, HMMALIGN+pplacer, and SEPP(50,50) (i.e., SEPP ran with the default setting on

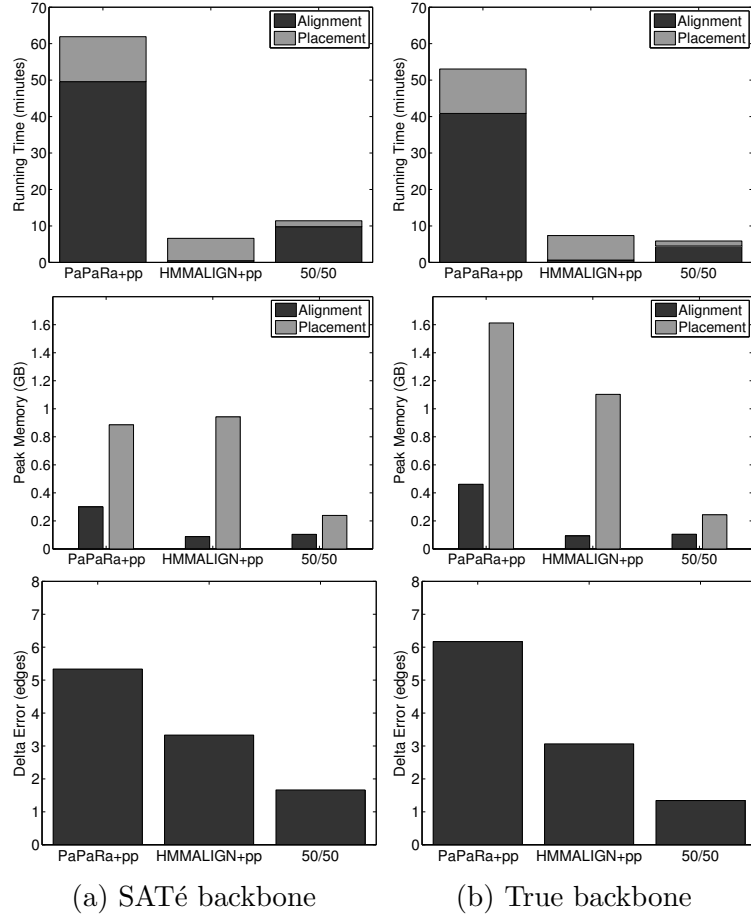


Figure 4.4: Results on simulated datasets for model M2. I show running time (top), peak memory usage (middle), and average number of additional missing branches per query sequence (bottom). Results for the SATé backbone alignment and tree are on the left, and results for the true backbone alignment and tree are on the right. The SATé backbone tree has 12.1% missing branch rate and the backbone tree based upon the true alignment has 0.09% missing branch rate. The number of additional missing branches shown (bottom) is the increment above that amount.

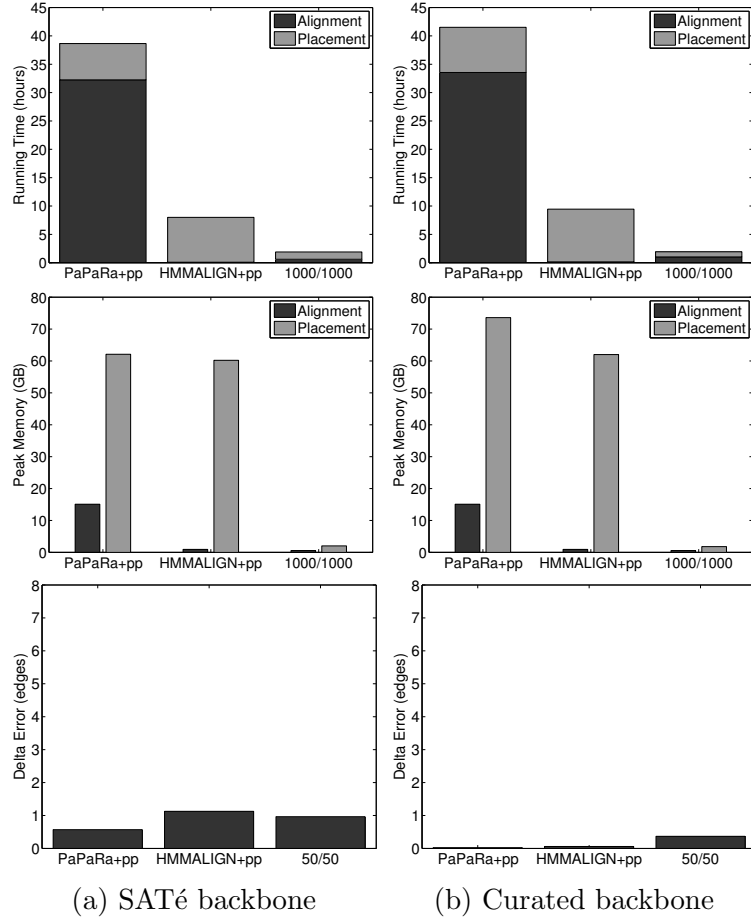


Figure 4.5: Results on 16S.B.ALL. I show running time (top), peak memory usage (middle), and average number of additional missing branches per query sequence (bottom). Results for the SATé backbone alignment and tree are on the left, and results for the curated backbone alignment and tree are on the right. The SATé missing branch rate is 7.64%, and the missing branch rate for the backbone tree defined by the true alignment is 1.835%. The number of additional missing branches shown (bottom) is the increment above that amount.

Table 4.2: Mean delta-error for all query sequences. I show the mean delta-error for each method on each model condition for all the query sequences. Count refers to the number of query sequences processed and placed, HMM refers to HMMALIGN+pplacer, PPR refers to PaPaRa+pplacer, and SEPP refers to SEPP run in default mode.

	All reads			
	bio.	M2	M3	M4
	SATé Backbone			
count	13819	99998	99999	99999
HMM	1.1	3.4	1.4	0.3
PPR	0.6	5.4	3.6	0.4
SEPP	1.0	1.7	1	0.4
	True or Curated Backbone			
count	13818	99997	99999	99999
HMM	0.0	3.2	1.2	0.0
PPR	0.0	6.2	3.8	0.2
SEPP	0.4	1.4	0.7	0.1

this model of $a = p = 50$). Note that SEPP(50,50) has the lowest delta-error of the three methods by far, followed by HMMALIGN+pplacer, and then by PaPaRa+pplacer. Furthermore, the differences are substantial. The methods are clearly also distinguished by running time and peak memory usage. HMMALIGN+pplacer is the fastest, SEPP(50,50) is somewhat slower, and PaPaRa uses much more time. Both PaPaRa+pplacer and HMMALIGN+pplacer use more memory than our method.

Results for M3 (see Table 4.2) are quite similar to M2, as HMMALIGN+pplacer was much more accurate than PaPaRa+pplacer and SEPP(50/50) produced more accurate placements than HMMALIGN+pplacer. However, the gap between SEPP(50,50) and HMMALIGN+pplacer.

Table 4.3: Mean delta-error for different categories of query sequences. I show the mean delta-error for each method on each model condition, as a function of the level of difficulty for the query sequence, as estimated by HMMER. Count refers to the number of query sequences processed and placed, HMM refers to HMMALIGN+pplacer, PPR refers to PaPaRa+pplacer, and SEPP refers to SEPP run in default mode.

	Hard reads				Very hard reads			
	bio.	M2	M3	M4	bio.	M2	M3	M4
	SATé Backbone							
count	104	79510	58924	3989	21	63613	40495	844
HMM	2.4	4.2	2.3	0.7	3.9	5.2	3.2	1.5
PPR	0.8	5.8	4.4	1.0	0.9	6.2	4.9	1.5
SEPP	2.4	2.0	1.4	0.8	3.8	2.4	1.8	1.0
	True or Curated Backbone							
count	104	79511	58924	3989	21	63614	40495	844
HMM	0.5	4.0	2.0	0.2	2.2	5.0	2.9	0.9
PPR	0.1	6.7	4.7	0.5	0.0	7.1	5.2	0.9
SEPP	1.1	1.7	1.1	0.4	1.5	2.0	1.4	0.5

LIGN+pplacer was reduced to only half an edge. On M4 (see Table 4.2), however, the relative performance between SEPP(50,50) and HMMALIGN+pplacer depended on the backbone tree. For the SATé alignment/tree, SEPP(50,50) was more accurate but slightly slower than HMMALIGN+pplacer. For the true alignment/tree, HMMALIGN+pplacer was somewhat more accurate and took less time. Note that the difference in placement accuracy between SEPP(50,50) and HMMALIGN+pplacer was extremely small - less than one-ninth of an edge for both backbones.

4.3.4 Results on 16S.B.ALL

The datasets based upon 16S.B.ALL, presented a different kind of challenge. Each dataset had 13,820 query sequences and a backbone tree with 13,822 sequences. Thus, these datasets had much larger backbone trees, but the backbone trees and alignments reflected lower rates of evolution.

The default setting for SEPP on this dataset is $a=p=1000$; therefore, I ran SEPP(1000,1000) for both backbones. Results on these datasets are shown in Figure 4.5. Note that PaPaRa+pplacer provides a small improvement in placement accuracy (slightly more than half an edge) in comparison to the other methods. However, PaPaRa+pplacer is enormously computationally intensive, using 40 hours to analyze these data, much longer than either other method. Also, HMMALIGN+pplacer and PaPaRa+pplacer have very large peak memory usage, near or above 60GB on both backbone trees. Thus, PaPaRa+pplacer is computationally extremely intensive, and possibly the improvement in placement accuracy is insufficient given the additional running time costs.

A comparison of SEPP(1000,1000) to HMMALIGN+pplacer shows that both have extremely good placement accuracy, with delta-error approximately one edge for both methods on the SATé backbone tree and well under half an edge on the curated backbone tree. HMMALIGN+pplacer produces more accurate placements than our method for the curated backbone and SEPP(1000,1000) produces more accurate placements for the SATé backbone, but the differences between the two methods are small in both cases (less than

a third of an edge). The methods are, however, distinguished by their computational requirements, as HMMALIGN+pplacer is much slower (at least 4 times as much time) and uses dramatically more memory (60GB as compared to about 2GB).

4.3.5 Comparing methods on query sequences of different levels of difficulty

Tables 4.2 and 4.3 compares methods in terms of their placement accuracy as a function of the level of difficulty in placing the query sequence, as predicted by HMMER (see the discussion in Section 4.3.6). Note that error increases as the reads become more difficult, as HMMER predicts. I show that SEPP, run in default mode, performs very well in general (as observed earlier) in comparison to HMMALIGN+pplacer and PaPaRa+pplacer, but has a particularly strong advantage on the hard and very hard reads. Interestingly, PaPaRa+pplacer does well on hard and very hard reads for 16S.B.ALL but not on the simulated datasets.

4.3.6 Summary

There are several observations we can make. First, the methods I evaluated for phylogenetic placement—PaPaRa+pplacer, HMMALIGN+pplacer, and SEPP methods—often produce placements that are extremely accurate, increasing the topological error over the input backbone tree by at most an edge (often much less than an edge) on average. Furthermore, while these methods do sometimes have differences in placement accuracy that go beyond

an edge, these differences are sometimes still small enough to be relatively unimportant, compared to the computational cost required to obtain the improved placement accuracy.

However, I did observe conditions in which the differences in placement accuracy were quite large, suggesting that increased effort in placing query sequences correctly was merited. For example, I see big differences in placement accuracy on model M2, resulting in several edges improvement produced by SEPP(50,50) over HMMALIGN+pplacer. The conditions under which accuracy differences are substantial are characterized by large evolutionary distances between some pairs of full-length sequences. I conjecture that in such conditions, the HMMs produced by HMMER on the full set of taxa may not be sufficient to produce highly accurate alignments for the query sequences, and will result in degraded placement accuracy. The technique I introduce here avoids this problem by using HMMER to produce HMMs only on smaller, less diverse, subsets of the taxa. As a result, the HMMs may produce more accurate alignments to the query sequences, and result in improved phylogenetic placement.

I note the interesting differences between HMMALIGN+pplacer and PaPaRa+pplacer. Only on the slowest evolving dataset, 16S.B.ALL, does PaPaRa+pplacer produces more accurate placements than HMMALIGN+pplacer, while PaPaRa+pplacer has substantially less accurate placements for the faster evolving datasets. This is consistent with the need to estimate transition state matrices on each edge, an estimation that may only

be highly accurate under sufficiently low rates of evolution.

Furthermore, these methods differ dramatically with respect to running time, with PaPaRa+pplacer much more computationally intensive than HMMALIGN+pplacer and the default setting for SEPP, thus suggesting that PaPaRa+pplacer is unlikely to be useful in large-scale metagenomic analyses.

The comparison between HMMALIGN+pplacer and SEPP is more complex, because SEPP is parameterized by the two algorithmic parameters a and p . Here I see that some very simple settings for these parameters ($a=p$, both set to about 10% of the number of taxa in the backbone tree) produces very fast results with generally very good accuracy, coming close to the accuracy obtained by the best methods (or improving on them), but in a fraction of the time. Other settings for the parameters can improve the placement accuracy but require greater running time and memory usage.

4.4 Conclusion and future work

In this chapter, I presented SEPP, a technique for boosting the accuracy and/or speed of a phylogenetic placement method. I showed that SEPP using fHMM for alignment and pplacer for placement resulted in improvements in placement accuracy and/or running time. Given the plans to analyze millions of reads, the speed-ups that SEPP provides could be essential to providing scalability for phylogenetic placement methods. In addition, I showed that using fHMM resulted in comparable or better accuracy than using HMM for alignment.

I plan to explore other methods for estimating the extended alignment. For example, improved accuracy might be obtained by coupling SEPP with PaPaRa for those cases where the backbone tree and alignment has slow evolutionary rates to enable PaPaRa to produce highly accurate extended alignments. Another potential method to examine is Mafft-profile [35]. Mafft-profile takes in a backbone alignment and a sequence of query sequences and aligns each query sequence to the backbone alignment. Mafft-profile can be run under an accurate setting (“addfragments” and “L-INS-I”), however, the most accurate setting can only be run on a backbone alignment of 1000 sequences. More accurate placements may be obtained if Mafft-profile is used within SEPP, using the decomposition technique to allow Mafft-profile to scale to larger datasets.

Based on the placement of the query sequence, the evolutionary relationship between the query sequence and the sequences in the backbone alignment can be inferred. Thus, SEPP can be used to taxonomically identify unknown reads based upon the placement of the sequence in the backbone tree. In Chapter 5, I will show a modification of SEPP that results in improved classification sensitivity and precision over the single HMM approach.

Chapter 5

TIPP: Taxonomic identification and phylogenetic profiling using families of Hidden Markov Models

In the previous chapter, I presented SEPP, a method for phylogenetic placement using families of HMMs. I presented results that show SEPP has improved phylogenetic placement accuracy on evolutionarily divergent datasets. In this chapter, I will show how SEPP can be used for taxonomic identification, and how SEPP has high sensitivity in classifying reads, but at the cost of high false positive classifications on reads from novel taxa or on reads with high rates of error.

In this chapter I will present TIPP, a modification of SEPP that incorporates statistical support measures to control the precision and sensitivity of classification. I will show that TIPP classifies more fragments correctly compared to leading taxonomic identification methods, and that TIPP maintains high precision and sensitivity under difficult conditions. In addition, I show experimental results that TIPP also improves taxonomic profiling accuracy.

Section 5.1 shows how phylogenetic placement can be applied toward taxonomic identification and profiling and present results on taxonomic iden-

tification using SEPP. In Section 5.2, I describe TIPP, a modification of SEPP for taxonomic identification and profiling. In Section 5.3, I describe the simulation study designed to evaluate the performance of TIPP toward taxonomic identification and profiling. In Section 5.4, I present the results comparing TIPP for taxonomic identification and taxonomic profiling. I show that TIPP outperforms other taxonomic identification methods under difficult conditions and that TIPP generally results in better profiles than leading profiling methods. Finally, in Section 5.5, I discuss possible ways of improving TIPP, and future studies using TIPP.

5.1 Taxonomic Identification through Phylogenetic Placement

One approach toward taxonomic identification is through phylogenetic placement. The evolutionary relationship between the the query sequence and the backbone sequences can be inferred from the placement location. For example, in Figure 5.1, $Q1$ is placed closest to Species $A1$, and thus, it can be inferred that $Q1$ is more closely related to Species $A1$. Similarly, $Q2$ is more closely related to Species $A1$ and $A2$ than to Species $B1$ and $B2$. Thus, one simple technique for identifying a query sequence is to classify it by the lowest common ancestor (LCA) of its sibling leaf nodes. Thus, $Q1$ would be classified as Species $A1$ (its only sibling leaf node is Species $A1$) and $Q2$ would be classified as Genus A (its sibling leaf nodes are Species $A1$ and Species $A2$).

Using this approach, I compared SEPP and HMMALIGN+pplacer for

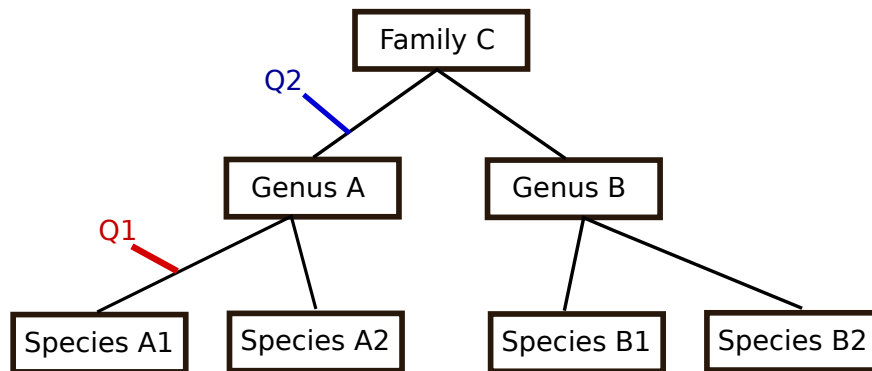


Figure 5.1: Taxonomic classification using phylogenetic placement. The leaf nodes of the rooted backbone tree are labeled with the species name. Each query sequence is placed onto an edge in the backbone tree and is classified by the LCA of its sibling leaf nodes.

taxonomic identification under a *leave-species-out* experiment. Under a leave-species-out experiment, the species of the query sequence is removed from the backbone alignment and tree, simulating the classification of a novel species. Thus, while the species of the query sequence cannot be correctly identified, the remaining taxonomic lineage (genus, family, etc...) can still be correctly classified. The fragments were simulated under differing models and rates of sequencing error.

Figure 5.2 shows that SEPP is more sensitive than HMMALIGN+pplacer under the hardest model condition (“454_3”), classifying more fragments correctly, especially at the phylum level. Both methods tend to classify the large majority of the fragments, leaving very few fragments unclassified. This results in a very high false positive rate, especially under more difficult conditions.

To understand why this is the case, it’s important to note that pplacer outputs multiple possible locations for the placement of each query sequence. Each placement has an associated likelihood weight ratio. However, when HMMALIGN+pplacer or SEPP is used for taxonomic classification, only the placement with the likelihood weight ratio is used, ignoring the fact that there may be other placements with comparable weight. This was one of the key insights in the development of TIPP. By taking into account different sources of uncertainty, both in the alignment and placement steps, the false positive rate could be reduced.

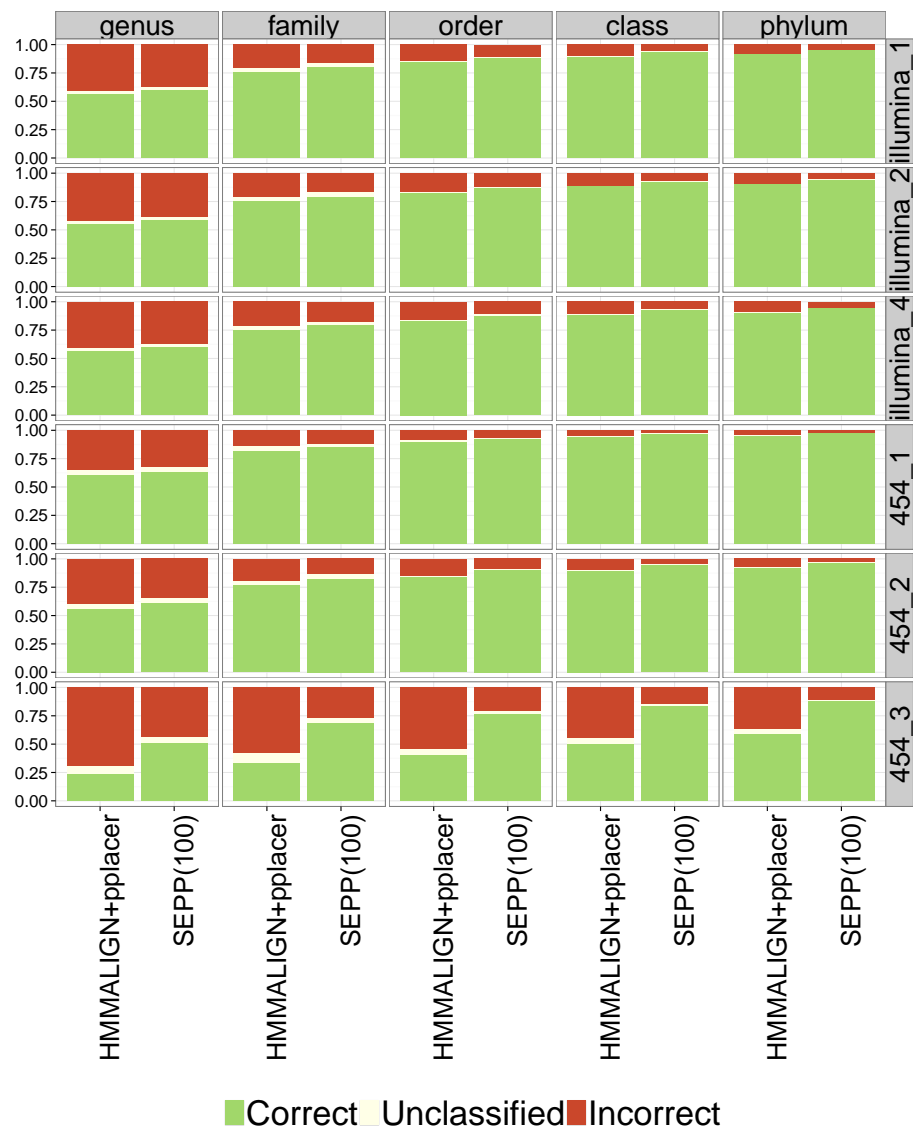


Figure 5.2: Leave-species-out experiments comparing SEPP and HMMALIGN+pplacer taxonomic classification accuracy on the rpsB gene under sequences simulated under different error model conditions. Fragments were simulated with either Illumina-like or 454-like errors, and with varying rates of error. Models denoted with a “1” have the lowest rates of error, and models with a “3” or “4” have the highest rates of error. SEPP is run using an alignment decomposition size of 100 and placing on the entire backbone tree.

5.2 TIPP Algorithm

TIPP is an extension of SEPP designed specifically for taxonomic classification and profiling of metagenomic reads. TIPP is a marker-based method and can only classify reads that originate from one of the gene markers. Assuming that the reads have already been binned to the specific gene markers, TIPP uses the SEPP algorithm to place the query sequences into the backbone tree, but with some modifications.

In contrast to SEPP which uses a single extended alignment for placement of a query sequence, TIPP selects multiple extended alignments and computes multiple placements for a query sequence. This reduces the potential for false positive classification (and hence improves the precision of the taxon identification) by taking into account uncertainty in the alignment and placement steps. TIPP allows the user to specify minimum statistical support levels for both the alignment and the placement steps. Given those thresholds, TIPP computes a set of alignments and placements that suffices to meet the required statistical support levels, and returns the LCA (least common ancestor) of these placements as the final placement. The result is a statistically-based method that can be tuned for precision or recall, and which has better recall (even for its more conservative setting) compared to other marker-based methods such as HMM+pplacer and MetaPhyler.

As a pre-processing step, TIPP builds the backbone alignment and tree on the full-length gene sequences for each marker gene using SATé [46, 47]. Next, TIPP uses a statistical pipeline to perform taxon identification,

as follows. For simplicity, I will first describe the algorithm for classifying fragments that have already been binned to the marker genes. I will later explain how to extend this algorithm to the case where the fragments have not yet been binned.

TIPP’s technique for taxonomic identification for a single marker gene. I now describe in greater detail the technique used by TIPP to taxonomically identify the fragments matching a given marker gene. The input is (1) a set Q of fragmentary sequences from a single gene, (2) a set S of full-length sequences for the same gene, (3) a backbone reference alignment A and backbone reference tree T estimated on S , and (4) a refined taxonomy T^* .

Parameter settings for default TIPP. I describe the simplest use of TIPP, which is used for all cases where n , the number of full-length sequences obtained for the marker gene, is not very large (see Section 5.2.3 for a description of how I modify TIPP when n is very large). In this simple version of TIPP, I do not constrain the portion of the taxonomy into which the fragment can be inserted. I also need to set m_a , the maximum alignment subset size, whether EPA or pplacer is used, and the statistical support thresholds s_a and s_p for alignment and placement support, respectively.

I now describe how TIPP performs a taxonomic identification of a single query sequence q :

Step 1: Build fHMM. TIPP decomposes the reference alignment A and tree

T into alignment subsets with at most m_a leaves using the fHMM decomposition technique. This produces a partition of S into subsets S_1, S_2, \dots, S_k , each of size at most m_a .

Step 2: Compute extended alignments. In contrast to the fHMM alignment algorithm which selects the single extended alignment with the best bit score, TIPP uses the bit scores to select as many extended alignments as necessary to reach the alignment support s_a . Section 5.2.1 describes this process in more details.

Step 3: Placement. For each extended alignment, I use the selected method (EPA or pplacer) to place q into the refined taxonomy T^* . I thus obtain multiple placements and their likelihood weight ratios (a value computed by pplacer and EPA, which I treat as probabilities) for each extended alignment. I combine all placement results of different extended alignments, and normalize the placement probabilities across results from all extended alignments. This results in multiple placements and their associated probabilities for each placement for the fragment.

Step 4: Classification. I assign the statistical support to each node in the taxonomy by adding the probabilities of all placements at or below the edge above the node; this allows us to classify the fragment at all taxonomic levels for which it has support of at least s_p . If $s_p > 0.5$ then this yields a unique taxon identification; otherwise, TIPP outputs multiple identifications, along with their support. The fragment is left unclassified at levels where support of s_p is not reached.

Thus, TIPP has many algorithmic parameters, some determining how the decomposition is run (m_a), and others determining the statistical support thresholds (s_a and s_p) and the taxonomy that is used. In my experiments, I use either the NCBI or the RDP taxonomy (depending on the dataset), restricted to the species in S , SATé to produce the reference alignment and tree on S , and RAxML to refine the specified taxonomy with respect to the SATé alignment.

5.2.1 Alignment Support Calculation

The fHMM alignment algorithm aligns the query sequence against each HMM, which in turn produces a bit score for each alignment. The HMM that produces the best bit score is selected, and the query sequence is inserted in the subalignment that generated the HMM. However, taking the HMM that produced the best bit score ignores the fact that there may be other HMMs with nearly as good bit scores.

Rather than taking the single best alignment with the highest bit score, TIPP takes as many alignments as necessary so that together they provide a total support above a certain threshold. To do this rigorously, I use the HMMER output to calculate the probability that a given fragment is generated by one of the models from a set of models, each associated with an alignment subset. These calculations are all based on the assumptions that 1) alignment subsets are disjoint, so that at most one subset generates the fragment, and 2) the fragment does indeed belong to the gene, so that it is generated by some

alignment subset.

Minimum Alignment Support Threshold. I now show how TIPP computes the probability that a fragment is generated by a set of alignment subsets. For a given HMMER model H and fragment x , HMMER calculates a bit-score, defined as:

$$BS(H) = \log_2 \frac{P(x|H)}{P(x|R)} \quad (5.1)$$

where $BS(H)$ is the bit-score for x on H , $P(x|H)$ is the probability of model H generating fragment x , and $P(x|R)$ is the probability of a random model R generating fragment x . Thus models producing higher bit-scores are more likely to have generated the fragment.

Assuming that a fragment x is generated by exactly one of the HMMs H_1 to H_n (each corresponding to a different alignment subset), the probability that H_i generated x is:

$$P(H_i|x) = \frac{P(x|H_i)P(H_i)}{\sum_{j=1}^n P(x|H_j)P(H_j)}. \quad (5.2)$$

Assuming that uniform prior probability (i.e. $P(H_i) = P(H_j)$), I can rewrite Equation 5.2 as:

$$P(H_i|x) = \frac{1}{\sum_{j=1}^n \frac{P(x|H_j)}{P(x|H_i)}}. \quad (5.3)$$

By Equation 5.1,

$$BS(H_j) - BS(H_i) = \log_2 \frac{P(x|H_j)}{P(x|R)} - \log_2 \frac{P(x|H_i)}{P(x|R)} \quad (5.4)$$

$$= \log_2 \frac{P(x|H_j)}{P(x|H_i)} \quad (5.5)$$

Substituting into Equation 5.3, the result is the formula for computing the probability of H_i using bit-scores:

$$P(H_i|x) = \frac{1}{\sum_{j=1}^n 2^{BS(H_j)-BS(H_i)}} \quad (5.6)$$

Thus, assuming that the bit-scores are sorted such that $BS(A_i) \geq BS(A_{i+1})$ ($i = 1, 2, \dots, n - 1$), to reach a specified threshold s_a , I find the smallest m such that $\sum_{k=1}^m P(H_k|x) \geq s_a$.

5.2.2 Abundance Profile Estimation

The previous description of the TIPP algorithm requires that the fragments have already been binned to the marker genes. However, in a metagenomic shotgun sequencing experiment, the output is reads from all the different genomes in the sample. Thus, reads from the marker genes must first be identified and binned before TIPP can be applied.

To bin the reads, BLAST [3] is used to map the fragments to the marker genes. Only fragments that have been binned to a marker gene are used to estimate the abundance profiles; fragments that fail to match to any of the marker genes are discarded.

Next, TIPP is applied to classify all the reads that have been binned. Some of the reads will fail to be classified at any taxonomic level; these reads are discarded and are not included in the abundance profile estimation. The binned reads that can be classified are then pooled, and the distribution is estimated from the pooled collection.

5.2.3 TIPP on Larger Markers

I modified how I ran TIPP on the bacterial 16S RNA dataset due to the large number of taxa (9197), as follows: I used the SATé alignment as the reference alignment, the refined taxonomy as the reference tree, and rather than placing into the entire refined taxonomy, I used SEPP to decompose the refined taxonomy into both alignment subsets and placement subsets using decomposition parameters of $m_p = 1000$ and $m_a = 100$. Thus, the reads were placed into subtrees of the 16S marker, each of which contains at most 1000 leaves.

5.3 Performance Evaluation

I initially evaluated the impact of algorithmic parameters on taxonomic classification and phylogenetic profiling, based on which I selected default settings for TIPP; these are reported in the Appendix. I then evaluated TIPP in comparison to other phylogenetic profiling methods on a collection of datasets. I also performed experiments evaluating the impact of sequencing error on taxonomic classification, the effect of TIPP's algorithmic parameter settings on

the taxonomic identification accuracy, and finally the ability of different taxonomic classification methods to identify “dark matter” (i.e., sequences that come from novel phyla).

Methods studied. I compared MetaPhyler, MetaPhlAn, PhymmBL, and NBC as abundance profiling methods. TIPP, MetaPhyler, and MetaPhlAn are marker-based methods and can classify fragments from their marker reference dataset. TIPP and MetaPhyler use the same set of universal housekeeping markers that are unlikely to undergo duplication and horizontal gene transfer. MetaPhlAn, on the other hand, selects markers that uniquely identify specific taxonomic groups. NBC and PhymmBL are composition-based methods and can classify fragments originating from any region in the genome.

MetaPhyler classifies every fragment at each taxonomic level and assigns the classification a confidence score. For MetaPhyler, I used a version that classifies a fragment at the most specific classification yielding a confidence score of 90% or higher. I used MetaPhyler version 1.25 (downloaded from <http://metaphyler.cbcb.umd.edu>), an extension of the originally published algorithm that also provides a confidence for each classification. The chosen confidence cutoff of 90% roughly corresponds to a mis-classification rate of 10%, chosen as a reasonable trade-off between precision and recall. For PhymmBL, I classified a fragment at the most specific classification yielding a confidence score of 95% or higher; however, PhymmBL does not give confidence scores at the species level, and so cannot be used to perform taxonomic

identification and abundance profiling at the species level. Finally, NBC gives a confidence score of the fragment matching to a genome. I accepted the classification if the confidence score is above the species threshold suggested by the NBC authors. Thus, a fragment will either be classified at the species level or be completely unclassified. See Section A8.2 for commands used for training and running these methods.

Except where indicated, I used the following “default” settings for TIPP. The alignment subset size m_a is set to 100, and I place fragments into the refined taxonomy (described above) after I compute the extended alignments. In all experiments shown here I use pplacer for the placement step (see Section A3.2 for results on using EPA inside TIPP). The remaining parameters are the alignment subset size m_a and the alignment and placement support thresholds s_a and s_p , respectively. I refer to TIPP with this parameter setting by TIPP(s_a, s_p, m_a). All results shown in this paper use 95% as the alignment and placement support threshold.

Reference marker datasets. In order to classify metagenomic samples, TIPP uses the reference sequence dataset obtained from [43, 44], which consists of 30 phylogenetic marker genes that span the Bacteria and Archaea domains. The marker genes selected were believed to be single copy genes and universally present across the Bacteria domain, making them resistant to horizontal gene transfer and gene duplication. Only species whose genomes have been sequenced were present in the reference dataset. The number of sequences

in each marker ranges from 65 to 1555 sequences, with an average of 1312 sequences per marker gene. See Section A6.1 for the list of marker genes and the empirical statistics of the reference alignments on these datasets.

Simulated taxonomic identification datasets. Datasets used in the taxonomic identification experiments were generated by simulating fragments from biological data and then adding errors for either Illumina or 454 sequencing technologies, varying the error rates from low to high. I used MetaSim [70] to generate fragments with Illumina or 454 errors, starting from the reference datasets of 30 marker genes and the 16S gene. Both 100-bp Illumina-type fragments and 300-bp 454-type fragments were generated, with different levels of error, thus, I have Illumina_1, Illumina_2, and Illumina_4 models, and 454_1, 454_2, and 454_3 models (in the increasing order of error rates). Illumina-type fragments contained only substitution errors, and 454-type fragments contained only indel errors, biased toward insertions.

These error models allow us to explore the impact of varying sequencing error on taxonomic identification, and the higher error models improves the realism of the non-leave-out experiments. These error models range from low amounts of error, with the default average number of error events per fragment, up to 7.6 times the average number of substitutions per fragment (for Illumina data) and up to 4.2 times the number of indels (for 454 data); see the Table A25 for the fragment statistics for the different error models. Current sequencing data, *when properly filtered*, do not exhibit the levels of error

shown in the highest error model conditions (Illumina_4 and 454_3); therefore, this experiment represents a stress test of the methods, testing robustness to increased error in the data, that may indicate performance under future sequencing technologies, or under unfiltered data.

In total, the leave-one-out experiments had 600,000 fragments simulated from the 30 marker genes and 240,000 fragments simulated from the 16S genes. The non-leave-one-out experiments had 600,000 fragments simulated from the 30 marker genes.

Simulated abundance profiling datasets. I used several datasets from previous studies in the abundance profiling experiments. The simulated abundance profiling datasets can be grouped by the complexity of the abundance profiles and the average fragment length. High complexity (HC) datasets have roughly uniform distributions of the species. Low complexity (LC) datasets have staggered distributions of the species; typically low complexity distributions have a power law distribution. Medium complexity (MC) datasets fall in between LC and HC datasets. Datasets either have short average read length (at most 100 nucleotides) or long average read length (200-1000 nucleotides).

I used simulated abundance profile datasets from 4 different studies: the MetaPhlAn HC and LC datasets [74], the FACS HC dataset [83], the FAMeS LC, MC and HC simulated datasets [53], and the WebCarma HC dataset [24]. Of these datasets, only the MetaPhlAn datasets had short sequences. To better examine performance on datasets with short sequences, I

generated Illumina-like reads from the genomes used in the WebCarma and FACS datasets. Finally, the FACS dataset originally contained both human and viral sequences. These were removed from the datasets so that profiling performance was tested only on bacterial and archaeal sequences. Table 5.1 shows the summary of the datasets examined. A more in-depth description of the datasets can be found in Section A7. .

Table 5.1: Summary of all simulated abundance datasets. Complexity refers to the distribution of species in the profile. High complexity datasets have an even distribution of species. Low complexity datasets have a staggered distribution of species. Medium complexity datasets fall in between. Datasets with a “*” were generated by generating Illumina-like reads from an existing abundance profile using MetaSim. Datasets labeled with “DOE-JGI” used fragments generated from genome sequencing projects at the Department of Energy Joint Genome Institute. “Length” refers to the average length of the reads, and “Complex.” refers to the complexity (High, Medium, or Low).

Dataset	# Genomes	Complex.	Seq. Model	Reads	Length
MetaPhlAn HC	100	High	NA	1000000	88
MetaPhlAn LC	25	Low	NA	240000	88
FAMeS HC	113	High	DOE-JGI	116771	949
FAMeS MC	113	Medium	DOE-JGI	114457	969
FAMeS LC	113	Low	DOE-JGI	97495	951
FACS HC	19	High	454	26984	268
FACS HC Illumina*	19	High	Illumina	300000	100
WebCarma	25	High	454	25000	265
WebCarma Illumina*	25	High	Illumina	300000	100

Taxonomies. The taxonomic trees for the 30 marker genes were estimated by using RAxML [77] to refine the NCBI taxonomy using the SATé alignment of the reference datasets (i.e., the reference alignments). The taxonomic trees for the 16S RNA gene were estimated by using RAxML to refine the RDP

taxonomy using the reference SATé alignment.

Taxonomic identification results presented in this paper are based on reads simulated from 32 marker genes (30 genes used in the MetaPhyler study and two additional 16S marker genes). Note the marker-based methods TIPP and MetaPhyler are trained specifically on this reference dataset. Thus, the main foci of the taxonomic identification experiments are parameter exploration for TIPP and the comparison of TIPP versus MetaPhyler.

Experiments. I performed both *leave-one-out* experiments and *non-leave-one-out* experiments. In *leave-one-out* experiments, a particular taxonomic group is removed from the reference set, and then fragments belonging to the left-out taxonomic group are classified using various tools. In *non-leave-one-out* experiments, the fragments being classified are obtained by adding sequencing errors to short substrings from the full-length sequences.

The leave-one-out experiments are useful at understanding whether methods are able to identify novel organisms or taxonomic clades (an important focus of recent studies [97]). However, the non-leave-one-out experiments (especially under the higher error models) are useful at understanding how well methods are able to identify sequences from organisms with close relatives in public databases. Thus, evaluating performance under both conditions is helpful at characterizing how well the methods work. Real metagenomic samples are likely a mix of species, only some of which are not present in the reference datasets, and therefore will fall somewhere in between the two extremes of

leave-one-out and non-leave-one-out in terms of ease of classification.

Abundance profiling results presented in this paper are based on reads simulated from metagenomes. Abundance profiles for each method were estimated by examining the set of fragments classified by the profiling method. As noted earlier, marker-based methods require the fragments to first be binned to the reference markers. BLAST is used internally by MetaPhyler and MetaPhlAn with settings specific to the software; the BLAST setting used by TIPP can be found in Section A8.2 .

MetaPhlAn and MetaPhyler both output an abundance profile from a set of sequences. All other methods studied output the classification of the fragments; abundance profiles for these methods were estimated by using the relative abundance of the fragment classification results. Abundance profiles for all methods were then normalized by including only fragments that were classified at the taxonomic level of interest. For example, species-level abundance profiles are computed only on fragments that have species level classification; fragments left unclassified at the species level are excluded. Details on the computation of the abundance profiles can be found in Section A5 .

Experiment 1: Algorithmic parameter exploration. TIPP has many several parameters, and so my initial experiment evaluated the impact of these parameters on the sensitivity and precision of TIPP used as a taxon identification method, and then on the accuracy of the phylogenetic profiles it produces. I set default values for these parameters based on these initial experiments.

Experiment 2: Abundance profiling experiments. I performed abundance profiling experiments, separating the datasets into two different groups: datasets with short reads (88-100 bps) and datasets with long reads (265-969 bps). I explored performance on datasets with uniform (HC datasets) and staggered distributions (MC and LC datasets) of species.

Experiment 3: Evaluating the impact of sequencing error on taxonomic identification methods. I performed a non-leave-one-out simulation study to compare NBC, PhymmBL, and MetaPhyler to TIPP on taxonomic identification of fragments simulated from marker genes (MetaPhlAn was excluded because it uses a disjoint reference set), evaluating the impact of sequencing error on taxonomic classification.

Performance evaluation. For the taxonomic classification methods, the true lineage of each fragment is known, so I compute the percentage of fragments classified correctly, incorrectly, and left unclassified at each taxonomic rank.

For the abundance profiling experiments, the true abundance of the metagenomes is known, so I compute the root-mean-squared error ($RMSE$) of the estimated taxonomic profile. Let C_l be the set of clades found in the true profile and all the estimated profiles for the taxonomic level l , R_x be the abundance of clade x for the reference profile, and E_x be the abundance of clade x for the estimated profile. Then $RMSE_l$ (root-mean-squared-error for a taxonomic level l) is:

$$RMSE_l = \sqrt{\sum_{x \in C_l} \frac{(R_x - E_x)^2}{|C_l|}} \quad (5.7)$$

I normalize the all the *RMSE* of the other methods by the *RMSE* of TIPP(0%,0%,100) to infer the relative performance of the methods.

5.4 Results

Experiment 1: Parameter Exploration Experiments. Initial experiments evaluated the impact of the algorithmic parameters (support thresholds, alignment subset size, and placement subset size) on TIPP for fragment taxon identification (Sections S5.1 and S5.3) and phylogenetic profiling (Fig. S15 and S16). The results show that using 0% for both the alignment and placement support thresholds increased the sensitivity substantially, but also decreased the precision; thus, more fragments were classified at each level, but some of these classifications were incorrect, compared to using a threshold of 95%. Using an fHMM rather than a single HMM increases the true classification rate and reduces the false classification rate, with the biggest improvements observed for lower taxonomic levels under the higher error conditions. TIPP(95%,95%,100) (that is, alignment and placement support threshold of 95%, and using an HMM family with alignment subsets of size 100) is the default setting for TIPP when used as a taxonomic classifier for individual reads. Interestingly, when the objective is to estimate the phylogenetic profile (i.e., the distribution of taxa within a dataset, for some specific taxonomic level),

then reducing the alignment and placement support thresholds improves the estimated distribution. That is, the increase in true positives (sequences that are correctly classified) is sufficiently larger than the increase in false positives (sequences that are incorrectly classified), so that the resultant distribution that is estimated has higher accuracy. Thus, for the purpose of estimating phylogenetic profiles, I used TIPP(0%,0%,100) as the default setting. For full details on these experiments, see the Supplementary materials.

Table 5.2: The average *RMSE* on the short and long fragment datasets, normalized by TIPP’s *RMSE* for each model condition and each taxonomic rank. Thus methods with *RMSE* > 1 have worse performance than TIPP, and methods with *RMSE* < 1 have better performance than TIPP. Note that PhymmBL does not output any species level classifications. I use TIPP(0%,0%,100) for abundance profiling (see SOM for results using other variants). The best results for each level and fragment length are in boldface.

Short Fragments	Species	Genus	Family	Order	Class	Phylum
NBC	1.595	1.991	2.435	2.440	2.038	2.661
PhymmBL	NA	1.993	2.403	2.386	1.934	2.487
MetaPhlAn	0.931	1.029	1.128	1.184	1.103	1.333
MetaPhyler	6.143	3.642	2.310	1.604	1.460	1.278
TIPP	1.000	1.000	1.000	1.000	1.000	1.000
Long Fragments	Species	Genus	Family	Order	Class	Phylum
NBC	1.161	1.250	1.264	1.236	1.059	1.888
PhymmBL	NA	1.194	1.075	1.045	0.823	1.373
MetaPhlAn	1.802	1.372	1.202	1.168	0.986	1.463
MetaPhyler	4.582	1.779	1.343	1.228	1.239	1.520
TIPP	1.000	1.000	1.000	1.000	1.000	1.000

Experiment 2: Abundance profiling experiments. I show results comparing TIPP to NBC, PhymmBL, MetaPhlAn, and Metaphyler in Table 5.2

(see Tables S21 and S22 for results on individual datasets).

On the short fragment datasets, methods differ on particular datasets, and some datasets are harder than others (for example, MetaPhlAn-LC presents a more difficult challenge than MetaPhlAn-HC). MetaPhyler’s accuracy is extremely poor at the more specific levels, and has among the worst accuracy at the species and genus level. MetaPhlAn has the best accuracy at every level on the two MetaPhlAn datasets (MetaPhlAn HC and MetaPhlAn LC). TIPP is the best on the WebCarma Illumina dataset at every level. On the FACs HC Illumina dataset, TIPP is also best on the lower levels (species through order), MetaPhlAn is best at the class level, and MetaPhyler is best at the phylum level. Averaging across these four datasets, TIPP and MetaPhlAn have the best results of all methods, and are very close in performance (MetaPhlAn is better at the species level, slightly less accurate than TIPP at the genus level, and less accurate than TIPP at the family through class levels) with the exception of the phylum level (MetaPhlAn has 33.3% worse RMSE).

On the long fragment sequences, the absolute and relative performance of methods can differ substantially between datasets. TIPP has the best accuracy at all levels except for class (where MetaPhlAn is best) for the FACs HC dataset. On the FAMeS HC dataset, MetaPhlAn is generally best, but NBC is best at the species level. On the FAMeS LC dataset, NBC and TIPP are competitive for the best at the species and genus level, but PhymmBL is best at the other levels. On the FAMeS MC dataset, TIPP is best at the

species, genus, and phylum levels, but MetaPhlAn is best at the other levels. Finally, on the WebCarma dataset, NBC is best (or close to the best) at the species and genus levels, PhymmBL is best at the family through class levels, and TIPP, PhymmBL, and MetaPhlAn are the best at the phylum level. Examining average performance by level, I see the following trends. At the species level, TIPP has the best average performance, NBC is second (16.1% worse than TIPP), and MetaPhlAn is third (80.2% worse than TIPP). At the genus level, TIPP is best, PhymmBL is second (19.4% worse than TIPP), and NBC is third (25.0% worse than TIPP). At the family level, TIPP is first, PhymmBL is second (7.5% worse than TIPP), and MetaPhlAn is third (20.2% worse than TIPP). At the order level, TIPP is first, PhymmBL is second (4.5% worse than TIPP), and MetaPhlAn is third (16.8% worse than TIPP). At the class level, PhymmBL is first (17.7% better than TIPP), MetaPhlAn is second (1.4% better than TIPP), and TIPP is third. Finally, at the phylum level, TIPP is best, PhymmBL is second (37.3% worse than TIPP), and MetaPhlAn third (46.3% worse than TIPP). Thus, TIPP has the best average accuracy at all levels except the class level, where PhymmBL is best. More generally, PhymmBL has excellent performance on these datasets, and is typically in second place. Also, although MetaPhlAn and TIPP are close in some levels, there are large differences at the species genus, and phylum levels.

MetaPhlAn is close to TIPP on the short fragment datasets but less accurate for the long fragment datasets, or at the phylum level for the short fragments. PhymmBL had excellent results on the long fragment datasets (and is

best at the class level) but was not as accurate on the short fragment datasets. NBC had variable performance - excellent on many long fragment conditions, but not as accurate for the short fragment datasets. Overall MetaPhlAn and TIPP have the best average performance on short fragment datasets, while TIPP and PhymmBL have the best average performance on the long fragment datasets.

Experiment 3: Exploring robustness to sequencing error on taxonomic identification experiments.

I used non-leave-one-out results on fragments simulated from all 30 marker genes, comparing TIPP(95%, 95%, 100), MetaPhyler, PhymmBL, and NBC under varying error models (Figure 5.3 for 454-like error models, Fig. S17 for Illumina-like errors). The higher error models (Illumina_4, 454_2, and 454_3) produce datasets that do not contain any full-length sequences with a close match to the query sequence.

PhymmBL performed reasonably well on the Illumina error-model conditions and the easier 454 error-model condition. However, PhymmBL typically has more false classifications; this is most noticeable on the harder 454 model conditions.

While NBC had excellent precision, it had the worst recall of the methods. On the easiest Illumina model condition, NBC classified less than 60% of the fragments at the phylum level. On the 454 model conditions, NBC's recall dropped by a considerable amount. On the easiest 454 model condition, less

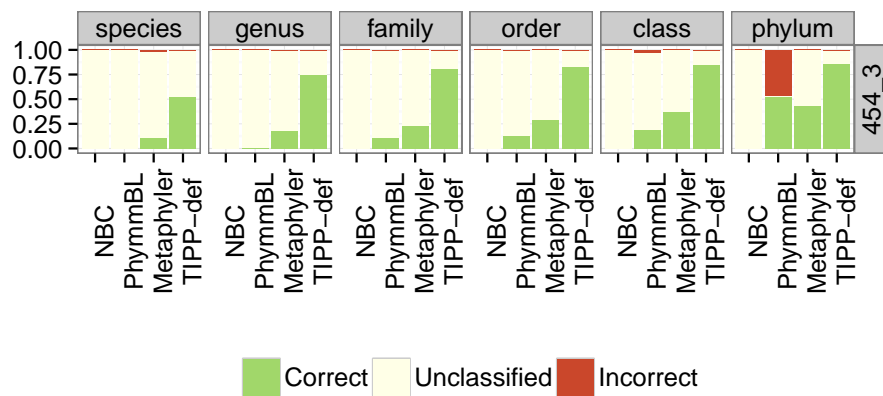


Figure 5.3: Non-leave-one-out experiments comparing the classification accuracy for NBC, PhymmBL, MetaPhyler and TIPP-default (i.e., TIPP-default refers to TIPP(95%,95%,100)) for fragments simulated from the 30 marker genes under different rates of 454-like errors. Note that PhymmBL does not classify below the genus level and thus has 100% unclassified rate at the species level.

than 20% of the fragments could be classified at the phylum level.

MetaPhyler had very good performance on Illumina error models, but poorer performance on the harder 454-error models. For the Illumina error-model conditions, MetaPhyler classified more than 90% of the fragments correctly at all taxonomic levels for the first two Illumina error-model conditions, but dropped to 70% or higher on the hardest Illumina error-model condition. On the 454 error-model condition, however, the percentage of fragments correctly classified by MetaPhyler rapidly dropped as the error rate increases, with less than 50% of the fragments classified correctly at the phylum level on the hardest model condition.

TIPP had very few incorrect classifications for any error-model at any level. TIPP also had very good recall except at the species level. MetaPhyler had better recall at lower taxonomic ranks (species and genus) with Illumina_1 and Illumina_2 error models.

Not surprisingly, methods trained on the marker dataset vastly outperform the composition-based methods on sequences from these markers for taxon identification. However, under the higher error models (especially with 454 errors, which involve indels), we see substantial differences between methods, with TIPP showing high robustness to indels.

One interesting question is whether taxonomic classification methods can correctly identify “dark matter” (sequences that do not belong to known phyla [71]). Since all fragments in my datasets come from known phyla, fail-

ure to classify these fragments could be interpreted as an assertion that the fragments belong to new phyla, and so would be a “false positive” with respect to identifying dark matter. Figure 1 allows us to explore this in a non-leave-out experiment with indel errors under 454 models (low indel rates for 454.1 and higher rates for 454.3). Under low indel errors, TIPP and PhymmBL have generally low rates (less than 2%) of incorrectly identifying sequences as “dark matter”, Metaphyler has slightly higher rate (6%), and NBC has an extremely high rate (72%). Incorrect dark matter identification under the 454.3 error model is much higher: 100% for NBC, 56% for MetaPhyler, 14% for TIPP, and 2% for PhymmBL. These results show the challenges in interpreting failure to classify sequences as indicative of membership in novel phyla, especially for taxonomic identification methods (such as NBC and MetaPhyler) that attempt to minimize false positive identifications.

Summary. The taxonomic identification experiments showed interesting differences between TIPP, NBC, MetaPhyler, and PhymmBL. On average TIPP had higher recall than MetaPhyler, with exceptions only on the non-leave-one-out experiments on the easier Illumina error models. On all other model conditions (454 non-leave-out experiments and all leave-out-experiments), TIPP had greater recall than MetaPhyler, with the largest improvement in recall at the lower taxonomic levels. At the same time, TIPP also had lower precision in some (but not all) cases, but in most cases the reduction in precision was not as large as the improvement in recall. By design, NBC had very good pre-

cision, but on these data NBC also had poorer recall than the other methods. PhymmBL was typically not as accurate as either MetaPhyer or TIPP (lower precision and recall), but was more accurate than NBC.

The experiments on abundance profiling included MetaPhlAn and explored accuracy with respect to RMSE. These experiments showed somewhat different trends. On short fragment datasets, TIPP and MetaPhlAn had better accuracy than the other methods. TIPP and MetaPhlAn had very similar average accuracy, with MetaPhlAn better on the MetaPhlAn datasets, and TIPP generally better on the other datasets. On long fragment datasets, TIPP had generally the best accuracy of all methods. PhymmBL was overall second best, and had the best accuracy at the class level, and MetaPhlAn was in third place.

Since TIPP and MetaPhlAn are marker-based methods, and only classify a fraction of the full set of fragments, this shows that good performance for abundance profiling does not rest on the ability to classify all fragments, or even most fragments. Instead, highly accurate classification of marker genes can provide excellent estimations of taxonomic abundance.

More generally, abundance profiling can be improved by somewhat more aggressive taxonomic profiling techniques, provided that proportionally more correct than incorrect classifications result. My comparison of the different TIPP variants suggest that the choice between which TIPP version to use depends on the context of the analysis; taxonomic identification analyses may benefit by minimizing false positive classifications by us-

ing TIPP(95%,95%,100), whereas abundance profiling analyses may benefit by using TIPP(0%,0%,100).

5.5 Conclusions and Future Work

In this chapter, I presented TIPP, a new marker-based method for taxonomic identification and abundance profiling. TIPP combines SEPP, a recent method for phylogenetic placement, with statistical methods for evaluating the support for a given alignment and phylogenetic placement, in order to give a highly accurate taxon identification of each read. Furthermore, by modifying algorithmic parameters (such as the required statistical support), the user can control for precision and recall, and depending on the context of the analysis, can optimize TIPP for taxonomic identification or for abundance profiling.

SEPP's technique of using fHMM is a key part to TIPP's improved accuracy as a taxonomic identification method, and suggests that similar improvements for other HMM-based classification methods might also be achievable. Therefore, in my future work, I plan to compare TIPP against mOTU [85], a new marker-based profiling method that maps metagenomic reads to a reference dataset generated by HMMs, and to explore the impact on mOTU's performance by using SEPP's decomposition strategy in generating mOTU's reference dataset.

One important area of interest is the taxonomic identification of deeply branching phyla [71]. The key step in detecting deeply branching phyla is searching for cellular organisms with very different 16S sequences. Cells that

have different enough 16S sequences are targeted for single cell sequencing. TIPP can be used to aid in this endeavour. 16S fragments can be selected from metagenomic samples and then classified using TIPP. Since the focus is for searching on very divergent 16S sequences, the decomposition strategy used within TIPP may be helpful in obtaining more accurate classifications of the 16S fragments. Samples that have high amounts of “dark matter” 16S fragments can then be targeted for single cell assembly.

Chapter 6

UPP: Ultra-large alignments using family of Hidden Markov Model

In Chapter 4, I showed that SEPP resulted in improved phylogenetic placement accuracy compared to HMMALIGN+pplacer and PaPaRa+pplacer. In this chapter, I will show a modification to SEPP for the large-scale alignment of unaligned sequences. This new technique called **Ultra-large Alignments using Phylogenetic Profiles (UPP)** allows accurate alignment of datasets containing both fragmentary and full-length sequences, is fast and highly parallelizable, and can generate an accurate alignment on datasets containing 1,000,000 sequences.

In Section 6.1, I describe UPP, a modification of SEPP for ultra-large alignment. In Section 6.2, I describe the simulation study designed to evaluate the alignment and phylogeny estimation accuracy of UPP on both simulated and biological datasets. In Section 6.3, I present the results comparing UPP and other alignment techniques. I show that UPP can be tuned for accuracy or speed, and that UPP generally results in better alignments than other methods, and that UPP is the only method can align the largest datasets in less than 24 hours on a 12 core machine. Finally, in Section 6.4, I discuss

future improvements and ongoing studies for UPP.

6.1 UPP: Ultra-large alignment using Phylogeny-aware Profiles

UPP begins with a set of unaligned sequences and uses a subset of the sequences to build the fHMM. As the fHMM model requires a backbone alignment and tree, UPP selects a subset of the sequences to be the “backbone set”; the remaining sequences are called the “query set”. UPP preferentially selects the backbone sequences from sequences that are considered to be “full-length” in order to provide robustness to fragmentary data. UPP uses PASTA [56] to compute a backbone alignment and tree on the backbone sequences. UPP then uses the backbone alignment and tree to build the fHMM as described in Chapter 3, with some minor modifications. Rather than computing HMMs on the alignment subsets with less than m_a sequences, UPP computes HMMs at every stage of the decomposition step to create a set of nested hierarchical HMMs. Thus, the size of the subsets range from at most m_a sequences to up to the full dataset, and all but the smallest subsets contain other subsets (see Fig. 6.1 for diagram). UPP then applies the fHMM alignment algorithm to align the remaining query sequences to the backbone alignment.

UPP can also be used iteratively. In the first iteration, the UPP alignment is estimated, and a ML tree is estimated using FastTree [65]. 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 re-

sampling technique is generally beneficial, it is particularly helpful when 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.

The UPP algorithm. I describe a single iteration of UPP run in its default mode where the maximum alignment subset size m_a is set to 10; see Appendix B1.2.4 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 the 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). Next, 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”.

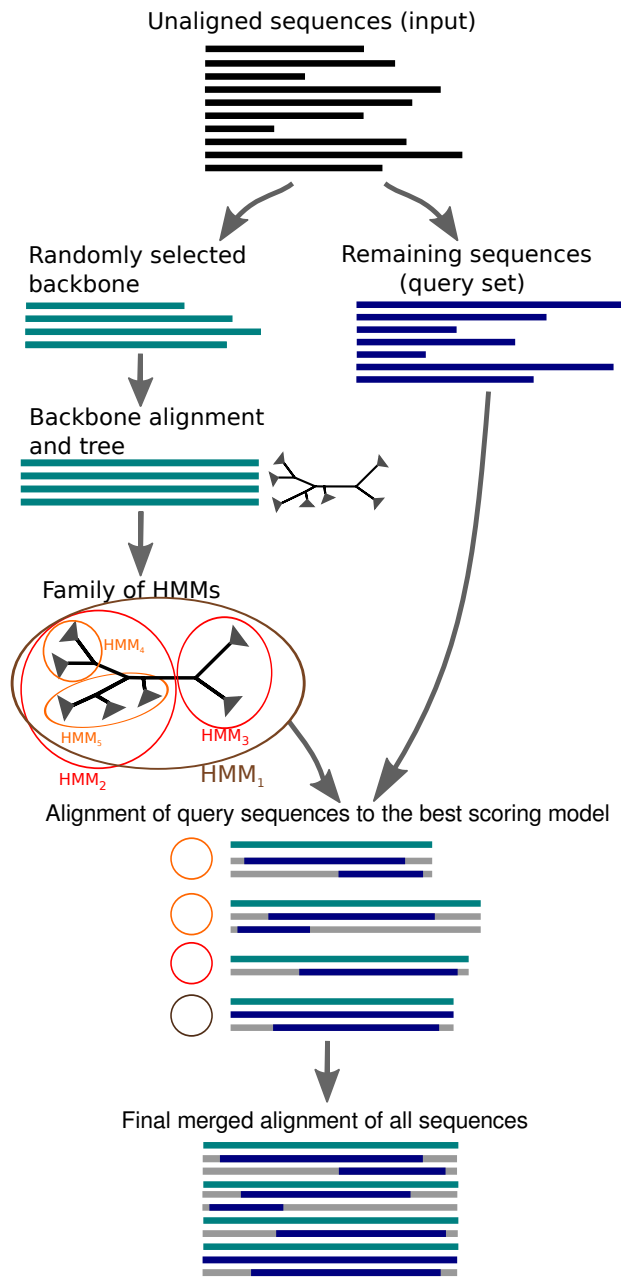


Figure 6.1: The UPP algorithm and the HMM Family technique.

The backbone tree is then used to produce a collection \mathcal{C} of subsets of the backbone dataset, as follows. In contrast to the original fHMM model, \mathcal{C} is initialized to include S_0 . The centroid is removed and the leaf sets of the two subtrees produced by removing the centroid edge are also 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.

I then compute the backbone alignment restricted to each subset of sequences in \mathcal{C} ; these are called the subset alignments. I 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. I 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 I 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, I 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 Appendix B1.2.1, B1.2.2, and B1.2.4 for

additional details.

6.2 Performance Evaluation

I demonstrate UPP’s accuracy on a collection of biological and simulated datasets, in comparison to leading multiple sequence alignments. I compare estimated alignments to reference (true or curated) alignments, and ML 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 [30, 69], I 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 [46] that were generated using ROSE [82], and used to evaluate SATé in comparison to other MSA methods on large datasets; 10,000-sequence datasets we generated using Indelible v. 1.03 [22] with average sequence length 1000; and subsets of the million-sequence RNASim [26] dataset with average sequence length 1554.5. RNASim is a simulator for RNA sequence evolution that I 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 from [65], which

were generated using ROSE based on proteins from the COG database [87], and had average sequence lengths varying from 179.4 to 346.9.

The biological datasets include the three largest datasets from the Comparative Ribosomal Website (CRW) [11], each a set of 16S sequences. I 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. I include ten large amino acid datasets (10 AA) with curated multiple sequence alignments (the eight largest BAliBASE datasets [89] and IGADBL_100 and coli.epi_100 from [25]); these range in size from 320 to 807 sequences and have average sequence lengths that range from 56.7 to 886.3. I also used 19 of the largest 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 [75] to evaluate protein multiple sequence alignment methods on large datasets, and have Homstrad [58] reference alignments on very small subsets (5-20 sequences, median 7) of their sequences.

I generated fragmentary datasets by selecting a random subset of sequences and a random substring (of a desired length) for each selected sequence (see Appendix 6.2 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 B2 and B3.

Reference alignments and trees. For simulated datasets, the reference alignment is the true alignment (known because I simulate evolution and record the events); for biological datasets, the reference alignment is the curated structural alignment. Reference trees for the simulated datasets 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 I collapse all branches with less than 75% support; this is the same technique used in [47] 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) [72] 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, I report FN rates for all analyses, using [64].

Fragment Simulation In order to test the robustness of different alignment methods to fragmentary sequences, I generated datasets with both full-length and fragmentary sequences from the 1000-taxon 1000M2, 1000M3, and 1000M4

datasets, the CRW datasets, the Indelible datasets, and the RNASim 10K dataset. For each dataset, a fraction of the sequences (12.5%, 25%, or 50%) were made fragmentary by selecting a contiguous substring (length drawn from a normal distribution with mean length of 500 bps and standard deviation of 60 bps) from a random position (drawn uniformly, at random) from the original full-length sequence. The remaining sequences that were selected to be full-length were left unmodified.

For each of the 1000-taxon fragmentary datasets, 5 replicates were generated. For the larger CRW, Indelible, and RNASim 10K datasets, only 1 replicate was generated.

Methods. I use Clustal-Omega [75] version 2.1, MAFFT [36, 38] version 6.956b, Muscle [16] version 3.8.31, Opal [93], PASTA, SATé-II [47], and UPP to compute multiple sequence alignments. I show results for only iteration of UPP in this chapter; see Appendix B2.2 for results using more than one iteration. I use FastTree [65] and RAxML [78] to compute maximum likelihood trees on estimated and reference alignments.

Performance Metrics. I compare estimated alignments and their maximum likelihood (ML) trees to reference alignments and trees. I use FastSP [57] 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 posi-

tive (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 Appendix B, and the means of these two alignment error rates are given in this chapter. I 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. I 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, I 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. I also performed a limited number of experiments on TACC with checkpointing, to explore performance when time is not limited.

6.3 Results

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

Since UPP computes its backbone using PASTA, I compared UPP to PASTA, and included a comparison to SATé-II, focusing on the RNASim datasets. As shown in [56], 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 (figs. B23-B30) and was much more robust to fragmentation (fig. B31). UPP produced more accurate trees than SATé-II (fig. B24). PASTA had a small advantage over UPP with respect to tree estimation on datasets without fragments (fig. B24), but was less accurate than UPP on datasets with fragments (fig. B32). 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 (Appendix B1.3). I focus the remainder of

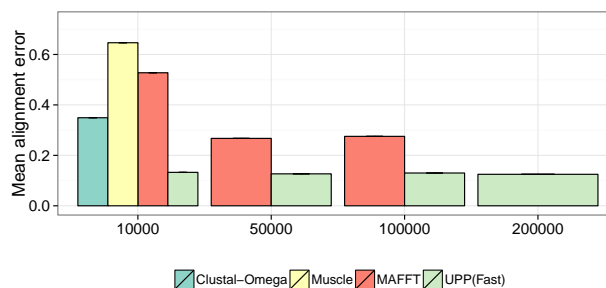
the discussion on Clustal-Omega, Muscle, MAFFT, and UPP; results for the full set of methods can be found in Section B2.9.

6.3.1 Phylogenetic Alignment Accuracy

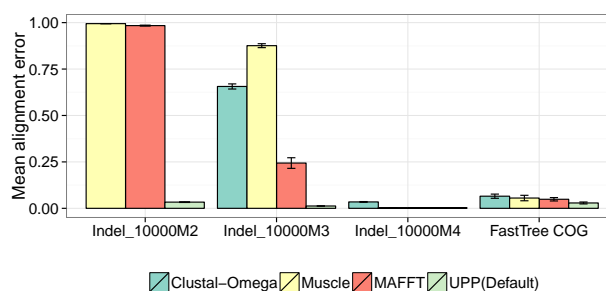
I begin by evaluating UPP for phylogenetic estimation purposes. I use simulated datasets, since these provide true alignments and true trees, and thus allow us to exactly quantify error in identifying true positional homology (i.e., descent from a common ancestor [69]).

Alignment estimation error on RNASim datasets. I 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 6.1). I compare UPP(Fast) to MAFFT (default MAFFT on the 10K and 50K datasets, MAFFT-PartTree on 100K dataset), Clustal-Omega, and Muscle (Fig. 6.2(a)). Default MAFFT produced less accurate alignments than MAFFT-PartTree on the RNASim 10K dataset (fig. B20) and failed to complete on the 100K and larger datasets (Section B1.3).

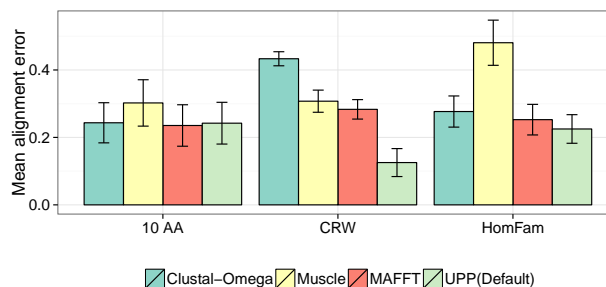
UPP(Fast) succeeded in analyzing all the datasets within the 24 hour time limit, MAFFT-PartTree succeeded in analyzing the datasets with up to 100K sequences, and default MAFFT successfully analyzed datasets with up to 50K sequences; however, the other methods failed to align RNASim datasets with more than 10K sequences. Muscle failed because it required more than 24



(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 6.2: **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.

GB of memory on these larger datasets, and Clustal-Omega failed to return an alignment but without giving an error message (see Appendix B1.3 for details). On the RNASim 10K dataset (Fig. 6.2(a)), the error rates were 13.3% for UPP(Fast), 34.9% for Clustal-Omega, 52.7% for default MAFFT, 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%.

I 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% alignment error, respectively; see Table B1). 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, I refer to these model conditions as easy (10000M4), medium (10000M3), and hard (10000M2). I ran UPP(Default), MAFFT-Default, Muscle, and Clustal-Omega on ten replicates for each model condition.

UPP had very low average alignment error across all three model con-

ditions: 3.3%, 1.3%, and 0.1% on the hard, medium, and easy model conditions, respectively (Fig. 6.2(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-Default had 97.9% error and Clustal-Omega failed to generate an alignment (Fig. 6.2(b)). Under the medium model condition, MAFFT-Default had 22.8% error, Clustal-Omega had 65.6% error, and Muscle had 87.6% error (Fig. 6.2(b)). Finally, under the easy model condition, MAFFT-Default, Muscle and UPP all had very low error (below 0.4%), and Clustal-Omega had 3.4% error (Fig. 6.2(b) and figs. B40 and B41).

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% (Fig. 6.2(b), figs. B48-B49).

Results on 1000-sequence simulated nucleotide datasets. The nine 1000-sequence model conditions studied in [46, 47] 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.

I 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 (fig. B37). 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%.

Impact of MSA estimation technique on phylogenetic tree estimation. Next, I evaluated the impact of MSA estimation on phylogenetic tree estimation. I 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 (fig. B39). For example, on the hardest model condition (1000M1), the Δ FN error rates were 4.2% for UPP(Default), 15.4%

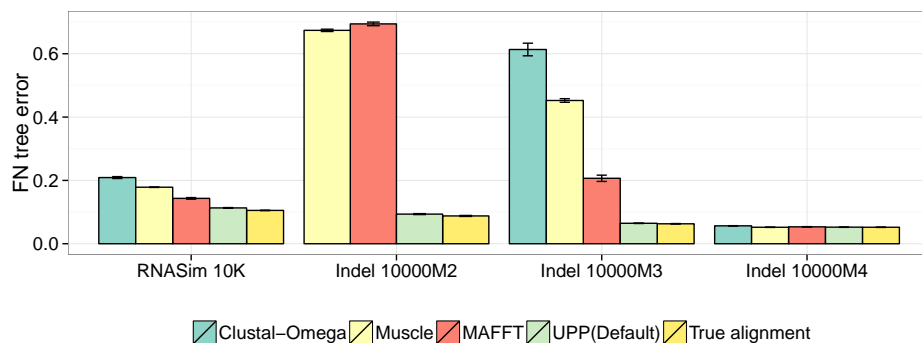


Figure 6.3: **Tree error on simulated datasets with 10,000 sequences.** I show FN tree error results on the RNASim 10K and Indelible datasets. ML trees were estimated using FastTree under the GTR model. All MSA methods were run with 24 GB of memory and 12 CPUs and given 24 hours to complete. MAFFT was 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.

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), Δ FN 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% Δ FN).

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 Δ FN (12.2%) on the other datasets; in comparison, UPP(Default) had the most accurate results (1.9% Δ FN), followed by Muscle with 3.1% and Clustal-Omega with 4.1% (fig. B50).

Performance on the Indelible and RNASim datasets with 10,000 sequences (Fig. 6.3) 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 0.6% Δ FN error, MAFFT had 58.1%, Muscle had 58.6%, and Clustal-Omega failed to generate an alignment (Fig. 6.3 and section B1.3).

I computed ML trees using FastTree on three UPP alignments (UPP(Fast), UPP(Fast,NoDecomp), and UPP(Default,NoDecomp)) of the million-sequence RNASim dataset. Despite the large number of sequences and relatively few sites (1500 average sequence length), 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), so that Δ FN was between 2.0-2.8% for all UPP variants I tested. The phylogenetic accuracy of these trees is noteworthy, given that the true alignment; see Table B1) given 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,NoDecomp), but FastTree's ability to produce reasonable results on extremely large datasets.

I used the RNASim datasets to explore the impact of increased taxon sampling, which is expected to improve phylogenetic accuracy [100]. As expected, tree error was reduced with increased taxon sampling when using true

alignments: ML trees had FN error rates of 10.6%, 8.1%, 6.9%, and 6.1% on the RNASim 10K, 50K, 100K, and 200K datasets, respectively. I 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 6.1).

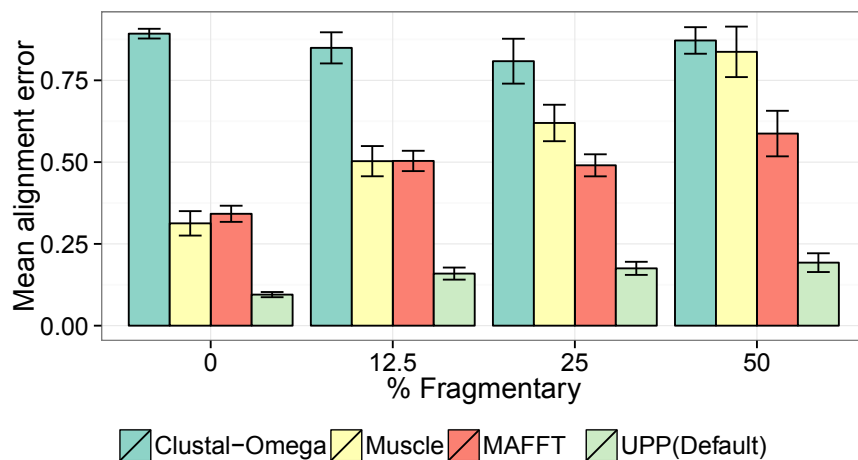
6.3.2 Structural Alignment Accuracy

I 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% (Fig. 6.2(c), figs. B45 and B46). The 19 HomFam datasets and three CRW datasets are too large for MAFFT-L-INS-i or Opal, and so I 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,

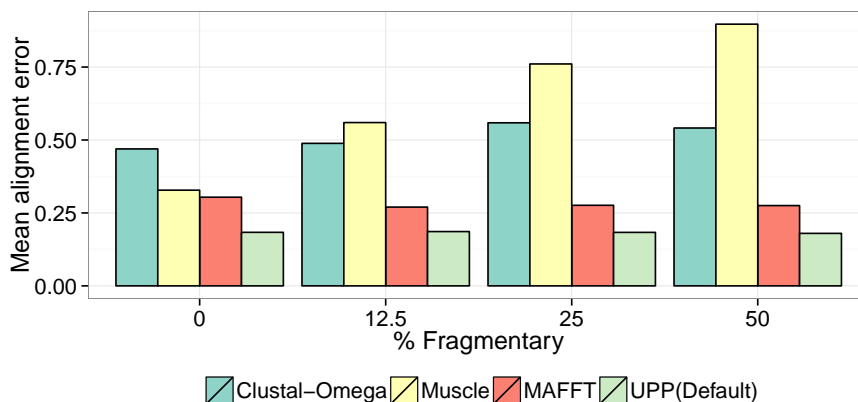
UPP had 22.5% alignment error, followed by MAFFT-Default with 25.3% error, Clustal-Omega with 27.7% error, and Muscle with 48.1% average error (Fig. 6.2(c), see also figs. B43 and B44 for additional results). MAFFT-default failed to run on the 16S.B.ALL CRW dataset (see Section B1.3), and so I used MAFFT-PartTree for that dataset; however, I report MAFFT-default for 16S.3 and 16S.T. 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% (Fig. 6.2(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.

6.3.3 Results on Fragmentary Datasets.

Figure 6.4 shows alignment error 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 for both datasets. 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



(a) Fragmentary 1000M2 model conditions



(b) Fragmentary CRW 16S.T datasets

Figure 6.4: **Impact of fragmentary sequences on alignment error.** I 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.

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 I introduce fragmentation, but less accurate when 12.5% of the sequences were fragmentary (Fig. 6.4(b)). 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 (Appendix B2.11).

Phylogenetic accuracy was also impacted by fragmentary data, but responses varied by the alignment method. Results on the RNASim 10K datasets (Fig. 6.5) 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 (Δ FN 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% Δ FN on full-length sequences, and then 35.6-49.0% Δ FN under all the fragmentary conditions. Clustal had 9.1% Δ FN on full-length sequences, and then 25.4%-25.8% on the fragmentary conditions. Furthermore, while both UPP(Default) and MAFFT-default were highly robust to fragmentary data, UPP(Default)

had better accuracy under all levels of fragmentation on this model condition: 0.8% Δ FN on full-length sequences, and at most 3.9% Δ FN even when half the sequences are fragmentary. MAFFT-default had 5.9% Δ FN for full-length sequences, and Δ FN between 5.8% and 7.1% on the fragmentary sequences.

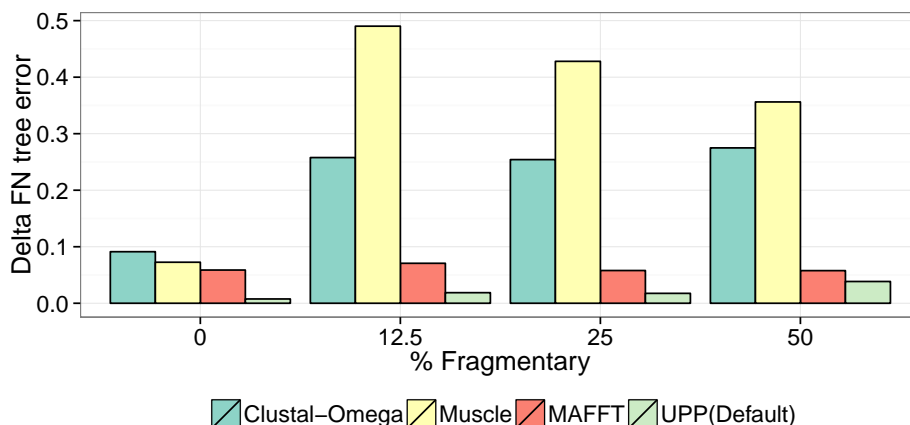


Figure 6.5: **Impact of fragmentation on tree error for the RNASim 10K datasets.** I show the Δ FN error rates of maximum likelihood trees computed using FastTree under the GTR model on alignments computed using Clustal-Omega, Muscle, MAFFT-Default, and UPP(Default), on the RNASim 10K dataset, including results on versions of the dataset where I make some of the sequences fragmentary. All fragments have average length 500, but I vary the percentage of the dataset that is fragmentary.

Figure B54 shows similar trends for the fragmentary 1000M2 and 1000M3 datasets. Δ FN on the fragmentary 1000M2 datasets ranged for UPP(Default) from 2.5-4.7%, from 34.7-65.0% for Muscle, and from 55.8-70.8% for Clustal-Omega (fig. B54). Results on fragmentary versions of the 1000M3 model condition, which has a lower rate of evolution than 1000M2, show Δ FN for UPP(Default) ranging from 0.2-2.2%, while Δ FN ranged from 12.7-27.8%

for Muscle and from 18.8-57.2% for Clustal-Omega. MAFFT-L-INS-i showed somewhat better tolerance to fragmentary data than Clustal-Omega or Muscle, but still had high Δ FN rates: 18.1-43.5% error on the 1000M2 model condition, and 4.0-14.1% for 1000M3.

6.3.4 Factors Influencing Accuracy

The choice of alignment method to compute the backbone alignment has an impact on final alignment and tree accuracy: comparing PASTA, MAFFT-L-INS-i, Muscle, and Clustal-Omega for backbone alignment estimation, I found that PASTA and Muscle backbones yielded the best alignment accuracy (fig. B6) but PASTA and MAFFT-L-INS-i backbones yielded the best tree accuracy (fig. B7). The choice of technique used to align query sequences to the backbone alignment also impacts alignment and tree error (Table 1 and figs. B8, B10, B13, B9, and B12), with the HMM Family technique giving the best results compared to a single HMM or MAFFT-Profile with either `--add` or `--addfragments`.

I found that the error of the backbone alignment and the alignment generated by three different ways of running UPP (default setting, and aligning query sequences using MAFFT-Profile) were strongly correlated (Fig. B33, Pearson's correlation coefficient 0.897; p-value=2.29e-10). Furthermore, alignment errors for UPP(Default) were very close to the backbone alignment error, with root mean square difference (RMSE) of 0.020, and closer to the backbone alignment error than those produced by UPP using MAFFT-profile (RMSE

of 0.024 for MAFFT-profile--addfragment and 0.051 for MAFFT-profile--add, fig. B33). Thus, while the three versions of UPP all showed good correlations between backbone alignment error and final alignment error, the use of the HMM Family technique gave the best correlation, and helps UPP to scale alignment accuracy obtained on small subsets to large datasets.

6.3.5 Running Time

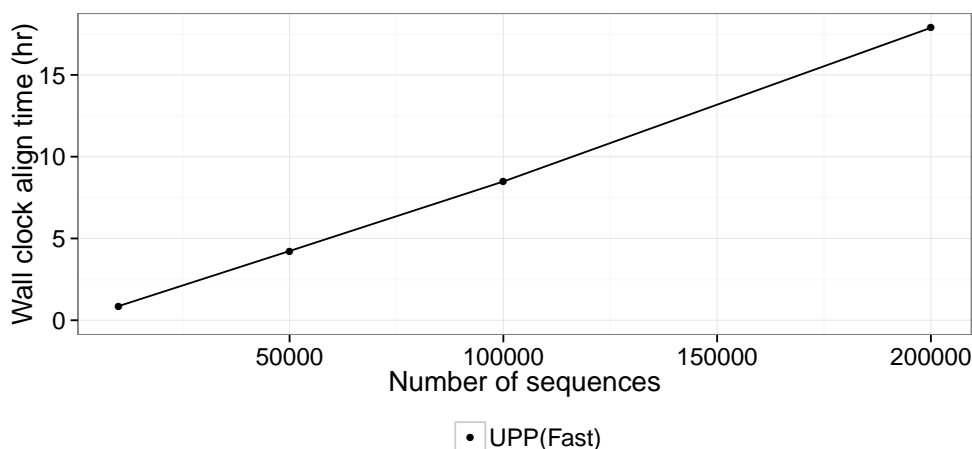


Figure 6.6: **Running time for UPP(Fast) on the RNASim datasets.** I show running time for UPP(Fast) on RNASim datasets with 10K, 50K, 100K, and 200K sequences. UPP(Fast) uses a backbone of size 100, computes the backbone alignment using PASTA, and then aligns the remaining sequences using the HMM Family technique. All analyses were run on TACC with 24 GB of memory and 12 CPUs.

Running times for UPP(Fast) on the RNASim datasets 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

Table 6.1: **Results for UPP variants on the RNASim datasets.** I show results for different variants of UPP on the RNASim datasets with 10,000 to 200,000 sequences. I 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 Default; it uses a backbone of size 1000, uses PASTA to compute the backbone alignment, and the HMM Family technique; Fast is obtained by using backbones of size 100 and keeping all other settings constant. The “ND” versions of these two methods replace the HMM Family technique with a single HMM. Default-MP uses MAFFT-Profile (with --add, denoted “A”, or with --addfragments, denoted “AF”) to add the query sequences into the backbone alignment, and otherwise is identical to Default; Fast-MP differs from this only by using a backbone of size 100.

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

hours, on 100K sequences in about 8.5 hours, and on 200K sequences in about 17.8 hours (Fig. 6.6). Table 6.1 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 produces only 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.

6.4 Conclusion and Future Work

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 ML trees based on PASTA alignments (for fragment-free datasets) are typically more accurate than ML trees based on UPP alignments. However, on large datasets, ML trees estimated on UPP alignments are typically more accurate than ML 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 I tested. Finally, UPP was the only method I 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, my study showed the alignment error on the backbone alignment is quite close to the alignment error on the alignment returned by UPP(Default) (Section B2.8), and this close relationship between the accuracy of the backbone alignment and the final alignment is weaker when I use MAFFT-Profile second iteration) and I didn't fully explore using a single HMM other than RNASim 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 simpler versions of UPP (e.g., using MAFFT-L-INS-i instead of PASTA for the backbone alignment, or using a single HMM rather than the HMM

Family Technique to align query sequences), the best accuracy was obtained using my 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 [69] 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 my 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 [9, 84]) 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 [100]. For example, the avian genome project is planning to eventually sequence all roughly 10,000 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.

Chapter 7

Conclusion and future work

7.1 Conclusion

Sequence alignment is a vital step in many bioinformatics pipelines. From the alignment, the phylogenetic relationship between the different sequences in the alignment can be inferred. Under the context of phylogenetic placement, I have shown that the standard approach of using a single HMM for aligning sequences to an existing alignment degrades when the sequences come from distantly related taxa, and that new methods are necessary for aligning evolutionarily divergent sequences. I present fHMM as a new statistical model for representing an MSA in Chapter 3, and I show how it can be used to align sequences to an existing backbone alignment.

In Chapter 4, I implemented the fHMM technique within SEPP and apply SEPP toward the phylogenetic placement problem. I presented a simulation study and showed that SEPP resulted in better placement accuracy than HMMALIGN+pplacer and PaPaRa+pplacer on difficult datasets. More importantly, I validated the hypothesis that using multiple HMMs can result in significantly better phylogenetic placement accuracy than using a single HMM. This result forms the basis for the remaining chapters of my dissertation.

In Chapter 5, I presented TIPP, an extension of SEPP by including statistical support measures to control the false classification rate, and showed its performance on taxonomic identification and profiling. I showed that using multiple HMMs resulted in better classification accuracy than using a single HMM. Most interestingly, I showed that under the context of taxonomic identification, requiring a high statistical support threshold for classification resulted in the best overall results, however, under the context of taxonomic profiling, using the minimum support threshold resulted in the best overall taxonomic profiles. Thus, the choice of the statistical support threshold depends on the application.

In Chapter 6, I presented UPP, a modification of SEPP for “de novo” sequence alignment. SEPP requires a backbone alignment as input. I presented a new technique to intelligently select the set of sequences to form the backbone alignment, and then applied the family of HMMs technique to complete the alignment on the entire set of sequences. I showed that UPP typically resulted in better alignments on both DNA and amino acid sequences, which in turn, resulted in more accurate phylogenies compared to other methods. In addition, I showed that UPP could accurately estimate an alignment on 1,000,000 sequences without the need of a supercomputer in less than 2 days.

7.2 Future Work

SEPP, TIPP, and UPP all use a similar pipeline for sequence alignment: a backbone alignment is decomposed into closely related subalignments

using a phylogenetic tree, and the query sequences are aligned to the subalignments. Any improvement to this pipeline may potentially improve accuracy of these three techniques. Possible ways to improve the accuracy of the pipeline include:

- Using different methods for aligning the query sequence to the backbone alignment. For example, we saw in Chapter 6 that Mafft-profile run under the most accurate settings resulted in accurate alignments on datasets containing both fragmentary and full-length sequences. However, this setting of Mafft could not be run on the larger datasets. The subalignments generated by the fHMM decomposition could be made sufficiently small enough such that Mafft-profile can be run under the most accurate setting. This may result in more accurate alignments of the query sequences to the subalignments.
- Applying different methods for decomposing the backbone alignment. In all three methods presented, the backbone tree was decomposed using the centroid edge decomposition. Using a different technique, such as the longest edge decomposition used in SATé [46], may result in better HMMs. Similarly, using a clade-based decomposition may group taxonomically similar sequences together and result in better HMMs.
- Using the hierarchical family of HMMs within SEPP and TIPP. Currently, only UPP has been tested with the hierarchical family of HMMs.

However, the hierarchical family may also lead to better placement results and taxonomic profiling results. In both SEPP and TIPP, the HMMs are computed on subalignments of roughly the same size. However, we saw in Chapter 6 that the HMM that yields the best HMMER score is not always the HMM based upon the smallest alignment subsets. By using the hierarchical family of HMMs, we allow fragments to align to both small and large HMMs which may result in improved alignment accuracy.

Future research for TIPP includes simple changes such as expanding the reference dataset, and algorithmic changes such as modifications to identification and profiling. Future work includes:

- Expanding the marker gene set. TIPP currently uses a set of 30 marker genes for taxonomic profiling. A simple extension would be to expand the marker gene set to the 40 marker genes used in [85], as well as update existing marker genes to include more recently sequenced genomes. By expanding the marker gene set, TIPP may be able to estimate more accurate profiles on metagenomic datasets.
- Exploring taxonomic identification of viral sequences. Viruses are difficult to identify because there are no genes that are found across all viruses. Instead, viruses are typically identified using group specific genes. To make the problem more difficult, viruses can have higher rates of mutations and horizontal gene transfer, making alignment estimation

and phylogeny estimation difficult. Thus, viral identification would be a good test case for TIPP's ability to classify very divergent sequences.

- Combining abundance profiles. TIPP currently uses a simplistic algorithm for computing the taxonomic profile from the marker genes; all reads that are binned to any of the marker genes are pooled together, and the abundance profile is estimated on the pooled reads. This process ignores the fact that the source gene of the reads is known. Better profiles may be obtained if separate abundance profiles are estimated from the reads binned to each individual marker gene, and the profiles are combined using a mixture modeling approach.
- Improved detection of rare species in a sample. One difficulty in taxonomic profiling is determining whether a low abundance species is truly present, or the abundance estimation is a false positive. While TIPP treats each read independently, the reads themselves are not independent; they come from the population of species present in the metagenomic sample. Thus, inferences about the abundance profile of the reads can be used to filter out false positives, as well as detect rare species. For example, if a rare species is detected across multiple different markers, it's likely to be present in the sample. However, if the species is only present in very few markers, then it is more likely to be a false positive.

Future work on UPP include:

- Incorporating iteration within UPP. The quality of the final alignment can be heavily dependent on the initial selection of the backbone sequences. The initial step of filtering short sequences from the backbone selection process may exclude entire clades from the backbone set, making it difficult to align sequences from the excluded clade. I have already shown preliminary work that iteration within UPP can result in better alignments when the initial backbone set is sampled non-uniformly from the phylogenetic tree. Re-sampling may also be necessary due to the process of random sampling failing to include any sequences from smaller clades. Better results could be obtained by examining different re-sampling strategies, as well as examining different ways of selecting the backbone set.

Appendices

Appendix A

TIPP

A1 Precision and Recall Comparisons and Statistical Significance

In this section we compare techniques two at a time according to precision and recall. For each comparison we show tables with differences between precision and recall values of the two techniques being compared, and indicate whether the differences are statistically significant.

A1.1 HMMER+pplacer versus HMMER+EPA

Table A1 shows that pplacer and EPA are indistinguishable in terms of both precision and recall (i.e., the differences are not statistically significant).

A1.2 Experiment 1: TIPP variants

In this section we provide comparisons of different TIPP variants.

A1.2.1 HMMER+pplacer versus SEPP

We first compare HMMER+pplacer (which is TIPP(0%,0%,ALL)) against SEPP (which is TIPP(0%,0%,100)) based on the leave-species-out study.

Table A1: The difference between precision and recall of HMMER+pplacer and HMMER+EPA in the leave-species-out experiments on the rpsB gene. Negative values indicate HMMER+EPA is better, while positive values indicate that HMMER+pplacer is better. None of the differences were statistically significant according to the Pearson’s chi-square contingency table test (as implemented in R [68]). p-values are shown in parentheses below each comparison.

	recall	precision
genus	0.005 (0.535)	0.008 (0.386)
family	-0.001 (0.930)	-0.001 (0.865)
order	0.004 (0.525)	0.004 (0.517)
class	0.005 (0.350)	0.005 (0.297)
phylum	0.003 (0.405)	0.004 (0.329)

Table A2 shows the difference between precision and recall of SEPP and HMMER+pplacer (positive values mean SEPP performs better, and negative values mean that TIPP performs better). Compared to HMMER+pplacer, SEPP always results in both better precision and better recall. All differences are statistically significant.

Table A2: The difference between precision and recall of SEPP and HMMER+pplacer in the leave-species-out experiments on the rpsB gene. Negative values indicate HMMER+pplacer is better, while positive values indicate that SEPP is better. Differences in bold are statistically significant according to the Pearson’s chi-square contingency table test (as implemented in R [68]). SEPP always results in better precision and recall. All differences are statistically significant.

	genus	family	order	class	phylum
Recall					
Illumina_1	0.035	0.043	0.039	0.041	0.037
Illumina_2	0.033	0.042	0.045	0.040	0.039
Illumina_4	0.034	0.048	0.050	0.050	0.045
454_1	0.027	0.033	0.029	0.027	0.025
454_2	0.055	0.064	0.064	0.058	0.051
454_3	0.276	0.349	0.367	0.341	0.294
mean	0.077	0.097	0.099	0.093	0.082
Precision					
Illumina_1	0.036	0.042	0.037	0.039	0.035
Illumina_2	0.034	0.042	0.044	0.038	0.039
Illumina_4	0.032	0.042	0.048	0.045	0.041
454_1	0.028	0.026	0.022	0.020	0.021
454_2	0.055	0.060	0.059	0.051	0.045
454_3	0.286	0.354	0.359	0.320	0.273
mean	0.079	0.094	0.095	0.086	0.076

A1.2.2 TIPP(0%,0%,100) versus TIPP(0%,95%,100)

Next, we compare TIPP(0%,0%,100) (which is SEPP) versus TIPP(0%,95%,100) (which is SEPP plus consideration of placement support) based on the leave-species-out study to study the effects of placement support considerations. Table A3 shows difference between precision and recall of TIPP(0%,95%,100) versus TIPP(0%,0%,100) (positive values mean TIPP(0%,95%,100) performs better, and negative values mean that TIPP(0%,0%,100) performs better). Our results show that on average, gains in precision due to the consideration of placement support are larger than losses in recall.

A1.2.3 TIPP(0%,95%,100) versus TIPP(95%,95%,100)

Finally, we compare TIPP(0%,95%,100) versus TIPP(95%,95%,100) based on the leave-species-out study to study the effects of alignment support considerations.

Table A4 shows difference between precision and recall of TIPP(0%,95%,100) versus TIPP(95%,95%,100) (positive values mean TIPP(0%,95%,100) performs better, and negative values mean that TIPP(95%,95%,100) performs better). Many of the differences, especially at lower levels, are not statistically significant. On average gains in precision and losses of recall due to the consideration of alignment support are very close. Therefore, the decision of whether to include alignment uncertainty should depend on the application, and whether recall or precision is more important.

Table A3: The difference between precision and recall of TIPP(0%,95%,100) versus TIPP(0%,0%,100) in the leave-species-out experiments on the rpsB gene. Negative values indicate TIPP(0%,0%,100) is better, while positive values indicate that TIPP(0%,95%,100) is better. Differences in bold are statistically significant according to the Pearson’s chi-square contingency table test (as implemented in R [68]). TIPP(0%,95%,100) always results in better precision, but worse recall compared to TIPP(0%,0%,100). All differences are statistically significant. On average, gains in precision due to the consideration of placement support are larger than losses in recall.

	genus	family	order	class	phylum
Recall					
Illumina_1	-0.068	-0.055	-0.031	-0.021	-0.015
Illumina_2	-0.073	-0.055	-0.031	-0.021	-0.013
Illumina_4	-0.075	-0.057	-0.030	-0.019	-0.016
454_1	-0.047	-0.035	-0.017	-0.013	-0.009
454_2	-0.058	-0.044	-0.022	-0.012	-0.011
454_3	-0.097	-0.077	-0.050	-0.034	-0.022
mean	-0.070	-0.054	-0.030	-0.020	-0.014
Precision					
Illumina_1	0.225	0.111	0.059	0.029	0.018
Illumina_2	0.229	0.105	0.060	0.035	0.021
Illumina_4	0.239	0.119	0.060	0.031	0.021
454_1	0.161	0.069	0.032	0.013	0.008
454_2	0.200	0.089	0.044	0.021	0.012
454_3	0.301	0.181	0.099	0.046	0.026
mean	0.226	0.112	0.059	0.029	0.018

Since our goal was to reduce TIPP’s false classifications, we set the default setting for TIPP to be TIPP(95%,95%,100). This is indicated by TIPP-def (or TIPP-default).

Table A4: The difference between precision and recall of TIPP(0%,95%,100) and TIPP(95%,95%,100) in the leave-species-out experiments on the rpsB gene. Negative values indicate TIPP(95%,95%,100) is better, while positive values indicate that TIPP(0%,95%,100) is better. Differences in bold are statistically significant according to the Pearson’s chi-square contingency table test (as implemented in R [68]). TIPP(95%,95%,100) always results in better precision, but worse recall compared to TIPP(0%,95%,100). Many of the differences are not statistically significant. On average gains in precision and losses of recall due to the consideration of alignment support are very close.

	genus	family	order	class	phylum
Recall					
Illumina_1	0.006	0.015	0.021	0.023	0.028
Illumina_2	0.007	0.010	0.013	0.016	0.017
Illumina_4	0.009	0.014	0.018	0.021	0.021
454_1	0.001	0.004	0.005	0.006	0.005
454_2	0.005	0.009	0.010	0.009	0.009
454_3	0.035	0.044	0.058	0.065	0.070
mean	0.011	0.016	0.021	0.023	0.025
Precision					
Illumina_1	-0.009	-0.012	-0.017	-0.020	-0.021
Illumina_2	-0.014	-0.016	-0.018	-0.020	-0.022
Illumina_4	-0.012	-0.012	-0.015	-0.016	-0.019
454_1	-0.005	-0.005	-0.006	-0.005	-0.005
454_2	-0.006	-0.006	-0.011	-0.009	-0.008
454_3	-0.037	-0.044	-0.064	-0.069	-0.064
mean	-0.014	-0.016	-0.022	-0.023	-0.023

A1.3 Leave-one-out experiments: TIPP versus MetaPhyler

In this section we present comparisons of TIPP and MetaPhyler to directly compare the two methods that use the same set of marker genes. Here, negative values mean that TIPP is better, and positive values mean that MetaPhyler is better. Tables A5 to A10 shows results based on 30 marker genes and 16S RNA, and under both Illumina and 454 error models.

TIPP has better recall on all genes. On the 30 marker genes, TIPP generally has better precision, but in some cases (especially at the diagonal), MetaPhyler has better precision. On 16S RNA datasets, MetaPhyler usually has better precision. In most cases, TIPP's gains in recall are much greater than its losses in precision.

Table A5: The difference between precision and recall of MetaPhyler and TIPP-def in the leave-one-out experiments on 30 marker genes with Illumina error models. Negative values indicate TIPP-def is better, while positive values indicate that MetaPhyler is better. Differences in bold are statistically significant (in this case all results) according to the Pearson’s chi-square contingency table test (as implemented in R [68]). TIPP-default always has better recall, and in many cases also has better precision. Note that in all but the genus level, TIPP’s gains in recall are on average much greater than its losses in precision.

	genus	family	order	class	phylum
Recall					
species	-0.090	-0.150	-0.154	-0.153	-0.145
genus		-0.154	-0.255	-0.278	-0.264
family			-0.148	-0.262	-0.263
order				-0.213	-0.281
class					-0.212
mean	-0.090	-0.152	-0.186	-0.227	-0.233
Precision					
species	0.106	0.011	-0.006	-0.017	-0.017
genus		0.125	-0.030	-0.045	-0.037
family			0.102	-0.042	-0.032
order				0.023	-0.040
class					0.039
mean	0.106	0.068	0.022	-0.020	-0.017

Table A6: The difference between precision and recall of MetaPhyler and TIPP-def in the leave-one-out experiments on 30 marker genes with 454 error models. Negative values indicate TIPP-def is better, while positive values indicate that MetaPhyler is better. Differences in bold are statistically significant according to the Pearson’s chi-square contingency table test (as implemented in R [68]). TIPP-default always has better recall, and in many cases has better precision, too. Note that TIPP’s gains in recall are on average greater (often many times) than its losses in precision.

	genus	family	order	class	phylum
Recall					
species	-0.202	-0.222	-0.197	-0.176	-0.156
genus		-0.253	-0.325	-0.285	-0.232
family			-0.216	-0.272	-0.223
order				-0.240	-0.265
class					-0.215
mean	-0.202	-0.237	-0.246	-0.243	-0.218
Precision					
species	0.156	0.023	-0.011	-0.027	-0.026
genus		0.134	-0.039	-0.066	-0.056
family			0.070	-0.072	-0.059
order				-0.001	-0.077
class					-0.018
mean	0.156	0.079	0.007	-0.042	-0.047

Table A7: The difference between precision and recall of MetaPhyler and TIPP-def in the leave-one-out experiments on 16S RNA gene on the bacterial dataset, with Illumina error models. Negative values indicate TIPP-def is better, while positive values indicate that MetaPhyler is better. Differences in bold are statistically significant according to the Pearson’s chi-square contingency table test (as implemented in R [68]). TIPP-default always has better recall, but MetaPhyler always has better precision. Note that TIPP’s gains in recall are on average many times greater than its losses on precision.

	genus	family	order	class	phylum
Recall					
species	-0.322	-0.427	-0.262	-0.116	0.008
genus		-0.237	-0.246	-0.203	-0.086
family			-0.147	-0.226	-0.153
order				-0.183	-0.159
class					-0.169
mean	-0.322	-0.332	-0.219	-0.182	-0.112
Precision					
species	0.070	0.025	0.018	0.011	0.007
genus		0.173	0.053	0.021	0.010
family			0.203	0.058	0.028
order				0.182	0.047
class					0.205
mean	0.070	0.099	0.092	0.068	0.059

Table A8: The difference between precision and recall of MetaPhyler and TIPP-def in the leave-one-out experiments on the 16S RNA gene on bacteria, under the 454 error models. Negative values indicate TIPP-def is better, while positive values indicate that MetaPhyler is better. Differences in bold are statistically significant according to the Pearson’s chi-square contingency table test (as implemented in R [68]). TIPP-default always has better recall, except for phylum level classification with leave-out-species, and MetaPhyler always has better precision. TIPP’s gain in recall is on average greater than its loss in precision for lower taxonomic levels (genus, family, and order).

	genus	family	order	class	phylum
Recall					
species	-0.543	-0.487	-0.141	-0.008	0.003
genus		-0.451	-0.266	-0.088	-0.036
family			-0.190	-0.133	-0.035
order				-0.125	-0.038
class					-0.121
mean	-0.543	-0.469	-0.199	-0.088	-0.045
Precision					
species	0.074	0.020	0.011	0.005	0.003
genus		0.119	0.040	0.011	0.005
family			0.186	0.041	0.014
order				0.204	0.102
class					0.270
mean	0.074	0.070	0.079	0.065	0.079

Table A9: The difference between precision and recall of MetaPhyler and TIPP-def in the leave-one-out experiments on 16S RNA gene on archaea with Illumina error models. Negative values indicate TIPP-def is better, while positive values indicate that MetaPhyler is better. Differences in bold are statistically significant according to the Pearson’s chi-square contingency table test (as implemented in R [68]). TIPP-default always has better recall, but MetaPhyler has better precision in most cases. Note that at leave-class-out level, TIPP-def has better precision, and in some other cases, the differences between precision values are not statistically significant.

	genus	family	order	class	phylum
Recall					
species	-0.524	-0.353	-0.317	-0.250	-0.062
genus		-0.420	-0.506	-0.476	-0.183
family			-0.178	-0.300	-0.568
order				-0.190	-0.634
class					-0.620
mean	-0.524	-0.387	-0.333	-0.304	-0.413
Precision					
species	0.058	0.007	0.004	0.002	0.001
genus		0.045	0.009	0.001	0.000
family			0.476	0.386	0.007
order				0.524	0.006
class					-0.259
mean	0.058	0.026	0.163	0.228	-0.049

Table A10: The difference between precision and recall of MetaPhyler and TIPP-def in the leave-one-out experiments on 16S RNA gene on archaea with 454 error models. Negative values indicate TIPP-def is better, while positive values indicate that MetaPhyler is better. Differences in bold are statistically significant according to the Pearson’s chi-square contingency table test (as implemented in R [68]). TIPP-default always has better recall, but MetaPhyler always has better precision. In all but three cases (leave-out-order and family at order and class levels) the differences between recall values are greater than differences between precision values.

	genus	family	order	class	phylum
Recall					
species	-0.646	-0.233	-0.111	-0.073	-0.022
genus		-0.307	-0.278	-0.219	-0.065
family			-0.213	-0.308	-0.506
order				-0.224	-0.610
class					-0.522
mean	-0.646	-0.270	-0.200	-0.206	-0.345
Precision					
species	0.084	0.006	0.002	0.002	0.002
genus		0.066	0.004	0.001	0.002
family			0.370	0.356	0.013
order				0.448	0.012
class					0.021
mean	0.084	0.036	0.126	0.202	0.010

A2 Leave-one-out Results in Tabular Format

In this section we present the leave-one-out results in a tabular format. In each table, we show true positive and false positive classification rates (in that order). When these two numbers do not add up to one, the remaining fraction of fragments are unclassified.

A2.1 Experiment 1: TIPP Variants

Tables A11 to A13 show the leave-one-out results corresponding to Section A3.1. These show leave-one-out results comparing variants of TIPP on the *rpsB* gene, with varying error model conditions. Table A11 shows leave-species-out, Table A12 shows leave-genus-out, and Table A13 shows leave-family-out results.

A2.2 Leave-one-out experiments: TIPP versus MetaPhyler

Tables A14 and A15 show the leave-one-out results for the 30 marker genes with Illumina and 454 error models respectively. Tables A16 to A19 similarly show leave-one-out results for 16S RNA under both error model conditions.

Table A11: Leave-species-out results comparing TIPP variants on *rspB* marker gene and various error models. Rows show the TIPP variants for different error models and columns show the classification ranks. Each cell of the table shows (true positive, false positive) classification rates for the corresponding method at the corresponding level.

leaveout.higher.species	genus	family	order	class	phylum
illumina_1					
(0%,0%,ALL)	(0.564,0.407)	(0.760,0.205)	(0.845,0.141)	(0.894,0.097)	(0.919,0.080)
(0%,0%,100)	(0.599,0.372)	(0.803,0.165)	(0.884,0.105)	(0.935,0.058)	(0.955,0.045)
(0%,95%,ALL)	(0.495,0.099)	(0.707,0.052)	(0.808,0.045)	(0.868,0.036)	(0.898,0.037)
(0%,95%,100)	(0.531,0.100)	(0.749,0.048)	(0.853,0.042)	(0.915,0.028)	(0.941,0.026)
(95%,95%,100)	(0.525,0.092)	(0.734,0.037)	(0.832,0.026)	(0.891,0.009)	(0.913,0.005)
illumina_2					
(0%,0%,ALL)	(0.553,0.417)	(0.752,0.206)	(0.830,0.159)	(0.884,0.108)	(0.909,0.089)
(0%,0%,100)	(0.585,0.383)	(0.794,0.167)	(0.874,0.116)	(0.924,0.070)	(0.948,0.051)
(0%,95%,ALL)	(0.479,0.098)	(0.695,0.052)	(0.798,0.057)	(0.860,0.039)	(0.892,0.040)
(0%,95%,100)	(0.512,0.102)	(0.739,0.054)	(0.844,0.051)	(0.903,0.034)	(0.935,0.029)
(95%,95%,100)	(0.505,0.091)	(0.729,0.040)	(0.830,0.034)	(0.888,0.015)	(0.918,0.008)
illumina_4					
(0%,0%,ALL)	(0.568,0.404)	(0.750,0.213)	(0.830,0.156)	(0.883,0.107)	(0.904,0.090)
(0%,0%,100)	(0.602,0.374)	(0.797,0.174)	(0.879,0.110)	(0.933,0.062)	(0.949,0.050)
(0%,95%,ALL)	(0.491,0.086)	(0.684,0.054)	(0.790,0.053)	(0.857,0.041)	(0.882,0.042)
(0%,95%,100)	(0.527,0.089)	(0.740,0.047)	(0.849,0.046)	(0.914,0.030)	(0.933,0.028)
(95%,95%,100)	(0.517,0.079)	(0.726,0.036)	(0.831,0.031)	(0.893,0.014)	(0.912,0.009)
454_1					
(0%,0%,ALL)	(0.608,0.350)	(0.819,0.140)	(0.896,0.087)	(0.944,0.046)	(0.956,0.040)
(0%,0%,100)	(0.635,0.323)	(0.852,0.116)	(0.925,0.066)	(0.971,0.026)	(0.981,0.019)
(0%,95%,ALL)	(0.548,0.103)	(0.766,0.035)	(0.865,0.030)	(0.923,0.016)	(0.943,0.013)
(0%,95%,100)	(0.589,0.126)	(0.817,0.044)	(0.908,0.032)	(0.957,0.013)	(0.971,0.011)
(95%,95%,100)	(0.587,0.121)	(0.814,0.039)	(0.903,0.026)	(0.951,0.008)	(0.967,0.005)
454_2					
(0%,0%,ALL)	(0.559,0.395)	(0.766,0.191)	(0.837,0.146)	(0.894,0.093)	(0.920,0.072)
(0%,0%,100)	(0.613,0.344)	(0.830,0.134)	(0.901,0.089)	(0.952,0.043)	(0.972,0.028)
(0%,95%,ALL)	(0.481,0.090)	(0.699,0.041)	(0.789,0.043)	(0.861,0.030)	(0.896,0.028)
(0%,95%,100)	(0.555,0.105)	(0.786,0.042)	(0.879,0.042)	(0.940,0.022)	(0.960,0.015)
(95%,95%,100)	(0.551,0.100)	(0.777,0.036)	(0.870,0.032)	(0.931,0.013)	(0.952,0.006)
454_3					
(0%,0%,ALL)	(0.239,0.691)	(0.335,0.580)	(0.405,0.541)	(0.499,0.441)	(0.591,0.367)
(0%,0%,100)	(0.515,0.434)	(0.684,0.266)	(0.772,0.209)	(0.840,0.148)	(0.885,0.109)
(0%,95%,ALL)	(0.167,0.079)	(0.255,0.100)	(0.321,0.125)	(0.405,0.142)	(0.482,0.149)
(0%,95%,100)	(0.418,0.078)	(0.608,0.067)	(0.722,0.093)	(0.806,0.093)	(0.863,0.080)
(95%,95%,100)	(0.383,0.052)	(0.563,0.033)	(0.664,0.036)	(0.741,0.027)	(0.793,0.017)

Table A12: Leave-genus-out results comparing TIPP variants on *rspB* marker gene and various error models. Rows show the TIPP variants for different error models and columns show the classification ranks. Each cell of the table shows (true positive, false positive) classification rates for the corresponding method at the corresponding level.

leaveout.higher.genus	family	order	class	phylum
illumina_1				
(0%,0%,ALL)	(0.450,0.474)	(0.709,0.265)	(0.820,0.166)	(0.868,0.128)
(0%,0%,100)	(0.466,0.458)	(0.738,0.237)	(0.848,0.141)	(0.901,0.098)
(0%,95%,ALL)	(0.340,0.142)	(0.620,0.085)	(0.768,0.063)	(0.836,0.059)
(0%,95%,100)	(0.359,0.151)	(0.654,0.090)	(0.805,0.062)	(0.875,0.051)
(95%,95%,100)	(0.352,0.132)	(0.639,0.063)	(0.783,0.030)	(0.847,0.021)
illumina_2				
(0%,0%,ALL)	(0.449,0.471)	(0.696,0.285)	(0.811,0.178)	(0.867,0.131)
(0%,0%,100)	(0.481,0.439)	(0.741,0.238)	(0.852,0.141)	(0.907,0.092)
(0%,95%,ALL)	(0.338,0.142)	(0.609,0.095)	(0.751,0.064)	(0.828,0.058)
(0%,95%,100)	(0.367,0.159)	(0.654,0.096)	(0.799,0.062)	(0.874,0.047)
(95%,95%,100)	(0.359,0.135)	(0.638,0.068)	(0.779,0.036)	(0.850,0.021)
illumina_4				
(0%,0%,ALL)	(0.424,0.503)	(0.684,0.295)	(0.804,0.184)	(0.855,0.141)
(0%,0%,100)	(0.454,0.475)	(0.731,0.248)	(0.850,0.143)	(0.898,0.101)
(0%,95%,ALL)	(0.318,0.144)	(0.589,0.094)	(0.738,0.067)	(0.812,0.057)
(0%,95%,100)	(0.346,0.147)	(0.636,0.098)	(0.795,0.065)	(0.865,0.051)
(95%,95%,100)	(0.339,0.126)	(0.617,0.068)	(0.767,0.035)	(0.833,0.022)
454_1				
(0%,0%,ALL)	(0.501,0.397)	(0.779,0.202)	(0.893,0.095)	(0.923,0.074)
(0%,0%,100)	(0.534,0.373)	(0.822,0.158)	(0.926,0.068)	(0.948,0.050)
(0%,95%,ALL)	(0.402,0.130)	(0.705,0.074)	(0.848,0.032)	(0.895,0.024)
(0%,95%,100)	(0.451,0.140)	(0.760,0.074)	(0.888,0.034)	(0.924,0.024)
(95%,95%,100)	(0.448,0.131)	(0.753,0.062)	(0.877,0.022)	(0.914,0.015)
454_2				
(0%,0%,ALL)	(0.453,0.450)	(0.700,0.273)	(0.822,0.160)	(0.870,0.122)
(0%,0%,100)	(0.519,0.375)	(0.791,0.184)	(0.901,0.091)	(0.932,0.066)
(0%,95%,ALL)	(0.341,0.116)	(0.603,0.080)	(0.764,0.049)	(0.826,0.040)
(0%,95%,100)	(0.418,0.132)	(0.723,0.075)	(0.863,0.045)	(0.905,0.032)
(95%,95%,100)	(0.409,0.117)	(0.709,0.056)	(0.845,0.026)	(0.888,0.017)
454_3				
(0%,0%,ALL)	(0.152,0.739)	(0.289,0.649)	(0.422,0.511)	(0.538,0.416)
(0%,0%,100)	(0.361,0.530)	(0.615,0.354)	(0.757,0.231)	(0.832,0.166)
(0%,95%,ALL)	(0.090,0.133)	(0.193,0.152)	(0.312,0.163)	(0.420,0.162)
(0%,95%,100)	(0.250,0.145)	(0.522,0.149)	(0.705,0.130)	(0.803,0.106)
(95%,95%,100)	(0.233,0.088)	(0.475,0.068)	(0.648,0.047)	(0.739,0.032)

Table A13: Leave-family-out results comparing TIPP variants on *rspB* marker gene and various error models. Rows show the TIPP variants for different error models and columns show the classification ranks. Each cell of the table shows (true positive, false positive) classification rates for the corresponding method at the corresponding level.

leaveout.higher.family	order	class	phylum
illumina_1			
(0%,0%,ALL)	(0.425,0.524)	(0.736,0.245)	(0.816,0.180)
(0%,0%,100)	(0.446,0.503)	(0.769,0.211)	(0.853,0.146)
(0%,95%,ALL)	(0.345,0.199)	(0.673,0.096)	(0.775,0.083)
(0%,95%,100)	(0.369,0.205)	(0.711,0.091)	(0.819,0.072)
(95%,95%,100)	(0.357,0.172)	(0.689,0.056)	(0.791,0.037)
illumina_2			
(0%,0%,ALL)	(0.418,0.538)	(0.721,0.257)	(0.823,0.171)
(0%,0%,100)	(0.449,0.506)	(0.763,0.218)	(0.862,0.138)
(0%,95%,ALL)	(0.317,0.194)	(0.623,0.092)	(0.741,0.076)
(0%,95%,100)	(0.366,0.203)	(0.692,0.095)	(0.814,0.068)
(95%,95%,100)	(0.352,0.171)	(0.669,0.060)	(0.787,0.034)
illumina_4			
(0%,0%,ALL)	(0.413,0.542)	(0.726,0.252)	(0.809,0.185)
(0%,0%,100)	(0.438,0.515)	(0.765,0.218)	(0.847,0.151)
(0%,95%,ALL)	(0.322,0.192)	(0.640,0.092)	(0.751,0.081)
(0%,95%,100)	(0.352,0.213)	(0.689,0.098)	(0.802,0.076)
(95%,95%,100)	(0.335,0.173)	(0.662,0.060)	(0.769,0.040)
454_1			
(0%,0%,ALL)	(0.493,0.451)	(0.819,0.158)	(0.878,0.117)
(0%,0%,100)	(0.527,0.419)	(0.862,0.121)	(0.914,0.082)
(0%,95%,ALL)	(0.416,0.183)	(0.747,0.057)	(0.830,0.045)
(0%,95%,100)	(0.465,0.206)	(0.810,0.061)	(0.878,0.043)
(95%,95%,100)	(0.459,0.184)	(0.796,0.044)	(0.867,0.029)
454_2			
(0%,0%,ALL)	(0.417,0.529)	(0.745,0.231)	(0.826,0.168)
(0%,0%,100)	(0.475,0.464)	(0.824,0.155)	(0.896,0.101)
(0%,95%,ALL)	(0.325,0.179)	(0.662,0.078)	(0.760,0.062)
(0%,95%,100)	(0.409,0.212)	(0.765,0.074)	(0.850,0.055)
(95%,95%,100)	(0.398,0.178)	(0.746,0.047)	(0.828,0.033)
454_3			
(0%,0%,ALL)	(0.172,0.760)	(0.371,0.562)	(0.515,0.439)
(0%,0%,100)	(0.373,0.584)	(0.692,0.292)	(0.806,0.191)
(0%,95%,ALL)	(0.104,0.196)	(0.270,0.178)	(0.397,0.170)
(0%,95%,100)	(0.298,0.281)	(0.634,0.180)	(0.776,0.131)
(95%,95%,100)	(0.266,0.171)	(0.581,0.070)	(0.708,0.039)

Table A14: Leave-one-out results for 30 marker genes Illumina error model. Rows show the left-out clade and columns show the classification rank. For each rank (true positive, false positive) rates are shown.

leaveout.illumina	genus	family	order	class	phylum
Metaphyler					
species	(0.429,0.024)	(0.586,0.027)	(0.673,0.031)	(0.745,0.026)	(0.776,0.023)
genus		(0.190,0.032)	(0.353,0.060)	(0.495,0.048)	(0.570,0.042)
family			(0.177,0.060)	(0.387,0.062)	(0.493,0.050)
order				(0.234,0.061)	(0.365,0.062)
class					(0.181,0.059)
TIPP-def					
species	(0.519,0.099)	(0.735,0.042)	(0.827,0.033)	(0.898,0.015)	(0.921,0.011)
genus		(0.345,0.127)	(0.609,0.078)	(0.773,0.035)	(0.834,0.028)
family			(0.325,0.178)	(0.649,0.070)	(0.756,0.049)
order				(0.448,0.134)	(0.645,0.075)
class					(0.393,0.158)

Table A15: Leave-one-out results for 30 marker genes 454 error model. Rows show the left-out clade and columns show the classification rank. For each rank (true positive, false positive) rates are shown.

	genus	family	order	class	phylum
Metaphyler					
species	(0.385,0.018)	(0.595,0.021)	(0.705,0.033)	(0.781,0.031)	(0.815,0.028)
genus		(0.224,0.026)	(0.447,0.062)	(0.617,0.060)	(0.697,0.055)
family			(0.240,0.072)	(0.518,0.082)	(0.638,0.072)
order				(0.340,0.085)	(0.514,0.091)
class					(0.306,0.099)
TIPP-def					
species	(0.587,0.147)	(0.817,0.050)	(0.902,0.031)	(0.957,0.010)	(0.971,0.007)
genus		(0.477,0.150)	(0.772,0.069)	(0.902,0.021)	(0.929,0.016)
family			(0.456,0.196)	(0.790,0.055)	(0.861,0.037)
order				(0.580,0.143)	(0.779,0.061)
class					(0.521,0.153)

Table A16: Leave-one-out results for 16S RNA gene on bacteria, under the 454 error model. Rows show the left-out clade and columns show the classification rank. For each rank (true positive, false positive) rates are show.

	genus	family	order	class	phylum
Metaphyler					
species	(0.149,0.001)	(0.381,0.002)	(0.770,0.002)	(0.954,0.001)	(0.979,0.001)
genus		(0.145,0.003)	(0.524,0.006)	(0.811,0.005)	(0.897,0.003)
family			(0.312,0.011)	(0.624,0.009)	(0.804,0.008)
order				(0.355,0.009)	(0.593,0.013)
class					(0.313,0.028)
TIPP- large					
species	(0.692,0.061)	(0.868,0.022)	(0.910,0.013)	(0.961,0.007)	(0.975,0.004)
genus		(0.596,0.098)	(0.790,0.042)	(0.899,0.015)	(0.933,0.008)
family			(0.502,0.143)	(0.756,0.045)	(0.839,0.020)
order				(0.480,0.142)	(0.632,0.088)
class					(0.434,0.236)

Table A17: Leave-one-out results for 16S RNA gene on bacteria, under the Illumina error model. Rows show the left-out clade and columns show the classification rank. For each rank (true positive, false positive) rates are show.

	genus	family	order	class	phylum
Metaphyler					
species	(0.031,0.001)	(0.090,0.002)	(0.391,0.003)	(0.638,0.003)	(0.825,0.002)
genus		(0.035,0.002)	(0.269,0.005)	(0.456,0.007)	(0.655,0.005)
family			(0.189,0.008)	(0.327,0.010)	(0.523,0.009)
order				(0.113,0.008)	(0.296,0.015)
class					(0.104,0.012)
TIPP- large					
species	(0.353,0.041)	(0.517,0.026)	(0.654,0.017)	(0.754,0.013)	(0.817,0.007)
genus		(0.272,0.080)	(0.515,0.041)	(0.658,0.024)	(0.741,0.014)
family			(0.337,0.109)	(0.552,0.053)	(0.676,0.032)
order				(0.296,0.096)	(0.455,0.047)
class					(0.274,0.122)

Table A18: Leave-one-out results for 16S gene on archaea under the 454 error model. Rows show the left-out clade and columns show the classification rank. For each rank (true positive, false positive) rates are show.

	genus	family	order	class	phylum
Metaphyler					
species	(0.101,0.000)	(0.734,0.000)	(0.877,0.000)	(0.923,0.000)	(0.975,0.000)
genus		(0.523,0.001)	(0.689,0.001)	(0.773,0.000)	(0.931,0.000)
family			(0.107,0.003)	(0.174,0.006)	(0.468,0.000)
order				(0.039,0.005)	(0.354,0.000)
class					(0.305,0.014)
TIPP- def					
species	(0.747,0.070)	(0.966,0.007)	(0.988,0.002)	(0.996,0.002)	(0.997,0.002)
genus		(0.830,0.061)	(0.968,0.005)	(0.991,0.002)	(0.997,0.002)
family			(0.320,0.208)	(0.482,0.309)	(0.974,0.013)
order				(0.262,0.343)	(0.964,0.012)
class					(0.827,0.058)

Table A19: Leave-one-out results for 16S RNA gene on archaea, under the Illumina error model. Rows show the left-out clade and columns show the classification rank. For each rank (true positive, false positive) rates are show.

	genus	family	order	class	phylum
Metaphyler					
species	(0.025,0.001)	(0.562,0.002)	(0.647,0.001)	(0.739,0.000)	(0.934,0.001)
genus		(0.353,0.003)	(0.405,0.002)	(0.483,0.001)	(0.810,0.001)
family			(0.034,0.002)	(0.088,0.005)	(0.364,0.003)
order				(0.035,0.004)	(0.277,0.004)
class					(0.149,0.087)
TIPP- def					
species	(0.549,0.058)	(0.916,0.010)	(0.964,0.005)	(0.989,0.002)	(0.996,0.001)
genus		(0.773,0.044)	(0.911,0.012)	(0.959,0.003)	(0.994,0.002)
family			(0.212,0.247)	(0.388,0.303)	(0.932,0.014)
order				(0.226,0.360)	(0.910,0.018)
class					(0.768,0.096)

A3 Results Omitted from Chapter 5

A3.1 Experiment 1: Leave-one-out TIPP Variants

In Chapter 5, in *Experiment 1: TIPP variants* section, we discussed results on different variants of TIPP. Here we compare different variants of TIPP under 3 different leave-one-out experiment settings for the rpsB marker gene: leave-species-out (Figure A1), leave-genus-out (Figure A2), and leave-family-out (Figure A3).

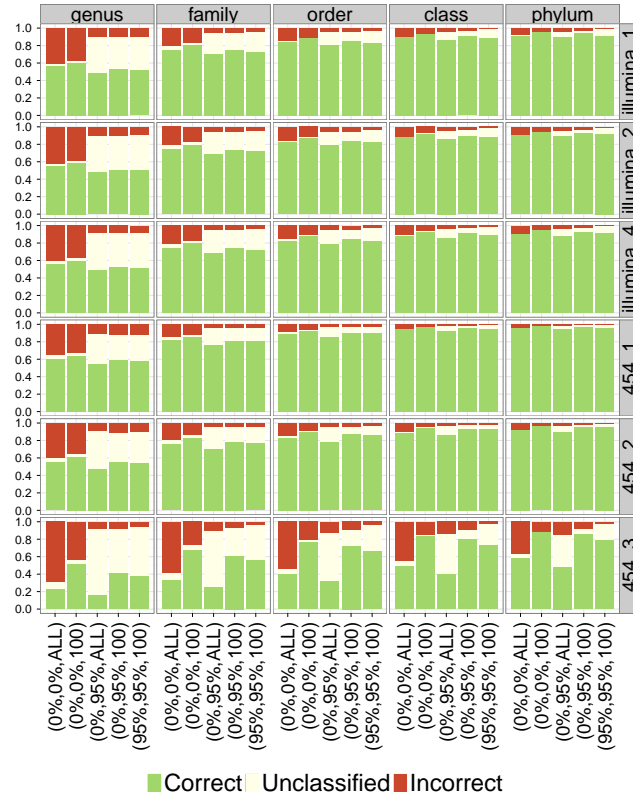


Figure A1: Leave-species-out experiment on the rpsB marker gene, comparing the classification accuracy of different variants of TIPP. Each variant is labeled by (X,Y,Z), where X refers to alignment support (s_a), Y refers to placement support (s_p), and Z refers to alignment subset size (m_a). Note that SEPP with $m_a = 100$ is identical to TIPP(0%,0%,100), and that HMMER+pplacer is identical to TIPP(0%,0%,ALL).

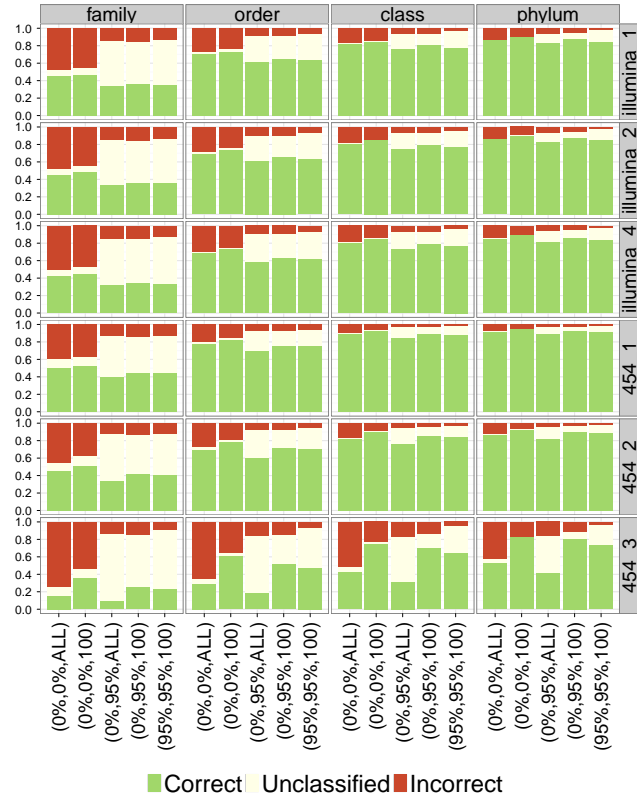


Figure A2: Leave-genus-out experiment on the rpsB marker gene, comparing the classification accuracy of different variants of TIPP. Each variant is labeled by (X,Y,Z), where X refers to alignment support (s_a), Y refers to placement support (s_p), and Z refers to alignment subset size (m_a).

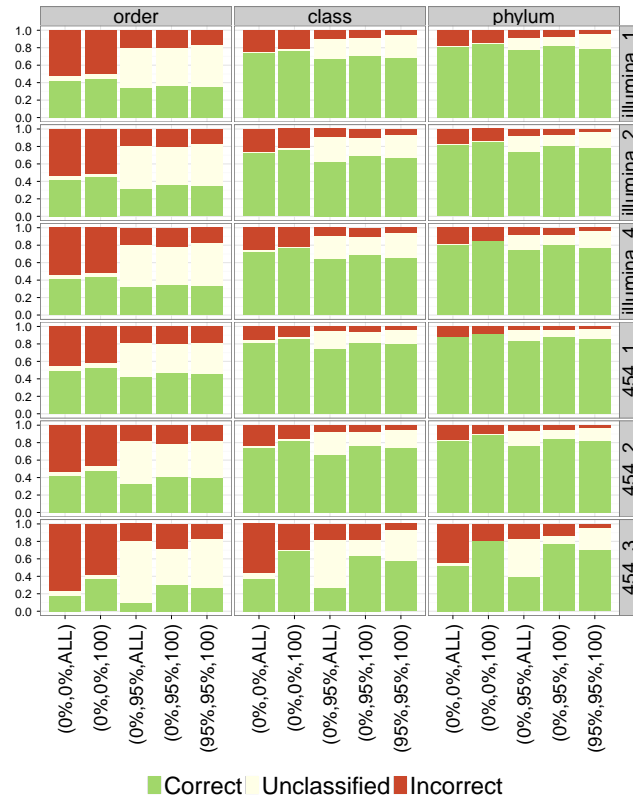


Figure A3: Leave-family-out experiment on the *rpsB* marker gene, comparing the classification accuracy of different variants of TIPP. Each variant is labeled by (X,Y,Z), where X refers to alignment support (s_a), Y refers to placement support (s_p), and Z refers to alignment subset size (m_a).

A3.2 TIPP Boosting of EPA versus pplacer

TIPP requires an external placement tool for its placement step. While the initial submission only used pplacer, the current current implementation of TIPP can use both pplacer and EPA. The results in Chapter 5 are based on using pplacer internally, and here we show results on using EPA inside TIPP, compared to using pplacer inside TIPP, in a leave-species-out experiment on the *rpsB* marker gene. We observe that in this experiment, TIPP using pplacer and EPA are almost identical (Figure A4); the differences in recall between the two techniques are statistically significant only when placing at the class or phylum level and the differences between precision is never statistically significant (Table A20).

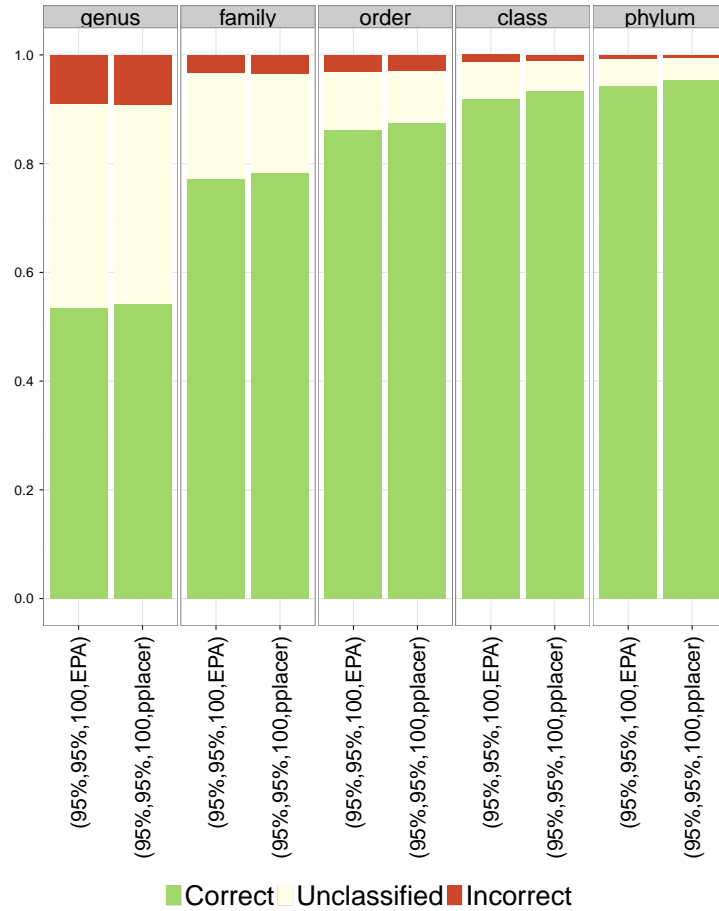


Figure A4: Leave-one-out experiment comparing the classification accuracy for TIPP with default settings when it uses EPA or pplacer internally for the placement step. Results are for a leave-species-out experiment on the rpsB marker gene with Illumina-like errors. Differences in recall are statistically significant at the class and phylum levels, but not below the class level.

Table A20: Precision and recall of TIPP when it uses EPA or pplacer internally for the placement step. The table shows the difference between precision and recall values of the two techniques (delta) and p-value of a statistical test showing whether the differences are statistically significant according to the Pearson’s chi-square contingency table test (as implemented in R [68]). Positive values mean TIPP with pplacer was better than TIPP with EPA. Results are for a leave-species-out experiment on the rpsB marker gene with Illumina-like errors. Differences in recall are statistically significant at the class and phylum levels, but not below the class level. In all other cases differences are not statistically significant.

	genus	family	order	class	phylum
Recall					
Delta	0.006	0.012	0.013	0.013	0.011
p-value	0.5617	0.1975	0.0763	0.0244	0.0165
Precision					
Delta	0.000	-0.001	0.002	0.003	0.001
p-value	0.9960	0.8662	0.6282	0.2412	0.4704

A3.3 Non-leave-one-out Parameter Exploration Study

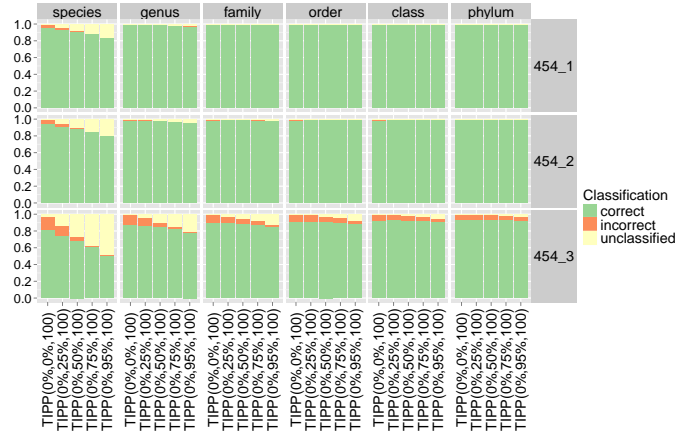
In this section we report on non-leave-one-out experiments performed on the rpsB marker gene in order to further understand the impact of parameter settings on the accuracy of TIPP. Note that results from these non-leave-one-out experiments should be interpreted in conjunction with leave-one-out results presented earlier. The impact of parameter settings could be quite different between non-leave-one-out and leave-one-out results, hence caution is required in interpreting the results. In general, changing TIPP parameters show a higher impact on classification accuracy in leave-one-out experiments. In non-leave-one-out experiments, the impact of changes to the TIPP parameters is often most observable at the highest error model conditions.

Placement Support. Placement support has a large impact on the overall classification accuracy (Fig. A5). For both types of sequencing error, the largest of varying placement support is at the species level; increasing the placement support results in fewer incorrect and correct classifications. The impact of placement support is most visible on 454 models with higher rates of error. A sizeable portion of the false positives can be removed by using higher placement support values.

Alignment Support. Figures A6 and A7 show the result of fixing the placement support to be 50% or 95%, and changing the alignment support threshold. Increasing the alignment support has a slight impact on the overall



(a) Illumina-like fragments



(b) 454-like fragments

Figure A5: Non-leave-one-out experiments showing the impact of changing placement support on the classification accuracy for fragments simulated from the *rpsB* gene with (c) Illumina-like errors and (d) 454-error like errors. Each column is the classification accuracy for a taxonomic rank and each row is the error model used. $TIPP(X\%,Y\%,Z)$ refers to TIPP run under the default settings with an alignment support of X , placement support of Y , and maximum alignment decomposition subset size of Z .

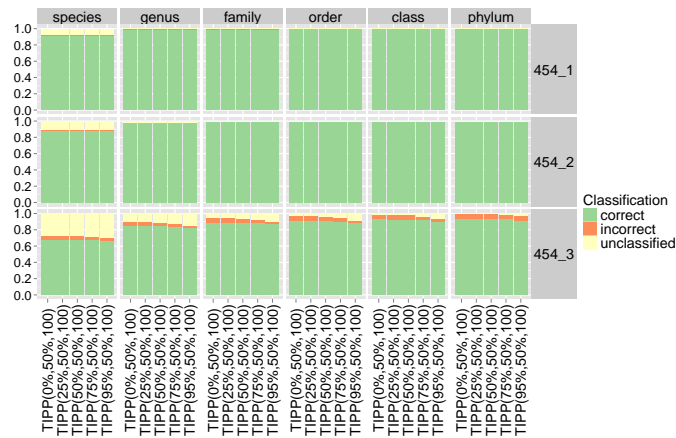
classification accuracy for the Illumina-type errors. The differences between the percentage of fragments classified at all levels for 0% alignment support and 95% alignment support is less than 5 percentage points. For the 454-type errors, the impact of alignment support is only noticeable for the higher error model conditions. Note that leave-one-out results were impacted more by varying alignment support.

Alignment Support and Placement Support. Figure A8 shows the result of changing alignment support and placement support together. TIPP(0%,0%,100) tends to over-classify, resulting in the largest percentage of incorrect classifications. Both TIPP(25%,25%,100) and TIPP(50%,50%,100) result in a drop of correct classifications at the species level, but, at the same time, a larger drop in incorrect classifications at the species level. The drop in correct classifications is not noticeable for the higher taxonomic levels on the error models with low rates of error. TIPP(95%,95%,100) has a large decrease of correct classifications at the species level, but also has the fewest incorrect classifications at all levels for all error models; nearly all the error for the lower error model conditions are eliminated.

Impact of maximum alignment subset size. We found that using smaller alignment subset sizes tends to improve accuracy, especially at the species level, but also increases the running time as there are more alignment subsets to analyze: the wall clock time to classify 200,000 fragments from the

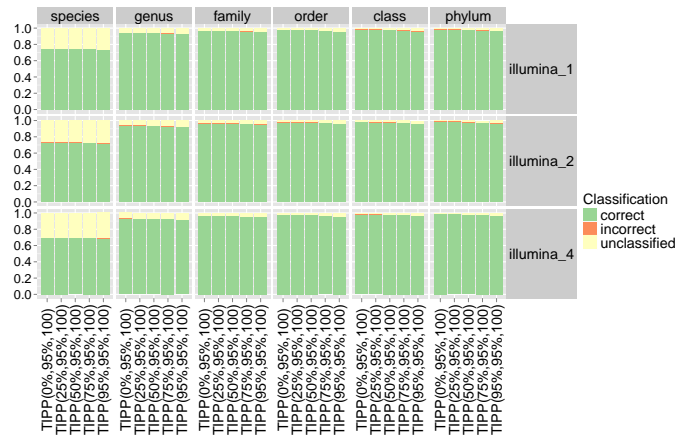


(a) Illumina-like fragments

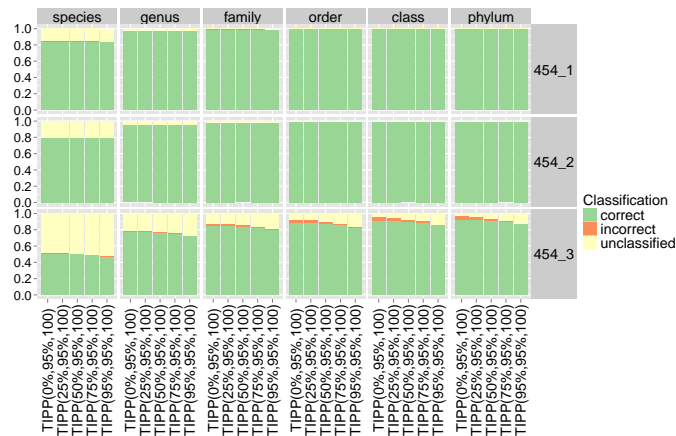


(b) 454-like fragments

Figure A6: Non-leave-one-out experiments showing the impact of changing alignment support while fixing the placement support to be 50% on the classification accuracy for fragments simulated under the (a) Illumina error model and (b) 454 error model for the rpsB marker gene. Each column is the classification accuracy for a taxonomic rank and each row is the error model used. TIPP($X\%$, $Y\%$, Z) refers to TIPP run under the default settings with an alignment support of X , placement support of Y , and maximum alignment decomposition subset size of Z .



(a) Illumina-like fragments

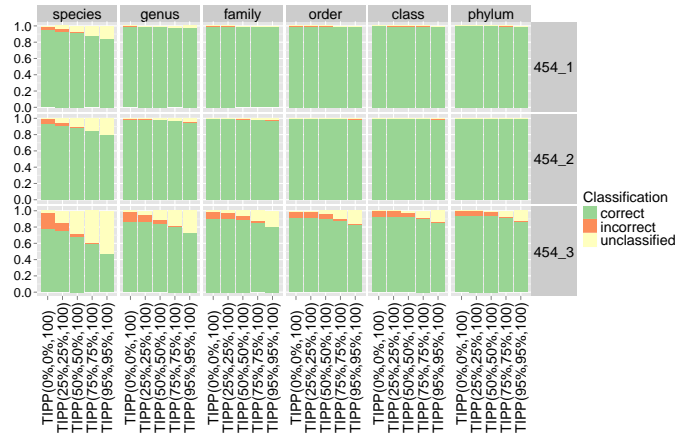


(b) 454-like fragments

Figure A7: Non-leave-one-out experiments showing the impact of changing alignment support while fixing the placement support to be 95% on the classification accuracy for fragments simulated under the (a) Illumina error model and (b) 454 error model for the rpsB marker gene. Each column is the classification accuracy for a taxonomic rank and each row is the error model used. TIPP($X\%, Y\%, Z$) refers to TIPP run under the default settings with an alignment support of X , placement support of Y , and maximum alignment decomposition subset size of Z .



(a) Illumina-like fragments



(b) 454-like fragments

Figure A8: Non-leave-one-out experiments showing the impact of changing both alignment support and placement support on classification accuracy for fragments simulated from the *rpsB* gene with (a) Illumina-like errors and (b) 454-error like errors. Each column is the classification accuracy for a taxonomic rank and each row is the error model used. TIPP($X\%$, $Y\%$, Z) refers to TIPP run under the default settings with an alignment support of X , placement support of Y , and maximum alignment decomposition subset size of Z .

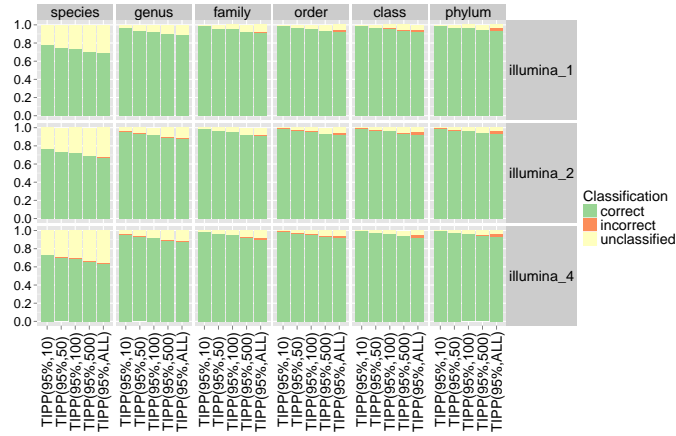
rpsB gene ranged from 2.1 hours for TIPP(95%,95%,ALL) to 3.6 hours for TIPP-default (TIPP(95%,95%,100)) (each run with 4 CPUs). The number of sequences in the reference alignment also impacts the running time for the TIPP(95%,95%,ALL) method, but this scales (at most) linearly with the number of sequences. The running time shown for the rpsB gene is a good case study, since the reference alignment has 1463 sequences and is one of the larger datasets in this study.

Figure A9 shows the impact of changing the alignment decomposition size on TIPP(95%,95%). The result shows that, in general, decreasing the alignment decomposition size increases the percentage of correctly classified fragments, as well as decreases the percentage of incorrectly classified fragments. In other words, smaller alignment subsets produce more accurate placements. However, using smaller alignment decomposition sizes results in an increase in running time.

A3.4 ROC Curves

Here we explore the impact of change in support threshold for alignment subsets of size 10 in a non-leave-one-out experiment on the rpsB marker gene (one of the hardest in the dataset), as alignment and placement support thresholds are increased progressively.

The ROC curves shown in Figure A10 show that the impact of the support thresholds is very visible at the species level, very reduced at the genus level, and then largely eliminated at the family level and above. The



(a) Illumina-like fragments



(b) 454-like fragments

Figure A9: Non-leave-one-out experiments showing the impact of changing the size of the alignment decomposition for TIPP(95%) for fragments simulated from the rpsB gene with (a) Illumina-like errors and (b) 454-error like errors. Each column is the classification accuracy for a taxonomic rank and each row is the error model used. TIPP($X\%$, Y) refers to TIPP run under the default settings with an alignment support and placement support of X and a maximum alignment decomposition subset size of Y .

curves for the species-level classification show that there is a tight relationship between the precision and recall as the support varies between 0% to 50%, but that the gain in precision moving from 50% support to 95% support is significantly smaller than the loss in recall (1-2% gain in precision but 6-10% drop in recall).

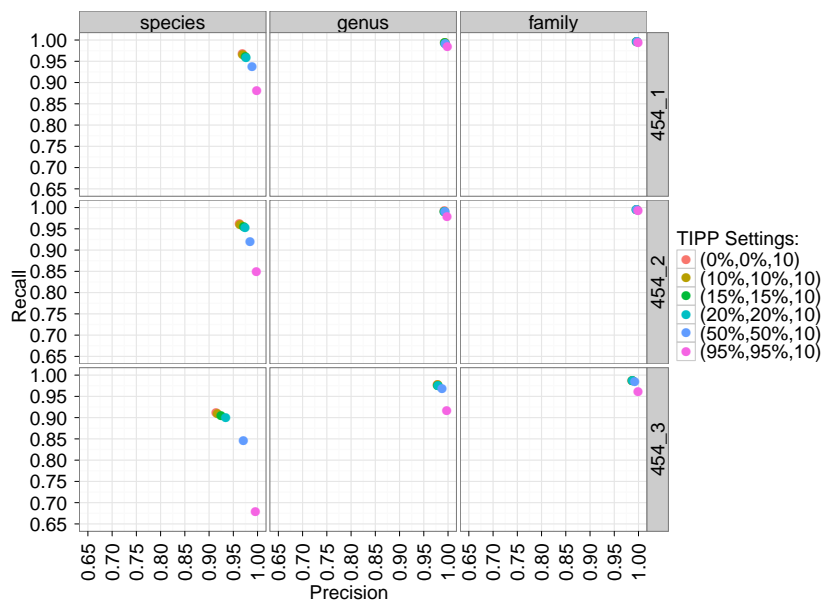


Figure A10: ROC curve showing the impact of the different support thresholds on precision and recall for species-level to family-level classification of fragments simulated from the rpsB marker genes in a non-leave-one-out experiment under the Illumina-error models. Note TIPP(0%,0%,10) is the same as SEPP with alignment subset size $m_a = 10$.

A3.5 Leave-one-out TIPP versus MetaPhyler

Next, to compare TIPP and MetaPhyler, we performed a leave-one-out study on the 30 marker genes used in the original MetaPhyler paper and on the 16S gene. Here we show results for the default setting for TIPP (i.e., TIPP(95%,95%,100)).

A3.5.1 Leave-one-out 30 marker genes

Figure A11 shows the result for the leave-one-out experiments for the 30 marker genes. TIPP has higher recall on these data than MetaPhyler, and the differences are substantial and statistically significant (p-values $\ll 10^{-5}$). The comparison with respect to precision is very interesting: while TIPP generally had better precision in about two-thirds of the cases, MetaPhyler had better precision one third of the time (except in one case, differences are statistically significant, p-value $\ll 10^{-5}$; see Section A1.3). Furthermore, the relative performance depended on the taxonomic level, so that MetaPhyler had better precision at the lower taxonomic levels, and TIPP had better precision at the higher taxonomic levels.

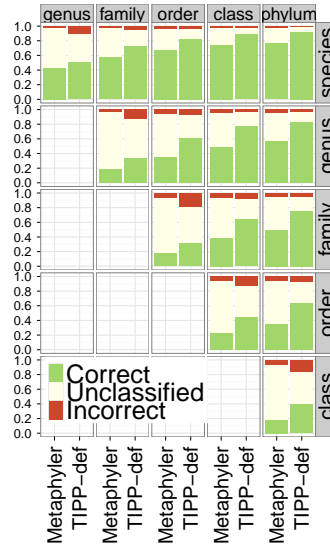
A3.5.2 Leave-one-out 16S RNA gene

Figures A12-A13 show the result for the leave-one-out experiments for the 16S rRNA gene. TIPP classifies more fragments correctly than MetaPhyler on these data, especially at the lower taxonomic levels. MetaPhyler generally had very low false classification rates. TIPP's false classification rates were

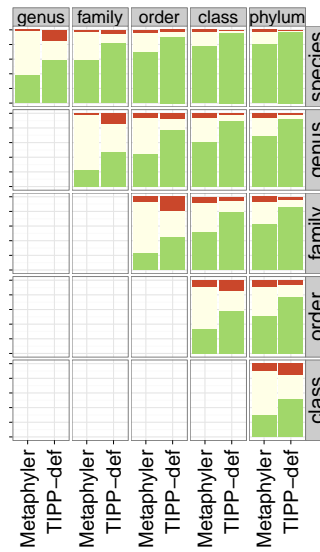
generally low, except when a taxonomic clade at the family or higher level is removed and classification is tested at the next taxonomic level. A detailed analysis of these cases revealed that the false classifications were mostly due to peculiarities in the taxonomy, potentially due to sparse taxonomic sampling.

For example, in the 16S Archaea taxonomy, the Halobacteria class has exactly one family. Therefore, the leave-one-family-out experiment results in an imbalanced taxonomy with no sequences from the Halobacteria class present in the taxonomy, and the nearest relatives are at the phylum level. As a result, it is impossible to correctly classify the fragments at the order or class level. Because TIPP tends to classify fragments if it can do so with some confidence, this results in a higher false classification rate (see Section A3.6 for more detailed discussion).

MetaPhyler generally had better precision than TIPP, but at a substantial cost in recall, especially at the lower taxonomic levels. For example, in the leave-out-species experiments for the 454 bacterial 16S rRNA fragments, MetaPhyler classified only 15% correctly at the genus level, and TIPP classified 69% correctly. On the other hand, the false classification rate for MetaPhyler was quite low, varying from less than 1% to 3%, while the false classification rate for TIPP was somewhat higher. On the bacterial 16S rRNA gene set, the false classification rate for TIPP was quite low (with the exception of the phylum level in the leave-one-class-out experiment). On the archaeal 16S rRNA genes, TIPP generally had low false classification rates, but there were a few cases where TIPP had high false classification rates.

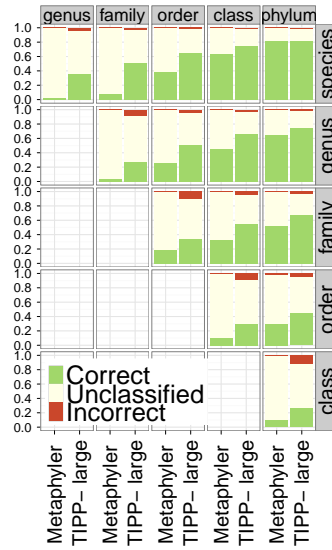


(a) Illumina error model

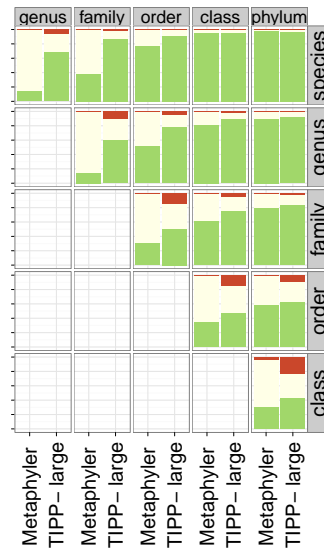


(b) 454 error model

Figure A11: Leave-one-out experiment comparing the classification accuracy for MetaPhyler versus TIPP-default (i.e., TIPP-default refers to TIPP(95%,95%,100)) on the 30 marker genes.



(a) 16S bacteria; Illumina error

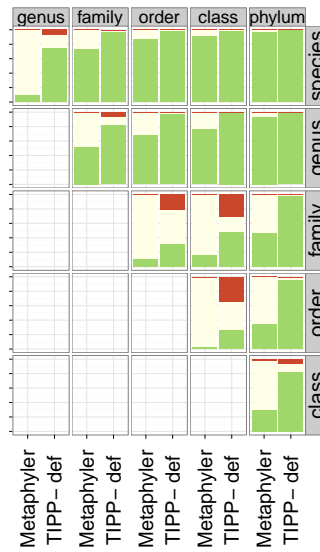


(b) 16S bacteria; 454 error

Figure A12: Leave-one-out experiment comparing MetaPhyler to TIPP on the 16S bacteria datasets, with both Illumina-like and 454-like error models. TIPP-large is similar to TIPP-def, except placement size is set to 1,000, and the taxonomic tree is used for alignment decomposition.



(a) 16S archaea; Illumina error



(b) 16S archaea; 454 error

Figure A13: Leave-one-out experiment comparing MetaPhyler to TIPP on the 16S archaea datasets, with both Illumina-like and 454-like error models. TIPP-def refers to TIPP run under the default settings (i.e. TIPP(95%,95%,100)).

A3.6 16S RNA on archaea, leave-one-out experiments; effects of Halobacteria

In this section we discuss the particularly high false classification rates for the leave-family-out and leave-order-out experiments for 16S RNA on archaea (). We find that the majority of the false classifications are caused by the Halobacteria class, and 37% of all fragments from the 16S RNA on Archaea dataset belong to this class. For the leave-family-out experiment, the total numbers of incorrect classifications for TIPP(95%,95%,100) at the order level and class level were 13,651 and 18,456, respectively. Of those incorrect classifications, 9,223 of the 13,651 (68%) and 15,187 of the 18,456 (82%) belong to fragments from this class.

Figure A14 highlights the reason for the large number of incorrect classifications. Most normal OTUs have more than direct child OTU, i.e. phyla typically have more than one class, and classes typically have more than one order. The Halobacteria class has only one order, and that order has only one family. Thus, removing either the family or the order prunes the entire class from the taxonomy. This makes it impossible to correctly classify fragments at either the order level or the class level. Note that this phenomenon is not unique to Halobacteria: any OTU that has exactly one direct child OTU will be removed completely when the child OTU is omitted in the leave-one-out experiment. This is most notable in Halobacteria because of the large number of fragments belonging to this class.

Figures A16 and A18 shows the result of omitting fragments from

this class from the leave-family-out and leave-order-out experiments. When Halobacteria is omitted, the incorrect classification rate drops and is in line with the incorrect classification rates for the 16S experiments.

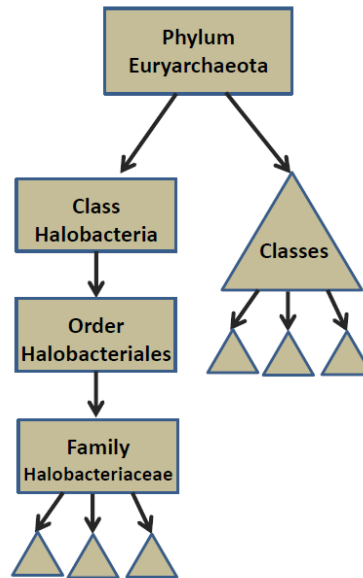


Figure A14: Taxonomy for Halobacteria class for the 16S RNA gene.

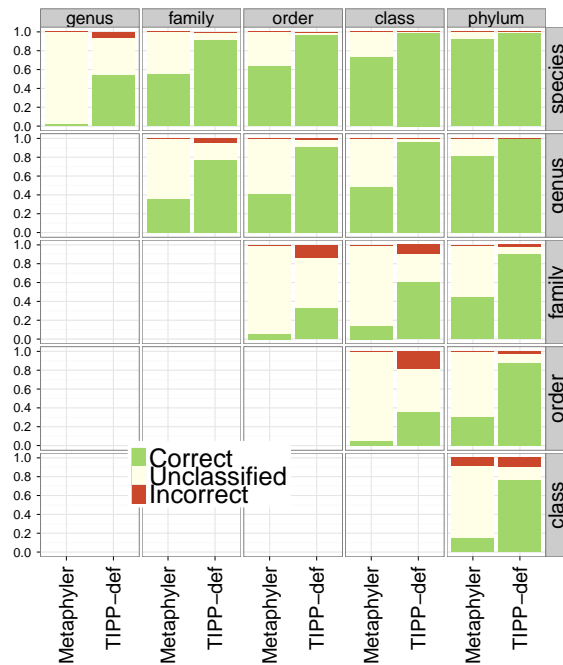


Figure A15: Illumina-like fragments

Figure A16: Removing Halobacteria class from leave-family-out and leave-order-out experiments for 16S RNA archaea gene for fragments simulated with Illumina-like errors . Each column is the classification accuracy for a taxonomic rank and each row is level being omitted.

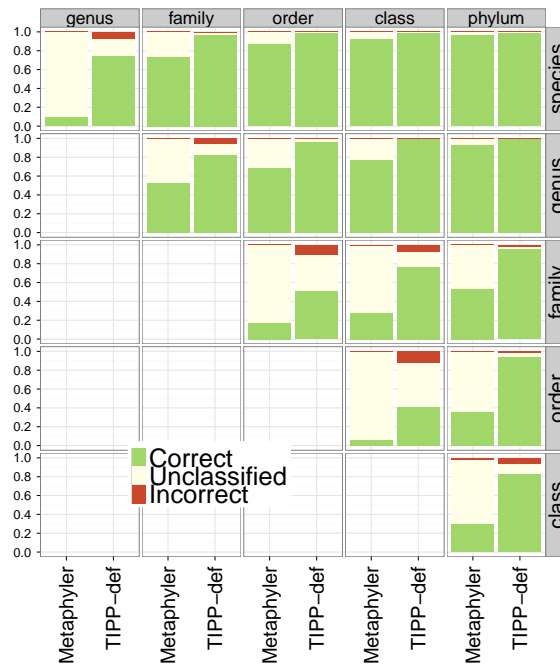


Figure A17: 454-like fragments

Figure A18: Removing Halobacteria class from leave-family-out and leave-order-out experiments for 16S RNA archaea gene for fragments simulated with 454-error like errors. Each column is the classification accuracy for a taxonomic rank and each row is level being omitted.

A3.7 Experiment 2: Abundance profiling experiments

We show the impact of alignment and placement support on abundance profiling (Fig. A19 and A20). Although results depended on the particular dataset, the following trends can be observed. On the short fragment datasets, on average using the 0% threshold improved average abundance profiles at the lower taxonomic levels and was neutral at the phylum level. On the long fragment datasets, the change in threshold had essentially no impact. This result lead us to select TIPP(0%,0%,100) for abundance profiling.

In Chapter 5, we compare TIPP-default against other abundance profiling methods. The tabular results for the figures are shown in Tables A21 and A22.

A3.8 Experiment 3: Exploring robustness to sequencing error on taxonomic identification experiments.

In Chapter 5 we showed non-leave-one-out results comparing TIPP(95%,95%,100), MetaPhyler, PhmmBL, and NBC on all marker genes under the 454.3 error model condition. Here we also show results on all remaining error model conditions. Figure A21 shows results under different rates of Illumina-like and 454-like errors.

Figure A22 show results for false positive detection of “dark matter” under the assumption that any read left unclassified at the phylum level comes from a novel phylum.

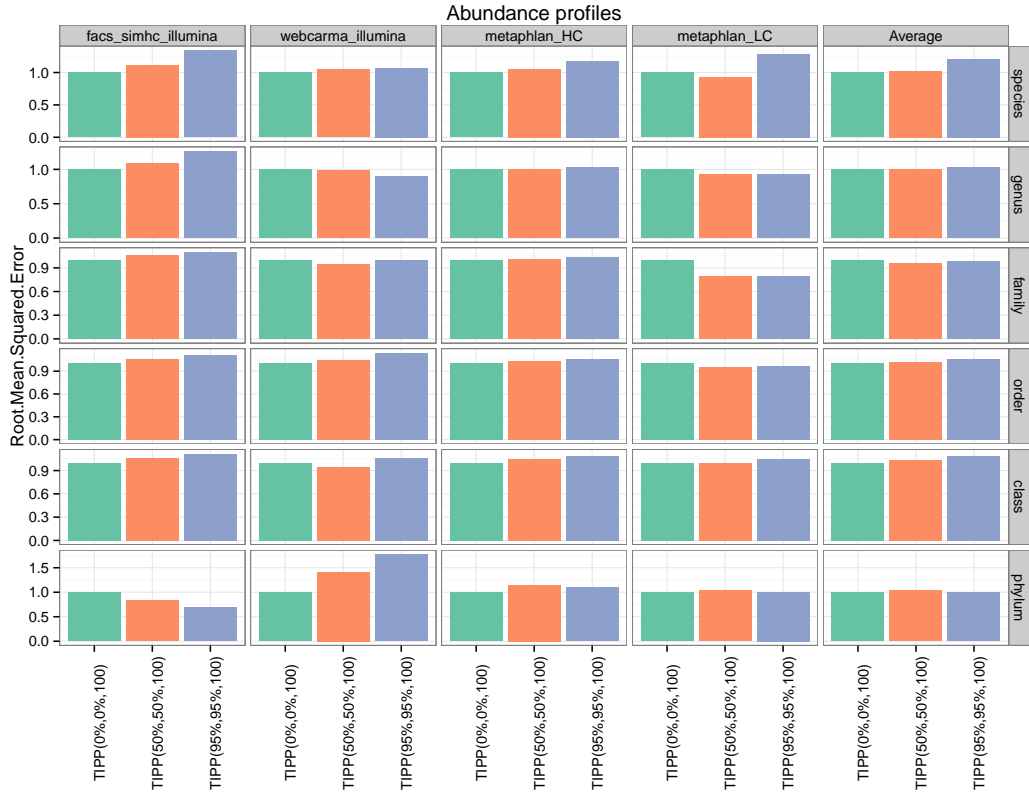


Figure A19: Abundance profiling results comparing different TIPP methods on short fragments. The RMSE has been normalized by TIPP(0%,0%,100)'s RMSE.

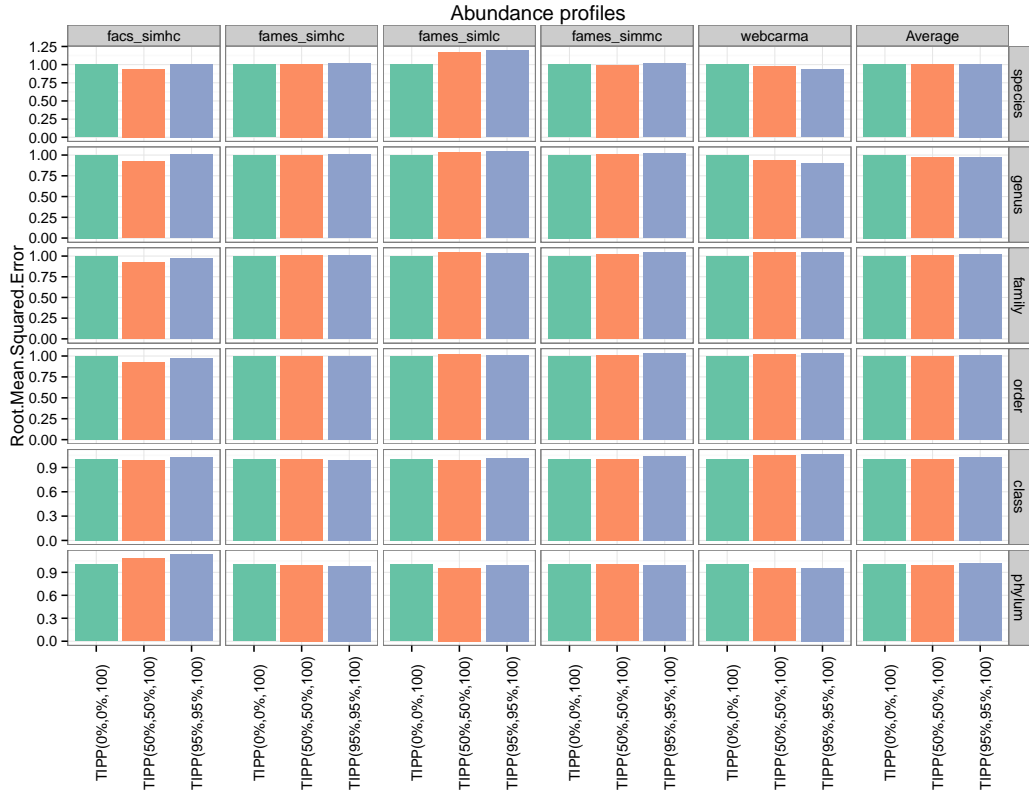


Figure A20: Abundance profiling results comparing different TIPP methods on long fragments. The RMSE has been normalized by TIPP(0%,0%,100)'s RMSE.

Table A21: The *RMSE* for different methods on the short fragment datasets, normalized by TIPP(0%,0%,100)'s *RMSE* for each model condition and each taxonomic rank. Thus methods with *RMSE* > 1 have worse performance than TIPP, and methods with *RMSE* < 1 have better performance than TIPP. Note that PhymmBL does not output species level classification.

Dataset	Species	Genus	Family	Order	Class	Phylum
FACs HC Illumina						
NBC	1.889	2.278	2.226	2.241	1.431	3.111
PhymmBL	NA	2.254	2.186	2.201	1.405	3.035
MetaPhlAn	1.134	1.101	1.403	1.054	0.967	2.018
MetaPhyler	5.324	2.095	1.496	1.351	1.279	0.743
TIPP(0%,0%,100)	1.000	1.000	1.000	1.000	1.000	1.000
TIPP(95%,95%,100)	1.341	1.264	1.101	1.104	1.119	0.691
WebCarm Illumina						
NBC	1.132	1.483	2.768	2.806	4.116	8.142
PhymmBL	NA	1.492	2.693	2.589	3.508	6.052
MetaPhlAn	1.321	1.576	1.377	2.229	2.720	5.595
MetaPhyler	3.443	1.816	2.181	1.738	1.759	1.630
TIPP(0%,0%,100)	1.000	1.000	1.000	1.000	1.000	1.000
TIPP(95%,95%,100)	1.066	0.899	0.999	1.141	1.065	1.773
MetaPhlAn HC						
NBC	1.200	2.066	2.192	2.474	2.234	1.985
PhymmBL	NA	2.061	2.210	2.500	2.217	2.023
MetaPhlAn	0.585	0.742	0.833	0.822	0.769	0.445
MetaPhyler	5.037	2.165	1.307	1.268	1.186	1.103
TIPP(0%,0%,100)	1.000	1.000	1.000	1.000	1.000	1.000
TIPP(95%,95%,100)	1.176	1.028	1.035	1.060	1.093	1.094
MetaPhlAn LC						
NBC	2.020	2.175	2.644	2.483	2.205	1.708
PhymmBL	NA	2.200	2.624	2.462	2.132	1.675
MetaPhlAn	0.492	0.527	0.725	0.952	0.787	0.628
MetaPhyler	10.000	7.744	4.169	2.051	1.880	1.803
TIPP(0%,0%,100)	1.000	1.000	1.000	1.000	1.000	1.000
TIPP(95%,95%,100)	1.286	0.919	0.789	0.967	1.044	1.004
Average						
NBC	1.595	1.991	2.435	2.440	2.038	2.661
PhymmBL	NA	1.993	2.403	2.386	1.934	2.487
MetaPhlAn	0.931	1.029	1.128	1.184	1.103	1.333
MetaPhyler	6.143	3.642	2.310	1.604	1.460	1.278
TIPP(0%,0%,100)	1.000	1.000	1.000	1.000	1.000	1.000
TIPP(95%,95%,100)	1.211	1.030	0.988	1.064	1.092	0.997

Table A22: The *RMSE* for different methods on the long fragment datasets, normalized by TIPP(0%,0%,100)’s *RMSE* for each model condition and each taxonomic rank. Thus methods with *RMSE* > 1 have worse performance than TIPP, and methods with *RMSE* < 1 have better performance than TIPP. Note that PhymmBL does not output species level classification.

Dataset	Species	Genus	Family	Order	Class	Phylum
FACs HC						
NBC	1.554	1.864	2.008	2.030	1.397	3.124
PhymmBL	NA	1.727	1.869	1.885	1.236	2.593
MetaPhlAn	1.220	1.087	1.533	1.239	0.661	1.720
MetaPhyler	5.338	2.048	1.441	1.366	1.176	1.268
TIPP(0%,0%,100)	1.000	1.000	1.000	1.000	1.000	1.000
TIPP(95%,95%,100)	1.007	1.002	0.967	0.977	1.028	1.134
FAMeS HC						
NBC	0.797	0.894	0.830	0.784	0.661	1.630
PhymmBL	NA	0.870	0.783	0.732	0.584	1.399
MetaPhlAn	1.206	0.834	0.761	0.608	0.478	0.877
MetaPhyler	4.159	1.624	1.194	1.109	1.249	1.777
TIPP(0%,0%,100)	1.000	1.000	1.000	1.000	1.000	1.000
TIPP(95%,95%,100)	1.025	1.002	1.005	0.998	0.988	0.983
FAMeS LC						
NBC	0.974	0.943	1.006	0.976	0.658	1.127
PhymmBL	NA	1.016	0.527	0.528	0.334	0.713
MetaPhlAn	3.849	2.714	1.813	1.868	1.349	1.683
MetaPhyler	6.489	2.197	1.442	1.221	1.175	1.331
TIPP(0%,0%,100)	1.000	1.000	1.000	1.000	1.000	1.000
TIPP(95%,95%,100)	1.195	1.046	1.036	1.008	1.015	0.997
FAMeS MC						
NBC	1.844	1.829	1.651	1.679	1.363	1.863
PhymmBL	NA	1.645	1.342	1.363	1.069	1.461
MetaPhlAn	2.332	1.521	0.690	0.859	0.647	2.463
MetaPhyler	4.997	1.741	1.291	1.260	1.019	2.366
TIPP(0%,0%,100)	1.000	1.000	1.000	1.000	1.000	1.000
TIPP(95%,95%,100)	1.014	1.015	1.049	1.037	1.038	0.987
WebCarma						
NBC	0.771	0.795	0.862	0.818	1.141	2.081
PhymmBL	NA	0.788	0.807	0.769	0.839	1.013
MetaPhlAn	1.384	1.153	1.127	1.207	1.665	1.003
MetaPhyler	3.205	1.467	1.318	1.186	1.567	1.260
TIPP(0%,0%,100)	1.000	1.000	1.000	1.000	1.000	1.000
TIPP(95%,95%,100)	0.932	0.894	1.039	1.038	1.071	0.957
Average						
NBC	1.161	1.250	1.264	1.236	1.059	1.888
PhymmBL	NA	1.194	1.075	1.045	0.823	1.373
MetaPhlAn	1.802	1.372	1.202	1.168	0.986	1.463
MetaPhyler	4.582	1.779	1.343	1.228	1.239	1.520
TIPP(0%,0%,100)	1.000	1.000	1.000	1.000	1.000	1.000
TIPP(95%,95%,100)	1.010	0.973	1.020	1.013	1.029	1.012



(a) Illumina error model



(b) 454 error model

Figure A21: Non-leave-one-out experiments comparing the classification accuracy for NBC, PhymmBL, MetaPhyler and TIPP-default (i.e., TIPP-default refers to TIPP(95%,95%,100)) for fragments simulated from the 30 marker genes under different rates of Illumina-like and 454-like errors. Note that PhymmBL does not classify below the genus level and thus has 100% unclassified rate at the species level.

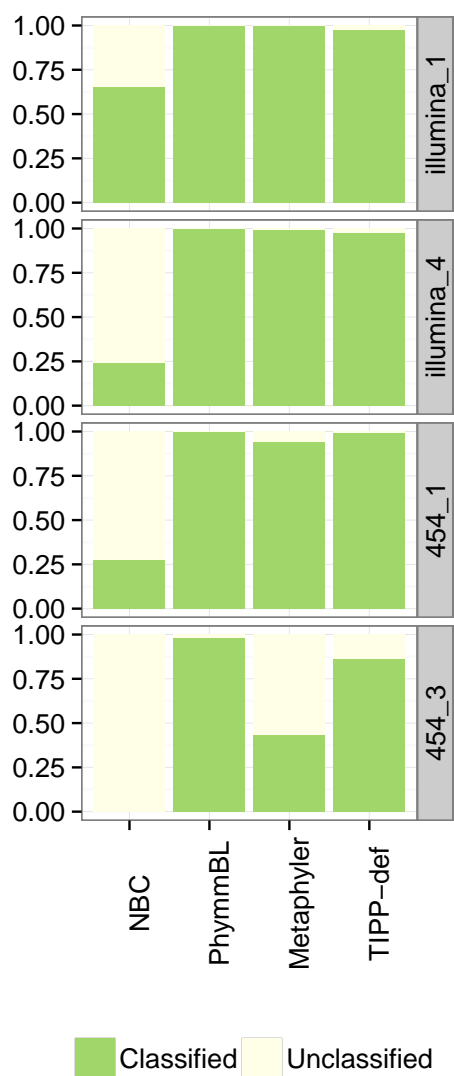


Figure A22: Non-leave-one-out experiments comparing the proportion of classified and unclassified reads at the phylum level for NBC, PhymmBL, MetaPhyer and TIPP-default (i.e., TIPP-default refers to TIPP(95%,95%,100)) for fragments simulated from the 30 marker genes under different error models. If reads that could not be classified at the phylum level are considered novel, then the unclassified rate identical to the false positive rate for detecting “dark matter” microbes.

A4 Non-leave-one-out Running Time Study.

In this section we report on running time experiments performed on the *rpsB* marker gene in order to examine the impact of the maximum alignment decomposition size on the running time. While the previous leave-one-out and non-leave-one out experiments had over million fragments, each individual TIPP run typically examined fewer than 5,000 fragments. Thus, computing the total running times across all the experiments would incur a substantial cost in setup time.

To obtain a better estimate of the running time, we ran TIPP on a very large simulated dataset. We simulated 200,000 fragments from *rpsB* with Illumina-like errors. We selected the *rpsB* gene because the number of sequences in its reference alignment is on the high end of the range (1463), so that its running time will be also at the high end of most analyses. We ran TIPP(95%,95%, X) on the fragments, with X ranging from 10 to the total number of sequences in *rpsB*, and we used pplacer within TIPP to place the fragments. Note that we could not run EPA inside of TIPP for these experiments, as we found EPA to be significantly slower than pplacer. For example, the time to place 200 fragments using pplacer within TIPP was roughly half a minute, while EPA took 55 minutes.

Each TIPP run was computed on an individual computer node with 32 GB of memory and was given 4 CPUs. We report the elapsed wall clock time in Table A23.

Table A23: Wall clock running time (in hours) to classify 200,000 fragments for different maximum alignment decomposition sizes, using four processors. The fragments were simulated using MetaSim with Illumina_1 errors from the rpsB marker gene.

Method	Wall clock time (hr)
TIPP(95%,10)	7.5
TIPP(95%,100)	3.6
TIPP(95%,500)	2.8
TIPP(95%,ALL)	2.1

A5 Abundance profile calculation

In Chapter 5, we briefly described how to compute an abundance profile of a method. We now provide more details on this procedure. Almost all the studied methods (lone exception of NBC) can leave a fragment partially classified. Abundance profiles at a specific level for a given method is computed by removing all unclassified fragments at the specific level and then computing the abundance profile on the remaining fragments. For example, if the abundance profile of a method at the species level is 30% species A, 30% species B, and 60% unclassified, the modified abundance profile would be 50% species A and 50% species B. Note that for the marker-based methods, the source gene of the fragment is ignored when computing abundance profiles, i.e., the profiles are computing on the entire set of classified fragments, ignoring that the fragments may be binned to different markers.

A6 Dataset

A6.1 Marker Genes and Empirical Statistics

Table A24 shows statistics for the marker genes used in this study. We show the maximum and average p-distances for each gene, which are defined as follows. We compute the SATé alignment on each gene, and we define the p-distance between two aligned sequences for a gene to be the fraction of the positions in which they both have nucleotides, but the nucleotides are different. The maximum of these pairwise distances is the “max p-distance”, and the average of these pairwise distances is “average p-distance”. Datasets that have maximum p-distances at 0.75 or larger are said to be “saturated”, and estimating alignments and trees on such datasets is very difficult.

A6.2 Fragments

We used MetaSim [70] to generate fragments, starting from the reference datasets of 30 marker genes and the 16S gene. Both 100-bp Illumina-type fragments and 300-bp 454-type fragments were generated, with different levels of error (thus, we have Illumina_1, Illumina_2, and Illumina_4 models for Illumina-type error, and similarly 454_1, 454_2, and 454_3 models). The index j in Illumina_ j error model is scaling factor for the substitution error rates; an index of 2 means all the substitution error rates per site are doubled. The index j in the 454_ j error model is the scaling factor for the negative flow error rate; an index of 2 means that insertions are twice more likely. Illumina-type fragments contained only substitution errors, and 454-type fragments contained

Table A24: Statistics for 30 marker genes and 16S RNA gene.

Marker	Number of sequences	Max p-distance	Average p-distance
16S_archaea	375	0.35	0.22
16S_bacteria	9197	0.36	0.20
dnaG	1555	1.00	0.59
frr	1313	0.67	0.47
infC	1338	0.73	0.46
nusA	1406	1.00	0.56
pgk	1544	0.81	0.49
pyrG	1501	1.00	0.43
pyrg	65	0.55	0.43
rplA	1396	0.70	0.45
rplB	1370	0.68	0.43
rplC	1400	0.73	0.48
rplD	1341	0.75	0.53
rplE	1399	0.72	0.42
rplF	1366	0.78	0.48
rplK	1421	1.00	0.41
rplL	1315	0.77	0.42
rplM	1365	0.71	0.45
rplN	1311	0.66	0.40
rplP	1351	0.75	0.43
rplS	1383	0.81	0.46
rplT	1279	0.73	0.44
rpmA	1223	0.63	0.42
rpsB	1463	0.74	0.45
rpsC	1293	1.00	0.46
rpsE	1316	0.69	0.45
rpsI	1174	0.74	0.47
rpsJ	1287	0.68	0.41
rpsK	1308	0.64	0.42
rpsM	1307	0.70	0.43
rpsS	1277	0.67	0.40
smpB	1278	0.71	0.49

only indel errors, biased toward insertions. The Illumina_1 and 454_1 models have the lowest error rates, and the Illumina_4 and 454_3 models have the highest error rates. Table A25 shows the amount of error simulated for each model condition.

Table A25: Statistics for non-leave-one-out fragment datasets. Fragments were simulated using MetaSim with Illumina-like error or with 454-like errors. Illumina-like fragments suffered from single bp substitution errors, while 454-like fragments suffered from indel errors, biased toward insertions.

Model name	Num. substitutions per fragment (avg)	Avg. length	Model name	Num. indel events per fragment (avg)	Avg. length
Illumina_1	0.5	100	454_1	14.2	272.5
Illumina_2	1.5	100	454_2	24.5	275.9
Illumina_4	3.8	100	454_3	60.2	284.9

A7 Abundance Profile Datasets

A7.1 Metaphlan Simulated Dataset

The Metaphlan simulated datasets [74] consist of 2 high complexity datasets and 8 low complexity datasets. The high complexity datasets contain 1,000,000 fragments each, and the low complexity datasets contain 250,000 fragments each. The average read length of the datasets was 88 bps. The fragments span the bacteria and archaea domains.

A7.2 FACs HC

The FACs simulated dataset [83] is a high complexity dataset containing 19 bacterial genomes, 3 viral genomes, and 2 human chromosomes. All genomes are present in equal amounts. MetaSim was used to simulate 454-like fragments from the genomes. The dataset contained 100,000 sequences total with an average length of 269 bps. The fragments in the original dataset span the bacterial, viral, and eukaryote domains. We used only the bacterial fragments from this dataset.

We also generated a high complexity dataset with 300,000 Illumina-like reads using MetaSim from this dataset. The fragments had an average length of 100 bps.

A7.3 FAMeS

The FAMeS simulated datasets [53] consist of a low complexity, medium complexity, and high complexity dataset. Fragments were obtain from isolate

genome sequencing projects at the Department of Energy Joint Genome Institute (DOE-JGI). Simulated abundance profiles were created by adding the desired proportion of fragments to achieve a desired profile. Thus, while the sequences are from a real study, the abundance profile itself is simulated and the true abundance is known. The datasets consist of 328,728 sequences total with an average length of 950 bps. The fragments span the bacterial and archaeal domains of life.

A7.4 WebCarma dataset

The WebCarma dataset simulated dataset [24] is a high complexity dataset containing 25 bacteria genomes. MetaSim was used to simulate 454-like fragments from the genomes. The dataset contained 25,000 fragments with an average length of 265 bps.

We also generated a high complexity dataset with 300,000 Illumina-like reads using MetaSim from this dataset. The fragments had an average length of 100 bps.

A8 Methods

A8.1 EPA and pplacer: Likelihood-based phylogenetic placement

EPA and pplacer are both tools for phylogenetic placement based on maximum likelihood, although pplacer can also use a Bayesian approach. Both of these tools evaluate the likelihood of placing the fragment on different reference tree edges, and optimize the length of the pendant edge and the position

of the pendant edge on the reference tree edge. They both return a set of near optimal placements along with the likelihood score that could be achieved with that placement. Thus, for a given extended alignment, a query sequence can be placed on multiple edges of the backbone tree, each with varying levels of confidence, and a comparison of the likelihoods of alternative placements can be used as a measure of placement uncertainty. Both pplacer and EPA uses heuristics to limit expensive likelihood calculations to the parts of the tree they consider to be most likely to contain optimal solution, but the exact heuristics used are very different between the two methods. EPA has the ability to use more models of sequence evolution than pplacer does.

We consulted Alexis Stamatakis, the author of EPA, regarding the differences between the two tools; his response is given below:

Date: Sun, 10 Mar 2013 17:21:53 -0700 From: Alexandros Stamatakis <jalexandros.stamatakis@gmail.com> To: Tandy Warnow <tandy@cs.utexas.edu> Subject: Re: difference between EPA and pplacer

Hi Tandy,

There is indeed no essential difference, you can quote this as pers. comm. with me.

Alexis

A8.2 Commands Used

Reference alignment: Each of the 30 marker genes were aligned using SATé-II [47] version 2.0.3. SATé is run using a configuration file (available upon request). The configuration options, listed below, indicate that we used SATé in its default mode.

Centroid edge decomposition, Maximum alignment size of 20% of the number of sequences, MAFFT to align, Muscle to merge alignments, FastTree to estimate trees, simple stopping rule.

Reference tree and refined taxonomy: Two sets of trees are used in all studies of marker genes: the reference tree (which is the SATé tree, and thus the RAxML tree on the reference alignment) and the refined taxonomy. Note that for the 16S analyses, the SATé alignment is used for the reference alignment, but the refined taxonomy also serves as the reference tree.

For refining a taxonomy the following RAxML command is used:

```
raxmlHPC -g [taxonomy] -s [SATé_alignment] -m GTRGAMMA -n  
[name]
```

Fragment simulation: MetaSim, by default, can generate fragments with 454-like errors. The command used to generate the 454 reads are shown below.

```
MetaSim cmd -4 -r<number_fragments> -f 300 -t 0
--454-multiplier 0.30 --454-logn-mean < 0.23 * j >
<sequence_names>
```

where j is the error model scaling factor. For the leave-out experiments on the 30 marker genes and 16S marker genes, j is 1; for the remaining experiments, j ranges from 1 to 3.

MetaSim does not have a default setting to simulate Illumina-like fragments. An empirical error model was downloaded from <http://ab.inf.uni-tuebingen.de/software/metasim/errormodel-80bp.mconf/>. The error model generates fragments of 80 bps. To generate 100 bp fragments, the error model for the last bp was repeated 20 extra times. For the higher error fragments, the error rates at each position was multiplied the by error factor (1, 2, or 4). The error model files used to generate the fragments are available upon request.

pplacer: pplacer v1.1.alpha13 was run with the following command.

```
pplacer --out-dir [output_directory] -j 1
-r [reference_alignment] -s [raxml_info_file] -t
[reference_placement] [extended_alignment]
```

EPA: EPA was run using version 7.4.2 of RAxML, and using `-f v` option.

```
raxmlHPC -f v -t [placement_tree] -s [extended_alignment]
-m GTRGAMMA -n [name]
```


HMMER: The following commands were used for building and aligning using HMMER.

```
hmmbuild --symfrac 0.0 --dna --informat afa [outputname]
[input_alignment]
```

```
hmmsearch --allcol --dna -o [outputname] [input_model]
[fragment_file]
```

Blast Binning: Fragments were binned to marker genes by blasting the fragments against the 30 marker reference dataset. The fragments were binned to the source gene of the sequence that gave the best match. The following commands were used to bin the fragments to marker genes.

```
blastn -db [blast_database] -outfmt 6 -query
[fragment_file] -out [outputname] -max_target_seqs 1
```

TIPP: TIPP can be run through a configuration file or through the command line. To run default TIPP, as described in Chapter 5, the following command can be used.

```
python run_tipp.py -t [taxonomic_tree] -a
[backbone_alignment] -r [raxml_info_file] -at 95 -pt 95 -tx
[taxonomy_file] -txm [taxonomy_mapping_file] -A 100 -adt
[ml_tree] -f [fragment_file] -o [outputname]
```

MetaPhyler: MetaPhyler version 1.25 was run using the following command.

```
perl runMetaPhyler.pl [fragment_file] blastn [outputname]
```

PhymmBL: PhymmBL version 4.0 was trained and run using the following commands.

```
perl phymmblSetup.pl
```

```
perl scoreReads.pl [fragment_file]
```

NBC: NBC version 1.1 was trained and run using the following commands. Fragments were classified at the best hit genome if the confidence score of the hit was above the species threshold.

```
countncbi [nbc_genome_directory] 15
```

```
score -a [fragment_file] -r 15 -j [nbc_genome_directory]
```

```
Species_threshold = -23.7* Read_length+490
```

MetaPhlAn: MetaPhlAn version 1.7.3 was run using the following command.

```
python metaphlan.py --blastdb [metaphlan_blast_database]  
[fragment_file] [outputname]
```

Appendix B

UPP

B1 Materials and Methods

Datasets used in this study are available at <http://www.cs.utexas.edu/~phylo/datasets/alignment.html>. (All software will be made freely available in open-source form upon acceptance.)

B1.1 Datasets

B1.1.1 CRW 16S biological datasets.

We used three ribosomal RNA datasets (6,323 to 27,643 taxa) from the Comparative Ribosomal Website [11]; these datasets have highly reliable, curated alignments based upon secondary and higher-order structures. The alignments were cleaned by removing any sequence that contained more than 50% gap characters, and then removing any site that consisted of all gapped characters. Reference trees containing only highly supported edges (contracting edges with less than 75% bootstrap support) were generated for these alignments by previous studies [46, 47]. The alignments and trees can be downloaded from <http://www.cs.utexas.edu/~phylo/datasets/phylogeny-topology.html>.

B1.1.2 FastTree COG simulated datasets.

We include seven simulated protein COG datasets with 5,000 sequences from [65]. The datasets were generated by aligning the gene families from the Clusters of Orthologous Groups (COG) database [88], and for each alignment, a random subalignment of 5000 sequences were sampled. The subalignment was cleaned (sites containing more than 25% gapped characters were removed from the subalignment) and an ML tree was estimated on the cleaned alignment using FastTree under JTT. The ML trees were then used as input to ROSE [82] for sequence simulation.

B1.1.3 Large AA datasets with full reference alignments.

We include ten large biological protein datasets with 353 to 807 sequences with curated reference alignments. We include eight datasets from the BALiBASE database (BALiBASE datasets RV100_BBA0039, 0067, 0081, 0101, 0117, 0134, 0154, and 0190) from [89] and two datasets (1GADBL_100 and coli_epi_100) from [25]). RAxML bootstrapping was performed on the curated alignments to obtain ML trees with branch support, and branches with less than 75% support were contracted and used as the reference tree for the datasets.

The model of amino acid evolution used to generate the reference trees was selected using RAxML-Light [79] version 1.0.5). The command used to find the best amino acid model is given below.

- RAxML-LIGHT: *raxmlLight-v1.0.5 -s [reference alignment] -T2 -m PROTCATAUTOF -n [name] -t [raxml parsimony tree]*

The following models were selected:

- RV100_BBA0067: VT
- RV100_BBA0081: VT
- RV100_BBA0101: WAG
- RV100_BBA0117: LG
- RV100_BBA0134: JTT
- RV100_BBA0154: WAG
- RV100_BBA0190: VT
- 1GAD.BL_100: LG
- coli_epi_100: LG
- RV100_BBA0039: LG

B1.1.4 HomFam datasets.

These are biological datasets that were assembled to evaluate protein MSA methods on large datasets in [75]; we use 19 of the 20 largest HomFam datasets (10,099 to 93,681 taxa). (We omit the “rhv” dataset due to the

warning on the Pfam website that the alignment of these sequences was very weak.)

The HomFam datasets were generated using the HomStrad [81] and Pfam [67] databases, as follows. Curated seed alignments on 5-20 sequences (median 7) from each protein family were downloaded from the HomStrad database, and for each protein family, homologous protein sequences from the Pfam database were added to the HomStrad seed sequences to produce each HomFam datasets.

The HomStrad seed alignments were used as the reference alignment; therefore estimated alignments were evaluated only with respect to the induced alignment produced on the seed sequences. See Table B3 for the number of seed sequences found in each dataset.

B1.1.5 1000-taxon simulated datasets.

We included 300 simulated 1000-taxon NT datasets that were used in [46, 47] to evaluate MSA methods on large nucleotide datasets. The average sequence length is 1000 under all model conditions. These were produced using ROSE under 15 different model conditions (20 replicates per model condition), varying the rates of substitutions, indels, and gap length distributions. The model conditions range in terms of difficulty (largely due to rate of evolution and relative frequency of indels versus substitutions). Thus, model conditions ending with “1” are the hardest, model conditions ending with “5” are the easiest, and model conditions ending in “2” or “3” are still somewhat difficult.

The letter (M, L, or S) refers to the gap length distribution (medium, long, or short). See [46] for further details on sequence generation.

B1.1.6 Indelible simulated datasets.

We used Indelible [22] version 1.03 to generate 30 NT datasets under 3 different model conditions (10 replicates per model condition) that had similar empirical statistics (percent gapped, average p-distance, and max p-distance) as the 1000-taxon 1000M2, 1000M3, and 1000M4 model conditions. We label these model conditions as 10000M2, 10000M3, and 10000M4. The average sequence length is 1000 under all model conditions.

B1.2 Methods

B1.2.1 Basic alignment methods

Each dataset was aligned (when possible) using Opal [93] version 2.1.2, Clustal-Omega [75] version 1.2.0, MAFFT[35–37] version 6.956b, MUSCLE [16] version 3.8.31, SATé-II [47] version 2.2.7 and PASTA version 1.0 [56]. Due to a bug in earlier versions of MAFFT 6.956b, MAFFT-Profile and MAFFT-default were run using MAFFT version 7.143.

We ran three different versions of MAFFT. MAFFT-L-INSI was run on datasets with 1,000 for fewer sequences. For datasets with more than 1,000 sequences, we ran MAFFT-default (“--auto”) and MAFFT-PartTree (using PartTree algorithm). All MAFFT variants included the “--ep 0.123” parameter.

MUSCLE was run with the “-maxiters 2” option on datasets of 3000 sequences or greater. UPP was run using a configuration file, available upon request (these will be made public upon acceptance.)

We ran two different versions of MAFFT-Profile: MAFFT-Profile-addfrag (“- -addfragments”) and MAFFT-Profile-add (“- -add”). On datasets with less than 1,000 sequences, the “L-INS-i” flag was also set.

SEPP version 1.0 [55] was simulated through UPP by excluding the “-S hierarchical” flag. Excluding this flag resulted in non-overlapping, approximately equally-sized alignment subsets instead of the nested hierarchical alignment subsets used within UPP.

PASTA was run using a MAFFT-PartTree starting tree for all but the RNASim datasets. For the RNASim datasets, we used the ML tree estimated on the UPP(Fast, No Decomp) alignment as the starting tree (MAFFT-PartTree was unable to run on the largest RNASim datasets). The remaining settings for PASTA are set using the “--auto” flag. PASTA is always run for 3 iterations or 24 hour time limit, whichever one came first. Commands for each method is given below.

- **Clustal-Omega:** *clustalo -i<input_sequence> -o <output_alignment>*
- **MAFFT-L-INS-i:** *mafft --ep 0.123 --localpair --maxiterate 1000 --quiet --anysymbol <input_sequence> > <output_alignment>*
- **MAFFT-default:** *mafft --ep 0.123 --auto --quiet --anysymbol <input_sequence> > <output_alignment>*

- **MAFFT-PartTree:** `mafft --ep 0.123 --parttree --retree 2 --partsize 1000 --quiet <input_sequence> > <output_alignment>`
- **MAFFT-profile:** `mafft [--localpair --maxiterate 1000] [- -addfragment | - -add] <query_file> <backbone_alignment> > <output_alignment>`
- **Opal:** `java -Xmx20g -jar opal.jar --in <input_sequences> --out <output_alignment>`
- **MUSCLE:** `muscle [-maxiters 2] -in <input_sequence> -out <output_alignment>`
- **PASTA:** `python run_pasta.py -o <output_directory> -i <input_sequences> -t <starting_tree> --auto --datatype=<molecule_type>`
- **UPP:** `python exhaustive_upp.py -S hierarchical -a <backbone_alignment> -t <backbone_tree> -s <query_sequences> -d <output_directory> -o <output_name> -x 12 -A <max_alignment_subsetSize> -m <molecule_type> -c <default_config_file>`
- **SEPP:** `python exhaustive_upp.py -a <backbone_alignment> -t <backbone_tree> -s <query_sequences> -d <output_directory> -o <output_name> -x 12 -A <max_alignment_subsetSize> -m <molecule_type> -c <default_config_file>`

B1.2.2 HMMER Commands

HMMER 3.0 [15] is used internally within UPP for building the HMM family, for searching for the best HMM for the alignment of a query sequence, and for inserting the query sequence into the alignment. We provide the HMMER commands used internally within UPP.

- **HMMBUILD:**

```
hmmbuild --symfrac 0.0 --informat afa --<molecule_type>  
<output_profile> <backbone_alignment>
```

- **HMMSEARCH:**

```
hmmsearch --noali -o <output_file> --cpu 1 -E 99999999 --max  
<input_profile> <query_file>
```

- **HMMALIGN:**

```
hmmalign --allcol --dna <output_profile> <query_file>  
<output_alignment>
```

B1.2.3 Maximum Likelihood Tree Estimation

We estimated Maximum Likelihood (ML) trees using FastTree [65] version 2.1.5 SSE3 and RAxML version 8.0.6, using the commands below. We ran each method in default mode for the nucleotide (NT) datasets, under the GTR substitution model.

Analyses of the amino acid (AA) datasets were performed differently by the two methods. For FastTree, we used the default setting which sets the

substitution model to JTT. For RAxML analyses of the simulated datasets, we also used JTT, with the GAMMA model for rates of evolution across sites. For the RAxML analyses of the biological datasets, we estimated the substitution models for each biological dataset, and then used that model with the GAMMA model for rates of evolution across sites.

- **FastTree AA:**

```
fasttree <input_fasta> > <output_tree>
```

- **FastTree NT:**

```
fasttree -nt -gtr <input_fasta> > <output_tree>
```

- **RAxML AA:**

```
raxmlHPC -T 12 -m PROTGAMMAJTT -j -n <output_name>  
<starting_tree> -s <input_fasta> > -w <output_directory> -p 1
```

- **RAxML NT:**

```
raxmlHPC -T 12 -m GTRGAMMA -j -n <output_name>  
<starting_tree> -s <input_fasta> > -w <output_directory> -p 1
```

B1.2.4 UPP alignment method

UPP is a novel *de novo* alignment technique that uses HMM Families to efficiently align of both large and fragmentary datasets. The input to UPP is a set of unaligned sequences. The output of UPP is the “unmasked” and “masked” alignments on the entire set of sequences. The “unmasked”

alignment preserves insertion columns generated during the alignment step. All characters in an insertion column are non-homologous and should not be used during tree inference. Thus, we also output the “masked” alignment in which insertion columns are removed for ML estimation. UPP proceeds in the following steps:

- partitioning the sequences into the backbone set and query set,
- estimating the backbone alignment and tree,
- decomposing the backbone sequence set into subsets, and computing the families of HMMs,
- aligning query sequences to HMMs, and merging the subalignments into the final “unmasked” and “masked” alignments.

We now provide details for each step below. Commands for building the HMM models, searching against the HMM models, and aligning the query sequences to each HMM, are provided in Section B1.2.2.

Partitioning the sequences. For datasets containing only full-length sequences and little sequence length heterogeneity, we randomly divide the sequences into the backbone set and the query set. For datasets with large sequence length heterogeneity, we want only full-length sequences to be in the backbone. Thus, we filter out sequences that might be fragmentary (too short)

or chimeric (too long), either by using a user-defined length threshold or by removing any sequence that is less than 25% shorter or longer than the median length of the the typical gene length. The backbone sequences are selected from the remaining unfiltered sequences. Details on backbone selection on the individual datasets can found in Section B1.2.4. All sequences that are not selected for the backbone set are placed into the query set.

Generating the backbone alignment and tree. We run PASTA on the backbone set using the commands give in Section B1.2.1; this produces the backbone alignment and backbone tree.

Decomposing the backbone tree into HMMs. We apply the recursive decomposition technique called the “centroid edge decomposition” [47] to build the family of HMMs. From the backbone tree, we select the centroid edge e (one whose removal separates the leaf set into approximately two equally sized subsets). We remove e (but not its endpoints) from the backbone tree to produce two subtrees. This process is recursively repeated on each subtree with more than ten leaves. The leaf set of each subtree generated by this process, including the initial starting tree, is added to a set \mathcal{C} of subsets of the backbone set. The backbone alignment restricted to each of these sets is called a “subset alignment”. For each alignment subset, we run HMMBUILD to produce an HMM, producing a match state for every site that is not completely gapped.

Aligning the query sequences. Each query sequence is scored against each HMM using HMMSEARCH, which reports a HMMER “bit score”, a measure of the quality of the match between the query sequence and the HMM. The HMM that yields the best bit score is selected and the query sequence is inserted into the subalignment using HMMALIGN. Once all the query sequences have been inserted into the subalignments, we merge all the subalignments back together using transitivity.

In the special case where a query sequence resulted in no scores against any of the HMMs (i.e., HMMSEARCH reports the sequence as non-homologous to all HMMs), the query sequence is omitted from the final alignment.

Backbone filtering In order to determine which sequences are acceptable to be in the backbone set, we require that backbone sequences must be within a certain length range. The range is defined as within 25% length of the typical gene sequence. In the case where the average gene length is not known, we use the median length the reference sequences as an approximation of the average gene length. In general, all but the 16S CRW datasets used the median length of the full-length reference sequences as an approximation of the average gene length. For example, on the HomFam dataset zf-CCHH where the median length of the seed sequences is 23, we only allow sequences between 23 residues \pm 6 residues to be selected into the backbone set. For the CRW datasets, we only allow sequences between 1500 bps \pm 375 bps (typical length of 16S gene

is approximately 1500 bps) to be selected into the backbone set.

For the fragmentary datasets, we applied the same protocol to select the backbone sequences using the length of the unmodified full-length sequences to define length range. Note that because the RNASim datasets and the 1000-taxon datasets had very homogenous length distributions, the simulated fragmentary sequences were perfectly partitioned from the full-length sequences, and the backbone was sampled from all the entire set of full-length sequences. The CRW datasets, on the other hand, had more sequence length heterogeneity, so many full-length (defined as an unmodified sequence) sequences were also filtered from backbone selection, and thus never had a chance to be sampled as a backbone sequence.

Using UPP iteratively. UPP can be used iteratively. In the first iteration, the UPP alignment is computed, and a tree is estimated on the alignment. The next iteration uses the tree to determine if a second iteration is recommended, and then (if needed) to do directed sampling for the backbone sequences using the tree. The major motivation for this iterative approach is highly uneven taxon sampling (e.g., a densely sampled in-group and very sparsely sampled outgroup). In this case the inclusion of the outgroup sequences into the backbone would suffice to provide the rebalancing that is needed. However, directed sampling is also helpful when fragmentary sequences are unevenly distributed throughout the phylogeny, or sequence lengths have changed substantially over evolutionary history; in these cases, the approach we use for selecting the back-

bone dataset (which only samples from the full-length sequences) could have substantially reduced phylogenetic diversity compared to the input dataset, so that some major clades may have few or no backbone sequences. When this happens, although the alignment estimated on the backbone dataset may be highly accurate, the ability of the HMM Family to align some of the query sequences may be reduced for those query sequences located in under-sampled major clades. This issue arose for the 16S.T dataset, as shown in Figure B1. There was a major clade that was primarily composed of shorter sequences, so that the backbone dataset contained very few sequences from that clade.

We developed a resampling algorithm to handle this problem that can be used after an alignment and tree is computed for the dataset. The resampling algorithm searches for clades that are undersampled, and if found, selects a new backbone set with more uniform sampling across the estimated tree.

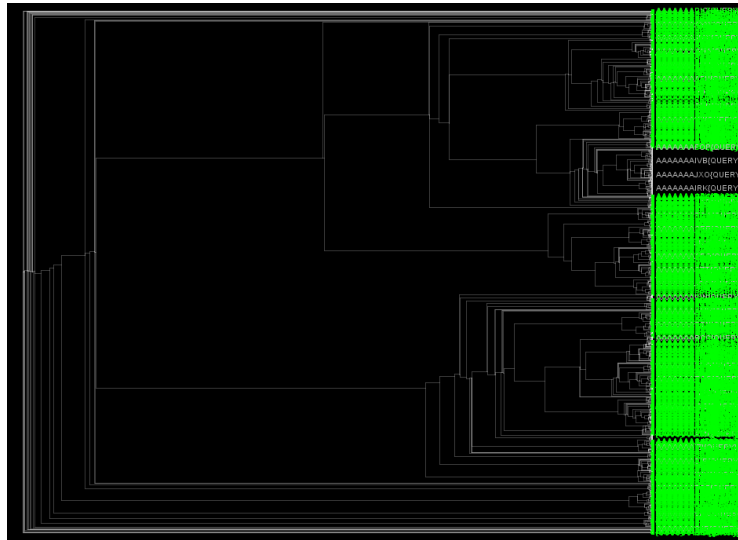
The first step is to detect undersampled clades, where the threshold t for under-sampling is a variable that the user can set; we explored this approach using threshold $t = 0.1$. UPP starts by using the estimated tree on the initial UPP alignment and rooting it arbitrarily. Next UPP performs a post order traversal of the internal nodes, and for each internal node v , it computes the proportion $p(v)$ of the leaves in the clade for v that are in the backbone. If this proportion is less than tB (where B is the desired proportion, set to be the total number of sequences in the backbone dataset divided by the number of leaves in the tree) and the clade is large enough (so that the expected number of leaves in the backbone would be at least 10, if the backbone was distributed

uniformly across the tree), then UPP considers the clade under-sampled. If any clade is found that is under-sampled, this triggers an additional iteration of UPP.

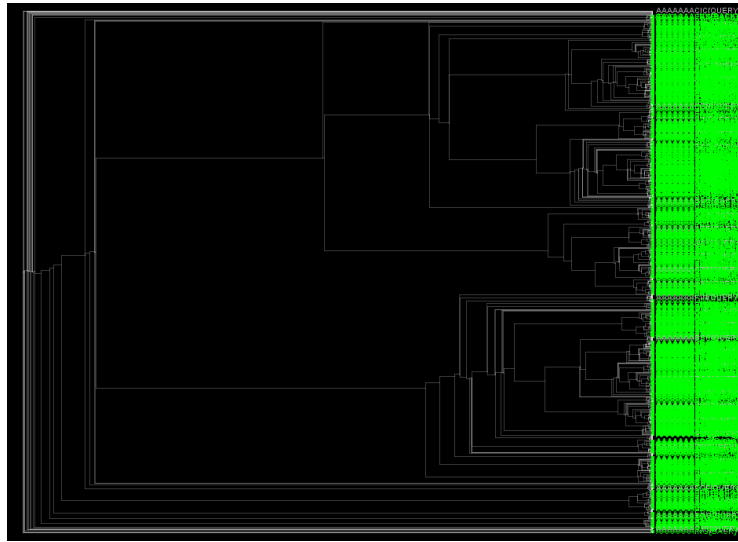
UPP uses directed sampling in the next iteration, performed as follows. UPP performs a post-order traversal of the internal nodes, and for each internal node v such that at least 10 sequences should have been put in the backbone set from the clade for v , it selects the backbone set from the clade at v , with the correct proportion (defined by the total number of backbone sequences desired, and the total number of leaves in the tree).

The selection of the backbone sequences begins by selecting randomly from the full-length sequences in the clade and then completes the set by sampling from the remaining sequences. However, the decision of what constitutes full-length is based on the sequence length distribution within the clade.

Then the entire clade at v is deleted from the tree. This process repeats until all the original leaves in the tree have been deleted. The end result of this process is a backbone set that has uniform sampling across the estimated tree.



(a) Initial backbone sequences



(b) Resampled backbone sequences

Figure B1: **Distribution of the backbone sequences in the estimated ML tree on 16S.T.** We show the distribution of the backbone sequences (highlighted in green) in the ML tree estimated on the UPP(Default) alignment. The initial selection of the backbone sequences resulted in sparse sampling throughout one clade. After applying the resampling technique, the distribution of the backbone sequences show much more uniform coverage.

B1.3 Early termination on large datasets

Many alignment methods failed to complete analyses on the larger datasets, but reasons varied. Some failed due to insufficient memory, or due to a bug in the software, or were simply unable to produce an alignment within the 24 hour time limit (i.e., they might have been able to produce an alignment if given more time). This section documents each case.

MAFFT-default. MAFFT-default terminated early on the CRW 16S.B.ALL dataset due to the following error message:

```
Cannot allocate 12568 character vector.
```

MAFFT-default also failed to produce an alignment on the RNASim 100K dataset within the 24 hour time limit on TACC. According MAFFT's output log, MAFFT was still running when the job was evicted.

MAFFT-PartTree. MAFFT-PartTree terminated with the following error message on the RNASim 200K dataset:

```
mafft: line 2028: 28963 Segmentation fault
"$prefix/splittbfast" $legacygapopt -Z
$algopt $splitopt $partorderopt $parttreeoutopt
$memopt $seqtype $model -f "$gop -Q
$spfactor -h $aof -p $partsize -s
$groupsize $treealg -i infile > pre 2>> "$progressfile"
```

MAFFT-addfragments. MAFFT “addfragments” terminated early on the RNASim datasets with 50,000 sequences or more, the CRW 16S.B.ALL dataset, and the HomFam aat, p450, rvp, and zfCCHH datasets with the following error message (note that the line number and location of the segmentation fault was different for each dataset):

```
mafft: line XXXX: YYYYY Segmentation fault
"$prefix/pairlocalalign" $localparam $addarg
-C $numthreads $seqtype $model -g~$lexp
-f~$lgop -Q~$spfactor -h $laof -Y
< infile > /dev/null 2>> "$progressfile"
```

MAFFT-add. MAFFT “add” terminated early on the Indelible 10000M2 dataset with the following error message:

```
bug! hairitsu ga kowareta!
```

On the RNASim 200K dataset, MAFFT “add” terminated with non-zero status, which signifies an error during execution.

Opal. Opal failed to align the FastTree COG438 dataset, most likely due to a memory problem the machine. The Opal log file shows that Opal terminated early during execution and that the peak virtual memory usage at the time was 31 GB (only 24 GB of RAM available on Lonestar machine).

MUSCLE. MUSCLE terminates early on the RNASim datasets with 50K sequences or more with the following error message:

```
*** OUT OF MEMORY ***  
Memory allocated so far 23718.4 MB  
No alignment generated
```

On the HomFam zf-CCHH and rvp, MUSCLE terminated with the following error message: *Segmentation fault*.

Clustal-Omega. Clustal-Omega failed to terminate within 24 hours on the RNASim datasets with 50,000 sequences or more. The log file showed that Clustal-Omega was still running, so given enough time, it may be possible for Clustal-Omega to produce an alignment on the larger RNASim datasets.

On the Indelible 10000M2 dataset, Clustal-Omega terminated early with the following error message:

```
HHalignWrapper:hhalign_wrapper.c:945: problem in  
alignment (profile sizes: 892 + 1540) (S1870 + S7661),  
forcing Viterbi  
      hh-error-code=3 (mac-ram=2048)  
+-----+  
| both sequences truncated right |  
+-----+
```

```
i2 = 2 != 6699 = qa->L, j2 = 10788 != 10846 = ta->L
PrintAlignments:hhitlist-C.h:199: qt_ali.Build failed
halign:halign.cpp:1216: Could not print alignments
HAlignWrapper:halign_wrapper.c:984: 2nd attempt
worked HAlignWrapper:halign_wrapper.c:945:
problem in alignment
(profile sizes: 833 + 2432) (S3589 + S1870), forcing Viterbi
      hh-error-code=3 (mac-ram=2048)
```

B2 Supplemental Figures and Tables

Table B1: **Results on the million-sequence RNASim dataset.** We show results for UPP with a backbone of 100 sequences and using HMM families (labeled “UPP(F)”) and using only a single HMM (labeled “UPP(F,ND)”). We also show results for UPP with a backbone of 1000 sequences but without any decomposition (labeled “UPP(D,ND)”). For this dataset, ΔFN is computed by comparing the difference in tree error between the estimated ML tree and the ML tree estimated on the masked true alignment (labeled “TA”), where any site with fewer than 1,000 ungapped characters was removed; this masking was performed to reduce the size of the true alignment (21,946 before masking and 3,887 sites after masking) and make the ML tree estimation computationally feasible. The tree error reported for the true alignment is also based on the masked true alignment. The UPP alignments were computed on a dedicated machine with 250 GB of memory and 12 CPUs; this is a different computational platform than TACC’s Lonestar Cluster machines, which we used to run the 10K to 200K RNASim experiments, and so the running times cannot be compared to results on the smaller RNASim datasets.

Dataset	Method	Align. Error	Tree error	ΔFN	Time (hrs)
1,000,000	UPP(F,ND)	13.0%	8.4%	2.8%	51.6
1,000,000	UPP(D,ND)	11.1%	7.7%	2.1%	64.7
1,000,000	UPP(F)	12.8%	7.5%	2.0%	286.4
1,000,000	TA	0.0%	5.6%	0.0%	0.0

Table B2: **Empirical statistics of the true alignments for the simulated datasets.** The p-distance between two sequences is the proportion of fully ungapped sites in which they have different letters. Average p-distance is computed by averaging all p-distances between all pairs of sequences. Max p-distance is the maximum p-distance over all pairs of sequences. Values marked with an “*” were estimated by subsampling 1000 sequences at random and computing the empirical statistics of the sample. This was repeated 10 times, and the average of the results is reported (with the exception of the max p-distance where we report the maximum value of the replicates).

Dataset	Number taxa	# Sites ref. align	Avg. Seq. length	Max p-dist.	Avg. seq. p-dist.	Prop. gapped characters	Avg. gap length
1000L1	1,000	3,517	1,019.7	0.77	0.69	0.71	12.34
1000L2	1,000	2,830	1,009.0	0.77	0.70	0.64	13.15
1000L3	1,000	7,142	1,023.8	0.77	0.69	0.86	19.79
1000L4	1,000	2,249	1,007.9	0.61	0.51	0.55	10.07
1000L5	1,000	1,859	1,007.7	0.60	0.50	0.46	11.13
1000M1	1,000	3,880	1012.9	0.76	0.69	0.74	9.99
1000M2	1,000	4,068	1,022.1	0.77	0.69	0.75	10.35
1000M3	1,000	2,361	1,006.2	0.75	0.66	0.57	6.90
1000M4	1,000	2,698	1,008.2	0.61	0.50	0.63	8.08
1000M5	1,000	1,690	1,004.1	0.61	0.50	0.41	5.77
1000S1	1,000	1,901	1,004.5	0.77	0.69	0.47	3.44
1000S2	1,000	1,441	1,000.1	0.76	0.69	0.31	2.61
1000S3	1,000	1,715	1,001.4	0.76	0.69	0.42	3.08
1000S4	1,000	1,288	1,000.3	0.61	0.50	0.22	2.38
1000S5	1,000	1,151	999.9	0.62	0.51	0.13	2.43
COG1028	5,000	251	233.8	0.92	0.62	0.07	3.05
COG1309	5,000	197	179.4	0.92	0.72	0.09	8.39
COG2814	5,000	384	346.9	1.00	0.65	0.10	12.57
COG438	5,000	380	337.0	0.88	0.68	0.11	7.64
COG583	5,000	296	284.2	1.00	0.67	0.04	2.95
COG596	5,000	281	254.3	1.00	0.71	0.09	5.93
COG642	5,000	322	276.1	0.89	0.65	0.14	12.22
10000M2	10,000	5,109	1,000.0	0.75	0.68	0.80	6.83
10000M3	10,000	3,088	1,000.3	0.70	0.63	0.68	5.89
10000M4	10,000	1,831	1,000.2	0.59	0.51	0.45	5.10
RNASim 10K	10,000	8,637	1,554.6	0.62	0.41	0.82	5.24
RNASim 50K	50,000	12,400	1,554.6	0.62	0.41	0.87	7.24
RNASim 100K	100,000	14,316	1,554.5	0.61*	0.41*	0.89	8.41
RNASim 200K	200,000	16,365	1,554.5	0.62*	0.41*	0.91	9.65
RNASim 1M	1,000,000	21,946	1,554.5	0.62*	0.41*	0.93	13.18*

Table B3: **Empirical statistics of the reference alignments of the biological datasets.** The p-distance between two sequences is the proportion of fully ungapped sites in which they have different letters. Average p-distance is computed by averaging all p-distances between all pairs of sequences. Max p-distance is the maximum p-distance over all pairs of sequences. Some statistics for the HomFam datasets are computed only on the seed alignment; thus the total number of taxa and average sequence length refer to the size and length of the HomFam seed sequences, and the total number of taxa and average sequence length of the entire HomFam dataset are shown in parentheses next to the seed size and seed length.

Dataset	Number taxa	Sites in reference alignment	Avg. Sequence length	Max p-distance	Avg. p-distance	Prop. gapped characters	Avg. gap length
1GADBL_100	561	490	324.9	0.71	0.46	0.34	7.04
coli_epi_100	320	150	133.1	0.87	0.58	0.11	2.71
RV100_BBA0039	807	2,696	395.1	1.00	0.42	0.85	57.41
RV100_BBA0067	410	1,092	463.7	0.92	0.78	0.58	23.26
RV100_BBA0081	353	1,693	585.8	1.00	0.86	0.65	38.29
RV100_BBA0101	509	4,214	492.3	1.00	0.78	0.88	144.10
RV100_BBA0117	460	110	56.7	1.00	0.75	0.48	13.14
RV100_BBA0134	717	3,186	470.2	1.00	0.73	0.85	88.70
RV100_BBA0154	303	1,275	518.5	0.85	0.66	0.59	28.46
RV100_BBA0190	397	2,547	886.3	1.00	0.69	0.65	26.57
CRW 16S.3	6,323	8,716	1557.2	0.83	0.32	0.82	9.43
CRW 16S.B.ALL	27,643	6,857	1371.9	0.77	0.21	0.80	4.94
CRW 16S.T	7,350	11,856	1492.1	0.90	0.34	0.87	12.09
HomFam aat	10 (25,100)	476	403.6 (337.8)	0.87	0.71	0.15	4.09
HomFam Acetyltransf	6 (46,285)	229	161.5 (83.0)	0.87	0.75	0.29	6.98
HomFam adh	5 (21,331)	375	373.6 (123.6)	0.47	0.36	0.00	1.17
HomFam aldosered	7 (13,277)	386	310.9 (268.6)	0.79	0.57	0.19	5.11
HomFam biotin_lipoyl	7 (11,833)	112	83.4 (71.9)	0.84	0.71	0.26	7.14
HomFam blmb	6 (17,200)	344	241.7 (192.5)	0.90	0.79	0.30	8.19
HomFam ghf13	10 (12,607)	626	469.8 (264.5)	0.84	0.72	0.25	5.06
HomFam gluts	14 (10,099)	235	215.6 (98.1)	0.81	0.60	0.08	3.08
HomFam hla	5 (13,465)	178	178.0 (153.1)	0.33	0.24	0.00	0.00
HomFam hom	8 (12,037)	98	63.5 (53.5)	0.84	0.64	0.35	9.52
HomFam myb_DNA-binding	5 (10,398)	61	53.6 (46.6)	0.77	0.59	0.12	2.47
HomFam p450	12 (21,013)	512	408.0 (331.5)	0.87	0.79	0.20	3.95
HomFam PDZ	6 (14,950)	110	93.0 (80.9)	0.84	0.69	0.15	3.19
HomFam Rhodanese	6 (14,049)	216	150.0 (102.3)	0.89	0.76	0.31	7.33
HomFam rrm	20 (27,610)	157	86.7 (67.5)	0.91	0.77	0.45	8.28
HomFam rvp	6 (93,681)	132	106.33 (94.3)	0.76	0.63	0.19	3.08
HomFam sdr	13 (50,157)	361	259.5 (163.3)	0.89	0.77	0.28	4.70
HomFam tRNA-synt.2b	5 (11,293)	467	307.8 (177.6)	0.88	0.81	0.34	8.65
HomFam zf-CCHH	15 (88,345)	39	29.1 (23.3)	0.85	0.65	0.25	2.71

B2.1 Sequence length distribution

We display the sequence length distribution for the biological datasets in Figures B2-B3. The results show that there can be large sequence length heterogeneity within the protein family (e.g., ghf13 in Fig. B2 and RV100_BBA0190 in Fig. B3).

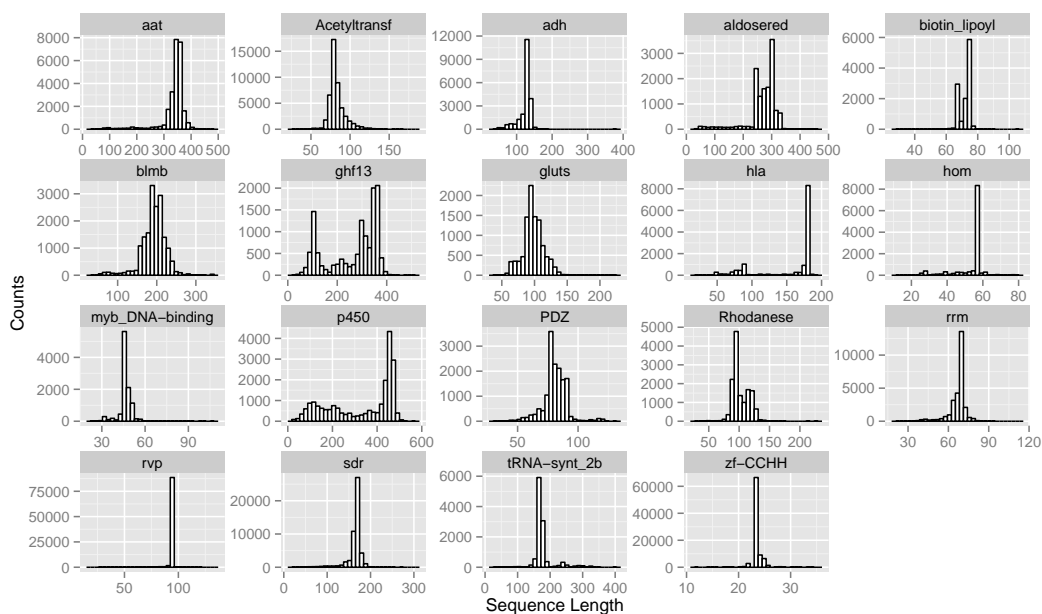


Figure B2: Sequence length distributions for the HomFam datasets.

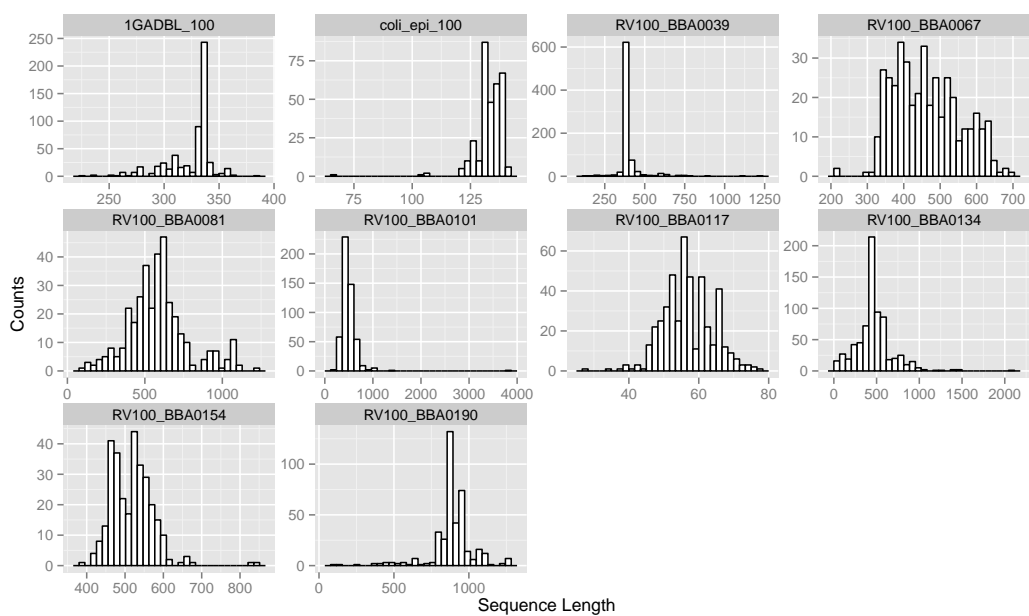
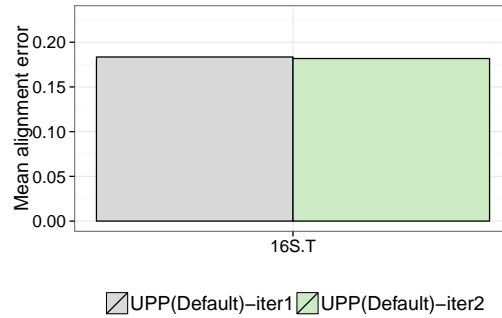


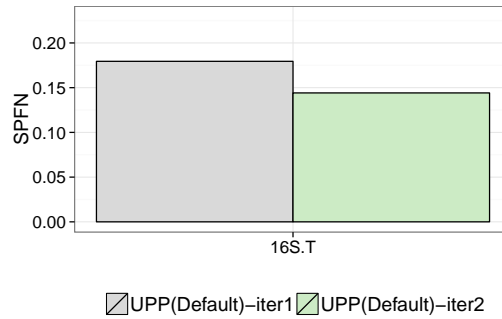
Figure B3: Sequence length distributions for the ten large AA datasets with full reference alignments.

B2.2 Resampling on the CRW 16S.T dataset

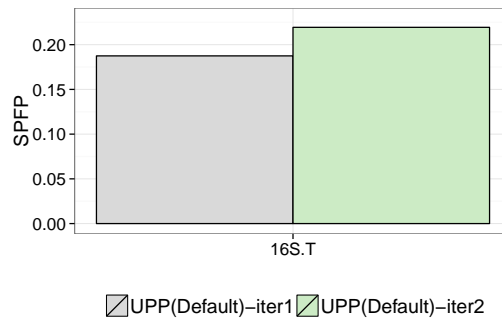
The initial UPP alignment and tree on CRW 16S.T dataset triggered the resampling algorithm due to a sparse sampling of a major clade. A new backbone set was automatically selected using the resampling algorithm. The new backbone set resulted in a more even sampling across the tree (Fig. B1). Fig. B4 shows the alignment error for the first iteration (backbone set selected by random sampling of the full-length sequences) and the second iteration of UPP (uniform sampling across the ML tree). While both iterations had very similar average alignment error (18.4% for the first iteration and 18.2% for the second iteration), the difference in ΔFN tree error was large (11.5% for the first iteration and 6.6% for the second).



(a) Average alignment error

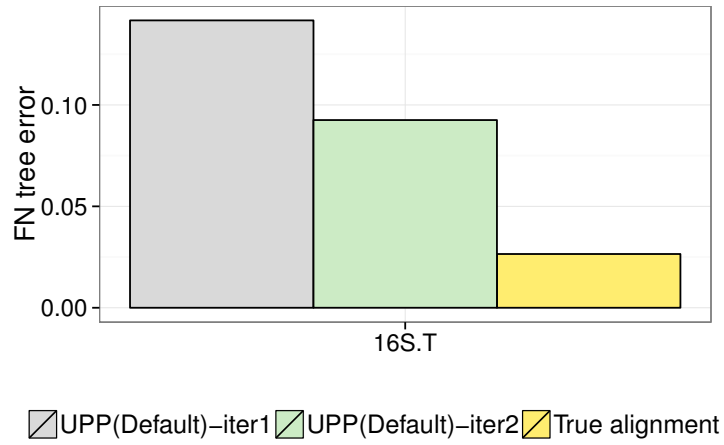


(b) Alignment SPFN error

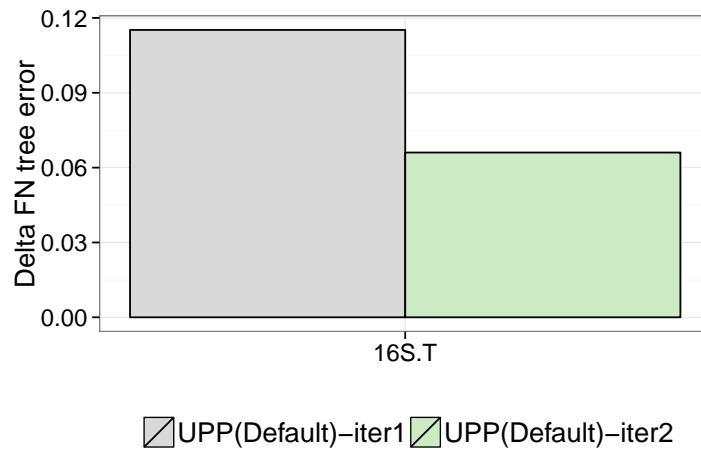


(c) Alignment SPFP error

Figure B4: **Alignment error rates for the first two iterations of UPP on the CRW 16S.T dataset.** The first iteration used a backbone sequence set selected by randomly sampling the full-length sequences from the dataset. The resulting ML tree was scored with respect to the distribution of the backbone sequences throughout the tree and triggered a second iteration of UPP. The second backbone was selected through the resampling algorithm. We show results for the first and second UPP iterations.



(a) FN tree error



(b) Delta FN tree error

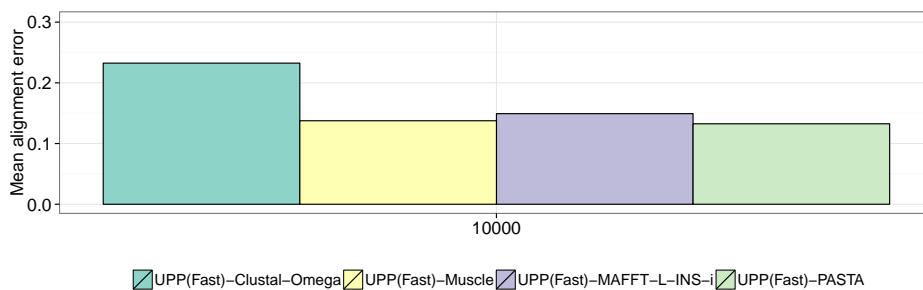
Figure B5: **Tree error rates for first two iterations of UPP on the CRW 16S.T dataset.** The first iteration used a backbone sequence set selected by randomly sampling the full-length sequences from the dataset. The resulting ML tree was scored with respect to the distribution of the backbone sequences throughout the ML tree and triggered a second iteration of UPP. The second backbone was selected through the resampling algorithm. We show results for the first and second UPP iterations. ML trees were estimated using FastTree under GTR.

B2.3 UPP pipeline exploration

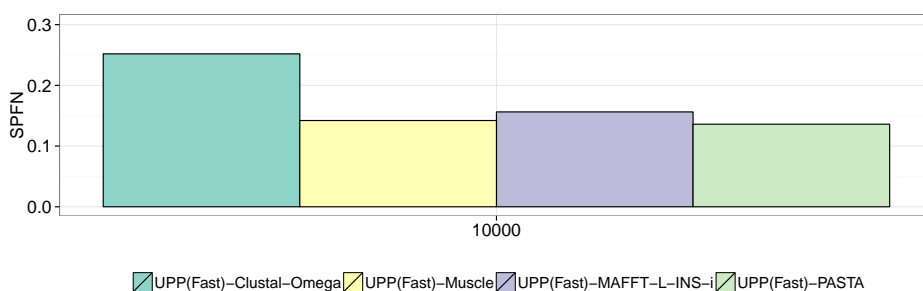
We examined different modifications to each stage of the UPP pipeline to examine the impact alignment and tree estimation accuracy. We now give a brief overview and summary of our findings.

B2.3.1 Backbone alignment method

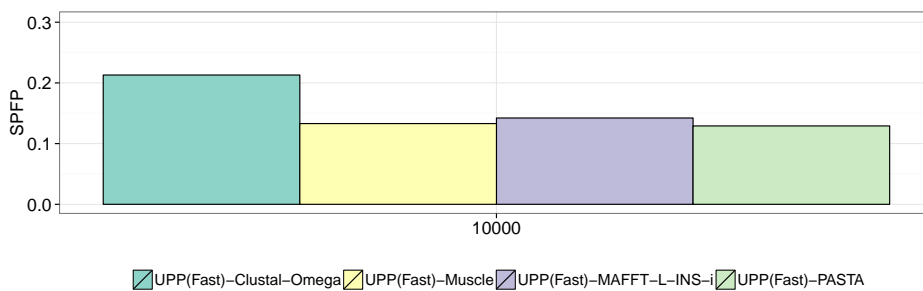
We ran UPP(Fast) on the backbone alignments estimated using Clustal-Omega, MAFFT-L-INS-i, MUSCLE, and PASTA on backbone sets of size 100 on the RNASim 10K dataset (Fig. B6 and B7). We found that UPP using PASTA and MUSCLE backbones resulted in the most accurate UPP alignments, followed very closely by UPP on the MAFFT-L-INS-i backbone. UPP on using the Clustal-Omega backbone, on the other hand, resulted in a distinctively worse alignment. Curiously, while UPP on PASTA and MUSCLE backbones resulted in the best alignments, UPP on PASTA and MAFFT-L-INS alignments resulted in the best trees. UPP on MUSCLE was close behind, and as before, UPP on Clustal-Omega was distinctly worse.



(a) Average alignment error

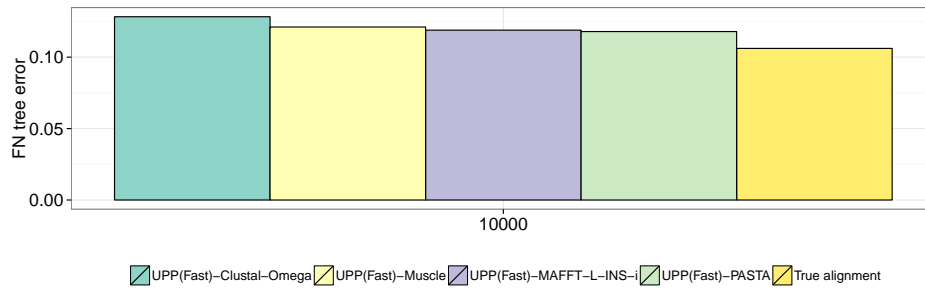


(b) Alignment SPFN error

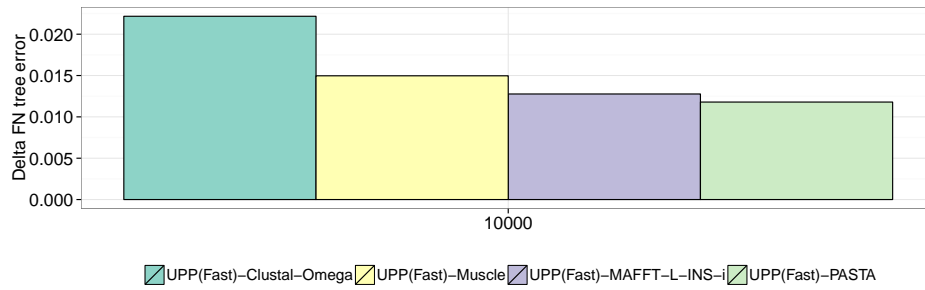


(c) Alignment SPFP error

Figure B6: **Alignment error rates of different UPP backbone alignments on the RNASim 10K dataset.** All backbones are of size 100. **NAM: fix the x-axis text size**



(a) FN tree error



(b) Delta FN tree error

Figure B7: **Tree error rates of different UPP backbone alignments on the RNASim 10K dataset.** All backbones are of size 100. ML trees were estimated using FastTree under GTR.

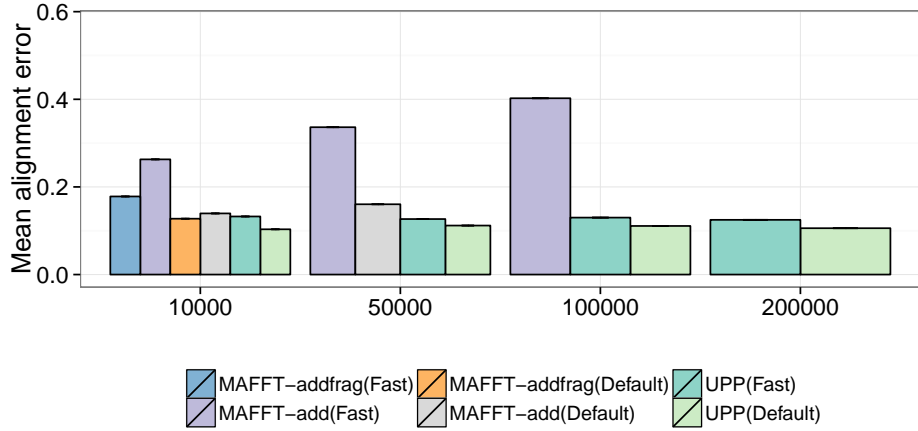
B2.3.2 Backbone size

We examined the impact the backbone size on alignment and tree accuracy. We compared the accuracy of UPP using the HMM Family technique (or a single HMM) with UPP using MAFFT-Profile on small (100) and large (1000) backbones. We denote methods that used the small backbone as “Fast” and methods that used the large backbone as “Default”.

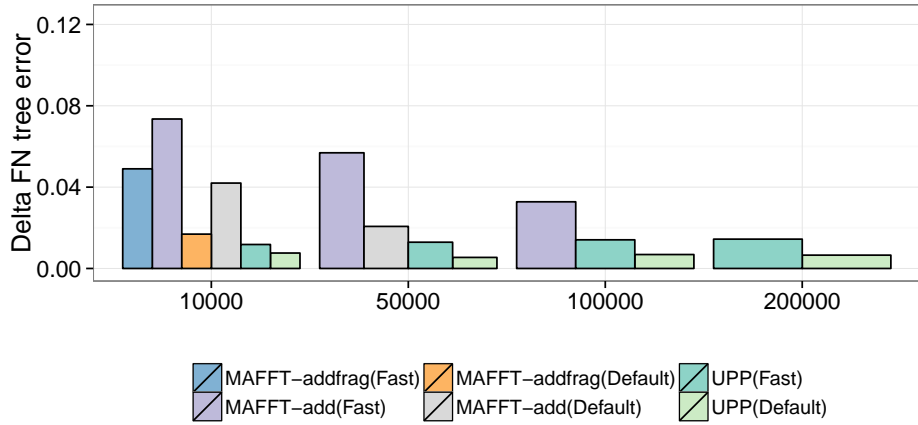
In general, using a larger backbone resulted in more accurate alignments and trees for both UPP and MAFFT-Profile-add (Fig. B8).

B2.3.3 Query sequence alignment method.

We compared three different techniques for aligning the query sequences to the backbone alignment within the UPP pipeline: using the HMM Family technique, using MAFFT-Profile “--add”, and using MAFFT-Profile “--addfragments”. Figure B8 showed that the HMM Family technique resulted in more accurate alignments and trees than MAFFT-Profile, whether using --add or --addfragments. In addition, UPP using the HMM Family technique made it possible to align 200,000 sequences within 24 hours, but UPP using MAFFT-Profile “--add” was unable to align the 200K dataset in that timeframe, and UPP using MAFFT-Profile “--addfragments” could only align up to 10,000 sequences (Table 1 from main document). Comparing MAFFT-Profile “--add” and MAFFT-Profile “--addfragments”, we found that MAFFT-Profile “--addfragments” resulted in more accurate alignments and trees than MAFFT-add (Fig. B8), at a large increase in running time (Table 1 from main document), however, neither MAFFT variants were as accurate as the HMM Family technique.



(a) Average alignment error



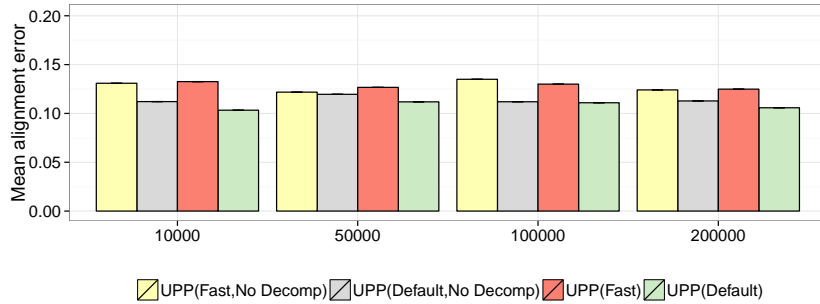
(b) Delta tree error

Figure B8: **Alignment and tree error for UPP variants on the RNASim datasets.** Methods labeled with “Default” use a backbone size of 1000. Methods labeled with “Fast” use a backbone size of 100. ML trees were estimated using FastTree under GTR.

B2.3.4 Impact of using the HMM Family technique or a single HMM

We explored in more detail the impact of using a single HMM to represent the backbone alignment (UPP with no decomposition) versus using the family of HMMs (UPP with decomposition) on both small (100) and large (1000) backbone sizes. In general, there are very minor differences between using a single HMM and the family of HMMs with respect to alignment error (Figs. B9 and B12). There are exceptions, however. The CRW 16S.T dataset cannot be fully aligned using a single HMM. The dataset contains a sequence which HMMALIGN cannot align to the HMM computed on the entire backbone for both “Fast” and “Default” versions of the backbones. However, when decomposition is applied, every sequence in 16S.T can be aligned. Thus, the family of HMMs technique is essential for generating an alignment on some datasets.

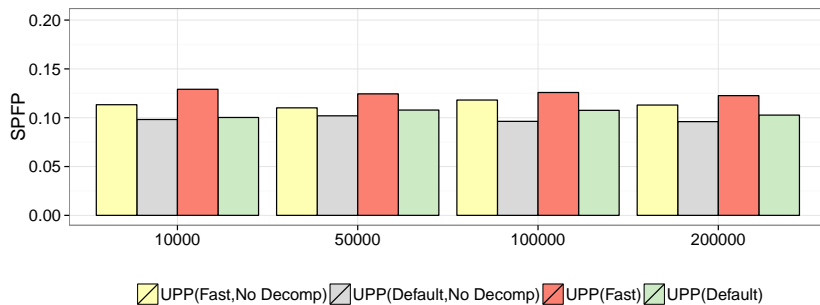
The impact of decomposition is much more substantial with respect to tree accuracy: UPP with decomposition resulted in significantly more accurate trees (Fig. B10 and B13). However, decomposition does come with a significant increase in running time (Fig. B11).



(a) Average alignment error

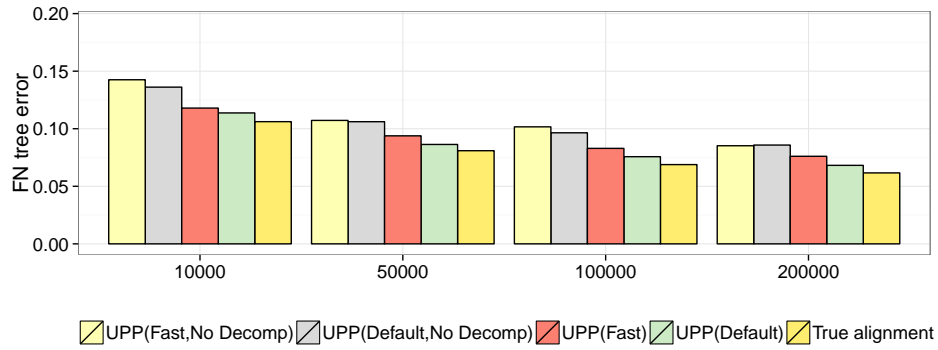


(b) Alignment SPFN error

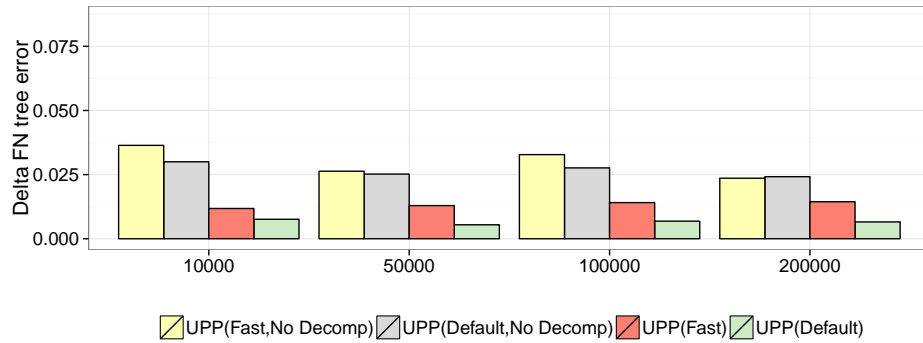


(c) Alignment SPFP error

Figure B9: **Alignment error for UPP variants on the RNASim datasets.** Methods labeled with “Default” use a backbone size of 1000. Methods labeled with “Fast” use a backbone size of 100.



(a) FN tree error



(b) Delta FN tree error

Figure B10: **Tree error of UPP variants on the RNASim datasets.** Methods labeled with “Default” use a backbone size of 1000. Methods labeled with “Fast” use a backbone size of 100. ML trees were estimated using FastTree under GTR.

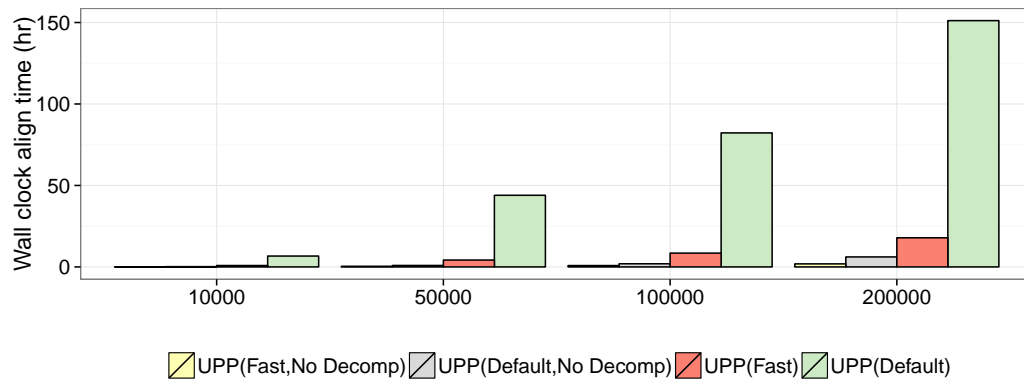


Figure B11: **Wall clock alignment time (hrs) of UPP variants on the RNASim datasets.** All methods were run on a machine with 12 CPUs and 24 GB of memory. Methods labeled with “Default” use a backbone size of 1000. Methods labeled with “Fast” use a backbone size of 100. ML trees were estimated using FastTree under GTR.



(a) Average alignment error

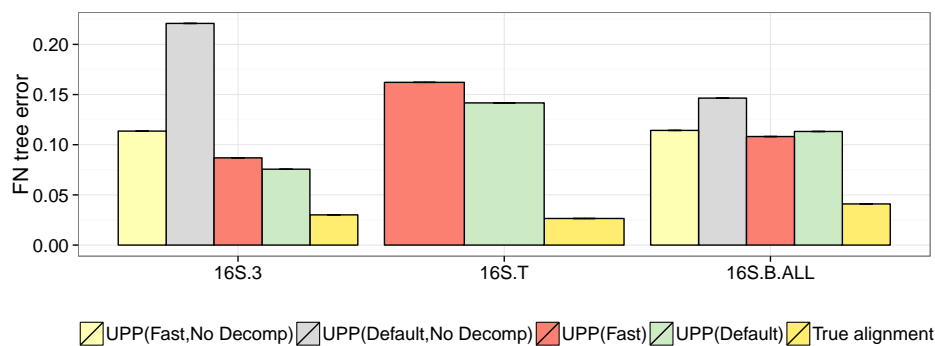


(b) Alignment SPFN error

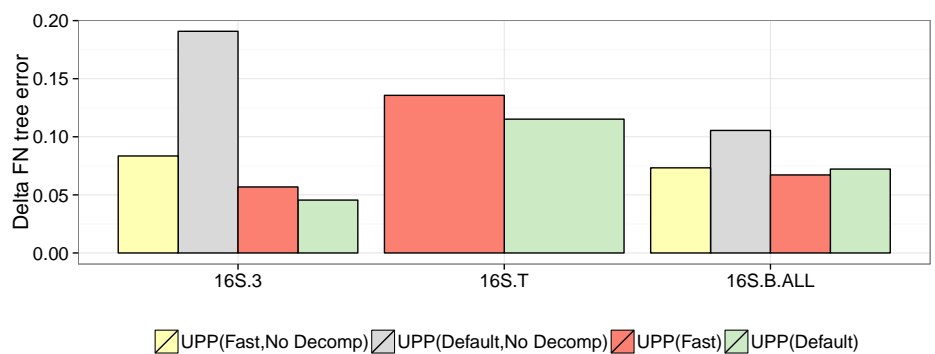


(c) Alignment SPFP error

Figure B12: **Alignment error for UPP variants on the CRW 16S datasets.** Methods labeled with “Default” use a backbone size of 1000. Methods labeled with “Fast” use a backbone size of 100. Both UPP(Default,No Decomp) and UPP(Fast,No Decomp) failed to align one of the 16S.T sequences and thus failed to generate an alignment on the entire dataset. UPP results are based on the first iteration of UPP.



(a) FN tree error



(b) Delta FN tree error

Figure B13: **Tree error of UPP variants on the CRW 16S datasets.** ML trees were estimated using FastTree under GTR. Methods labeled with “Default” use a backbone size of 1000. Methods labeled with “Fast” use a backbone size of 100. Both UPP(Default, No Decomp) and UPP(Fast, No Decomp) failed to align one of the 16S.T sequences and thus failed to generate an alignment on the entire dataset. UPP results are based on the first iteration of UPP.

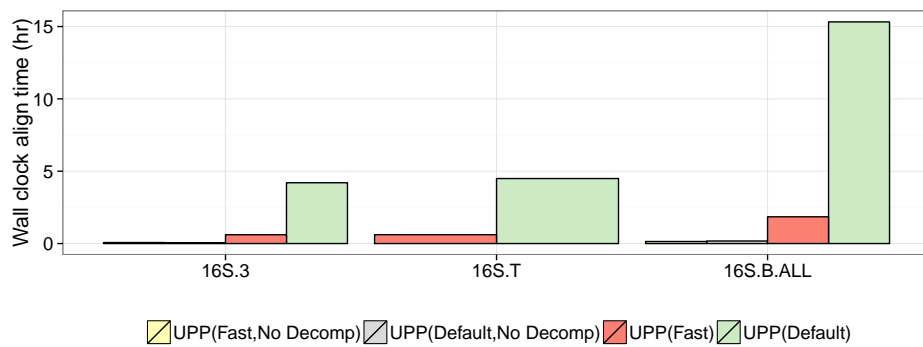


Figure B14: **Wall clock alignment time (hrs) of UPP variants on the RNASim datasets.** All methods were run on a machine with 12 CPUs and 24 GB of memory. UPP(Default) uses a backbone of size 1000. UPP(Fast) uses a backbone of size 100. Both UPP(Default, No Decomp) and UPP(Fast, No Decomp) failed to align one of the 16S.T sequences and thus failed to generate an alignment on the entire dataset. UPP results are based on the first iteration of UPP.

B2.4 SEPP vs. UPP

We now compare UPP and SEPP. Recall that both methods use the same backbone, but SEPP divides the dataset into approximately ten disjoint subsets of approximately equal size, and constructs HMMs on each subset alignment. In comparison, UPP uses the HMM Family technique, which will produce a much larger collection of HMMs.

SEPP(Default,10%) and UPP(Default) use the same backbone alignment, but SEPP decomposes the alignment into disjoint subsets (approximately 10 of them), using the centroid edge decomposition from SATé-II; “10%” refers to the requirement that every subset should have approximately 10% of the backbone sequences, the protocol used in [55].

Both UPP and SEPP had comparable alignment accuracy on both the RNASim and CRW 16S datasets (Figs. B15-B17). Differences between the methods were more distinct on the CRW 16S datasets, especially with respect to tree accuracy. UPP resulted in significantly better trees on all of the CRW datasets (Fig. B18). In addition, UPP was more robust to fragmentary datasets compared to SEPP (Fig. B19).



(a) Average alignment error

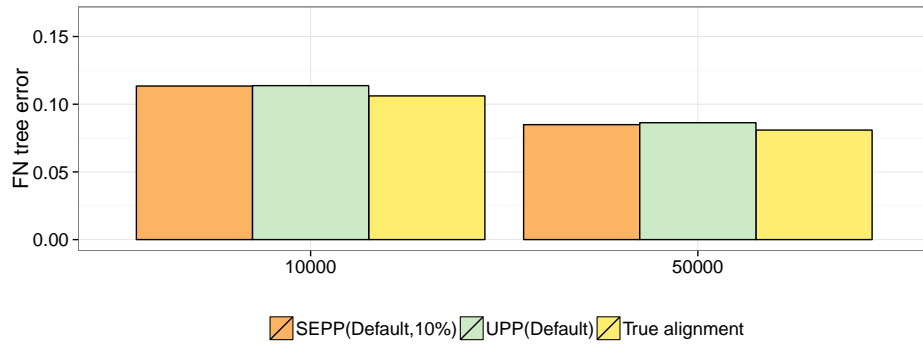


(b) Alignment SPFN error

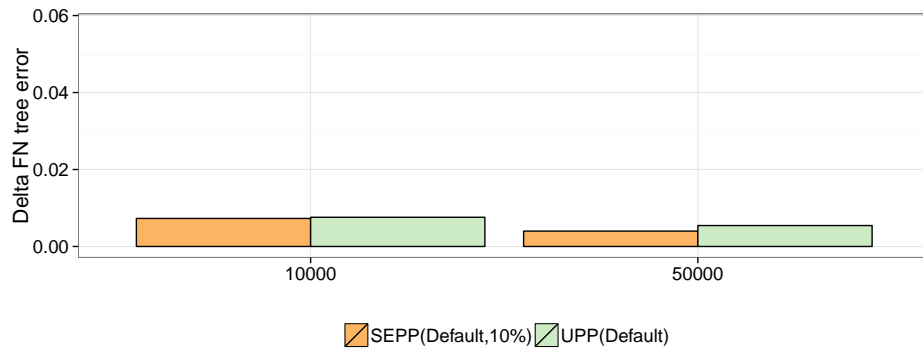


(c) Alignment SPFP error

Figure B15: **Alignment error for UPP and SEPP on the RNASim datasets with 10K and 50K sequences.** UPP(Default) and SEPP(Default,10%) both use the same backbone of size 1000. SEPP decomposes the backbone into (approximately) 10 subsets each of the same size.

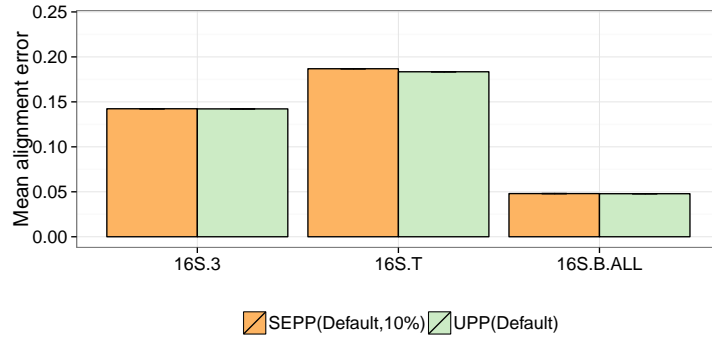


(a) FN tree error

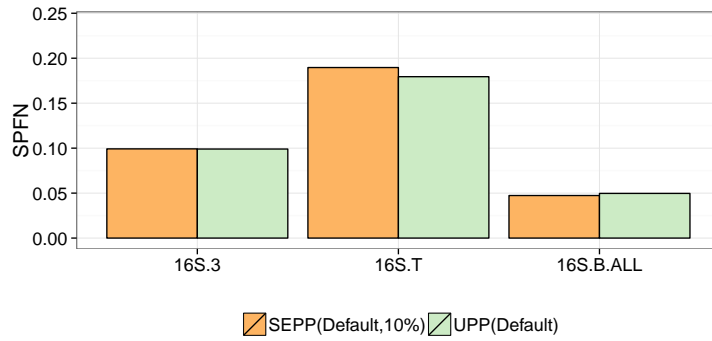


(b) Delta FN tree error

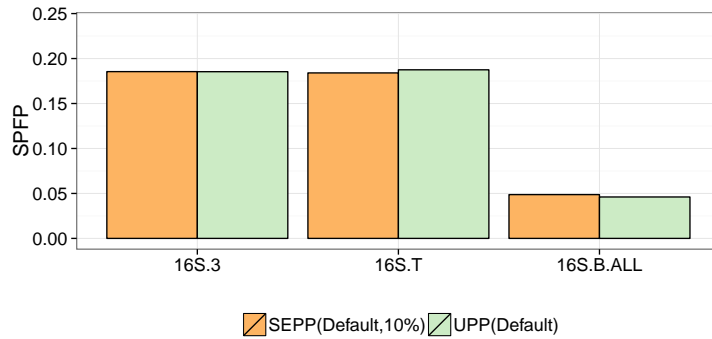
Figure B16: **Tree error of UPP and SEPP on the RNASim datasets with 10K and 50K sequences.** ML trees were estimated using FastTree under GTR. UPP(Default) and SEPP(Default,10%) both use the same backbone of size 1000. SEPP(Default,10%) decomposes the backbone into (approximately) 10 subsets each of the same size.



(a) Average alignment error

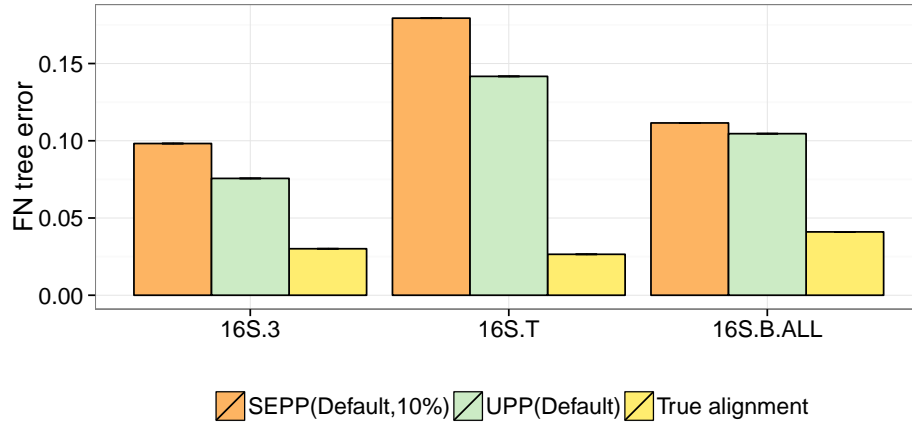


(b) Alignment SPFN error

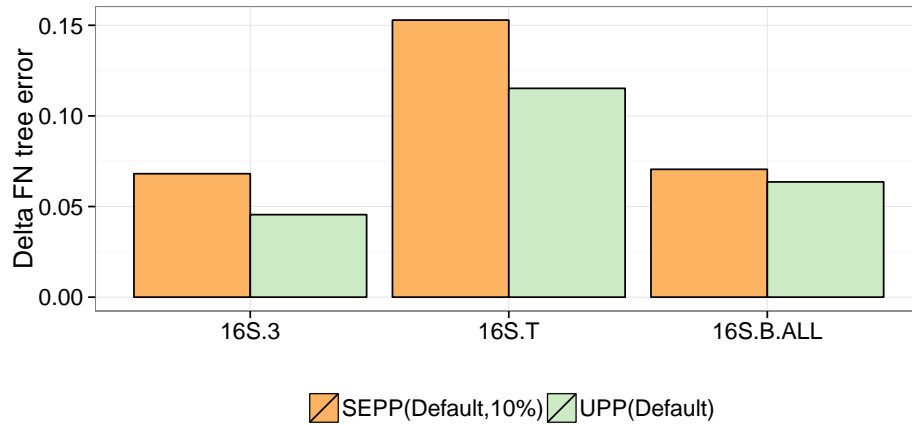


(c) Alignment SPFP error

Figure B17: **Alignment error for UPP and SEPP on the CRW 16S datasets.** UPP(Default) and SEPP(Default,10%) both use the same backbone of size 1000. SEPP(Default,10%) decomposes the backbone into (approximately) 10 subsets each of the same size. UPP and SEPP results shown are based on the first iteration.

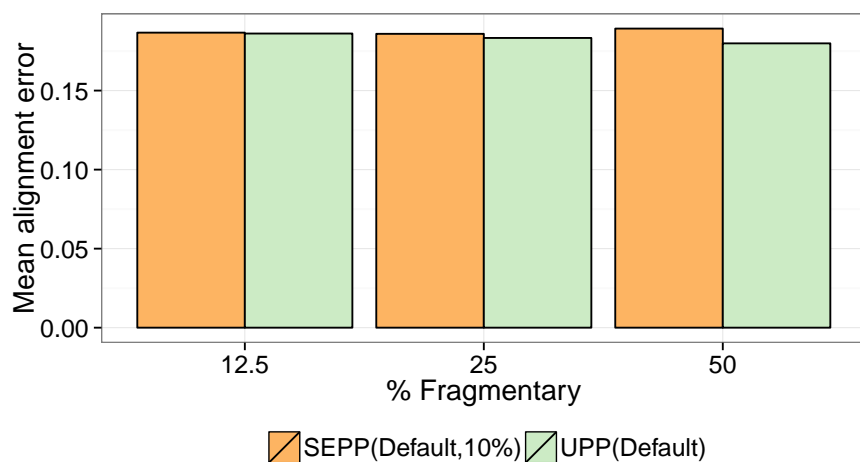


(a) FN tree error

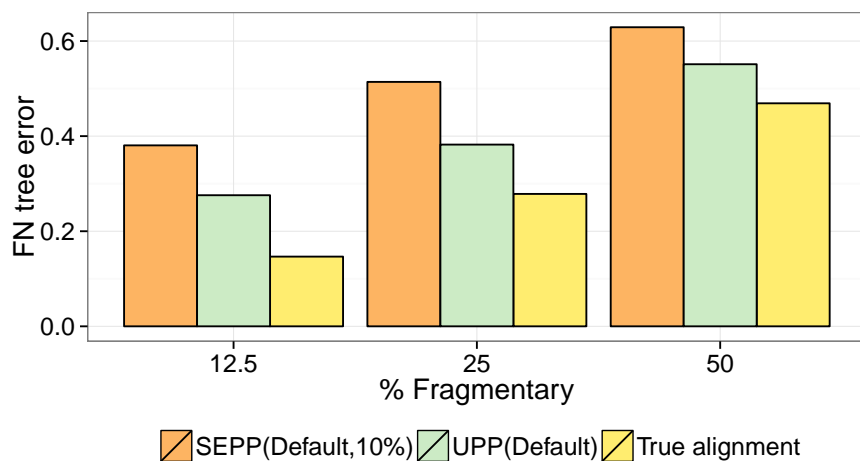


(b) Delta FN tree error

Figure B18: **Tree error of UPP and SEPP on the CRW 16S datasets.** ML trees were estimated using FastTree under GTR. UPP(Default) and SEPP(Default,10%) both use the same backbone of size 1000. SEPP(Default,10%) decomposes the backbone into (approximately) 10 subsets each of the same size. UPP and SEPP results shown on 16S.T are based off of one iteration.



(a) Average alignment error

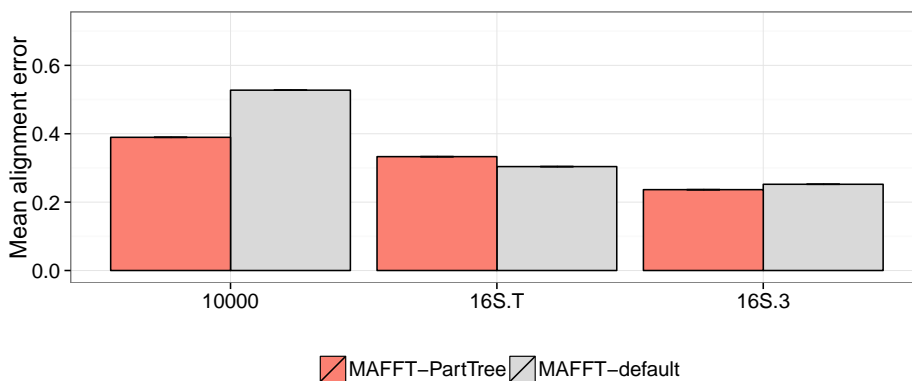


(b) Tree error

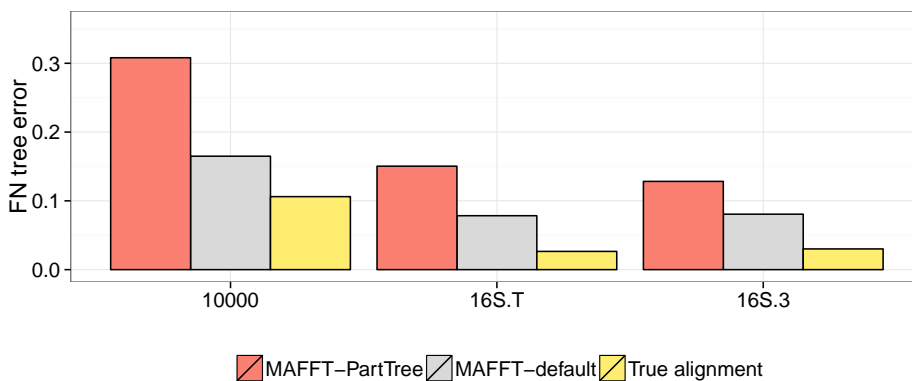
Figure B19: **Alignment and tree error for SEPP and UPP on fragmentary CRW 16S.T datasets.** We show alignment error and tree error for UPP and SEPP on the fragmentary CRW 16S.T datasets, varying the percentage of fragmentary sequences, each with an average length of 500 sites (i.e., approximately one third the average sequence length for 16S.T). UPP(Default) and SEPP(Default,10%) both use the same backbone of size 1000. SEPP(Default,10%) decomposes the backbone into (approximately) 10 subsets each of the same size. UPP results are based on the first iteration of UPP.

B2.5 MAFFT variants

We compared MAFFT-PartTree and MAFFT-Default on the large datasets. Fig. B20 shows that MAFFT-PartTree results in comparable or better alignments than default MAFFT, however, default MAFFT results in significantly better trees.



(a) Average alignment error



(b) FN tree error

Figure B20: **Results of default MAFFT and MAFFT-PartTree on the 16S.T, 16S.3, and RNASim 10K datasets.** All ML trees were estimated using FastTree under GTR.

B2.6 PASTA on the ten large AA datasets

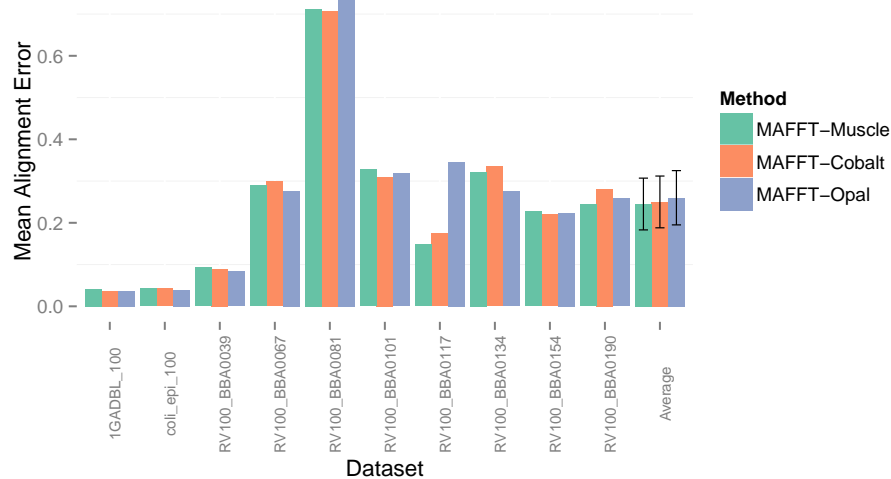
UPP’s alignment accuracy depends on the accuracy of the backbone alignment. PASTA is an improvement on SATé-II, and both have been studied extensively on NT datasets [56]; however, no published studies have studied PASTA, SATé-I or SATé-II on AA datasets.

We explored PASTA variants, varying the technique used to estimate alignments on subsets and then to merge alignments together, using the 10 AA datasets with full reference alignments. Initial analyses (data not shown) revealed that MAFFT-L-INS-i gave the best results for producing the subset alignments. We then evaluated techniques for merging alignments, including Opal [93], MUSCLE [16], or COBALT [63].

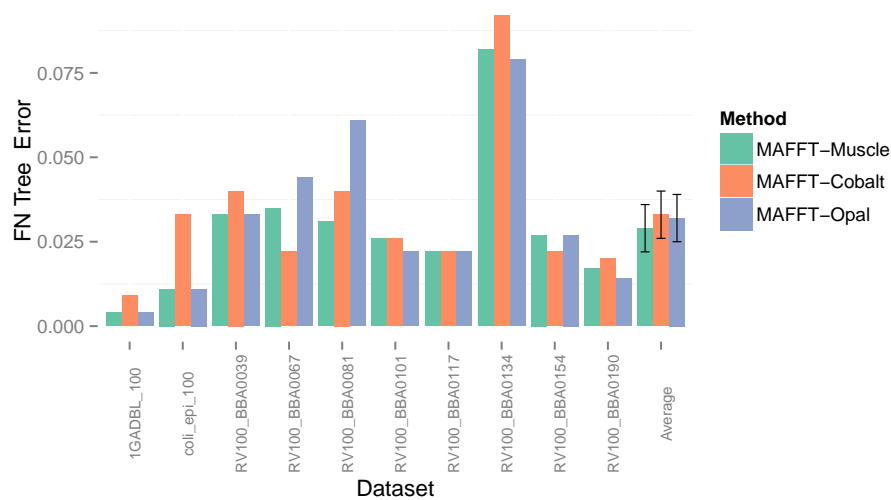
We ran PASTA under default settings (no starting tree, subset size 200, MAFFT-L-INS-i to align subsets, FastTree to compute trees in each iteration, and running for three iterations), varying only the alignment merger technique. The software version numbers and commands used within PASTA to align the sequences and merge the subsets are given in Section B2.6.1. ML trees were estimated on the alignments using RAxML under JTT, LG, or WAG models of protein evolution (model selection described in Section B2.6.2).

We found that while all PASTA variants resulted in alignments with comparable accuracy, RAxML maximum likelihood trees on PASTA using MUSCLE to merge subalignments resulted in the most accurate trees (Fig. B21). We refer to this version as “PASTA-MUSCLE.”

We then compared PASTA-MUSCLE to alignments and trees computed using standard MSA methods followed by RAxML for maximum likelihood. PASTA-MUSCLE and MAFFT-L-INS-i gave the most accurate alignments, but PASTA-MUSCLE resulted in the most accurate trees (Fig. B22). Thus, we used PASTA-MUSCLE to generate our backbone alignments for AA datasets.

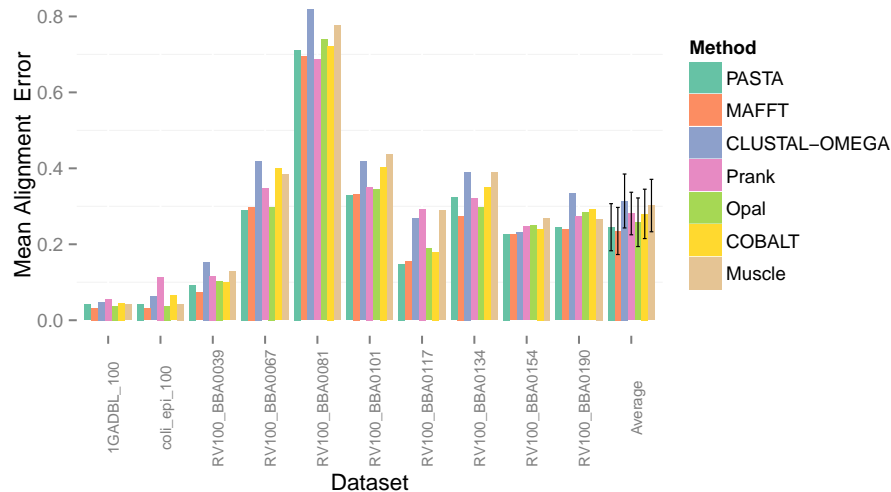


(a) Mean alignment error of PASTA variants

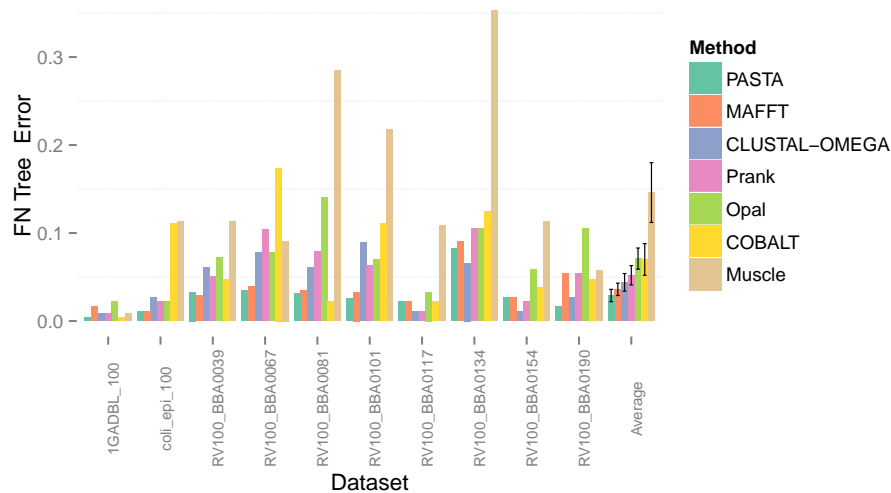


(b) Tree FN error of PASTA variants

Figure B21: **Alignment and tree error for PASTA variants on the ten large AA datasets with full reference alignments.** Subalignments were estimated using MAFFT-L-INS-i, and the resulting subalignments were merged with either MUSCLE, Opal, or Cobalt. ML Trees were estimated using RAxML under amino acid substitution models selected using PROTEST (see Section B2.6.2).



(a) Mean alignment error



(b) Tree FN error

Figure B22: **Alignment and tree error of different methods on the ten large AA datasets with full reference alignments.** PASTA used MAFFT-L-INS-i to align subalignments, and MUSCLE to merge subalignments. ML Trees were estimated using RAxML under amino acid substitution models selected using PROTEST (see Section B2.6.2).

B2.6.1 PASTA commands

The following software was used to generate the results presented in Section B2.6:

Each dataset was aligned (when possible) using Opal [93] version 2.0.0, Clustal-Omega [75] version 1.0.2, MAFFT[35–37] version 6.857b, Cobalt [63] version 2.0.1, MUSCLE [16] version 3.8.31, PRANK [49] version 100802 and PASTA version 1.0 [56]. Due to a bug in earlier versions of MAFFT 6.956b, MAFFT-Profile and MAFFT-default were run using MAFFT version 7.143.

The commands used for the experiments in Section B2.6 are given below.

- **Clustal-Omega:** `clustalo -align -i<input_sequence> -o <output_alignment>`
- **MAFFT:** `mafft --localpair --maxiterate 1000 --ep 0.123 <input_sequence> > <output_alignment>`
- **Opal:** `java -Xmx20g -jar opal.jar --in <input_sequences> --out <output_alignment>`
- **MUSCLE:** `muscle -in <input_sequence> -out <output_alignment>`
- **Cobalt:** `cobalt -i <input_sequence> -rpsdb <cdd_clique_0.75> > output_alignment>`

- **Prank:** *prank -once -noxml -notree -nopost +F -quiet -matinitsize=5 -protein -d=<input_sequence> -o=<output_alignement>*
- **RaxML:** *raxml -m PROTGAMMA<model> -n ml -s <output_phylip> -T2 -w <working_directory>*
- **PASTA:** *python run_pasta.py -o <output_directory> -i <input_sequences> -t <starting_tree> --auto --num-cpus=12 --datatype=<molecule_type>*

B2.6.2 Model selection for PASTA variants

Model selection for the ten large AA datasets with full reference alignments was performed with PROTEST, using the input parameters listed below:

```
Alignment file..... : [MAFFT-L-INS-i alignment]
Tree.....           : RAxML parsimony tree on
                        MAFFT-L-INS-i
StrategyMode.....   : Fast (optimize branch lengths & model)
Candidate models..... :
  Matrices.....     : JTT LG WAG
  Distributions..... : +G
  Number of rate categ... : 4
  Observed frequencies... : false
Statistical framework
  Sort models according to....: AIC
  Sample size.....         : 0.0 (not calculated yet)
  sampleSizeMode.....      : Total number of characters
                        (alignment length)
```

PROTEST selected the following AA models for the 10 AA datasets:

- 1GADBL_100: LG
- coli_100: LG

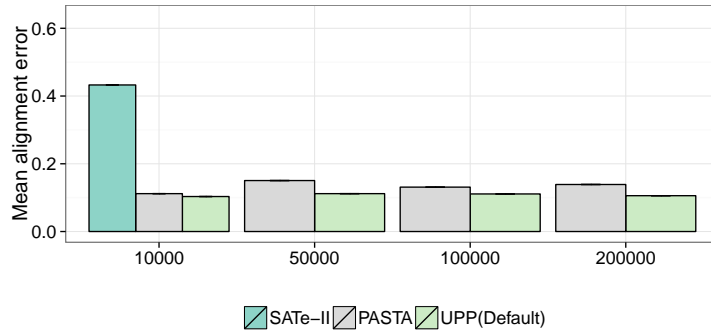
- RV100_BBA0039: LG
- RV100_BBA0067: WAG
- RV100_BBA0081: JTT
- RV100_BBA0101: WAG
- RV100_BBA0117: LG
- RV100_BBA0134: JTT
- RV100_BBA0154: WAG
- RV100_BBA0190: LG

B2.7 Comparisons between UPP, SATé-II, and PASTA

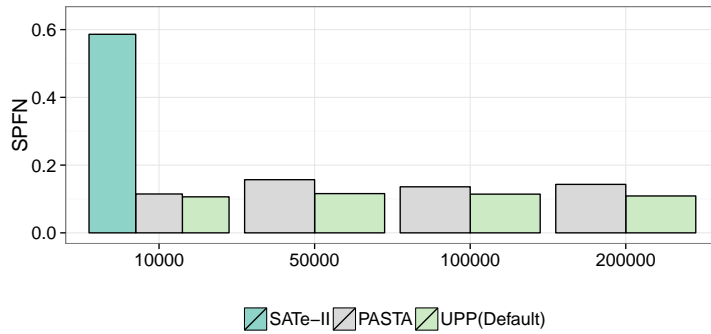
We compared UPP to SATé-II and PASTA on both full-length sequences and fragmentary sequences. PASTA is generally more accurate than SATé-II with respect to both trees and alignments (Fig. B23), and is also faster.

The comparison between UPP and PASTA on the full-length datasets shows that UPP typically results in comparable or better alignments (figs. B27, B23 and B30), but that PASTA results in comparable or better trees (figs. B24 and B26).

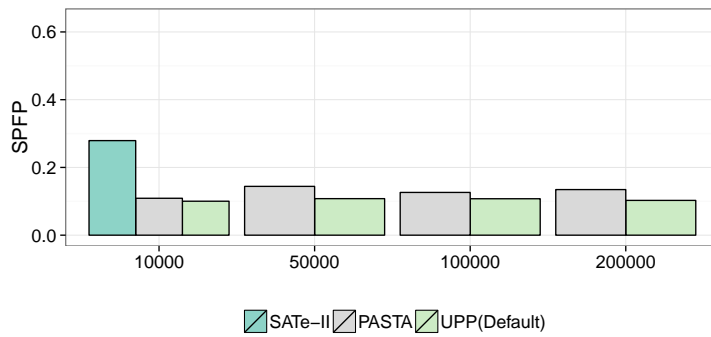
On fragmentary datasets, UPP consistently resulted in better alignments and trees than PASTA, especially as datasets became more fragmentary (fig. B32).



(a) Average alignment error

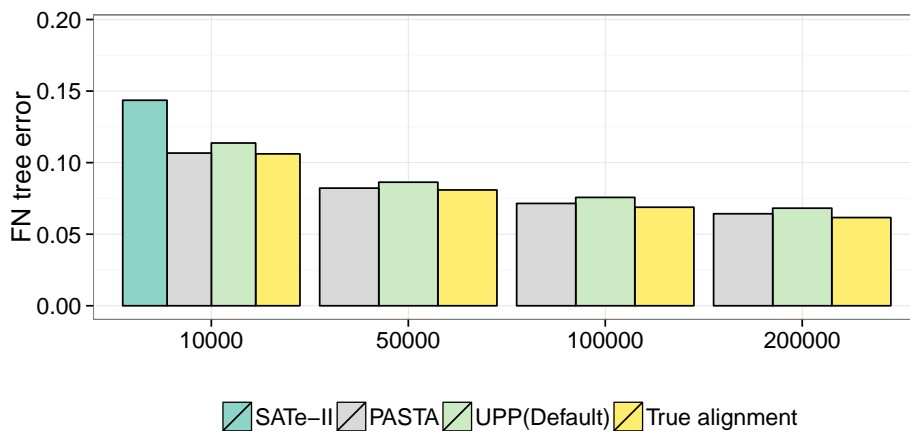


(b) Alignment SPFN error

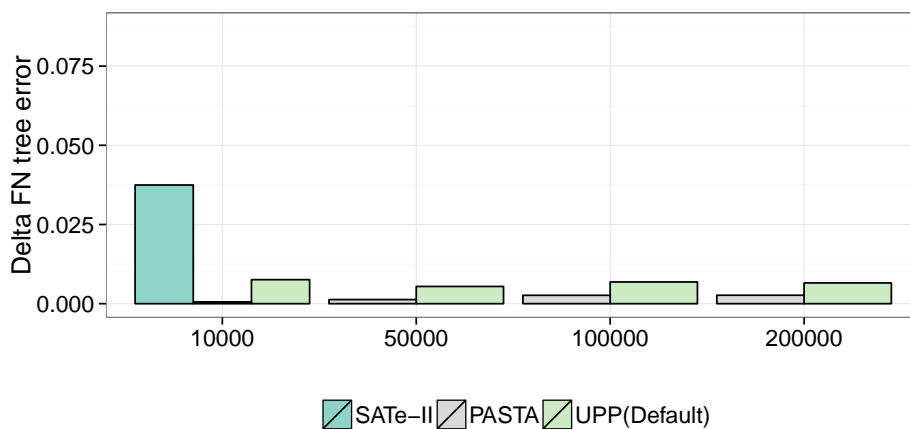


(c) Alignment SPFP error

Figure B23: **Alignment error of UPP, SATé-II, and PASTA on the RNASim datasets.** UPP(Default) uses a backbone of size 1000. Results not shown indicate failure to complete within 24 hours using 12 processors.

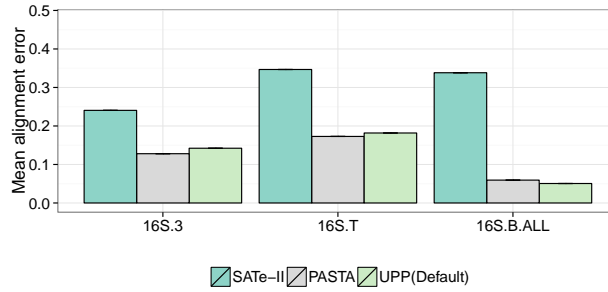


(a) FN tree error

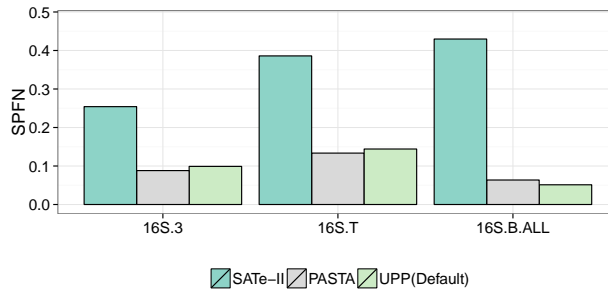


(b) Delta FN tree error on the RNASim datasets.

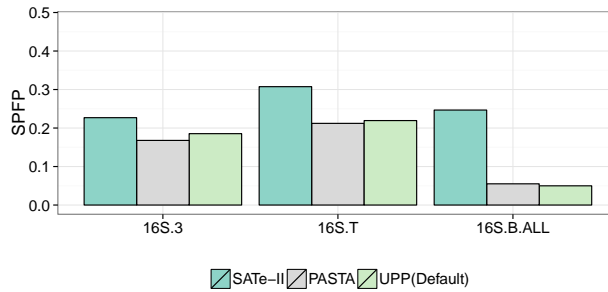
Figure B24: **Tree error of UPP, SATé-II, and PASTA for the RNASim datasets.** ML trees were estimated using FastTree under GTR. UPP(Default) uses a backbone of size 1000. Results not shown indicate failure to complete within 24 hours using 12 processors.



(a) Average alignment error

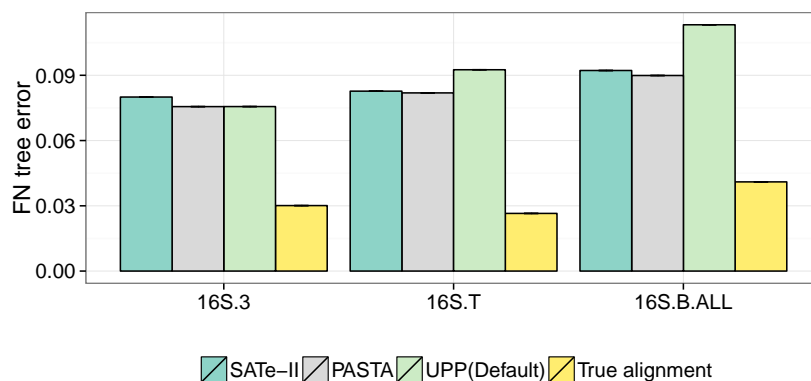


(b) Alignment SPFN error

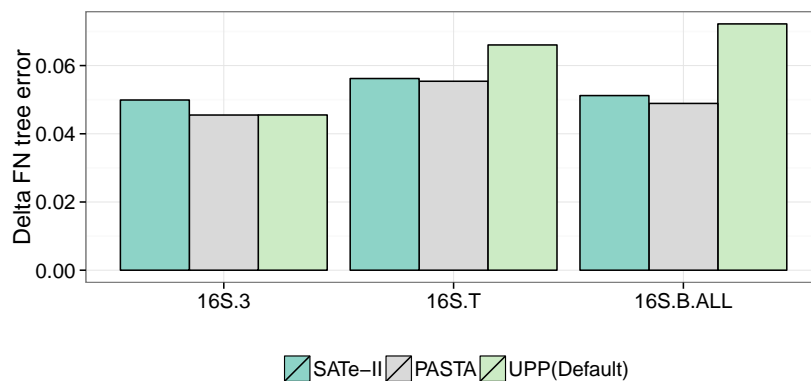


(c) Alignment SPFP error

Figure B25: **Alignment error of SATé-II, PASTA, and UPP on the CRW datasets.** All methods were allowed to run till termination and were not limited by the 24 hour time limit. UPP was run with 2 iterations on the 16S.T dataset. UPP(Default) uses a backbone of size 1000. Backbone sequences were selected from all sequences that were within $1500 \text{ bps} \pm 375 \text{ bps}$ in length.

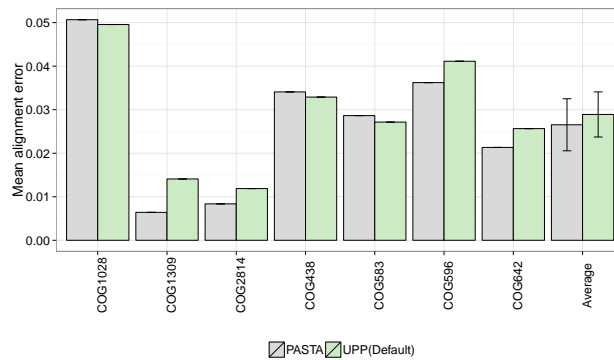


(a) FN tree error

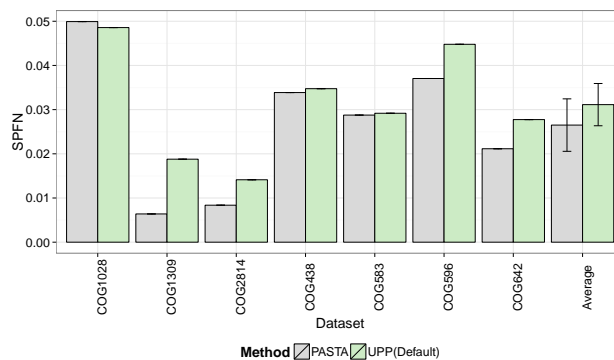


(b) Delta FN tree error

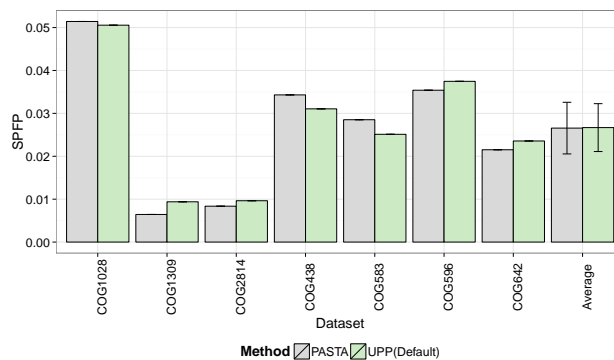
Figure B26: **Tree error for PASTA, SATé-II, and UPP on the CRW datasets.** ML trees were estimated using FastTree under GTR. All methods were allowed to run till termination and were not limited by the 24 hour time limit. UPP was run with 2 iterations on the 16S.T dataset. UPP(Default) uses a backbone of size 1000. Backbone sequences were selected from all sequences that were within $1500 \text{ bps} \pm 375 \text{ bps}$ in length.



(a) Average alignment error

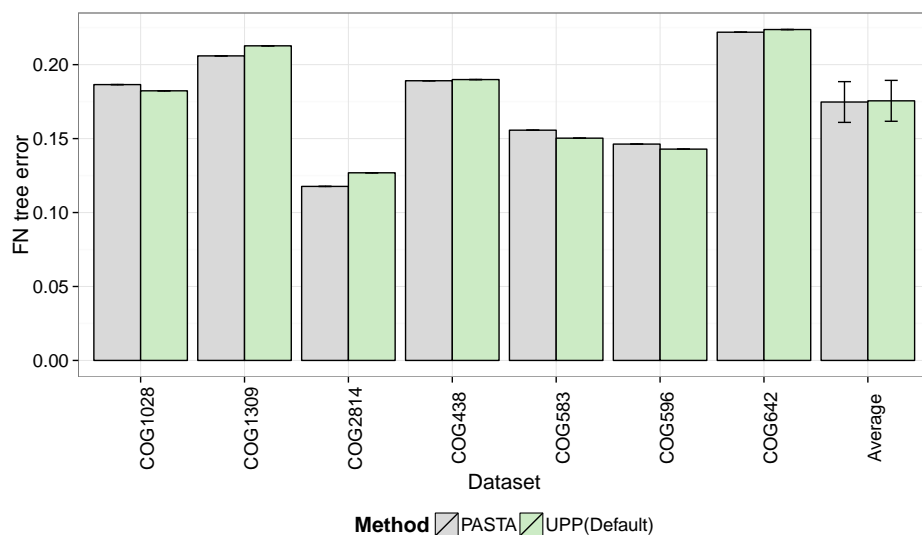


(b) Alignment SPFN error

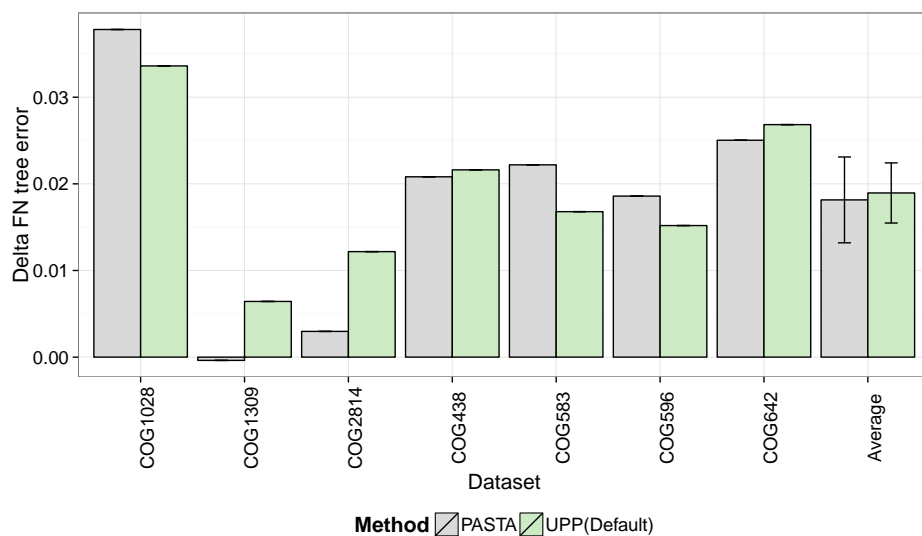


(c) Alignment SPFP error

Figure B27: **Alignment error of PASTA and UPP on the FastTree COG datasets.** Backbone sequences were filtered by only selecting from sequences within 75% to 125% in length of the median sequence length of the reference sequences. Standard error bars are shown.



(a) FN tree error



(b) Delta FN tree error

Figure B28: **Tree error of UPP and PASTA on the simulated Fast-Tree COG datasets.** ML trees were estimated using FastTree under JTT. Backbone sequences were filtered by only selecting from sequences within 75% to 125% in length of the median sequence length of the reference sequences. UPP(Default) uses a backbone of size 1000. Standard error bars are shown.

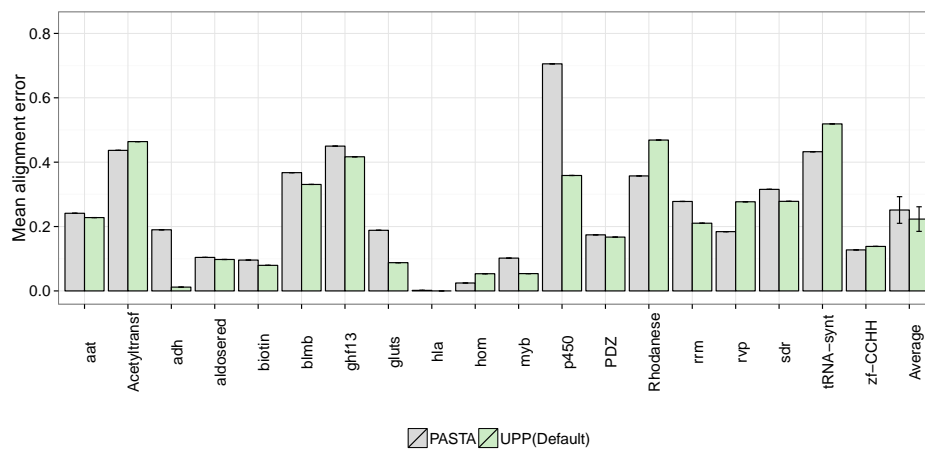
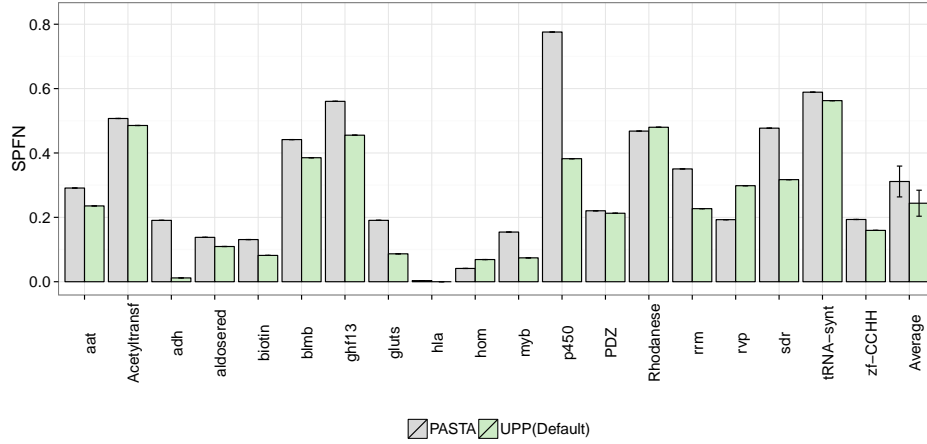
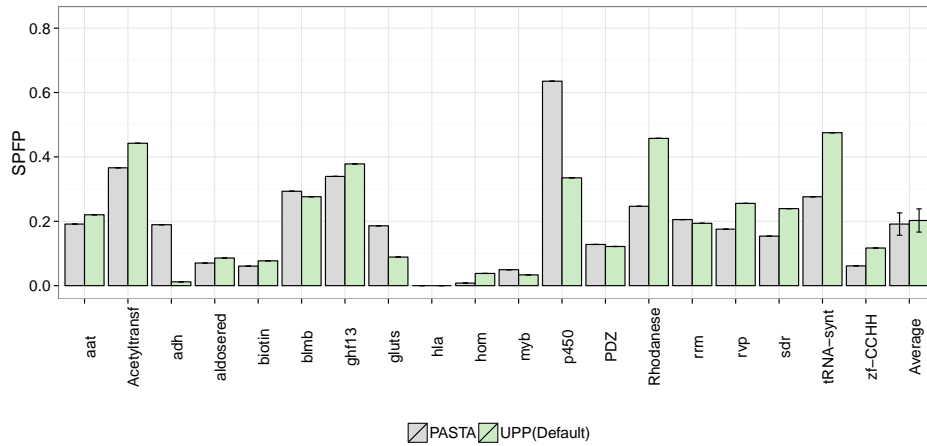


Figure B29: **Average alignment error of UPP and PASTA on the HomFam datasets.** UPP(Default) uses a backbone of size 1000. Backbone sequences were filtered by only selecting from sequences within 75% to 125% in length of the median sequence length of the seed sequences. Standard error bars are shown.

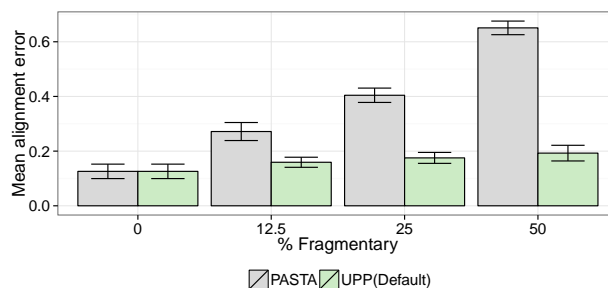


(a) Alignment SPFN error

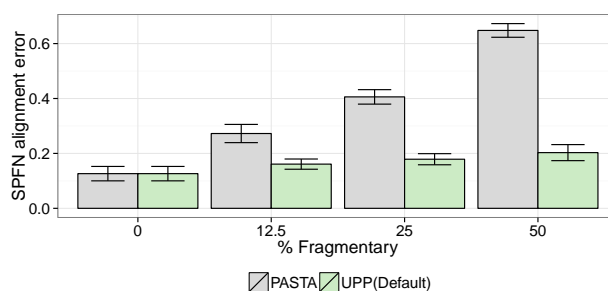


(b) Alignment SPFP error

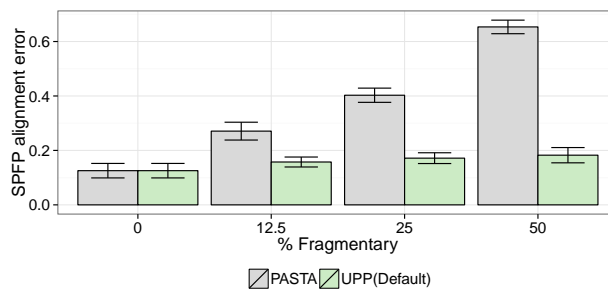
Figure B30: **SPFN and SPFP alignment error of UPP and PASTA on the HomFam datasets.** UPP(Default) uses a backbone of size 1000. Backbone sequences were filtered by only selecting from sequences within 75% to 125% in length of the median sequence length of the seed sequences. Standard error bars are shown.



(a) Average alignment error



(b) Alignment SPFN error



(c) Alignment SPFP error

Figure B31: **Alignment error of PASTA and UPP on the fragmentary 1000M2 datasets.** UPP(Default) uses a backbone size equal to the total number of full-length sequences. Backbone sequences were filtered by only selecting from sequences within 75% to 125% length of the typical 1000M2 length (1000 bps). Fragments had an average length of 500 bps, roughly one half the length of an average full length sequence from 1000M2. Note that at 0% fragmentation, UPP(Default) is identical to PASTA. Standard error bars are shown. Averages are computed over 5 replicates per dataset.

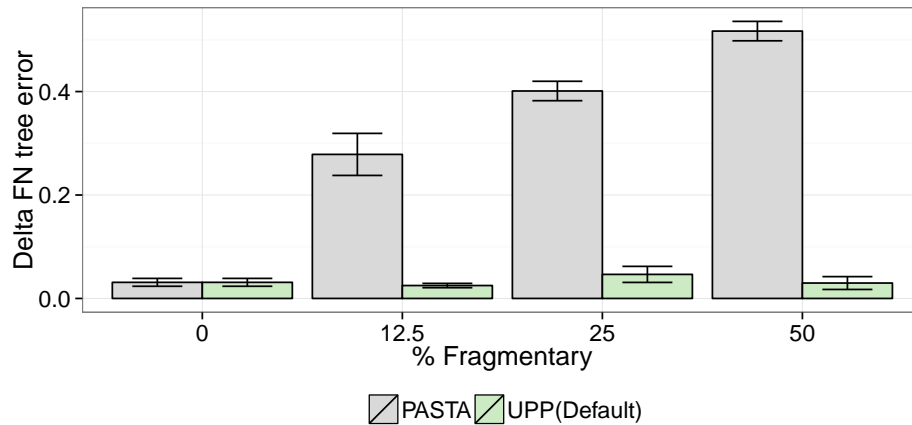


Figure B32: **Delta FN Tree error of UPP and PASTA on the fragmentary 1000M2 datasets.** UPP(Default) uses a backbone size equal to the total number of full-length sequences. Backbone sequences were filtered by only selecting from sequences within 75% to 125% length of the typical 1000M2 length (1000 bps). ML trees were estimated using FastTree under GTR. Fragments had an average length of 500 bps, roughly one half the length of an average full length sequence from 1000M2. Note that at 0% fragmentation, UPP(Default) is identical to PASTA. Standard error bars are shown. Averages are computed over 5 replicates per dataset.

B2.8 Backbone and final alignment error.

We examined the alignment error of the backbone alignment and the resulting alignment error of the alignment generated by UPP using the HMM Family technique, or using MAFFT-profile “add” or MAFFT-profile “addfragments” (Fig. B33).

We found that the backbone alignment error was statistically significantly correlated to the alignments generated by the UPP pipeline (Pearson’s correlation coefficient 0.897; p-value of 2.292e-10). We also found that alignment errors on alignments generated by the HMM Family technique (UPP(Default)) were closer to the original backbone alignment error than when they were generated using MAFFT-profile (root mean squared difference in alignment error of 0.020 for UPP(Default) versus 0.024 and 0.051 for UPP using MAFFT-profile “addfragments” and MAFFT-profile “add”, respectively).

This result shows that the HMM Family technique best preserves the alignment accuracy of the original backbone alignment and can be used as a way to scale existing alignments methods to ultra-large datasets.

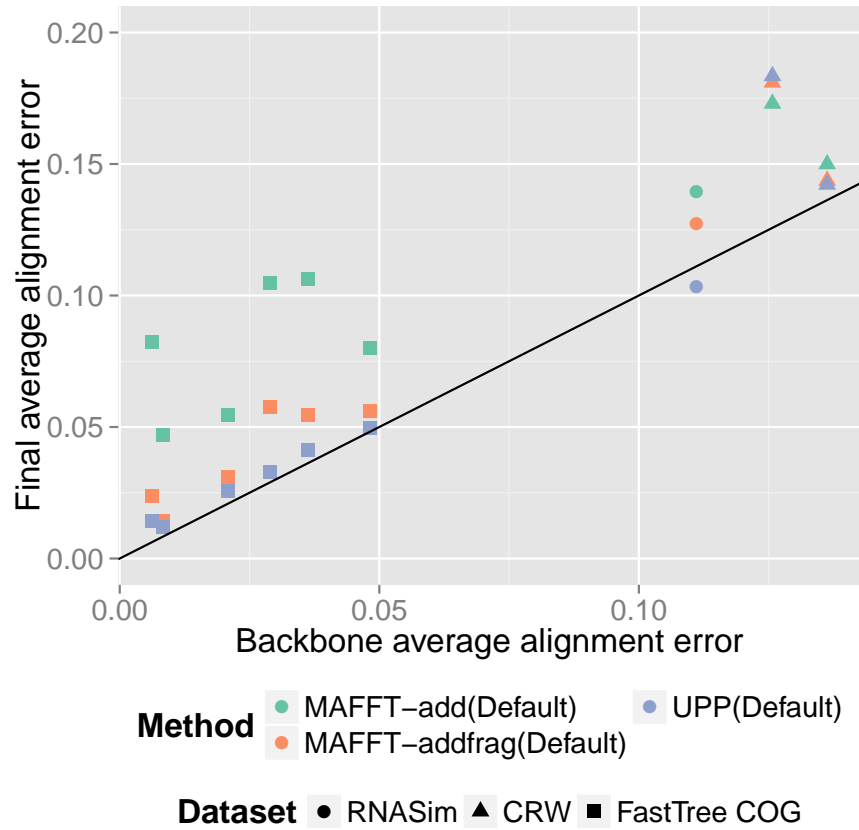
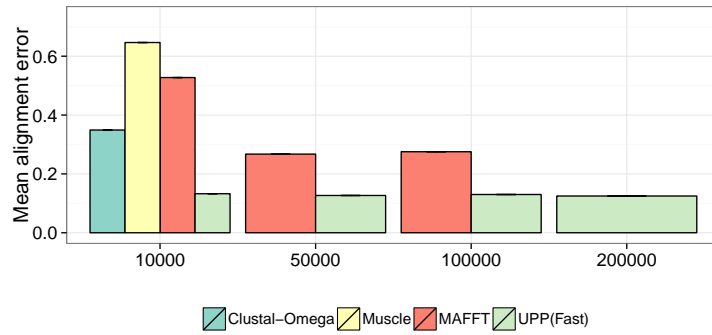


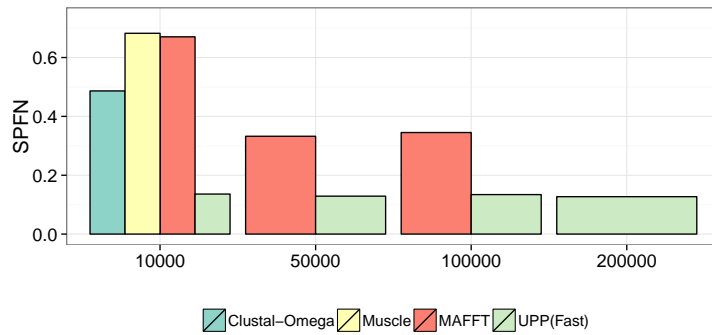
Figure B33: **Comparison of initial backbone error and final UPP alignment error, using PASTA backbones of size 1000.** Each point represents the alignment error for a specific method on a specific dataset. Points below the line represent alignment methods that have a lower alignment error relative to the backbone alignment. Points above the line represent alignment methods that have a higher alignment error relative to the backbone alignment. The Pearson’s correlation coefficient for the backbone alignment error versus the final alignment error for the entire collection of points is 0.897 and is statistically significantly correlated (p-value of 2.292e-10; Pearson’s product-moment correlation test).

B2.9 Results on full-length datasets

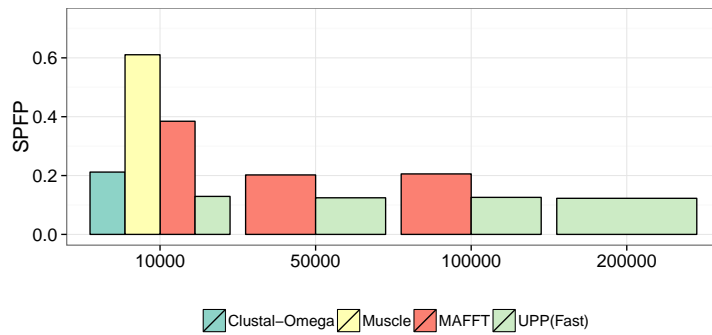
We present results on all the full-length sequence datasets.



(a) Average alignment error

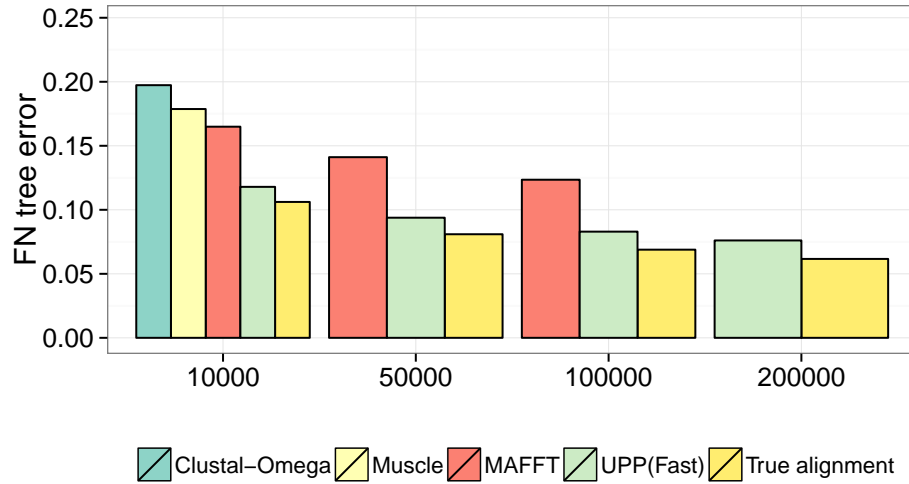


(b) Alignment SPFN error

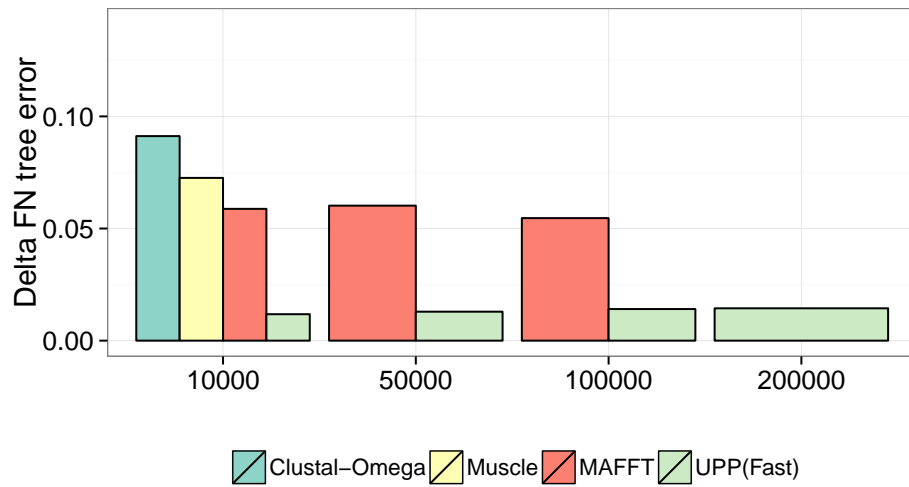


(c) Alignment SPFP error

Figure B34: Alignment error on the RNASim datasets. MAFFT is run under the default options on the RNASim 10K and 50K datasets and under “PartTree” for the RNASim 100K dataset. UPP(Fast) use a backbone size of 100.



(a) FN tree error



(b) Delta FN tree error

Figure B35: **Tree error on the RNASim datasets.** ML trees were estimated using FastTree under GTR. MAFFT is run under the default options on the RNASim 10K and 50K datasets and under “PartTree” for the RNASim 100K dataset. UPP(Fast) use a backbone size of 100.

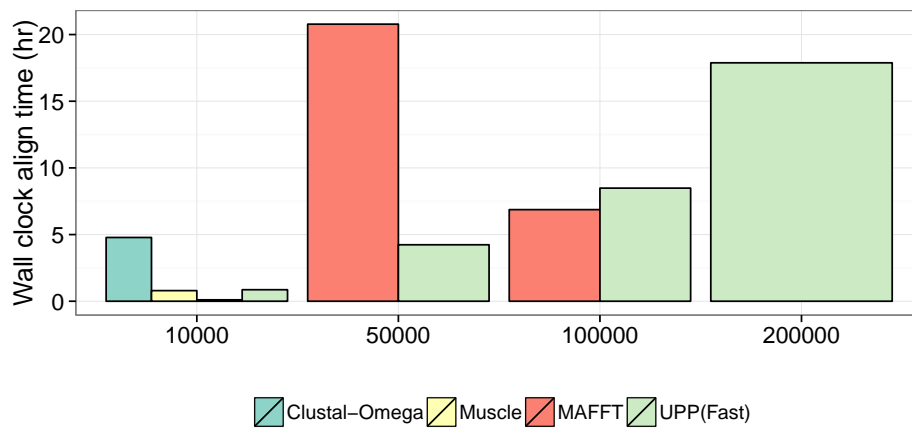


Figure B36: **Wall clock alignment time (hrs) on the RNASim datasets.** All methods were run on a machine with 12 CPUs and 24 GB of memory. MAFFT is run under the default options on the RNASim 10K and 50K datasets and under “PartTree” for the RNASim 100K dataset; the difference in how MAFFT was run explains the difference in time between 50K and 100K sequences. UPP(Fast) uses a backbone size of 100.

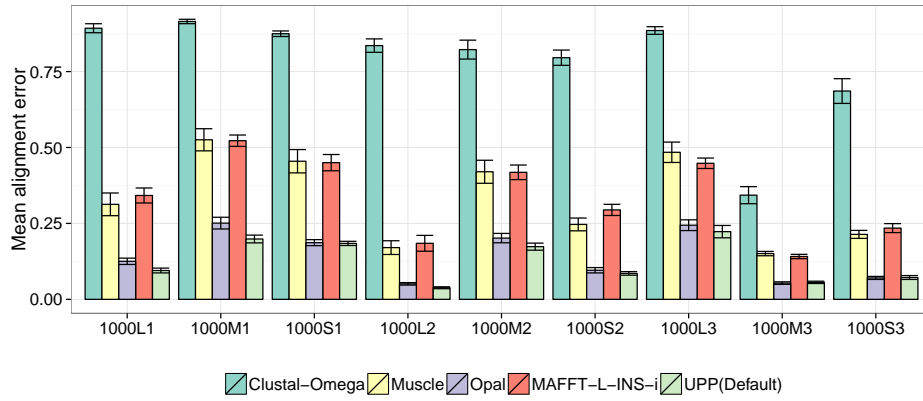
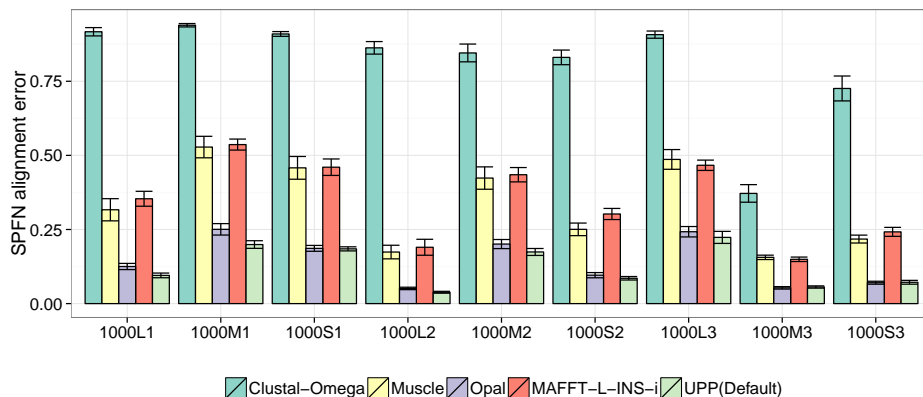
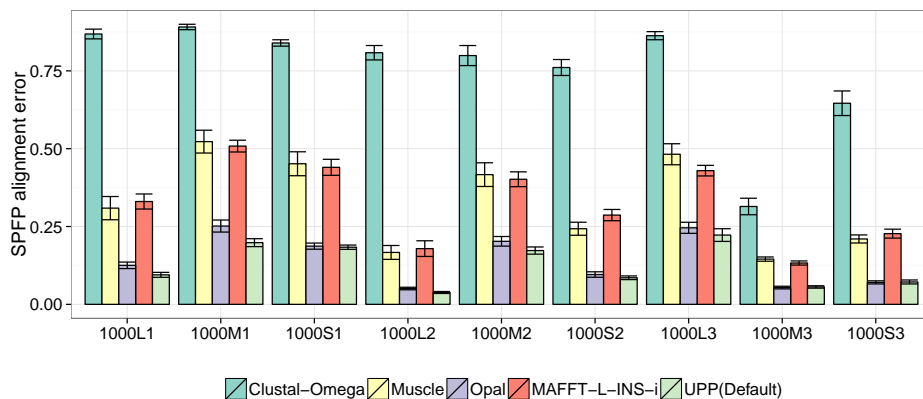


Figure B37: **Average alignment error of different methods on the hardest 1000-taxon datasets.** Standard error bars are shown. Averages are computed over 20 replicates per dataset. UPP(Default) is identical to PASTA on these datasets.

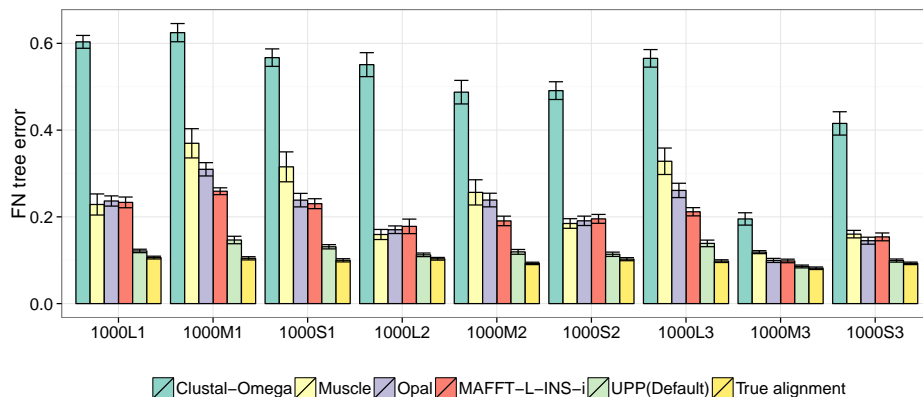


(a) Alignment SPFN error

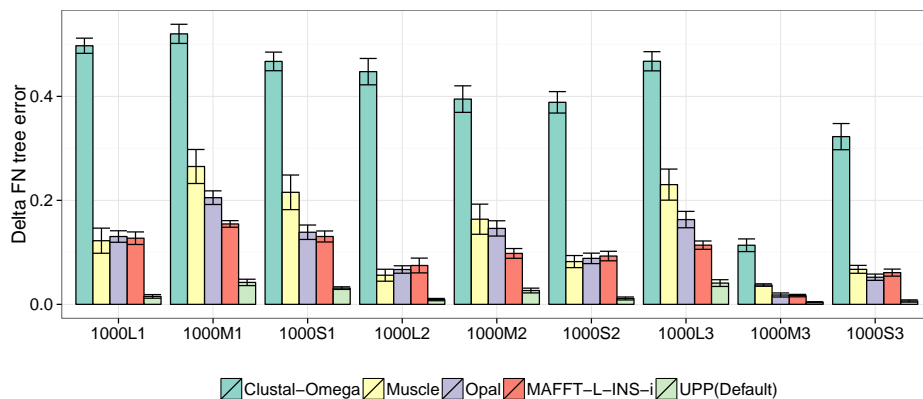


(b) Alignment SPFP error

Figure B38: **SPFN and SPFP alignment error of different methods on the hardest 1000-taxon datasets.** Standard error bars are shown. Averages are computed over 20 replicates per dataset. UPP(Default) is identical to PASTA on these datasets.



(a) FN tree error



(b) Delta FN tree error

Figure B39: **Tree error of different methods on the hardest 1000-taxon datasets.** ML trees were estimated using FastTree under GTR. Standard error bars are shown. Averages are computed over 20 replicates per dataset. UPP(Default) is identical to PASTA on these datasets.

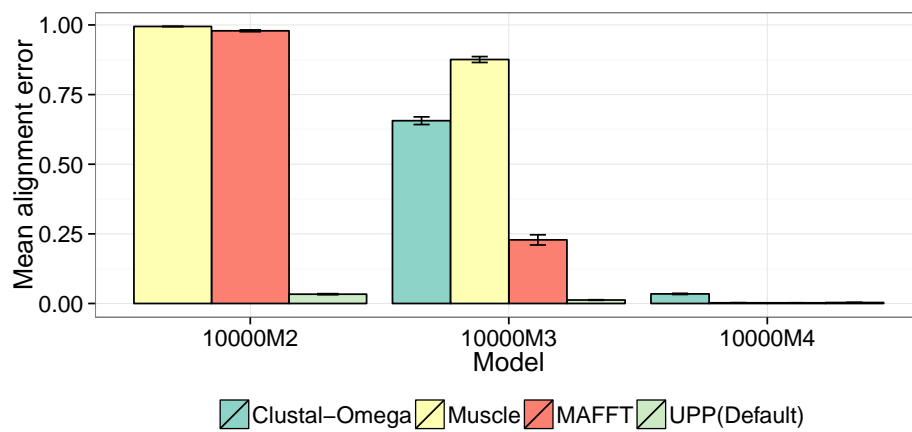
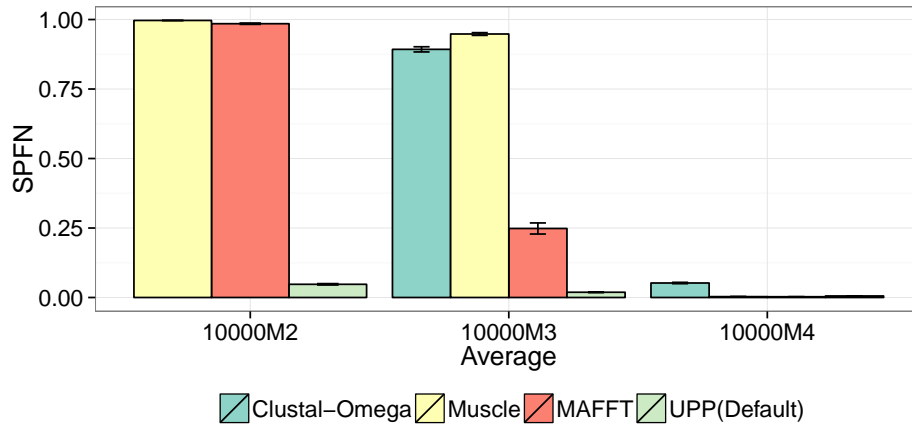
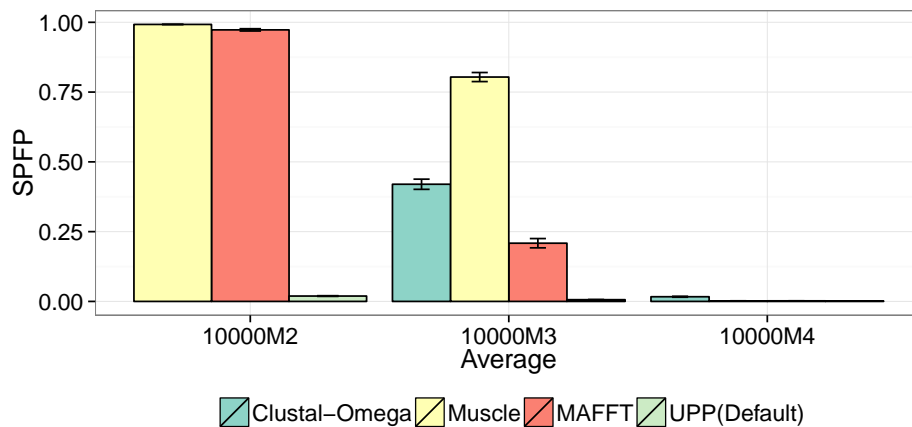


Figure B40: **Average alignment error on the Indelible datasets.** Clustal-Omega was unable to generate an alignment on the 10000M2 dataset (terminated with error message). MAFFT was run under the default options. Standard error bars are shown. Averages are computed over 10 replicates per model condition.

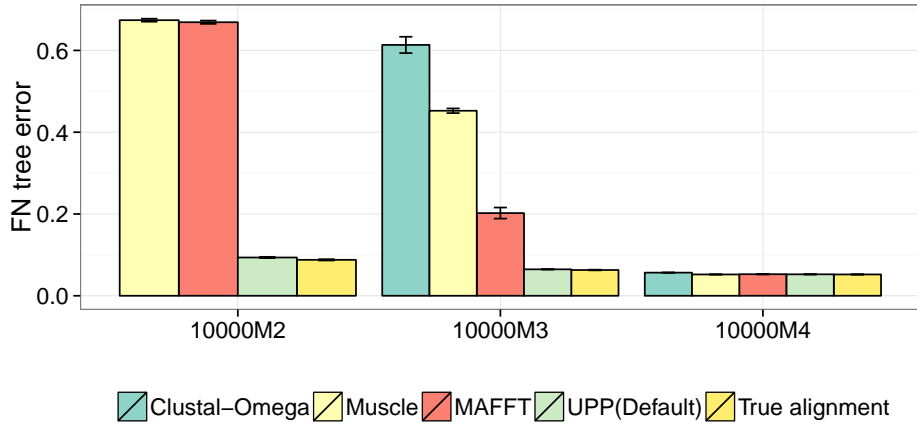


(a) Alignment SPFN error

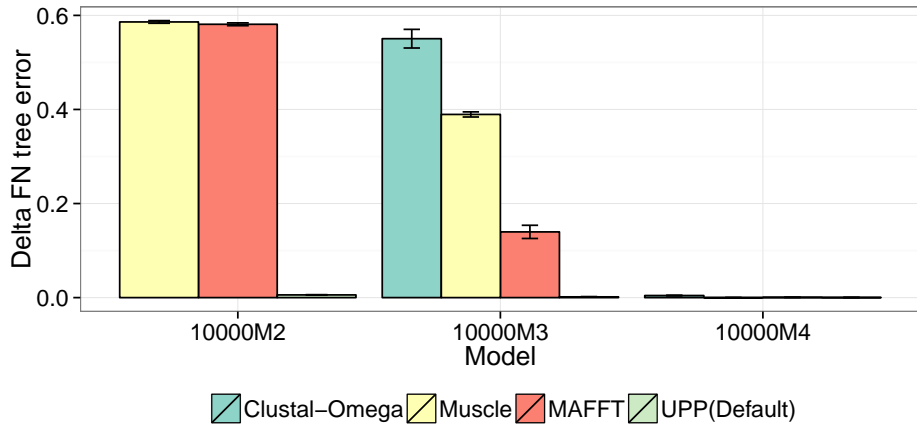


(b) Alignment SPFP error

Figure B41: **SPFN and SPFP alignment error on the Indelible datasets.** Clustal-Omega was unable to generate an alignment on the 10000M2 dataset (terminated with error message). MAFFT was run under the default options. Standard error bars are shown. Averages are computed over 10 replicates per model condition.



(a) FN tree error



(b) Delta FN tree error

Figure B42: **Tree error on the Indelible datasets.** Clustal-Omega was unable to generate an alignment on the 10000M2 dataset (terminated with error message). MAFFT was run under the default options. Standard error bars are shown. Averages are computed over 10 replicates per model condition.

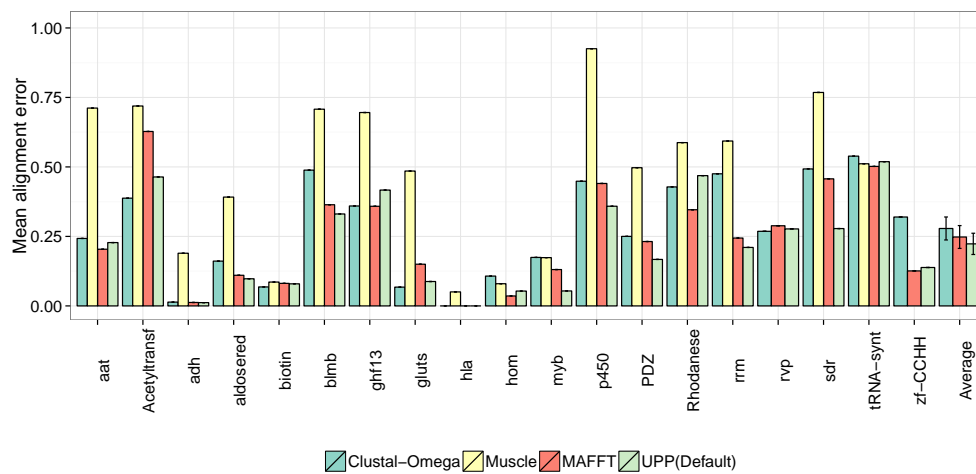
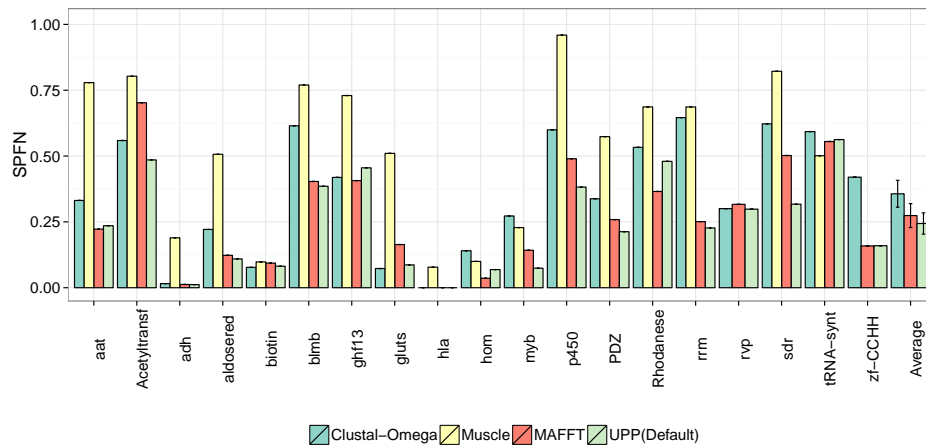
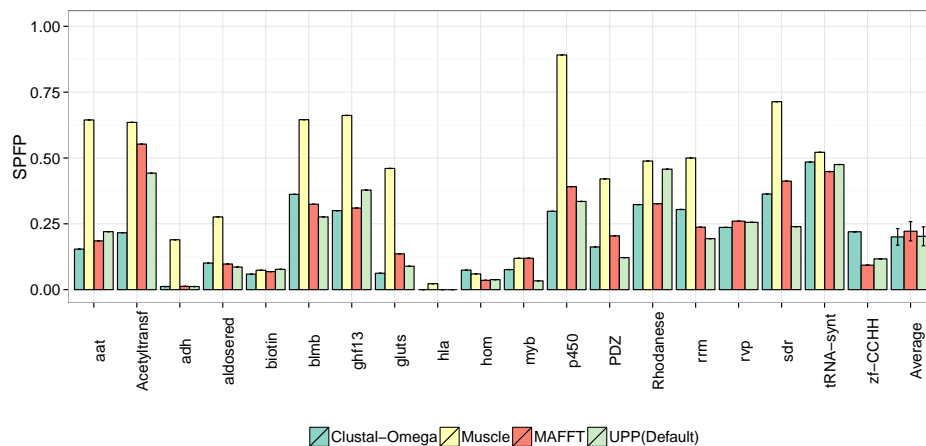


Figure B43: **Average alignment error on the HomFam datasets.** On the two largest HomFam datasets (zf-CCHH and rvp), MUSCLE terminated with a “segfault” error and was unable to produce an alignment. Thus, average alignment error for MUSCLE is excluded from the results. MAFFT is run under the default options.



(a) Alignment SPFN error



(b) Alignment SPFP error

Figure B44: **Alignment SPFN and SPFP error on the HomFam datasets.** On the two largest HomFam datasets (zf-CCHH and rvp), MUSCLE terminated with a “segfault” error and was unable to produce an alignment. Thus, average alignment error for MUSCLE is excluded from the results. MAFFT is run under the default options.

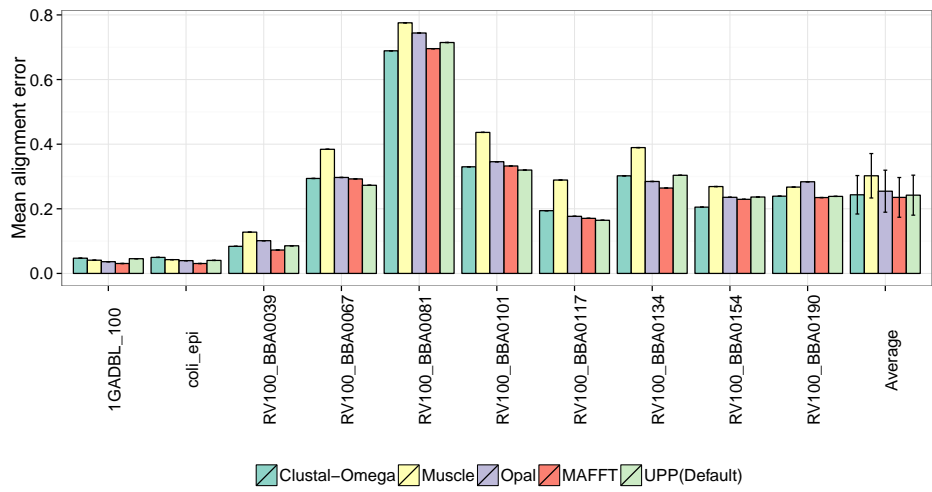
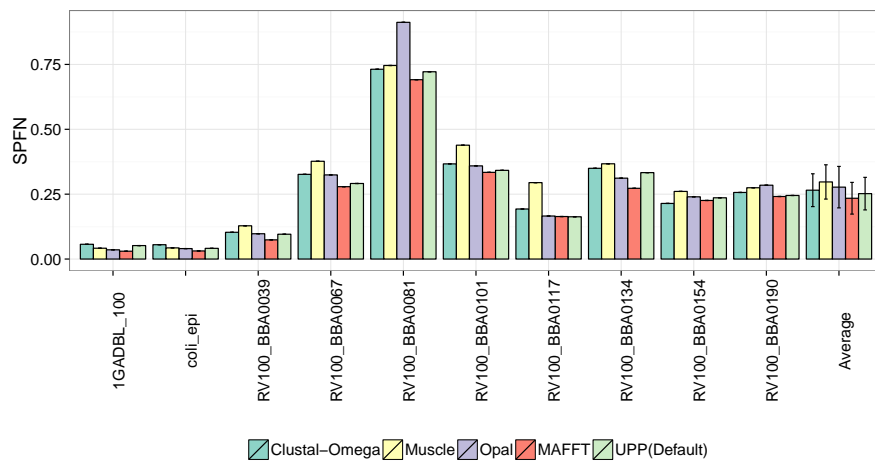
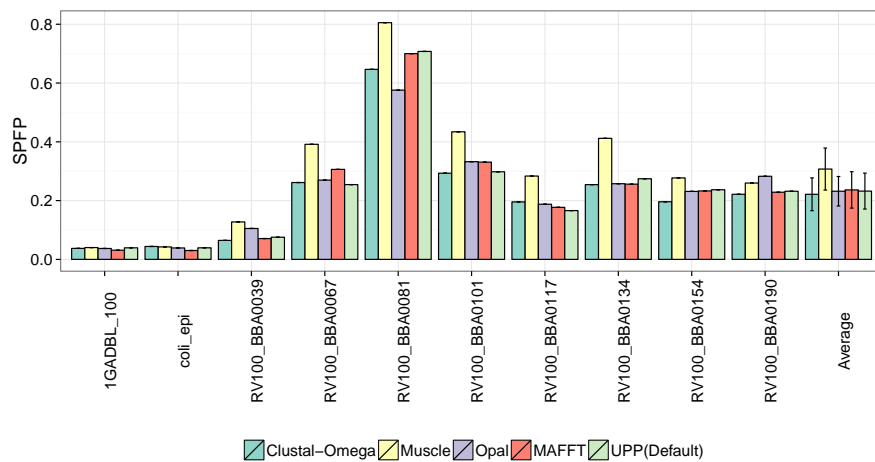


Figure B45: Average alignment errors on the ten large protein datasets with full alignments. MAFFT is run under “L-INS-I”



(a) Alignment SPFN error



(b) Alignment SPFP error

Figure B46: Alignment SPFN and SPFP error on the ten large protein datasets with full alignments. MAFFT is run under “L-INS-I”

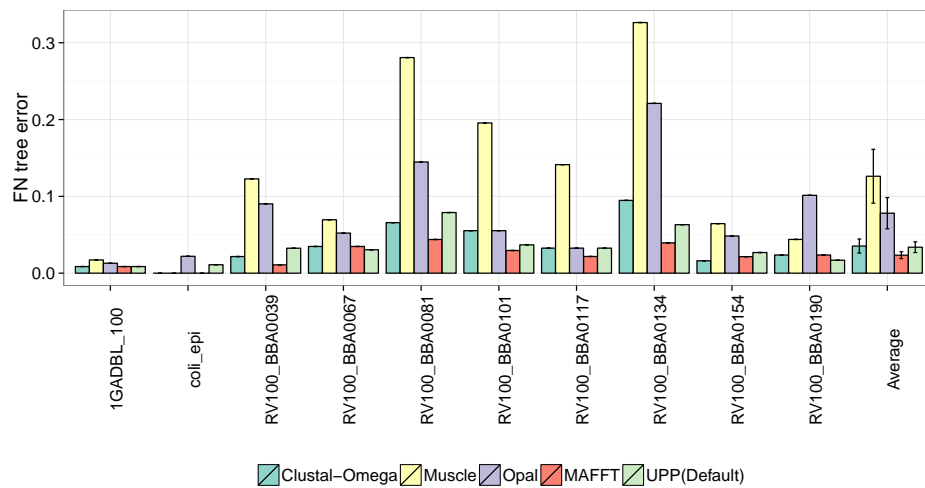


Figure B47: **Tree error rates on the large protein datasets with full alignments.** MAFFT is run under “L-INS-I”. ML trees were estimated using RAxML under JTT.

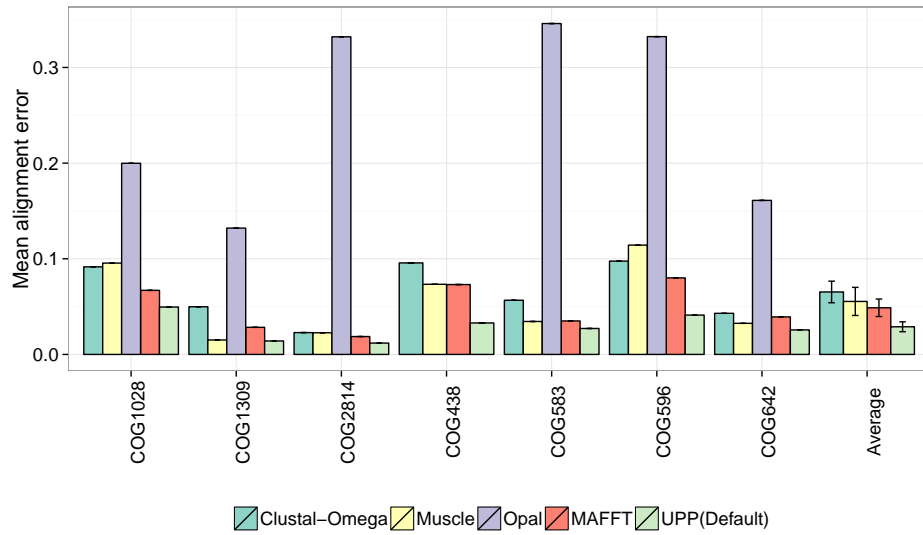
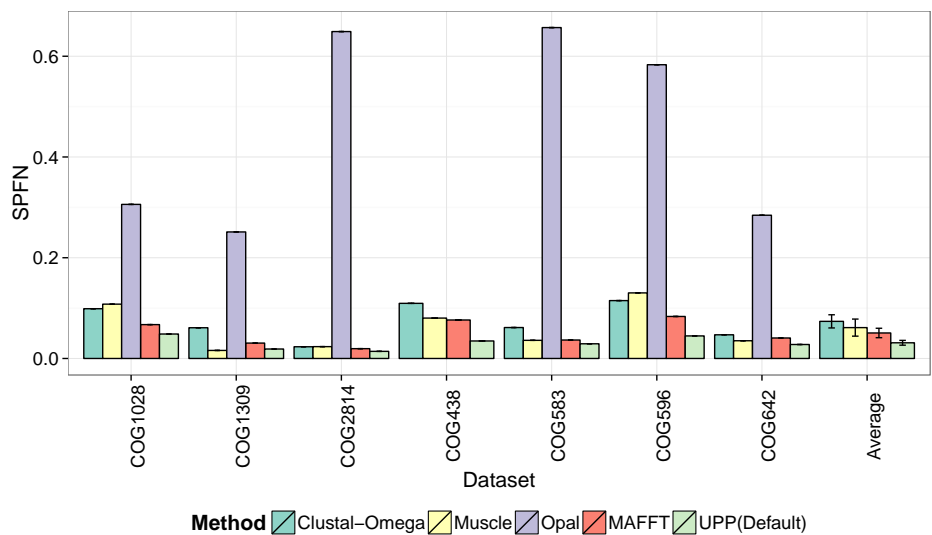
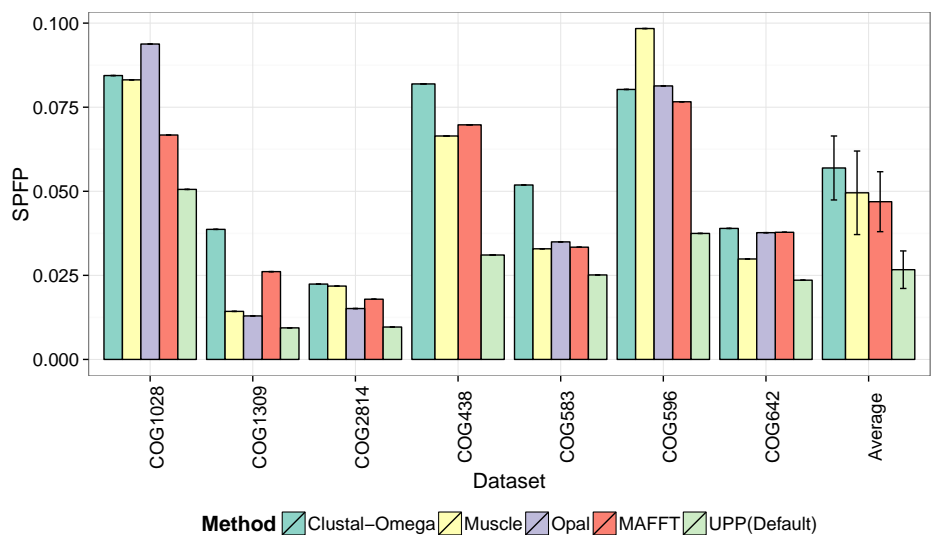


Figure B48: **Average alignment error on the FastTree COG datasets.** MAFFT is run under the default options. Opal failed to generate an alignment on COG438, and thus is excluded from the average results.

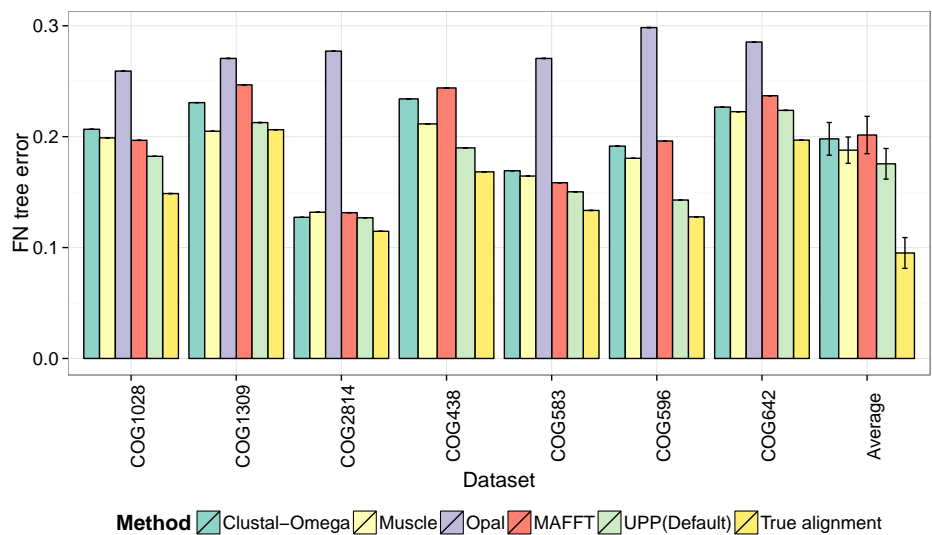


(a) Alignment SPFN error

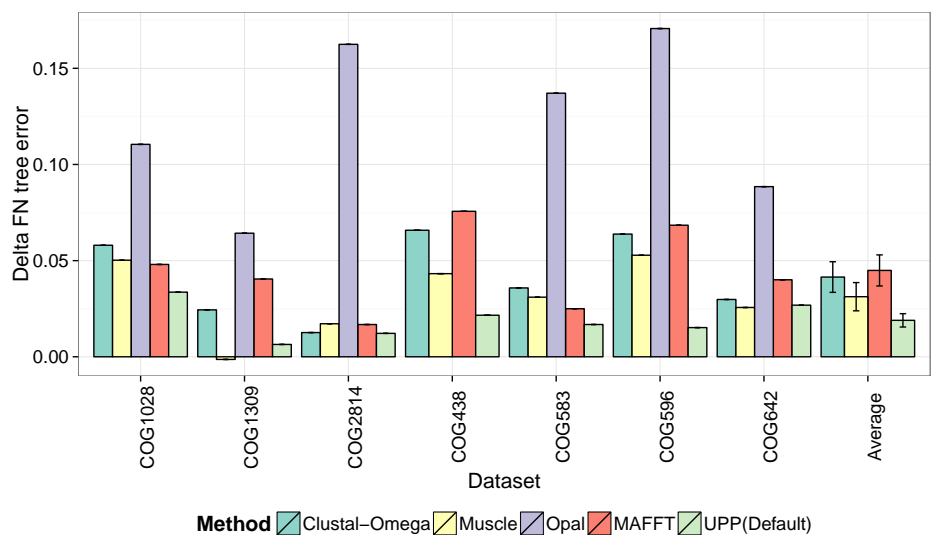


(b) Alignment SPFP error

Figure B49: **SPFN and SPFP alignment error on the simulated Fast-Tree COG datasets.** MAFFT is run under the default options. Opal failed to generate an alignment on COG438, and thus is excluded from the average results.

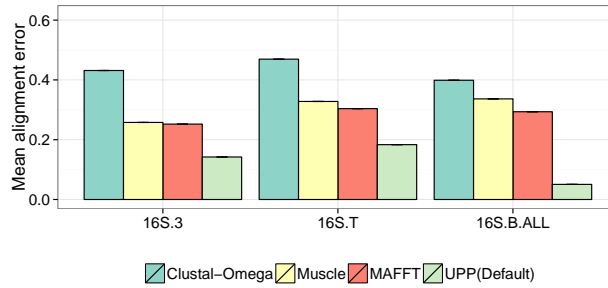


(a) FN tree error

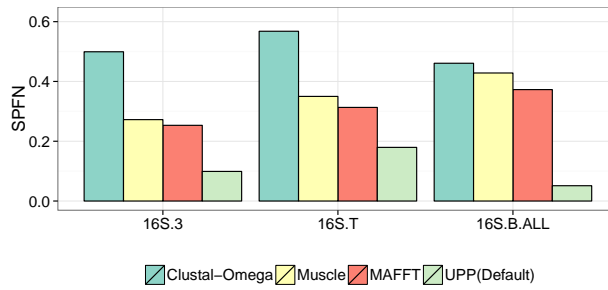


(b) Delta FN tree error

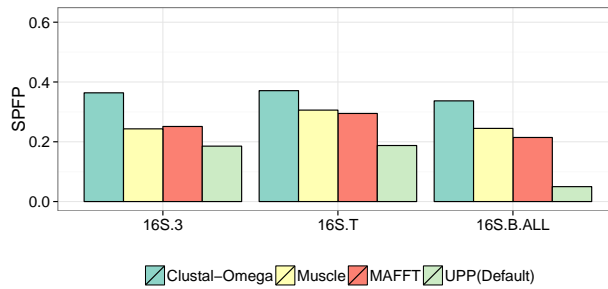
Figure B50: **Tree error rates on the simulated FastTree COG datasets.** ML trees were estimated using FastTree under JTT. MAFFT is run under the default options. Opal failed to generate an alignment on COG438, and thus is excluded from the average results.



(a) Average alignment error



(b) Alignment SPFN error



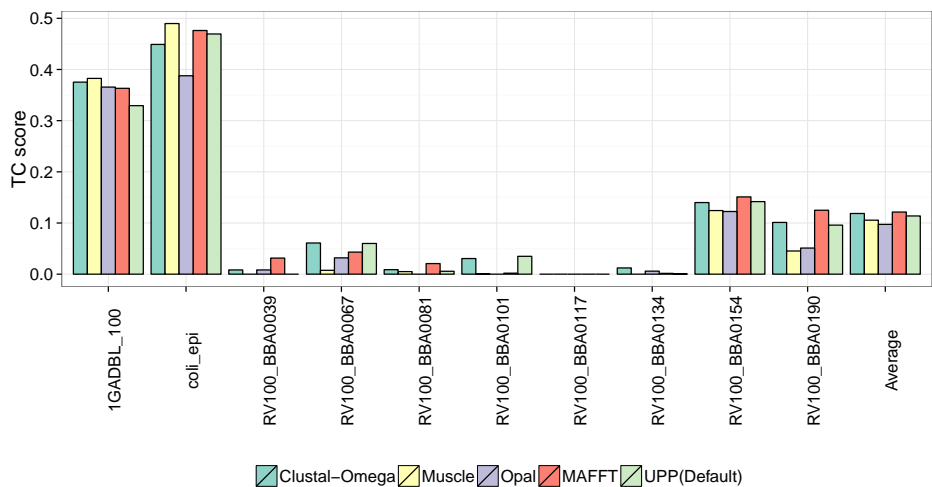
(c) Alignment SPFP error

Figure B51: **Alignment error rates on the CRW datasets.** MAFFT is run under the default options on the 16S.T and 16S.3 datasets and under “PartTree” on the 16S.B.ALL dataset. UPP results are based on the first iteration of UPP.

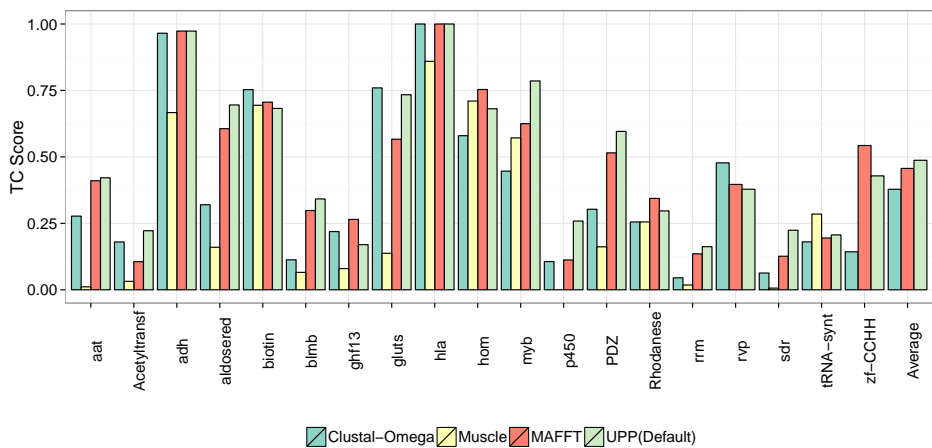
B2.10 TC Scores

We report total column (TC) scores on the protein datasets with structurally based reference alignments (Fig. B52). The TC score is the proportion of columns in the reference alignment that are recovered in the estimated alignment.

UPP, MAFFT, and Clustal-Omega have similar TC scores on the ten large AA datasets, followed closely by Opal and Muscle. On the HomFam datasets, the differences between the methods are more clear with UPP having the best TC score, followed by MAFFT, then Clustal-Omega.



(a) TC scores on the ten large biological datasets with full alignments

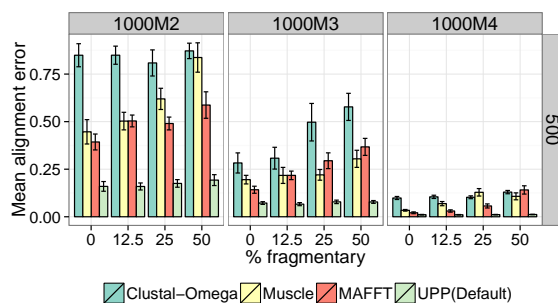


(b) TC scores on the Homfam datasets

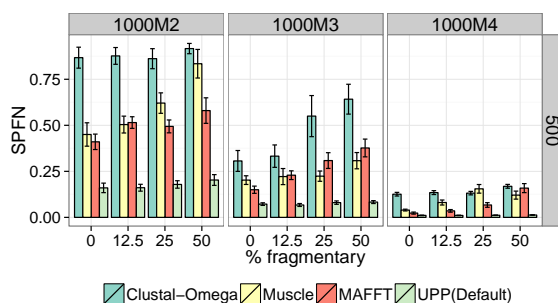
Figure B52: **TC (Total Column) scores of methods on the biological AA datasets.** MAFFT is run under “L-INS-i” option on the ten large AA datasets and under the default option on the HomFam datasets.

B2.11 Results on fragmentary datasets

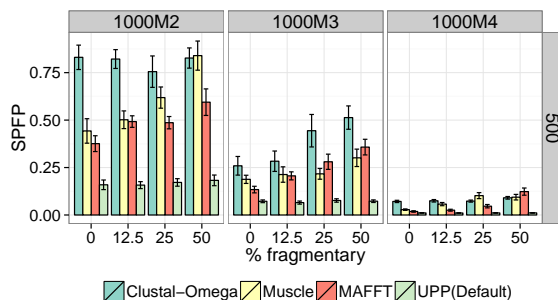
We present results on the individual fragmentary datasets.



(a) Average alignment error

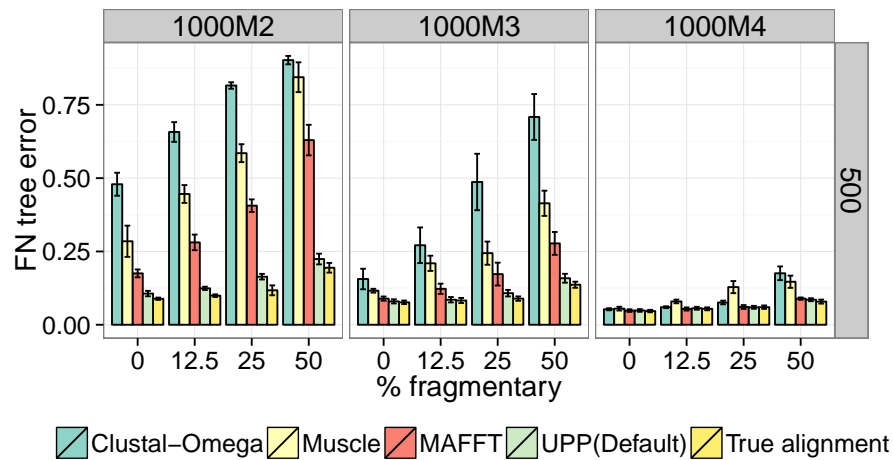


(b) Alignment SPFN error

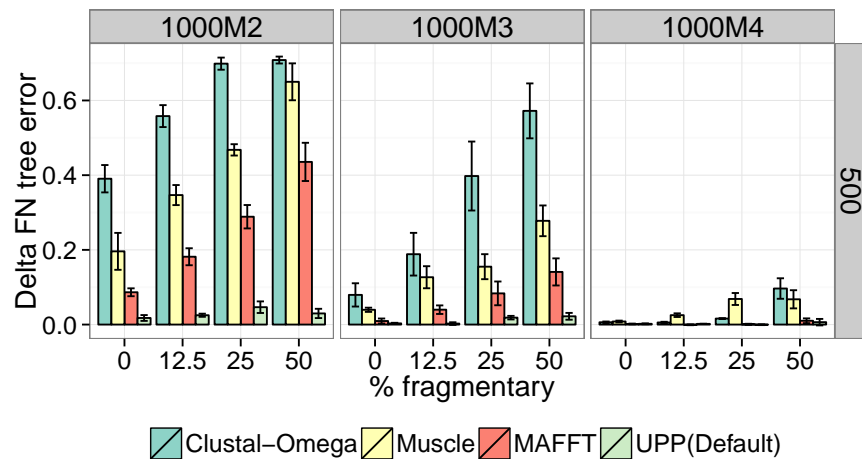


(c) Alignment SPFP error

Figure B53: **Alignment error rates on the fragmentary 1000-taxon model conditions.** We show alignment error rates for different methods on the 1000M2, 1000M3, and 1000M4 datasets, varying the percentage of fragmentary sequences, each with an average length of 500 sites (i.e., approximately half the average sequence length). MAFFT is run under the “L-INS-i”. Standard error bars are shown. Averages are computed over 5 replicates per dataset.

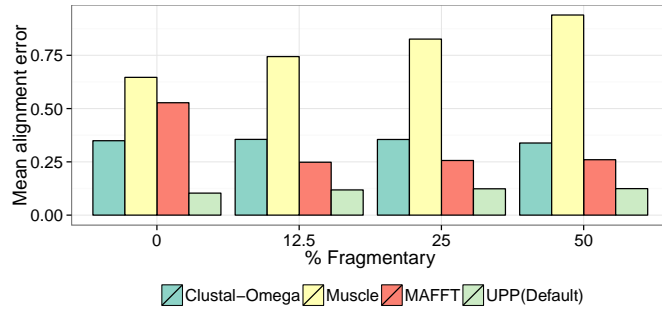


(a) FN tree error

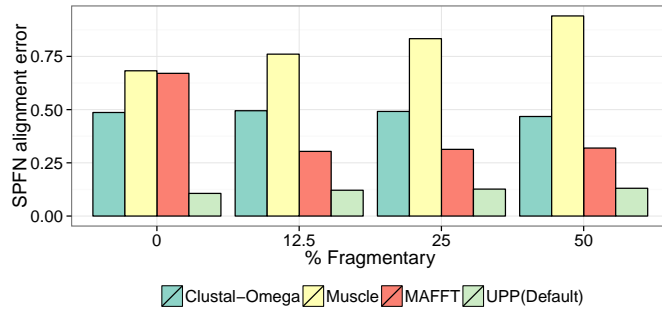


(b) Delta FN tree error

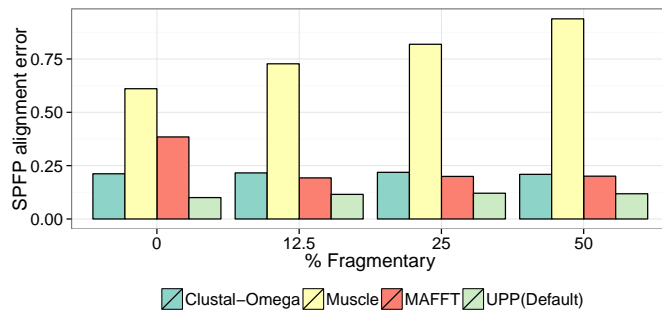
Figure B54: **Tree error rates on the fragmentary 1000-taxon datasets.** We show tree error rates for different methods on the 1000M2, 1000M3, and 1000M4 datasets, varying the percentage of fragmentary sequences, each with an average length of 500 sites (i.e., approximately half the average sequence length). MAFFT is run under the “L-INS-i”. ML trees were estimated using FastTree under GTR. Standard error bars are shown. Averages are computed over 5 replicates per dataset.



(a) Average alignment error

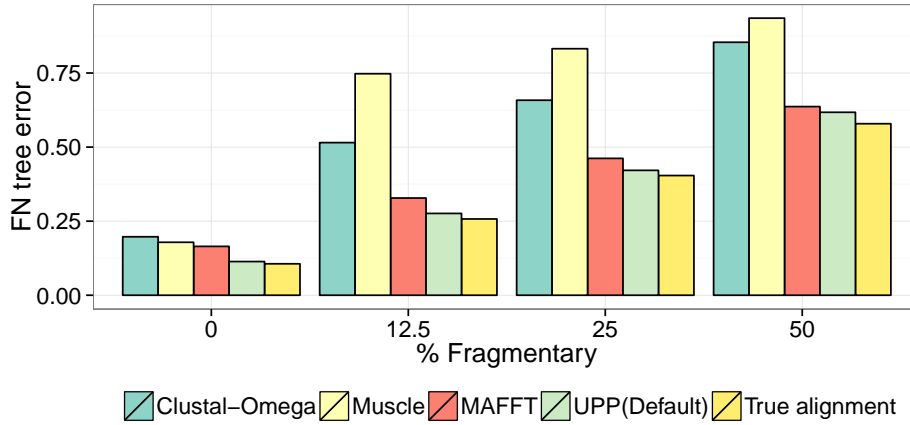


(b) Alignment SPFN error

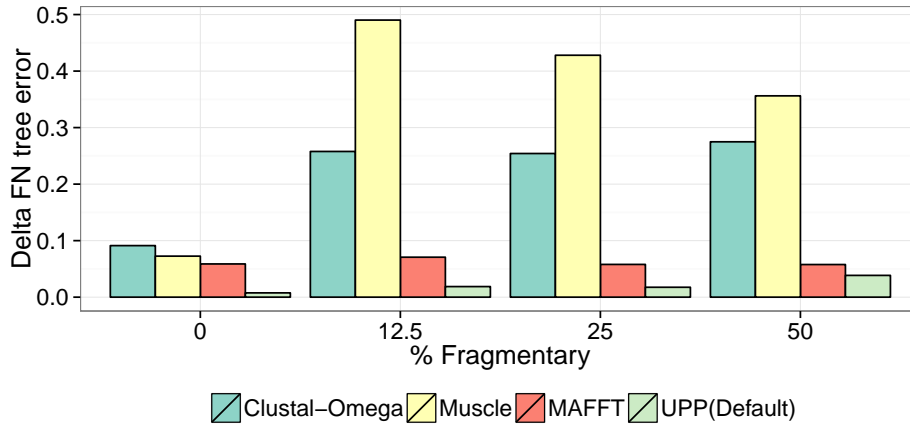


(c) Alignment SPFP error

Figure B55: **Alignment error rates on the fragmentary RNASim 10K datasets.** We show alignment error rates for for Clustal-Omega, MUSCLE, MAFFT-Default, and UPP(Default) on the RNASim 10K datasets, varying the percentage of fragmentary sequences, each with an average length of 500 sites. (i.e., approximately one third the average sequence length).



(a) FN tree error



(b) Delta FN tree error

Figure B56: **Tree error rates on the fragmentary RNASim 10K datasets.** We show tree error rates for Clustal-Omega, MUSCLE, MAFFT-Default, and UPP(Default) on the RNASim 10K datasets, varying the percentage of fragmentary sequences, each with an average length of 500 sites. (i.e., approximately one third the average sequence length). ML trees were estimated using FastTree under GTR.

Bibliography

- [1] C. Afrasiabi, B. Samad, D. Dineen, C. Meacham, and K. Sjolander. The PhyloFacts FAT-CAT web server: ortholog identification and function prediction using fast approximate tree classification. *Nucl. Acids Res.*, 41(W1):W242–W248, 2013.
- [2] D. J. Aldous. Stochastic models and descriptive statistics for phylogenetic trees, from Yule to today. *Statist. Sci.*, 16:23–34, 2001.
- [3] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215:403–410, 1990.
- [4] R. I. Amann, W. Ludwig, and K. H. Schleifer. Phylogenetic identification and in situ detection of individual microbial cells without cultivation. *Microbiological reviews*, 59:143–169, 1995.
- [5] A. L. Bazinet and M. P. Cummings. A comparative evaluation of sequence classification programs. *BMC bioinformatics*, 13(1):92, Jan. 2012.
- [6] S. A. Berger, D. Krompass, and A. Stamatakis. Performance, Accuracy, and Web Server for Evolutionary Placement of Short Sequence Reads

- under Maximum Likelihood. *Systematic Biology*, 60(3):291–302, May 2011.
- [7] S. A. Berger and A. Stamatakis. Aligning short reads to reference alignments and trees. *Bioinformatics*, 27(15):2068–2075, 2011.
- [8] P. Bonizzoni and G. D. Vedova. The complexity of multiple sequence alignment with SP-score that is a metric. *Theoretical Computer Science*, 259:63–79, 2001.
- [9] A. Bouchard-Côté and M. I. Jordan. Evolutionary inference via the poisson indel process. *Proceedings of the National Academy of Sciences*, 110(4):1160–1166, 2013.
- [10] D. Brown, N. Krishnamurthy, and K. Sjölander. Automated protein subfamily identification and classification. *PLoS computational biology*, 3(8), 2007.
- [11] J. Cannone, S. Subramanian, M. Schnare, J. Collett, L. D’Souza, Y. Du, B. Feng, N. Lin, L. Madabusi, K. Muller, N. Pande, Z. Shang, N. Yu, and R. Gutell. The comparative rna web (crw) site: an online database of comparative sequence and structure information for ribosomal, intron, and other rnas. *BMC Bioinformatics*, 3(1):2, 2002.
- [12] T. Clemen. Combining forecasts : A review and annotated. *International Journal of Forecasting*, 5:559–583, 1989.

- [13] C. B. Do and K. Katoh. Protein multiple sequence alignment. *Methods in molecular biology (Clifton, N.J.)*, 484:379–413, 2008.
- [14] S. Eddy. A new generation of homology search tools based on probabilistic inference. *Genome Inform*, 23:205211, 2009.
- [15] S. R. Eddy. Profile Hidden Markov Models. *Bioinformatics*, 14:755–763, 1998.
- [16] R. C. Edgar. MUSCLE: a multiple sequence alignment method with reduced time and space complexity. *BMC Bioinformatics*, 5:113, Aug. 2004.
- [17] J. Eisen and C. Fraser. Phylogenomics: intersection of evolution and genomics. *Science*, 300(5626):1706–1707, 2003.
- [18] J. A. Eisen. Phylogenomics: improving functional predictions for uncharacterized genes by evolutionary analysis. *Genome Research*, 8(3):163–167, Mar. 1998.
- [19] J. Felsenstein. *Inferring Phylogenies*. Sinauer Associates, Sunderland, Massachusetts, 2003.
- [20] R. D. Finn, J. Mistry, J. Tate, P. Coghill, A. Heger, J. E. Pollington, O. L. Gavin, P. Gunasekaran, G. Ceric, K. Forslund, L. Holm, E. L. L. Sonnhammer, S. R. Eddy, and A. Bateman. The Pfam protein families database. *Nucleic acids research*, 38(Database issue):D211–22, Jan. 2010.

- [21] W. M. Fitch, E. M. Peterson, and L. M. de la Maza. Phylogenetic analysis of the outer-membrane-protein genes of Chlamydiae, and its implication for vaccine development. *Molecular Biology and Evolution*, 10:892–913, 1993.
- [22] W. Fletcher and Z. Yang. Indelible: A flexible simulator of biological sequence evolution. *Molecular Biology and Evolution*, 26(8):1879–1888, 2009.
- [23] W. Fletcher and Z. Yang. The effect of insertions, deletions, and alignment errors on the branch-site test of positive selection. *Mol Biol Evolution*, 27(10):2257–2267, 2010.
- [24] W. Gerlach and J. Stoye. Taxonomic classification of metagenomic shotgun sequences with CARMA3. *Nucleic acids research*, 39:e91, 2011.
- [25] G. B. Gloor, L. C. Martin, L. M. Wahl, and S. D. Dunn. Mutual information in protein multiple sequence alignments reveals two classes of coevolving positions. *Biochemistry*, 44(19):7156–65, May 2005.
- [26] S. Guo, L.-S. Wang, and J. Kim. RNA evolution simulator: User manual version 1.0.
- [27] D. Haussler, S. O’Brien, and O. Ryder. Genome 10K. <https://genome10k.soe.ucsc.edu>.
- [28] M. Holder and P. O. Lewis. Phylogeny estimation: traditional and Bayesian approaches. *Nature reviews. Genetics*, 4:275–284, 2003.

- [29] J. Huelsenbeck. Performance of phylogenetic methods in simulation. *Systematic biology*, 44(1):17–48, 1995.
- [30] S. Iantomo, K. Gori, N. Goldman, M. Gil, and C. Dessimoz. Who watches the watchmen? An appraisal of benchmarks for multiple sequence alignment. In *Multiple Sequence Alignment Methods*, volume 1079 of *Methods in Molecular Biology*, pages 59–73. Springer, 2014.
- [31] R. Jacobs, M. Jordan, S. Nowlan, and G. Hinton. Adaptive mixtures of local experts. *Neural computation*, 3:79–87, 1991.
- [32] E. Jarvis, S. Mirarab, and *et al.* Whole genome analyses resolve the early branches in the tree of life of modern birds. *Science*, 2014. Submitted, main paper of the avian phylogenomics project.
- [33] G. Jordan and N. Goldman. The effects of alignment error and alignment filtering on the sitewise detection of positive selection. *Mol Biol Evolution*, 29(4):1125–1139, 2012.
- [34] K. Karplus. SAM-T08, HMM-based protein structure prediction. *Nucleic acids research*, 37:W492–W497, 2009.
- [35] K. Katoh and M. C. Frith. Adding unaligned sequences into an existing alignment using MAFFT and LAST. *Bioinformatics (Oxford, England)*, 28(23):3144–3146, Sept. 2012.

- [36] K. Katoh, K.-i. Kuma, H. Toh, and T. Miyata. MAFFT version 5: improvement in accuracy of multiple sequence alignment. *Nucleic acids research*, 33(2):511–8, Jan. 2005.
- [37] K. Katoh, K. Misawa, K.-i. Kuma, and T. Miyata. MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic acids research*, 30(14):3059–66, July 2002.
- [38] K. Katoh and H. Toh. PartTree: an algorithm to build an approximate tree from a large number of unaligned sequences. *Bioinformatics*, 23:372–374, 2007.
- [39] D. C. Koboldt, K. M. Steinberg, D. E. Larson, R. K. Wilson, and E. R. Mardis. The next-generation sequencing revolution and its impact on genomics. *Cell*, 155:27–38, 2013.
- [40] L. B. Koski and G. B. Golding. The closest BLAST hit is often not the nearest neighbor. *J Mol Evol*, 52(6):540–2, 2001. 0022-2844 (Print) Letter.
- [41] C.-S. Ku and D. H. Roukos. From next-generation sequencing to nanopore sequencing technology: paving the way to personalized genomic medicine. *Expert Rev. Med. Devices*, 10(1):1–6, 2013.
- [42] M. K. Kuhner and J. Felsenstein. A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. *Molecular Biology and Evolution*, 11(3):459–68, May 1994.

- [43] B. Liu, T. Gibbons, M. Ghodsi, and M. Pop. MetaPhyler: Taxonomic profiling for metagenomic sequences. In *Bioinformatics and Biomedicine (BIBM), 2010 IEEE International Conference on*, pages 95–100. IEEE, 2011.
- [44] B. Liu, T. Gibbons, M. Ghodsi, T. Treangen, and M. Pop. Accurate and fast estimation of taxonomic profiles from metagenomic shotgun sequences. *BMC Genomics*, 12(Suppl 2):S4, Jan. 2011.
- [45] K. Liu, C. Linder, and T. Warnow. Multiple sequence alignment: a major challenge to large-scale phylogenetics. *PLoS Currents*, page RRN1198, November [revised 2011 March 18] 2010. Version 2.
- [46] K. Liu, S. Raghavan, S. Nelesen, C. R. Linder, and T. Warnow. Rapid and accurate large-scale coestimation of sequence alignments and phylogenetic trees. *Science*, 324(5934):1561–1564, June 2009.
- [47] K. Liu, T. J. Warnow, M. T. Holder, S. M. Nelesen, J. Yu, A. P. Stamatakis, and C. R. Linder. SATE-II: very fast and accurate simultaneous estimation of multiple sequence alignments and phylogenetic trees. *Syst Biol*, 61(1):90–106, Jan. 2012.
- [48] S. Liu and J. M. Suflita. Ecology and Evolution of Microbial Populations for Bioremediation. *Trends in Biotechnology*, 11:344–352, 1993.

- [49] A. Löytynoja and N. Goldman. An algorithm for progressive multiple alignment of sequences with insertions. *Proceedings of the National Academy of Sciences of the United States of America*, 102:10557–10562, 2005.
- [50] A. Löytynoja, A. J. Vilella, and N. Goldman. Accurate extension of multiple sequence alignments using a phylogeny-aware graph algorithm. *Bioinformatics (Oxford, England)*, 28(13):1684–91, July 2012.
- [51] R. D. Martin. Primate Origins and Evolution: A Phylogenetic Reconstruction. *Proceedings. Biological sciences / The Royal Society*, 263:689–696, 1990.
- [52] F. Matsen, R. Kodner, and E. V. Armbrust. pplacer: linear time maximum-likelihood and Bayesian phylogenetic placement of sequences onto a fixed reference tree. *BMC Bioinformatics*, 11(1):538+, Oct. 2010.
- [53] K. Mavromatis, N. Ivanova, and K. Barry. Use of simulated data sets to evaluate the fidelity of metagenomic processing methods. *Nature methods*, 4(6):495–500, 2007.
- [54] S. Mirarab, M. S. Bayzid, B. Boussau, and T. Warnow. Statistical binning improves species tree estimation in the presence of gene tree incongruence. *Science*, 2014. Submitted, companion paper to the Avian Phylogenomics Project paper.

- [55] S. Mirarab, N. Nguyen, and T. Warnow. SEPP: SATé-Enabled Phylogenetic Placement. *Proceedings of the Pacific Symposium on Biocomputing*, pages 247–58, Jan. 2012.
- [56] S. Mirarab, N. Nguyen, and T. Warnow. PASTA: ultra-large multiple sequence alignment. In *Research in Computational Molecular Biology*, volume 8394, pages 177–191. Lecture Notes in Computer Science, Springer, 2014.
- [57] S. Mirarab and T. Warnow. FastSP: Linear-time calculation of alignment accuracy. *Bioinformatics*, 27(23):3250–3258, 2011.
- [58] K. Mizuguchi, C. Deane, T. Blundell, and J. Overington. HOMSTRAD: a database of protein structure alignments for homologous families. *Protein Sci*, 7:24692471, 1998.
- [59] E. N. Moriyama, P. K. Strobe, S. O. Opiyo, Z. Chen, and A. M. Jones. Mining the Arabidopsis thaliana genome for highly-divergent seven transmembrane receptors. *Genome biology*, 7(10):R96, Jan. 2006.
- [60] D. Morrison. Multiple sequence alignment for phylogenetic purposes. *Australian Systematic Botany*, 19:479–539, 2006.
- [61] K.-O. Mutz, A. Heilkenbrinker, M. Lonne, J.-G. Walter, and F. Stahl. Transcriptome analysis using next-generation sequencing. *Current Opinion in Biotechnology*, 24:22–30, 2013.

- [62] C. Notredame. Recent progress in multiple sequence alignment: a survey. *Pharmacogenomics*, 3:131–144, 2002.
- [63] J. S. Papadopoulos and R. Agarwala. COBALT: constraint-based alignment tool for multiple protein sequences. *Bioinformatics (Oxford, England)*, 23:1073–1079, 2007.
- [64] M. N. Price. Fast tree-comparison tools. Website, 2009. <http://www.microbesonline.org/fasttree/treecmp.html>.
- [65] M. N. Price, P. S. Dehal, and A. P. Arkin. FastTree 2—approximately maximum-likelihood trees for large alignments. *PloS one*, 5(3):e9490, Jan. 2010.
- [66] T. M. Prychitko and W. S. Moore. Alignment and phylogenetic analysis of β -fibrinogen intron 7 sequences among avian orders reveal conserved regions within the intron. *Molecular Biology and Evolution*, 20(5):762–771, 2003.
- [67] M. Punta, P. C. Coggill, R. Y. Eberhardt, J. Mistry, J. Tate, C. Boursnell, N. Pang, K. Forslund, G. Ceric, J. Clements, A. Heger, L. Holm, E. L. L. Sonnhammer, S. R. Eddy, A. Bateman, and R. D. Finn. The Pfam protein families database. *Nucleic Acids Research*, 40(Database issue):D290–301, Jan. 2012.
- [68] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012.

ISBN 3-900051-07-0.

- [69] G. Reeck, C. de Haen, D. Teller, R. Doolittle, W. Fitch, and e. al. “homology” in proteins and nucleic acids: a terminology muddle and a way out of it. *Cell*, 50:667, 1987.
- [70] D. C. Richter, F. Ott, A. F. Auch, R. Schmid, and D. H. Huson. MetaSim: a sequencing simulator for genomics and metagenomics. *PLoS One*, 3(10):e3373, Jan. 2008.
- [71] C. Rinke, P. Schwientek, A. Sczyrba, and *et al.* Insights into the phylogeny and coding potential of microbial dark matter. *Nature*, 499:431–436, 2013.
- [72] D. Robinson and L. Foulds. Comparison of phylogenetic trees. *Math. Biosci.*, 53:131–147, 1981.
- [73] D. J. Russell, editor. *Multiple Sequence Alignment Methods*, volume 1079 of *Methods in Molecular Biology*. Springer, 2014.
- [74] N. Segata, L. Waldron, A. Ballarini, V. Narasimhan, O. Jousson, and C. Huttenhower. Efficient metagenomic microbial community profiling using unique clade-specific marker genes. *Nature Methods*, 9(8):811–814, 2012.
- [75] F. Sievers, A. Wilm, D. Dineen, T. J. Gibson, K. Karplus, W. Li, R. Lopez, H. McWilliam, M. Remmert, J. Söding, J. D. Thompson,

- and D. G. Higgins. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Molecular Systems Biology*, 7(539), Oct. 2011.
- [76] P. Soltis and D. Soltis. iPToL: the iPLANT Tree of Life Project. <https://www.iplantcollaborative.org/challenge/iplant-tree-life>.
- [77] A. Stamatakis. RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. *Bioinformatics (Oxford, England)*, 22(21):2688–90, nov 2006.
- [78] A. Stamatakis. RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics (Oxford, England)*, pages 1–2, Feb. 2014.
- [79] a. Stamatakis, a. J. Aberer, C. Goll, S. a. Smith, S. a. Berger, and F. Izquierdo-Carrasco. RAxML-Light: a tool for computing terabyte phylogenies. *Bioinformatics (Oxford, England)*, 28:2064–6, 2012.
- [80] M. Stark, S. A. S. Berger, A. Stamatakis, and C. von Mering. MLTreeMap- accurate Maximum Likelihood placement of environmental DNA sequences into taxonomic and functional reference phylogenies. *BMC genomics*, 11:461, 2010.
- [81] L. a. Stebbings and K. Mizuguchi. HOMSTRAD: recent developments of the Homologous Protein Structure Alignment Database. *Nucleic acids research*, 32(Database issue):D203–7, Jan. 2004.

- [82] J. Stoye, D. Evers, and F. Meyer. Rose: generating sequence families. *Bioinformatics*, 14:157–163, 1998.
- [83] H. Stranneheim, M. Kaller, T. Allander, B. Andersson, L. Arvestad, and J. Lundeberg. Classification of DNA sequences using Bloom filters. *Bioinformatics*, 26(13):1595–1600, May 2010.
- [84] M. A. Suchard and B. D. Redelings. BAli-Phy: simultaneous Bayesian inference of alignment and phylogeny. *Bioinformatics*, 22:2047–2048, 2006.
- [85] S. Sunagawa, D. R. Mende, G. Zeller, F. Izquierdo-Carrasco, S. A. Berger, J. R. Kultima, L. P. Coelho, M. Arumugam, J. Tap, H. B. Nielsen, S. Rasmussen, S. Brunak, O. Pedersen, F. Guarner, W. M. de Vos, J. Wang, J. Li, J. Doré, S. D. Ehrlich, A. Stamatakis, and P. Bork. Metagenomic species profiling using universal phylogenetic marker genes. *Nature Methods*, (october), Oct. 2013.
- [86] N. Takahata and Y. Satta. Evolution of the primate lineage leading to modern humans: phylogenetic and demographic inferences from DNA sequences. *Proceedings of the National Academy of Sciences of the United States of America*, 94:4811–4815, 1997.
- [87] R. L. Tatusov, M. Y. Galperin, D. A. Natale, and E. V. Koonin. The COG database: a tool for genome-scale analysis of protein functions and evolution. *Nucleic Acids Research*, 28(1):33–36, 2000.

- [88] R. L. Tatusov, D. A. Natale, I. V. Garkavtsev, T. A. Tatusova, U. T. Shankavaram, B. S. Rao, B. Kiryutin, M. Y. Galperin, N. D. Fedorova, and E. V. Koonin. The cog database: new developments in phylogenetic classification of proteins from complete genomes. *Nucleic Acids Research*, 29(1):22–28, 2001.
- [89] J. D. Thompson, B. Linard, O. Lecompte, and O. Poch. A comprehensive benchmark study of multiple sequence alignment methods: current challenges and future perspectives. *PloS one*, 6(3):e18093, Jan. 2011.
- [90] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13(2):260–269, 1967.
- [91] L. Wang and T. Jiang. {On} the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1:337–348, 1994.
- [92] L.-S. Wang, J. Leebens-Mack, P. Kerr Wall, K. Beckmann, C. W. DePamphilis, and T. Warnow. The impact of multiple protein sequence alignment on phylogenetic estimation. *IEEE/ACM transactions on computational biology and bioinformatics / IEEE, ACM*, 8:1108–19, 2011.
- [93] T. J. Wheeler and J. D. Kececioglu. Multiple alignment by aligning alignments. *Bioinformatics (Oxford, England)*, 23:i559–i568, 2007.

- [94] A. Wilder-Smith, E.-E. Ooi, S. G. Vasudevan, and D. J. Gubler. Update on dengue: epidemiology, virus evolution, antiviral drugs, and vaccine development. *Current infectious disease reports*, 12:157–164, 2010.
- [95] K. E. Wommack, J. Bhavsar, and J. Ravel. Metagenomics: read length matters. *Applied and environmental microbiology*, 74:1453–1463, 2008.
- [96] K. M. Wong, M. A. Suchard, and J. P. Huelsenbeck. Alignment uncertainty and genomic analysis. *Science*, 319(5862):473–476, 2008.
- [97] D. Wu and J. A. Wu. Stalking the fourth domain in metagenomic data: Searching for, discovering, and interpreting novel, deep branches in marker gene phylogenetic trees. *PLoS ONE*, 6(3):e18011, 03 2011.
- [98] P. Yang, Y. H. Yang, B. B. Zhou, and A. Y. Zomaya. A review of ensemble methods in bioinformatics. *Current Bioinformatics*, 5(4):296–308, 2010.
- [99] S. E. Yuksel, J. N. Wilson, and P. D. Gader. Twenty years of mixture of experts. *IEEE transactions on neural networks and learning systems*, 23(8):1177–93, Aug. 2012.
- [100] D. J. Zwickl and D. M. Hillis. Increased taxon sampling greatly reduces phylogenetic error. *Syst. Biol.*, 51:588–598, 2002.