ABSTRACT

Title of dissertation: COMPUTATIONAL METHODS TO ADVANCE PHYLOGENOMIC WORKFLOWS

Adam Bazinet, Doctor of Philosophy, 2015

Dissertation directed by: Professor Michael Cummings
Center for Bioinformatics and Computational Biology
Affiliate Professor, Department of Computer Science

Phylogenomics refers to the use of genome-scale data in phylogenetic analysis. There are several methods for acquiring genome-scale, phylogenetically-useful data from an organism that avoid sequencing the entire genome, thus reducing cost and effort, and enabling one to sequence many more individuals. In this dissertation we focus on one method in particular — RNA sequencing — and the concomitant use of assembled protein-coding transcripts in phylogeny reconstruction. Phylogenomic workflows involve tasks that are algorithmically and computationally demanding, in part due to the large amount of sequence data typically included in such analyses. This dissertation applies techniques from computer science to improve methodology and performance associated with phylogenomic workflow tasks such as sequence classification, transcript assembly, orthology determination, and phylogenetic analysis. While the majority of the methods developed in this dissertation can be applied to the analysis of diverse organismal groups, we primarily focus on the analysis of transcriptome data from Lepidoptera (moths and butterflies), generated as part of a collaboration known as “Leptree”.

COMPUTATIONAL METHODS TO ADVANCE PHYLOGENOMIC WORKFLOWS

by

Adam Bazinet

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2015

Advisory Committee:
Professor Michael Cummings, Chair/Advisor
Professor Charles Mitter, Dean's Representative
Professor Mihai Pop
Professor Héctor Corrada Bravo
Professor Amitabh Varshney

Preface

This dissertation is based, in part, on the following publications, listed by chapter:

Chapter 2

Adam L. Bazinet and Michael P. Cummings. A comparative evaluation of sequence classification programs. *BMC Bioinformatics*, 13:92, 2012.

Chapter 3

Adam L. Bazinet, Michael P. Cummings, and Antonis Rokas. Homologous gene consensus avoids orthology-paralogy misspecification in phylogenetic inference. Unpublished.

Chapter 4

Adam L. Bazinet, Derrick J. Zwickl, and Michael P. Cummings. A gateway for phylogenetic analysis powered by grid computing featuring GARLI 2.0. *Systematic Biology*, 63(5):812-818, 2014.

Chapter 5

Adam L. Bazinet and Michael P. Cummings. Computing the tree of life: leveraging the power of desktop and service grids. In *Proceedings of the Fifth Workshop on Desktop Grids and Volunteer Computing Systems (PCGrid)*, 2011.

Chapter 6

Adam L. Bazinet and Michael P. Cummings. Subdividing long-running, variable-length analyses into short, fixed-length BOINC workunits. *Journal of Grid Computing*. Submitted.

Chapter 7

Adam L. Bazinet, Michael P. Cummings, Kim T. Mitter, and Charles W. Mitter. Can RNA-Seq resolve the rapid radiation of advanced moths and butterflies (Hexapoda: Lepidoptera: Apoditrysia)? An exploratory study. *PLoS ONE* 8(12):e82615, 2013.

Chapter 8

Adam L. Bazinet, Michael P. Cummings, Kim T. Mitter, and Charles W. Mitter. Can RNA-Seq resolve the rapid radiation of advanced moths and butterflies (Hexapoda: Lepidoptera: Apoditrysia)? A follow-up study. In preparation.

Dedication

I dedicate this dissertation to my wife Jennifer, and my two children Cassandra and Derek. A loving family makes every day worth living.

Acknowledgements

I am grateful to many, many people for making this dissertation work possible.

First, I would like to give wholehearted thanks to my advisor, Dr. Michael Cummings. For well over a decade, I have been privileged to be a member of the Laboratory of Molecular Evolution, which Dr. Cummings leads and directs. My experience in his research group has been truly rewarding, and I am exceedingly grateful to be able to work with and learn from Dr. Cummings on a daily basis. He is a tireless, dedicated scientist and a devoted mentor.

I would like to recognize Drs. Charles and Kim Mitter, who have been extremely pleasant colleagues to work with on lepidopteran systematics over the past several years. Thanks also to the rest of my committee members for their support and encouragement.

Thanks to the faculty members in computer science at the University of Maryland — particularly those affiliated with the Center for Bioinformatics and Computational Biology — for the many years of instruction that I have had the privilege to receive.

To the many colleagues, coworkers, collaborators, system administrators, and support staff with whom I have had the pleasure of working these many years, thank you for your dedication and cooperation.

Finally, I would like to thank my family for their love and unwavering support.

The acknowledgements that follow are specific to various chapters.

Chapter 2 We acknowledge all of the tool developers for making their programs, data sets, and documentation available. In particular, we thank the following authors for their assistance acquiring data sets, running various programs, or making sense of results: Bo Liu (MetaPhyler), Manuel Stark (MLTreeMap), Kaustubh Patil (PhyloPythiaS), Arthur Brady and Steven Salzberg (PhymmBL), and Ozkan Nalbantoglu (RAIphy). In addition, we thank the reviewers for their helpful comments and suggestions.

Chapter 4 We thank Barry Dutton, Yevgeny Deviatov, Derrick Hinkle, and Kevin Young for their efforts developing various aspects of the grid system and the GARLI web service; Charles Mitter for developing the statistical determination of the number of required GARLI search replicates; and Mike Landavere, Christopher Camacho, Ahmed El-Haggan, Taha Mohammed, Patrick Beach, Matthew Kweskin, Kevin Hildebrand, and Fritz McCall (and other UMIACS staff members) for connecting and administering grid system resources. We also thank the associate editor and one anonymous reviewer for their helpful suggestions. Funding for this work was provided by National Science Foundation award number DBI-0755048.

Chapter 5 U.S. National Science Foundation award DBI-0755048 provided funding for this project.

Chapter 6 We thank Derrick Zwickl for adding necessary features to GARLI that enabled this work. Similarly, we thank David Anderson for adding new features to BOINC that aided with testing. This work was funded by National Science Founda-

tion award DBI-1356562.

Chapter 7 We are greatly indebted to the following generous colleagues for providing specimens used in this study: Andreas Zwick, Marcus J. Matthews, Akito Y. Kawahara, Robert F. Denno, Bernard Landry, Timothy Friedlander, Daniel H. Janzen, Andrew Mitchell, Joaquin Baixeras, Ron Robertson, Jadranka Rota, David Adamski, Jaga Giebultowicz, Chris Bergh, Neil R. Spencer, James K. Adams, and Luis Pea. Our work was greatly facilitated by the expert help of Suwei Zhao of the University of Maryland-Institute for Bioscience and Biotechnology Research Sequencing Core. We thank the Editor as well as Niklas Wahlberg and two anonymous reviewers for helpful comments on the manuscript. This study was made possible by the foundation built by our esteemed colleague Jerome Regier, who has now transitioned to a new scholarly career.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

r	optimal job runtime
α	alpha
β	beta
χ	chi
ω	omega
ABySS	Assembly By Short Sequences
AToL	Assembling the Tree of Life
BLAST	Basic Local Alignment Search Tool
BLOSUM	BLOcks SUbstitution Matrix
BOINC	Berkeley Open Infrastructure for Network Computing
BP	Bootstrap
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
CDS	Coding sequence
CIPRES	Cyberinfrastructure for Phylogenetic Research
CMNS	College of Computer, Mathematical, and Natural Sciences
COI	Cytochrome c oxidase subunit I
CPU	Central Processing Unit
CV	Coefficient of variation
DBM-DB	Diamondback moth genome database
DIAMOND	Double Index AlignMent Of Next-generation sequencing Data
DNA	Deoxyribonucleic acid
EMR	Extended majority-rule
EST	Expressed Sequence Tag
FACS	Fast and Accurate Classification of Sequences
FAMeS	Fidelity of Analysis of Metagenomic Samples
FFT	Fast Fourier transform
GARLI	Genetic Algorithm for Rapid Likelihood Inference
GB	Gigabyte
GHz	Gigahertz
GPU	Graphics processing unit
GRAM	Grid Resource Allocation Manager
GTR	General time-reversible
GUI	Graphical user interface
HaMStR	Hidden Markov Model based Search for Orthologs using Reciprocity
HAV	Homogeneous app version
HMM	Hidden Markov model
HR	Homogeneous redundancy
HTC	High-throughput computing
IJE	Idle job equality
ILS	Incomplete lineage sorting
IMM	Interpolated Markov model
IUPAC	International Union of Pure and Applied Chemistry
KB	Kilobyte

Lambda	Local Aligner for Massive Biological Data
LCA	Lowest-common ancestor
LOWESS	Locally weighted scatterplot smoothing
MAFFT	Multiple Alignment using Fast Fourier Transform
MB	Megabyte
MDS	Monitoring and Discovery Service
MEGAN	MEtaGenome ANalyzer
MG-RAST	Metagenomic Rapid Annotations using Subsystems Technology
MHz	Megahertz
ML	Maximum likelihood
MP-EST	Maximum Pseudo-likelihood Estimate of the Species Tree
MPI	Message Passing Interface
NJ	Neighbor-joining
NJst	Neighbor-joining species tree
NBC	Naive Bayes Classifier
NCBI	National Center for Biotechnology Information
NGS	Next-generation sequencing
OpenMP	Open Multi-Processing
ORF	Open reading frame
PAUP	Phylogenetic Analysis Using Parsimony
PC	Personal Computer
PDF	Portable Document Format
PEG	Protein-encoding gene
RAM	Random-access memory
RAxML	Randomized Axelerated Maximum Likelihood
RBH	Reciprocal best BLAST hit
RDP	Ribosomal Database Project
RNA	Ribonucleic acid
RNA-Seq	RNA sequencing
RSD	Reciprocal smallest distance
RTC	Rooted triple consensus
SINA	SILVA Incremental Aligner
SLURM	Simple Linux Utility for Resource Management
STAR	Species Tree estimation using Average Ranks of coalescences
STEAC	Species Tree Estimation using Average Coalescence times
SVD	Singular value decomposition
TRAM	Target Restricted Assembly Method
TORQUE	Terascale Open-source Resource and QUEue Manager
UMIACS	University of Maryland Institute for Advanced Computer Studies
WU	Workunit
XML	Extensible Markup Language
YGOB	Yeast Gene Order Browser
bp	Base pair
cDNA	Complementary DNA
kb	Kilobase

mRNA	Messenger RNA
matK	Maturase K
mt	Mitochondrial
nt	Nucleotide
pHMM	Profile hidden Markov model
rRNA	Ribosomal RNA
sp.	Species (singular)
spp.	Species (plural)

Chapter 1: Introduction

1.1 Background on phylogenetics

The theory of evolution by natural selection laid out by Charles Darwin is the fundamental guiding principle in modern biology. Evolutionary theory now includes a detailed understanding of evolution at the molecular level. Molecular sequencing technologies developed over the course of the past several decades have enabled the recovery of the exact nucleotide sequence of biological macromolecules (e.g., DNA and RNA), and in particular, subsequences of these molecules — *genes* — that encode the fundamental functional units of cell biology (most commonly, proteins).

Phylogenetics — the study of evolutionary relationships among groups of organisms — has benefitted greatly from the availability of molecular sequence data, and consequently has come to rely less on morphological data (data derived from visual inspection of organisms and sometimes measurement of organismal features). *Molecular phylogenetics*, therefore, uses variation in genetic sequence data as the basis for proposing and revising *phylogenies* — hypotheses about the evolutionary relationships of organisms, most commonly represented as phylogenetic trees.

The next section describes molecular phylogenetic workflows in more detail.

1.2 “Genes to trees”: an overview of phylogenetic workflows

1.2.1 A traditional phylogenetic workflow

A traditional molecular phylogenetic analysis typically commences with the acquisition of specimens, either from a natural environment or an existing collection, followed by the isolation or purification of particular molecules that are to be sequenced (most commonly DNA or RNA). For each taxon included in the analysis, one or more genes are sequenced using primers (oligonucleotides) that are designed to amplify and sequence particular genomic loci (i.e., Sanger-style sequencing [1]). In this manner, one expects to recover orthologous sequence data — typically, parts of genes — for multiple taxa. (Orthology and paralogy are discussed in more detail in Chapter 3.) Sets of orthologous gene sequences are then analyzed either separately or together in a process that entails multiple sequence alignment and phylogenetic analysis. The final product is most commonly a phylogenetic tree relating the taxa in question.

1.2.2 The Leptree collaboration

Phylogenetic workflows were integral to Assembling the Tree of Life (ATOL) projects, which were funded by the National Science Foundation and typically involved collaboration among multiple research groups to study a particular clade (or group) of organisms. “Leptree”, which began as an ATOL project headquartered at the University of Maryland, was an ambitious effort to collect, sequence, and analyze the evolutionary relationships of hundreds of insects of the order Lepidoptera (moths and butterflies). During the first stage of the project, which lasted several years, the consortium acquired DNA for the majority of target specimens, and Sanger-sequenced either all or part of 26 genes that were deemed useful for resolving relatively deep phylogenetic relationships. Sequence alignments for various subsets of taxa were created and manually refined, and phylogenetic analysis was performed using maximum likelihood and Bayesian methods (as implemented in GARLI [2] and MrBayes [3, 4], respectively). The results and their implications for lepidopteran evolution are described in multiple publications [5–8].

More recently, the Leptree group has begun generating transcriptome data for various lepidopteran taxa (a process described in the following section, and more extensively in Chapters 7 and 8); Leptree-generated transcriptome data is the primary genomic sequence data analyzed in this dissertation.

1.2.3 Phylogenomics: whole-genome-inspired methodology

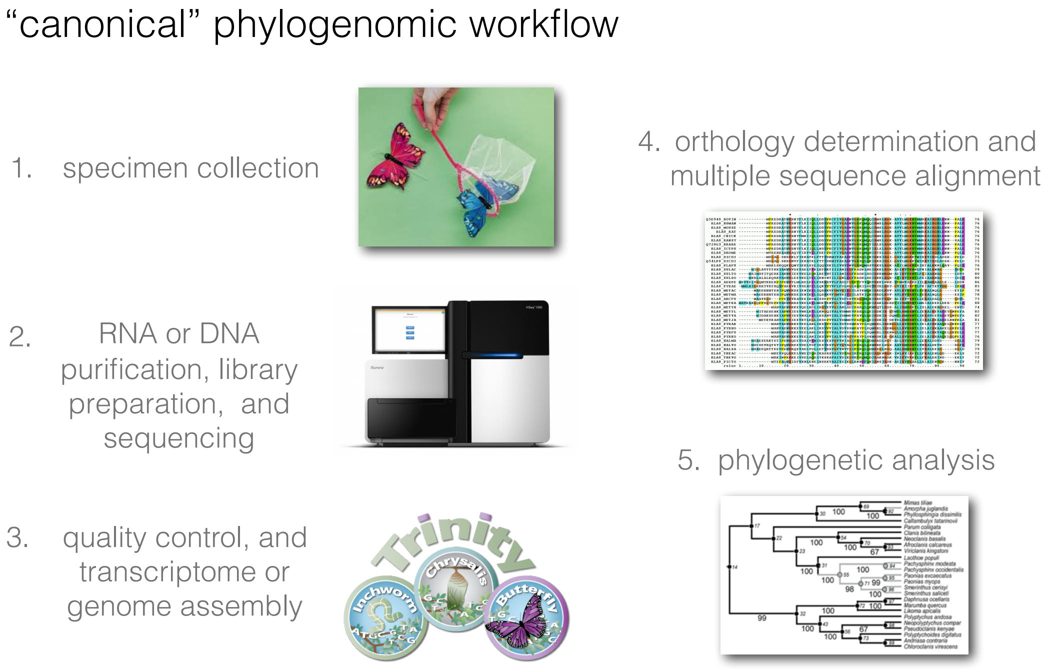
So-called "next-generation sequencing" technologies have enabled massive amounts of genome sequencing to be performed at relatively low cost compared to Sanger-based sequencing practices [9]. Next-generation sequencing was first used for *whole-genome shotgun sequencing*, in which the entire genome of an organism is sheared into small pieces of DNA that are individually sequenced, thus producing a large number of sequence "reads". These millions of reads are typically assembled into longer sequences called "contigs", which are then combined into "scaffolds" using mate-pair information. Scaffolds, finally, are arranged and oriented such that they accurately represent the sequence of the particular molecule from which they originated (e.g., a chromosome). Another, more recent assay is *RNA sequencing* (RNA-Seq) [10], in which complementary DNA (cDNA) is synthesized from RNA — most commonly, messenger RNA (mRNA) — and then sequenced. In this manner, a significant portion of the "transcriptome" of an organism (protein-coding genes expressed by the cell, as well as some non-coding RNAs) may be sequenced in a single experiment.

In a pioneering study, Hittinger et al. [11] showed how RNA-Seq could be used in a "*phylogenomic*" analysis of mosquitos. Using a non-normalized RNA-Seq protocol, they sequenced many highly-expressed genes thought to be phylogenetically useful (e.g., housekeeping and cell-cycle genes, which are well-conserved and have favorable evolutionary rates). Transcripts were assembled *de novo* (without the aid of a reference transcriptome) using VELVET [12]. Following quality control and filtering, a reciprocal best hit strategy using BLAST [13] was used to identify clusters of orthologous gene sequences. Multiple sequence alignment of these clusters was performed using DIALIGN2 [14] and custom Perl scripts. Phylogenetic analyses of these data matrices recovered robust and well-supported phylogenies of the various mosquito species, thus demonstrating the viability of RNA-Seq as a cost-effective way to obtain genome-scale data for use in phylogenetics.

Since then, phylogenomic studies of many organismal groups have been undertaken [15–25,25–42]. The methodology employed in these studies may vary from the Hittinger et al. workflow in certain respects. For example, the methods used to acquire genomic data may vary: instead of RNA-Seq, studies may employ "targeted sequencing" approaches to capture sequence from multiple genomic loci; additionally, studies may incorporate expressed sequence tag (EST) data from public databases. These various high-throughput sequencing approaches to generating data for systematics and phylogenetics are discussed in a comprehensive (although biased) review by Lemmon and Lemmon [43]. Other workflow details may vary as well, including methods for sequence assembly, orthology determination, data filtering, and phylogenetic analysis. In broad outline, however, most phylogenomic studies proceed according to the same basic steps, which we hereafter call the "canonical workflow" (Figure 1.1).

1.3 Computational methods in phylogenomic workflows

In the canonical phylogenomic workflow (Figure 1.1), once specimens have been acquired (step one) and RNA or DNA purification, library preparation, and sequencing (step two) have been completed, the remaining steps are purely computational. In the following sections we present the computational steps in the RNA-Seq-based workflow developed in this dissertation as well-defined computer science problems. Specifically, for each step we describe the input data, the algorithms used to compute results, and the resulting output data.



“canonical” phylogenomic workflow

1. specimen collection

2. RNA or DNA purification, library preparation, and sequencing

3. quality control, and transcriptome or genome assembly

4. orthology determination and multiple sequence alignment

5. phylogenetic analysis

Figure 1.1: (1) The canonical phylogenomic workflow begins with specimen acquisition. (2) RNA (or DNA) is extracted from the specimen and sequenced. (3) Low-quality sequence reads are trimmed or removed, and data of sufficient quality is assembled into transcripts (or contigs). (4) Orthologous gene sequences among taxa are determined, and then aligned to one another. (5) Finally, phylogenetic analysis is performed on the gene alignments either individually, or after they have been concatenated.

1.3.1 Transcriptome assembly

In *de novo* transcriptome assembly, one is given a set of sequencing reads, usually of uniform length, each of which is composed of a sequence of characters over an alphabet of four symbols {A, C, G, T}. The goal is to assemble the reads whose sequences overlap into a set of longer sequences called transcripts, each of which should correspond to the cDNA sequence of an mRNA molecule. In eukaryotes, there is a biological mechanism called *alternative splicing* that enables multiple gene products, called isoforms, to be produced from a single genomic locus. Thus, many recent transcriptome assemblers aim to assemble a set of transcripts corresponding to all of the isoforms in the sample, often by finding paths through a De Bruijn graph [44, 45]. In this data structure, the nodes consist of k-mers (sequences of length k), and the edges represent overlap between k-mers.

1.3.2 Orthology determination

As discussed in Chapter 3, there are variety of methods for determining orthologous gene sequences among extant genomes — i.e., sequences that derive from a common ancestral gene sequence. From a computational standpoint, the input is a set of transcripts (either whole or partial coding sequences) associated with each taxon in the analysis, and the output is a set of orthologous groups of sequences. An orthologous group frequently corresponds to an individual gene, or perhaps isoform. The process of clustering sequences into orthologous groups most commonly relies on sequence similarity searches [13] to identify sequences that are reciprocally most similar to one another. In addition, phylogenetic analysis of sequences can be very useful in orthology determination, as demonstrated in the Ensembl pipeline [46].

1.3.3 Multiple sequence alignment

Given a group of orthologous sequences, it is necessary to determine orthology down to the level of the individual nucleotide (or amino acid, in the case of a translated nucleotide sequence), as phylogenetic inference usually analyzes each character independently. Although we expect the sequences in an orthologous group to have a high degree of similarity (the exact degree depends on the evolutionary distance between the taxa), some variation among sequences is needed to make evolutionary inferences. Furthermore, the sequences frequently vary in length, so the sequence alignment process should ensure that putatively orthologous characters are placed in the same column of the alignment. Thus, where evolution has inserted characters in a particular sequence relative to the other sequences in the group (or equivalently, deleted characters from the other sequences relative to a particular sequence), this may be modeled with gap characters (often represented with a dash: "–"). The input to a multiple sequence alignment program in the context of a phylogenomic workflow, therefore, is a set of orthologous, unaligned sequences, and the output is a set of orthologous sequences that have been aligned according to an optimality criterion (usually the maximization of an alignment score, which is penalized for mismatched characters and the number and length of gaps that have been introduced). A performance comparison of multiple sequence alignment programs is given by Thompson et al. [47].

1.3.4 Phylogenetic analysis

The aligned orthologous sequence groups are subjected to phylogenetic analysis, either individually or in a combined analysis. The most powerful and computationally intensive class of phylogenetic analysis methods apply an evolutionary model to the sequence data, and evaluate possible topologies that relate the taxa together. The result of phylogenetic analysis is typically a hierarchical, bifurcating tree that gives the relationships among taxa, wherein the length of a branch corresponds to the amount of evolution a particular taxon has undergone relative to the others. The large number of possible topologies, branch lengths, and model parameter values makes finding the "*best tree*" — i.e., the tree that best explains the data — a challenging combinatorial optimization problem. Indeed, an exhaustive search of "tree space" is not possible for all but the smallest data sets. Instead, phylogenetic analysis programs such as GARLI [2] use heuristics for proposing and improving candidate trees (e.g., GARLI uses a heuristic resembling a genetic algorithm). GARLI is an example of a type of phylogenetic analysis program that uses maximum likelihood. The other major type of model-based phylogenetic analysis uses Bayesian inference, as implemented in a program such as MrBayes [4].

1.4 Applying high-throughput computing to phylogenetic analysis

Phylogenetic analysis is often extremely computationally intensive. The total amount of computation required for a particular analysis depends on the number of taxa included, the number of informative characters in the data set, the complexity of the evolutionary model being applied, and the heuristic used to search the solution space of possible phylogenetic trees, a space in which the number of possible topologies grows exponentially with the number of taxa. Like many problems in computer science, one can apply parallelism, in this case to phylogenetic analysis, to reduce the time needed to obtain results. In particular, the "embarrassingly parallel" paradigm, known more formally as high-throughput computing (HTC), may be used with the maximum likelihood type of phylogenetic analyses because they do not typically require inter-processor communication. Thus, a large number of searches for the "best tree" (the tree of highest likelihood) can be launched in parallel on processors that are distributed among various computer systems. When these analyses complete, the results are aggregated, compared, and presented. For this reason, maximum likelihood phylogenetic analysis is a good fit for *grid computing*, a model of distributed computing that uses geographically and administratively disparate resources. The user of grid computing is able to use a large number of computers without having to directly interact with them [48]. In the following section we describe The Lattice Project, the grid computing system that we use to perform computationally-intensive phylogenetic analyses.

1.4.1 The Lattice Project: a multi-model grid computing system

The Lattice Project is a grid computing system developed since 2003 at the University of Maryland under the guidance and direction of Dr. Michael Cummings and Adam Bazinet. The Lattice Project, which is based on Globus [49] software, incorporates volunteer computers running BOINC [50] as well as traditional grid computing resources such as Condor pools [51] and compute clusters. The architecture and functionality of the system is described extensively in Bazinet's master's thesis: *The Lattice Project: A Multi-model Grid Computing System* [52]. In recent years, we have enhanced the system for phylogenetic analysis by developing a web interface to the GARLI service [53, 54], currently available at `molecularevolution.org`. (The GARLI web service is described in greater detail in Chapter 4.) Our laboratory has also created a `C++` library that uses GPUs and other specialized hardware to speed up the likelihood calculations that form the kernel of many phylogenetic analysis programs [55]. (Synergistically, the BOINC pool of volunteer desktop computers is a substantial source of modern GPUs.) The Lattice Project has been used in studies of conservation biology [56], pandemic influenza [57], human evolution [58], protein binding [59], quantification of lineage divergence [60], and phylogenetics [5, 6, 6–8, 61–130]. The Lattice Project enables us to complete large-scale phylogenetic analyses in a reasonable amount of time, and is thus an essential part of the computing infrastructure used in this dissertation. Improvements to the grid system to better support phylogenetic analysis are discussed in Chapters 4–6.

1.5 Dissertation outline

The dissertation is structured as follows. Chapters 2–6 each describe work relevant to a particular stage of a phylogenomic workflow. Thus, Chapter 2 reviews and benchmarks sequence classification programs, which are useful for identifying the origin of reads and transcripts derived from the sequencing and assembly process, respectively. Chapter 3 describes a novel method for robust orthology determination, which is a critical and often challenging step in phylogenomic workflows. Chapter 4 describes the GARLI web service, our public-facing interface that facilitates the execution of GARLI analyses on our grid system. Chapter 5 describes several improvements to the grid system, many of which are aimed at improving the scheduling and execution of phylogenetic analyses. Chapter 6 describes a scheme that enables the use of BOINC for phylogenetic analyses that need to be completed relatively quickly, such as those submitted by users of the GARLI web service. Chapter 7 incorporates all of the foregoing work into a complete phylogenomic workflow that is used to analyze Leptree data. Chapter 8 improves on the methodology used in the phylogenomic workflow, and expands the scope and number of analyses that are performed. Finally, Chapter 9 seeks to identify the presence of endosymbiont sequences in transcriptome data derived from insect hosts.

Chapter 2: An evaluation of sequence classification programs

This chapter is based on the following publication: Adam L. Bazinet and Michael P. Cummings. A comparative evaluation of sequence classification programs. *BMC Bioinformatics*, 13:92, 2012. Corrections included.

2.1 Background

A fundamental problem in modern genomics is to taxonomically or functionally classify DNA sequences derived from environmental sampling (i.e., metagenomics). Many metagenomic studies are essentially community ecology studies, which seek to characterize communities statically or dynamically in terms of composition, structure, abundance, demography, or succession, and sometimes with consideration of other biotic or abiotic factors. Consequently many of these characterizations, and inferences derived from them, are sensitive to the accuracy and precision of taxonomic assignment of the metagenomic sequences involved. These sequences are often in the form of unassembled reads whose average length in a sample may vary by an order of magnitude depending on the sequencing technology used (e.g., ≈ 100 bp to $\approx 1{,}000$ bp). To classify these sequences of unknown origin, the basic strategy employed is to compare them to annotated sequences that reside in public databases (e.g., GenBank [131], Pfam [132]). On the basis of such comparisons, one may be able to say with some certainty that a particular sequence belongs to a specific taxon (of any taxonomic rank from domain to species; more specific classifications are usually more desirable). Sometimes the query sequence does not have a close relative in the database, which is problematic for all methods.

The classification of unlabeled sequences using previously labeled sequences is *supervised* learning; this approach is the focus of our evaluation. However, it is important to mention that *unsupervised* learning techniques exist for "binning" sequences in an environmental sample (e.g., LikelyBin [133], CompostBin [134]); i.e., clustering groups of similar sequences together. These techniques are useful when one desires a high-level characterization of their sample (e.g., classification of bacteria at the phylum rank). Binning may also be used to improve subsequent supervised classification of groups of sequences (PhyScimm [135]).

It is important to note that some supervised learning methods will only classify sequences that contain "marker genes". Marker genes are ideally present in all organisms, and have a relatively high mutation rate that produces significant variation between species. The use of marker genes to classify organisms is commonly known as DNA barcoding. The 16S rRNA gene has been used to greatest effect for this purpose in the microbial world (green genes [136], RDP [137]). For animals, the mitochondrial COI gene is popular [138], and for plants the chloroplast genes *rbc*L and *mat*K have been used [139]. Other strategies have been proposed, such as the use of protein-coding genes that are universal, occur only once per genome (as opposed to 16S rRNA genes that can vary in copy number), and are rarely horizontally transferred [140]. Marker gene databases and their constitutive multiple alignments and phylogenies are usually carefully curated, so taxonomic and functional assignments based on marker genes are likely to show gains in both accuracy and speed over methods that analyze input sequences less discriminately. However, if the sequencing was not specially targeted [141], reads that contain marker genes may only account for a small percentage of a metagenomic sample.

2.1.1 General approaches to sequence classification

We have identified three main supervised learning approaches that compare query sequences to database sequences for the purpose of assigning a taxon label: sequence similarity search-based methods (homology or alignment-based methods; e.g., BLAST [142]), sequence composition methods (e.g., Markov models, *k*-mer counting), and phylogenetic methods (which apply an evolutionary model to the query and database sequences and determine where the query best "fits" in the phylogeny). Most software programs use only one of these approaches, but some use a combination of two approaches. (None of the programs referenced in this study combine all three approaches.)

Programs that primarily use sequence similarity search include CARMA [143, 144], FACS [145], jMOTU/Taxonerator [146], MARTA [147], MEGAN [148], MetaPhyler [149], MG-RAST [150], MTR [151], and SOrt-ITEMS [152]. Most of these programs employ BLAST (most commonly, BLASTX), and several incorporate some version of the lowest-common ancestor (LCA) algorithm first pioneered by MEGAN.

After BLAST, the second most common method aligns a query sequence to a reference sequence represented by a profile hidden Markov model (pHMM); usually a Pfam domain. Alignment-based methods display great accuracy, even for short query sequences, but suffer from two general shortcomings: a) because the reference databases are very large, it can take a long time to search each query sequence against them; and b), if the query sequence is not represented in the database, as could often be the case, assignment accuracy may suffer more so than with other methods.

Programs that primarily use sequence composition models include Naive Bayes Classifier (NBC) [153, 154], PhyloPythia [155, 156], PhymmBL [157], RAIphy [158], RDP [159], Scimm [135], SPHINX [160], and TACOA [161]. Methods for building sequence models often make use of interpolated Markov models (IMMs), naive Bayesian classifiers, and *k*-means/*k*-nearest-neighbor algorithms. There is some overhead to computing sequence models of organismal genomes, but once models are built, query sequence classification is generally faster than with alignment-based methods. Accuracy, however, may still be able to be improved — this is why PhymmBL incorporates similarity search (the "BL" is for BLAST). As a result, PhymmBL achieves greater accuracy than either Phymm or BLAST alone. Finally, it was widely reported that the initial version of PhyloPythia performed poorly for query sequences less than 1,000 bp in length [157,158]; few current next-generation sequencing (NGS) technologies produce reads of that length or greater. However, composition-based methods are now perfectly capable of classifying short query sequences. For example, NBC obtained over 90% accuracy for 25 bp reads with five-fold cross-validation [153].

Programs that primarily use phylogenetic methods include epa [162], FastTree [163], and pplacer [164]. Phylogenetic methods attempt to “place” a query sequence on a phylogenetic tree according to a model of evolution using maximum likelihood (ml), Bayesian inference, or other methods such as neighbor-joining (nj). Some programs compute the length of the inserted branch, which represents the amount the query sequence has evolved relative to the rest of the sequences; most programs, however, are simply concerned with the placement (and hence classification) of the query sequence. Programs assign a specific taxon (and hence taxonomic rank) to a “placed” sequence using different algorithms, but they all make use of the basic observation that an inserted branch will be divergent from an internal node representing a species or higher rank. Since phylogenetic methods require a multiple alignment, and a fixed topology (either derived from the multiple alignment, or from some other source; e.g., the ncbi taxonomy), the first step in most phylogenetic workflows is to add a query sequence containing a marker gene to a reference alignment (amphora [165, 166], Treephyler [167], green genes [136]). Hence, most phylogenetic methods require the use of marker genes. One that does not, however, is sap [168], in which the first step is to construct a multiple alignment from the results of a blast search. Phylogenetic methods assume that using computationally intensive evolutionary models will produce gains in accuracy, and their inherent use of tree-based data structures makes taxon assignment to higher ranks as well as lower ones very straightforward. The additional algorithmic complexity means that phylogenetic workflows currently require substantial computing power to analyze large metagenomic samples, however; this is true even for methods that only use marker genes. Large-scale analyses will gradually become more practical as more efficient algorithms are developed, computational resources become more powerful, and through use of parallelization.

2.1.1.1 Additional considerations

It is important to consider if a sequence classification method offers a measure of assignment confidence. Such an uncertainty measure is extremely useful; assignments whose confidence score is below a certain threshold can be disregarded, for example. Phylogenetic methods tend to provide confidence of assignment through use of bootstrap values, posterior probabilities, or other techniques. Alignment-based methods generally do not provide a confidence estimate.

Another consideration is the availability and ease of use of the program — if it is a command line program, has a graphical user interface (GUI), is available as a web service, and so on. If the program is to be downloaded and installed, one must consider how much processing power, memory, and disk the program will need to analyze a particular data set. Some of these needs will prohibit local execution of the program for large data sets, perhaps instead necessitating use of a compute cluster. If there is a web service available for the program, one needs to find out how much computational power is allocated to a single user, and thus if the service can be used in practice to analyze large metagenomes. A further consideration is if the program continues to be actively developed and maintained after a paper is published and the code is initially released. Actively maintained programs are likely to be improved as a result of feedback from users, and may eventually become "standard" tools used by the community.

2.1.2 Program capability analysis

We identified 25 sequence classification programs that fall into one of the three primary analysis categories mentioned previously: sequence similarity or alignment-based (nine programs), sequence composition model-based (eight programs), and phylogenetic-based (eight programs). Our list is not exhaustive, but we do include a broad cross section of widely-used and interesting programs in our comparison.

The attributes and capabilities of each program are given in Table 2.1. For each program, we report the general analysis method it uses, and more detailed analysis characteristics, as applicable; if the program requires specific genes as input; and the type of interface to the program. For a given program attribute (a column in Table 2.1), it is possible to have multiple values. We defined a distance function and created a neighbor-joining tree that clusters the programs based on attribute similarity (Figure 2.1).

2.1.3 Program performance evaluation

When publishing their method, researchers typically compare their program to one or more existing programs. Presumably they attempt to choose programs that are most similar to their own, but we find that this is not always the case. Perhaps the researcher is simply not aware of all the tools in existence, or does not have the

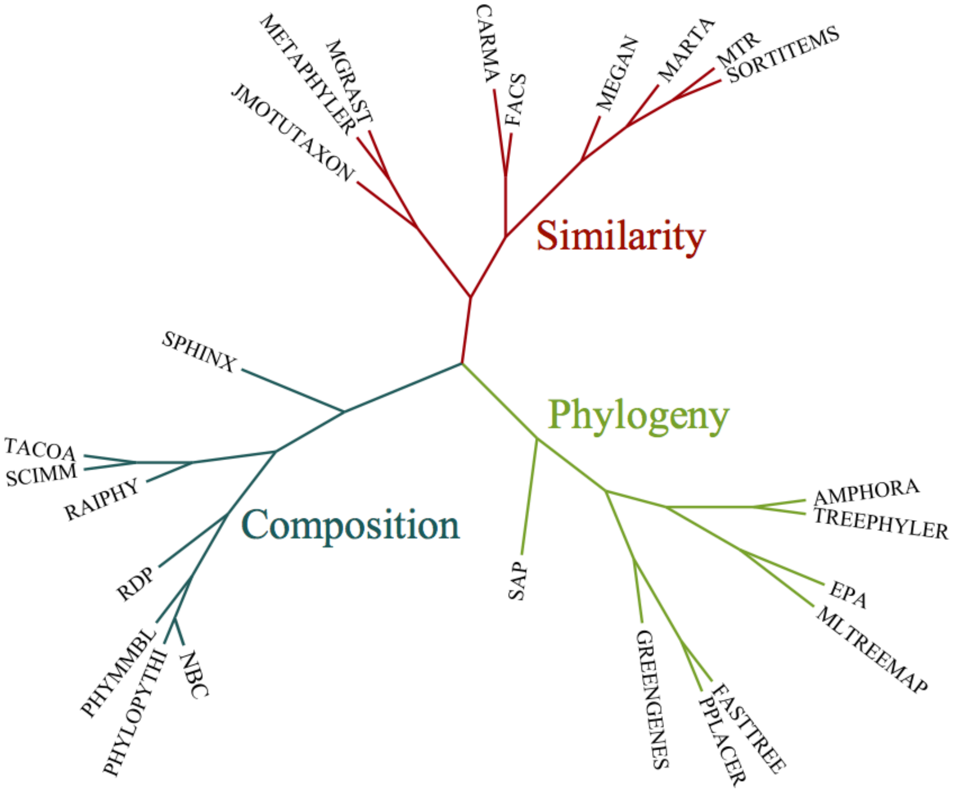


Figure 2.1: A neighbor-joining tree that clusters the sequence classification programs based on attribute similarity.

Similarity-based Methods

Program	Similarity Method	LCA	Specific Genes Req'd	Interface
CARMA	BLAST, HMM			command line, web-based
FACS	other			command line
jMOTU/Taxonerator	BLAST, other		multiple alignment	command line
MARTA	BLAST	LCA-like		command line
MEGAN	BLAST	LCA-like		GUI
MetaPhyler	BLAST		marker genes	command line
MG-RAST	BLAST		marker genes	web-based
MTR	BLAST	LCA-like		command line
SOrt-ITEMS	BLAST	LCA-like		command line

Composition-based Methods

Program	Composition Method	Machine Learning	Confidence Method	Specific Genes Req'd	Interface
Naive Bayes Classifier	NBC	supervised	other		command line, web-based
PhyloPythiaS	other	supervised			command line, web-based
PhymmBL	IMM	supervised	other		command line
RAIphy	other	semi-supervised			GUI
RDP	k-means/kNN, NBC	supervised	bootstrap	16S rRNA	command line, web-based
Scimm	IMM	semi-supervised			command line
TACOA	k-means/kNN	supervised			command line

Phylogeny-based Methods

Program	Phylogeny Method	Confidence Method	Specific Genes Req'd	Interface
EPA	ML	bootstrap, other	multiple alignment	command line, web-based
FastTree	other	bootstrap	multiple alignment	command line
green genes (NAST, Simrank)	other		16S rRNA	web-based
pplacer	ML, Bayesian	posterior probability, other	multiple alignment	command line

Combined Similarity and Composition-based Methods

Program	Similarity Method	Composition Method	Machine Learning	Specific Genes Req'd	Interface
SPHINX	BLAST	k-means/kNN	supervised		web-based

Combined Similarity and Phylogeny-based Methods

Program	Similarity Method	Phylogeny Method	Confidence Method	Specific Genes Req'd	Interface
AMPHORA	HMM	other	bootstrap	marker genes	command line
MLTreeMap	BLAST, HMM	ML	bootstrap, other	marker genes	command line, web-based
SAP	BLAST	Bayesian, other	posterior probability, other		command line
Treephyler	HMM	other	bootstrap	marker genes	command line

Table 2.1: Sequence classification program attributes and characteristics.

time to evaluate them all, so they pick a couple of popular or well-known tools. In contrast, we focused our comparisons on a single category at a time, which we believe generates more interesting and generally useful comparisons between programs that are conceptually similar.

We evaluated the performance of sequence classification programs in two main areas:

1. *assignment accuracy* — we tested assignment accuracy using data sets from the publications associated with each program, and analyzed each data set with as many programs from the corresponding category as possible. Specifically, we measured assignment sensitivity (*number of correct assignments* / *number of sequences in the data set*), precision (*number of correct assignments* / *number of assignments made*), the overall fraction of reads that were assigned, and the taxonomic rank at which assignments were made. (In general, more specific taxon assignments are more useful, although one usually expects sensitivity and precision to decrease as increasingly specific assignments are made.)

2. *resource requirements* (processing time, RAM, and disk requirements) — we monitored the resources consumed by each program during the analysis of each data set. Some programs have web services available that we used in program evaluation, which made it more difficult to precisely measure resource consumption.

2.2 Results

Within each category, we selected a subset of programs to evaluate. Programs were selected on the basis of several factors: if they were actively maintained, how popular they were, how recently they were published, if they were superseded by another program, and so on. From this standpoint, we attempted to make the comparisons in each category as interesting and useful to the current active community of researchers as possible.

2.2.1 Alignment

In the alignment category, we selected five programs to evaluate: CARMA (command line version 3.0), FACS (1.0), MEGAN (4.61.5), MG-RAST (3.0), and MetaPhyler (1.13). Based on our experience using these programs, we note the following:

1. FACS requires bloom filters to be built for the reference sequences that are to be searched, which is infeasible to do for large databases (e.g., GenBank's non-redundant nucleotide (nt) and protein (nr) databases). Therefore, we were unable to analyze the majority of data sets with FACS.

2. We ran BLASTX with default parameters against the nr database, and used this as input to CARMA and MEGAN. BLAST accounted for 96.40% and 99.97% of the total runtime for these workflows, respectively (Table 2.5).

3. MG-RAST has several different analysis options. We used the non-redundant multi-source annotation database, or M5NR, and their implementation of an

LCA algorithm for taxon assignment.

4. MG-RAST requires input sequences to contain protein-encoding genes (PEGs), and assigns each of these to a particular taxon. Not all query sequences in a random shotgun sample will contain a PEG, so MG-RAST typically classifies fewer overall sequences than other methods. In addition, it is possible for a single input sequence read to contain multiple PEGs. In order to be consistent with other methods that make classifications on a read-by-read basis, we map the PEG assignments back the read they came from, and make fractional read assignments to a particular taxon as necessary. (For example, a particular read could contain two PEGs: one PEG assigned to phylum A, and the other PEG assigned to phylum B. If only one of these is correct, the read would contribute 0.5 to a tally of "correct" assignments, and 0.5 to a tally of "incorrect" assignments.)

5. MetaPhyler requires input sequences to contain certain "marker genes" (protein-coding genes that are "universal" and occur only once per genome), an approach pioneered by AMPHORA. Very few query sequences in a random shotgun sample will contain marker genes, so MetaPhyler typically classifies fewer overall sequences than other methods; many fewer than even MG-RAST, for example.

Four data sets were selected for analysis with each of the alignment-based programs. Percentage of sequence classified, sensitivity, precision, and resource consumption are shown for the alignment-based programs in Table 2.5. What follows is a short description of each data set, and a summary of the results of analysis with each program.

2.2.1.1 FACS 269 bp high complexity 454 metagenomic data set

This data set, which consists of 10^5 sequences of average length 269 bp, originally used by Stranneheim et al. [145], was downloaded from the FACS web site. The sequences are from 19 bacterial genomes, three viral genomes, and two human chromosomes. The distribution of sequences is as follows: 73.0% Eukaryota, 25.6% bacteria, and 1.5% viruses.

It was reported that FACS assigned sequences to species with 99.8% sensitivity and 100% specificity using a k-mer size of 21 and a match cutoff of 35% sequence similarity [145]. However, we encountered technical difficulties using the FACS software and were unable to reproduce the results reported in the FACS paper.

Distribution of sequence assignments produced by the alignment-based programs is shown in Table 2.2.

2.2.1.2 MetaPhyler 300 bp simulated metagenomic data set

This data set, which consists of 73,086 sequences of length 300 bp, originally used by Liu et al. [149], was acquired from the authors. The sequences are simulated reads from 31 phylogenetic marker genes from bacterial genomes. The distribution of sequences into bacterial phyla is as follows: Proteobacteria, 47.0%; Firmicutes, 21.9%; Actinobacteria, 9.7%; Bacteroidetes, 4.8%; Cyanobacteria, 3.9%; Tenericutes, 2.2%; Spirochaetes, 1.9%; Chlamydiae, 1.3%; Thermotogae, 0.9%; Chlorobi, 0.9%.

Although a comparison of MetaPhyler, MEGAN, CARMA, and PhymmBL is already given for this data set [149], we decided to redo these analyses in a way that is consistent with our standard procedures (i.e., we did not exclude query reads from the reference database, as Liu et al. did with three out of four of their analyses; viz., MetaPhyler, MEGAN, and PhymmBL). Additionally, we restricted our analyses to the phylum rank.

The distribution of sequence assignments produced by the alignment-based programs is shown in Table 2.3.

2.2.1.3 CARMA 265 bp simulated 454 metagenomic data set

This data set, which consists of 25,000 sequences of average length 265 bp, originally used by Gerlach and Stoye [144], was acquired from the WebCARMA web site. The sequences are simulated 454 reads from 25 bacterial genomes. The distribution of sequences into bacterial phyla is as follows: Proteobacteria, 73.0%; Firmicutes, 12.9%; Cyanobacteria, 7.8%; Actinobacteria, 5.2%; Chlamydiae, 1.0%.

The distribution of sequence assignments produced by the alignment-based programs is shown in Table 2.4.

	actual	CARMA	MEGAN	MetaPhyler	MG-RAST
percentage of sequence classified		29.0	54.4	0.2	27.1
Eukaryota	73.0	30.3	42.0	0.0	21.0
Bacteria	25.6	62.8	52.0	84.0	71.5
Viruses	1.5	0.0	0.3	0.0	0.1
Archaea	0.0	6.9	5.7	16.0	7.3
percentage of sequence misclassified		8.0	12.2	16.0	7.6
correlation coefficient		0.45	0.72	-0.09	0.26

Table 2.2: Results for the FACS simHC metagenomic data set (10^5 sequences, 269 bp). The actual distribution of sequences compared to the distribution inferred by the alignment-based programs.

	actual	CARMA	MEGAN	MetaPhyler	MG-RAST
percentage of sequence classified		93.6	88.2	80.9	29.8
Proteobacteria	47.0	47.6	44.5	48.3	46.7
Firmicutes	21.9	22.2	24.0	21.8	23.1
Actinobacteria	9.7	8.7	8.8	9.1	9.3
Bacteroidetes	4.8	4.5	4.8	4.3	4.4
Cyanobacteria	3.9	3.6	3.8	3.9	3.7
Tenericutes	2.2	2.5	2.7	2.4	2.3
Spirochaetes	1.9	2.4	2.6	2.3	2.2
Chlamydiae	1.3	1.9	2.0	1.8	1.8
Thermotogae	0.9	1.2	1.2	1.1	1.2
Chlorobi	0.9	1.4	1.5	1.3	1.4
percentage of sequence misclassified		0.3	0.3	0.3	0.2
correlation coefficient		≈ 1.0	≈ 1.0	≈ 1.0	≈ 1.0

Table 2.3: Results for the MetaPhyler simulated metagenomic data set (73,086 sequences, 300 bp). The actual distribution of sequences compared to the distribution inferred by the alignment-based programs.

2.2.1.4 PhyloPythia 961 bp simMC data set

This data set, which consists of 124,941 sequences of average length 961 bp, originally used by Patil et al. [169], was downloaded from the FAMeS [170] web site. All classifications were performed at the genus rank.

2.2.1.5 Discussion

From the alignment-based analyses, we can make several observations.

1. The BLAST step completely dominates the runtime for alignment-based methods. It can use a fair amount of disk space in the process (as much as 17 GB for the MetaPhyler data set), and can use a considerable amount of RAM if analyzing a large number of sequences on a single node.

2. MetaPhyler is the one exception to the previous observation; its BLAST step and subsequent algorithmic steps run extremely quickly, but it generally only classifies a small fraction of reads in a typical sample. Also, Table 2.5 shows that MetaPhyler uses a large amount of RAM (5.6 GB); this is in part due to a memory leak that has been fixed in a subsequent release (personal correspondence with the author).

3. The MG-RAST web service showed a large variance in time required to receive results, although there is at least a weak correlation with data set size and analysis parameters. With a web service, it is difficult to know what other variables affect time to results (e.g., load on cluster queues), and currently the

MG-RAST server does not provide an estimate of how long a given submission will take.

4. For the FACS high complexity data set, none of the programs produced a taxonomic distribution that was remotely close to the known distribution (Table 2.2); all greatly underestimated the amount of eukaryotic DNA. The reason for this is unclear.

5. For the MetaPhyler 300 bp data set, all four alignment programs recapitulated the known distribution of bacterial phyla extremely well (Table 2.3). All had near-perfect precision, and sensitivity was greater than 80% for three out of four of the programs (Table 2.5). MG-RAST only had sensitivity of 30%, but this was still enough assignments to accurately estimate the taxonomic distribution (Pearson's $r \approx 1$).

6. For the CARMA 265 bp data set, CARMA, MEGAN, and MG-RAST recapitulated the known distribution of bacterial phyla extremely well (Table 2.4). MetaPhyler was slightly worse, but still quite good considering that it only classified 0.5% of sequences.

7. For the PhyloPythia 961 bp data set, all programs except MetaPhyler displayed comparable sensitivity and precision (Table 2.5).

8. Methods that use marker genes (MetaPhyler and MG-RAST) are generally less sensitive than methods that do not use marker genes (CARMA and MEGAN), but marker-based methods typically run faster (Table 2.5). All methods dis-

played comparable overall precision; CARMA and MG-RAST were the most precise (Table 2.5).

2.2.2 Composition

In the composition category, we selected four programs to evaluate: Naive Bayes Classifier (NBC, version 1.1), PhyloPythiaS (1.1), PhymmBL (3.2), and RAIphy (1.0.0). Based on our experience using these programs, we note the following:

1. All four programs need to be "trained" (classifiers built on training data) before they can be used to classify unknown query sequences. Training times for all four programs can be found in Table 2.6.

2. NBC, PhyloPythiaS, and PhymmBL were all trained on the latest microbial genomes in the RefSeq [171] database.

3. The database we used for RAIphy is the one currently available on the RAIphy web site, which was built from RefSeq in 2010. We built our own database using the latest version of RefSeq and retrained RAIphy with this updated database, but found that classification accuracy was drastically lower. We contacted the developers about the problem, but no satisfactory explanation was found.

4. Technical limitations having to do with memory usage or program bugs required us to break up our FASTA input files into multiple, smaller input files to use with PhyloPythiaS and PhymmBL.

5. NBC produces raw output as hundreds of large matrices, in which the rows represent genomes and the columns represent sequence reads. The value in a particular cell is the score given by the algorithm for assigning a particular sequence read to a particular genome. Therefore, it was necessary to parse this output to find the largest score in each column in order to assign each read to a particular taxon.

Three data sets were selected for analysis with each of the composition-based programs. Percentage of sequence classified, sensitivity, precision, and resource consumption are shown for the composition-based programs in Table 2.6. What follows is a short description of each data set, and a summary of the results of analysis with each program.

2.2.2.1 PhyloPythia 961 bp simMC data set

This data set, which consists of 124,941 sequences of average length 961 bp, originally used by Patil et al. [169], was downloaded from the FAMeS [170] web site. All classifications were performed at the genus rank.

2.2.2.2 PhymmBL 243 bp RefSeq data set

This data set, which consists of 80,215 sequences of average length 243 bp, originally used by Brady and Salzberg [157], was downloaded from the PhymmBL web site. All classifications were performed at the genus rank.

2.2.2.3 RAIphy 238 bp RefSeq data set

This data set, which consists of 477,000 sequences of average length 238 bp, originally used by Nalbantoglu et al. [158], was downloaded from the RAIphy web site. All classifications were performed at the genus rank.

2.2.2.4 Discussion

From the composition-based analyses, we can make several observations.

1. PhyloPythiaS took the longest to train (≈ three days), but its classification step was relatively fast (≈ 41× faster than PhymmBL). However, the fastest program was RAIphy, which took a negligible amount of time to train, and classified sequences ≈ 4× faster than PhyloPythiaS and ≈ 159× faster than PhymmBL (Table 2.6).

2. NBC displayed the highest average sensitivity and precision (97.4%), and PhymmBL displayed the second-highest average sensitivity and precision (≈ 76%) (Table 2.6).

3. PhyloPythiaS displayed very low average sensitivity (2.4%), but competitive average precision (70.9%) (Table 2.6).

4. Average precision is lower for composition-based programs than for alignment-based programs, but this is probably mainly due to the fact that classifications were made at the genus rank for composition-based classifications, and primarily at the phylum rank for alignment-based classifications (Tables 2.5 and 2.6).

5. Composition-based programs are supposed to excel at classifying sequences that are not exactly represented in the database, so it would be interesting to compare the performance of these programs in that type of analysis (cf. “clade-level exclusions” in Brady and Salzberg [157]).

2.2.3 Phylogenetics

In the phylogenetics category, we selected two programs to evaluate: MLTreeMap (version 2.061) and Treephyler (1.1). Based on our experience using these programs, we note the following:

1. The MLTreeMap web interface limits an analysis to 50,000 sequences, so we used the command line version. The MLTreeMap workflow makes callouts to BLAST, Gblocks [172], HMMER [173], and RAxML [174], and is very sensitive to the versions of these dependencies used, so it is important to use the specific versions of these programs that are bundled with MLTreeMap.

2. Treephyler requires the input sequences to be converted to amino acids, and the corresponding UFO [175] assignments to be provided. Thus, we performed a 6-frame translation of our DNA input sequences, and used the UFO web server to assign protein sequences to Pfam domains. These files were then used as input to Treephyler.

3. Treephyler is capable of utilizing multiple processing cores during analysis.

The only simulated data set associated with the MLTreeMap and Treephyler publications is the simulated medium complexity (simMC) PhyloPythia data set, so we analyzed this with both programs. Percentage of sequence classified, sensitivity, precision, and resource consumption are shown for the phylogenetics-based programs in Table 2.7.

2.2.3.1 PhyloPythia 961 bp simMC data set

This data set, which consists of 124,941 sequences of average length 961 bp, originally used by Patil et al. [169], was downloaded from the FAMeS web site. All classifications were performed at the genus rank.

2.2.3.2 Discussion

From the phylogenetics-based analyses, we can make several observations.

1. Treephyler took twice as long to run as MLTreeMap, but was ≈ 8× more sensitive and achieved higher precision. (Table 2.7).

2. MLTreeMap and Treephyler made some assignments at taxonomic ranks higher than genus that were not included in this analysis, but would otherwise be of interest.

3. MLTreeMap and Treephyler are capable of producing measures of confidence of assignment, which we did not include in this analysis but would be of practical use in most scenarios.

2.2.4 Comparison of all programs

All ten programs were used to analyze the simulated medium complexity (simMC) PhyloPythia data set, so it is of interest to compare their relative performance on this particular data set.

1. Composition-based programs displayed the highest average sensitivity (50.4%), and alignment-based programs displayed the highest average precision (93.7%) (Tables 2.5 and 2.6).

2. The two most computationally expensive programs, CARMA and MEGAN, achieved the highest precision (97.4% and 98.1%, respectively) (Table 2.5).

3. In terms of the best combined sensitivity and precision, NBC outperformed all other programs, achieving sensitivity and precision of 95.4% (Table 2.6).

2.3 Conclusions

The performance of a particular category of programs varied substantially between data sets. The precise reasons for this are likely a complex function of sample taxonomic composition and diversity, level of sequence representation in databases, read lengths and read quality. In general, however, if a data set was challenging for one program, it was challenging for the other programs in that category. The overall variance of the statistics makes it difficult to make definitive statements about the superiority of one program or method over another, but we can state some broad conclusions.

In general, high sensitivity is undesirable if corresponding precision is low. However, very precise methods that do not assign a large fraction of sequences may still be useful, depending on the application. For example, we have shown that in some cases, classifying only a small percentage of a sample may still be enough to recapitulate the correct organismal distribution, especially at a high rank (e.g., phylum). Methods that search for marker genes in a metagenomic sample interrogate relatively few sequences, but as a consequence run quickly and with high precision. In a targeted sequencing experiment, phylogenetic methods and other methods that use marker genes might thus be especially appropriate.

In general, composition-based programs classified sequences the fastest, once they were trained. Phylogenetic programs might be the most computationally intensive on a per-read basis, but owing to their use of marker genes only ran for an intermediate amount of time in our experiments. As expected, BLAST-based programs that did not use marker genes consumed the bulk of the computing resources in our study. Researchers should take note of the fact that programs vary by orders of magnitude in computational resource requirements, and should thus choose programs appropriately depending on the computing resources they have access to, the amount of data to analyze, and the particular bioinformatics application. In addition, some programs are much easier to set up and use than others. Of course, there is often a tradeoff between the level of flexibility and configurability of a program, and its ease of use.

Taxonomic sequence classification is a fundamental step in metagenomic analyses, as classification accuracy has a direct impact on downstream analyses and the conclusions drawn from them. Therefore, it is important to be aware of the wide variety of tools that currently exist to address this need, and to choose the best performing and most appropriate tools for a given analysis and set of resource constraints.

2.4 Methods

2.4.1 Program classification

We created and filled in Table 2.1 by hand using appropriate literature, program web sites, and documentation as necessary. In order to cluster the programs, we wrote a Perl script to construct a matrix containing a measure of similarity, or distance, for each possible pair of programs, defined as follows:

$$distance(program1, program2) = \sum_{a=1}^{n} distance(program1[a], program2[a])$$

where n is the number of program attributes (equal to the number of columns in the table). A program attribute may have multiple values.

Distances are calculated as follows:

```
if program1[a] == program2[a] then
    distance(program1[a], program2[a]) = 0
else if common(program1[a], program2[a]) == 0 then
    distance(program1[a], program2[a]) = 1
```

```
else
    $\text{distance}(\text{program1}[\text{a}], \text{program2}[\text{a}]) = \frac{\text{common(program1[a],program2[a])}}{\text{greater(program1[a],program2[a])}}$
end if
```

where $common(program1[a], program2[a])$ is equal to the number of values the two attributes share in common, and $greater(program1[a], program2[a])$ is equal to the number of values in the program attribute with the greater number of values.

We provided the distance matrix as input to the NEIGHBOR program from the PHYLIP package [176]. We plotted the resulting neighbor-joining tree in FigTree [177] and labeled it to produce Figure 2.1.

2.4.2 Tool usage and result processing

We wrote custom Perl scripts to parse the correct annotations out of the FASTA headers of the various input files for each data set. The PhymmBL data files did not contain annotations, so we used NCBI E-Utilities to access the NCBI taxonomy database and retrieve the scientific classification for each sequence. We used custom Perl scripts to parse out of program output files the classifications made by each program, and compared these to the correct annotations to calculate sensitivity and precision.

We used Pearson's correlation coefficient (via the `cor()` function in R [178]) to compare the known distribution of bacterial phyla to the classifications made by the various alignment programs.

We calculated runtimes in minutes of wall clock time; if a process ran in parallel, then we multiplied the runtime by the number of parallel processes. The runtimes are not directly comparable because the analyses used heterogeneous hardware. We calculated memory usage by manually inspecting process memory usage intermittently, which is error-prone. Despite their shortcomings, both measures should still be useful as the basis for a rough comparison.

2.5 Use of sequence classification programs in phylogenomics

Our research finds significant variability in classification accuracy, precision, and resource consumption of sequence classification programs when used to analyze various metagenomics data sets. However, the general trends and patterns that we observe are useful to be aware of when conducting various types of bioinformatics analyses.

In a typical metagenomics workflow, for example, a sequence classification program is used to identify the organism from which each read in the sample originated. In the context of an RNA-Seq-based phylogenomic workflow, it may be useful to treat the collection of assembled transcripts as a metagenome, because in addition to the organism one sets out to sequence, one may also (perhaps unknowingly) sequence other organisms that may be of interest. For example, when sequencing Lepidoptera, one may expect the majority of reads to be of lepidopteran origin. However, a fraction of the reads may belong to symbiotic organisms such as Microsporidia [179], or may be contaminants (human or bacterial). Therefore, before proceeding with downstream analysis steps, it can be useful to partition the reads or transcripts according to the organism from which they likely originated.

In our comparison of sequence classification programs, we found that the following programs exhibited an ideal balance between performance (assignment accuracy and precision) and computational resource requirements: CARMA [144], MEGAN [148, 180], and MG-RAST [150]. Hence, we would be most likely to use these particular programs in a phylogenomic workflow to assign reads or transcript fragments to various types of organisms. Upon doing this, we might proceed with downstream analysis steps using only the subset of reads or transcript fragments assigned to a particular organismal group (e.g., Lepidoptera).

	actual	CARMA	MEGAN	MetaPhyler	MG-RAST
percentage of sequence classified		68.7	90.5	0.5	80.2
Proteobacteria	73.0	73.2	73.0	69.2	73.2
Firmicutes	12.9	13.2	12.8	17.3	12.9
Cyanobacteria	7.8	7.3	7.8	6.8	7.6
Actinobacteria	5.2	5.0	5.3	2.3	5.4
Chlamydiae	1.0	1.2	1.1	4.5	0.9
percentage of sequence misclassified		0.3	0.2	0.0	0.1
correlation coefficient		≈ 1.0	≈ 1.0	≈ 1.0	≈ 1.0

Table 2.4: Results for the CARMA 454 simulated metagenomic data set (25,000 sequences, 265 bp). The actual distribution of sequences compared to the distribution inferred by the alignment-based programs.

Program	FACS 269 bp	MetaPhyler 300 bp	CARMA 265 bp	PhyloPythia 961 bp	Mean
			Percentage of sequence classified		
CARMA	29.0	93.6	68.7	61.3	63.2
MEGAN	48.4	88.2	90.5	62.2	72.3
MetaPhyler	0.2	80.9	0.5	0.6	20.6
MG-RAST	27.1	29.8	80.2	70.5	51.9
			Sensitivity (percentage)		
CARMA	26.7	93.4	68.5	59.8	62.1
MEGAN	42.5	87.9	90.3	61.0	70.4
MetaPhyler	0.1	80.7	0.5	0.5	20.5
MG-RAST	25.0	29.7	80.1	67.2	50.5
			Precision (percentage)		
CARMA	92.0	99.7	99.7	97.4	97.2
MEGAN	78.1	99.7	99.8	98.1	93.9
MetaPhyler	84.0	99.7	100.0	83.8	91.9
MG-RAST	92.4	99.8	99.9	95.3	96.9
			CPU Runtime (minutes)		
CARMA[1,2]	290,880	77,340	74,950	360,107	200,819
MEGAN[1,2]	288,020	72,060	72,010	351,060	195,788
MetaPhyler[3]	10	20	2	28	15
MG-RAST[4]	60	10,080	20,160	12,960	10,815
			Memory Usage (Megabytes of RAM)		
CARMA	100	100	100	120	105
MEGAN	1024	1024	1024	1410	1121
MetaPhyler	5734	5734	5734	5734	5734
MG-RAST[5]	-	-	-	-	-

[1] analysis performed on a 2.66 GHz Intel Core i7 MacBook Pro running Mac OS X 10.7.1 with 8 GB 1067 MHz DDR3 RAM.

[2] BLAST v2.2.18 analysis performed using ≈ 200 Opteron 2425 HE (2.1 GHz) cores; each node had 48 GB RAM.

[3] analysis performed on an AMD Opteron 250 (2.4 GHz) Sun Fire V40z with 32 GB RAM.

[4] used web service; recorded value is number of minutes to receive results, not actual CPU runtime.

[5] used web service; memory usage was unable to be determined.

Table 2.5: Performance of alignment-based programs. Measurements of sensitivity, precision, and resource consumption on four simulated data sets.

Program	PhyloPythia 961 bp	PhymmBL 243 bp	RAIphy 238 bp	Mean	Training
		Percentage of sequence classified			
NBC	100	100	100	100	
PhyloPythiaS	3.5	3.1	3.3	3.3	
PhymmBL	100	99.7	100	99.9	
RAIphy	100	100	100	100	
		Sensitivity (percentage)			
NBC	95.4	97.5	99.4	97.4	
PhyloPythiaS	3.1	1.8	2.2	2.4	
PhymmBL	48.4	96.8	81.9	75.7	
RAIphy	54.8	31.8	48.0	44.9	
		Precision (percentage)			
NBC	95.4	97.5	99.4	97.4	
PhyloPythiaS	88.1	58.5	66.1	70.9	
PhymmBL	48.4	97.0	81.9	75.8	
RAIphy	54.8	31.8	48.0	44.9	
		CPU Runtime (minutes)			
NBC[1]	13,496	3,595	17,573	11,555	1,217
PhyloPythiaS[2]	297	180	506	328	4,320
PhymmBL[1]	15,600	1,035	23,508	13,381	2,880
RAIphy[3]	105	25	122	84	30
		Memory Usage (Megabytes of RAM)			
NBC	200	200	200	200	
PhyloPythiaS[4]	100	100	100	100	
PhymmBL[4]	100	100	100	100	
RAIphy	500	335	400	412	

[1] analysis performed on an AMD Opteron 250 (2.4 GHz) Sun Fire V40z with 32 GB RAM.

[2] analysis performed on an AMD Opteron 248 (2.2 GHz) workstation with 8 GB RAM.

[3] analysis performed on a 2.66 GHz Intel Core i7 MacBook Pro running Mac OS X 10.7.1 with 8 GB 1067 MHz DDR3 RAM.

[4] input sequences were broken up into smaller files.

Table 2.6: Performance of composition-based programs. Measurements of sensitivity, precision, and resource consumption on three simulated data sets.

Program	% of sequence classified	Sensitivity (%)	Precision (%)	CPU Runtime (minutes)
MLTreeMap[1]	0.9	0.8	81.4	3,344
Treephyler[1]	6.6	6.3	95.7	7,444

[1] analysis performed on an AMD Opteron 250 (2.4 GHz) Sun Fire V40z with 32 GB RAM.

Table 2.7: Performance of phylogenetics-based programs. Measurements of sensitivity, precision, and resource consumption on the PhyloPythia 961 bp data set.

Chapter 3: Use of consensus sequences as an alternative to orthology determination

This chapter is based on the following publication: Adam L. Bazinet, Michael P. Cummings, and Antonis Rokas. Homologous gene consensus avoids orthology-paralogy misspecification in phylogenetic inference. Unpublished.

3.1 Background on orthology

Orthologs are homologous genes that have evolved from a single ancestral gene; i.e., the gene copies have arisen through a *speciation* event [181, 182]. As a consequence of this evolutionary relationship, one usually expects orthologous genes to have similar functions. Orthologs are frequently contrasted with paralogs [181, 182], genes that have arisen through a *duplication* event within a (possibly ancestral) species. The process of gene duplication and divergence has been proposed as a mechanism by which genes acquire new function [183, 184]. Figure 3.1 shows the evolution of the β-globin gene, which involves both duplication and speciation.

As a consequence of gene duplication and loss, incomplete lineage sorting, and lateral gene transfer, the evolutionary history of a particular gene (a "gene tree") may disagree with the species tree relating certain organisms [186]. These

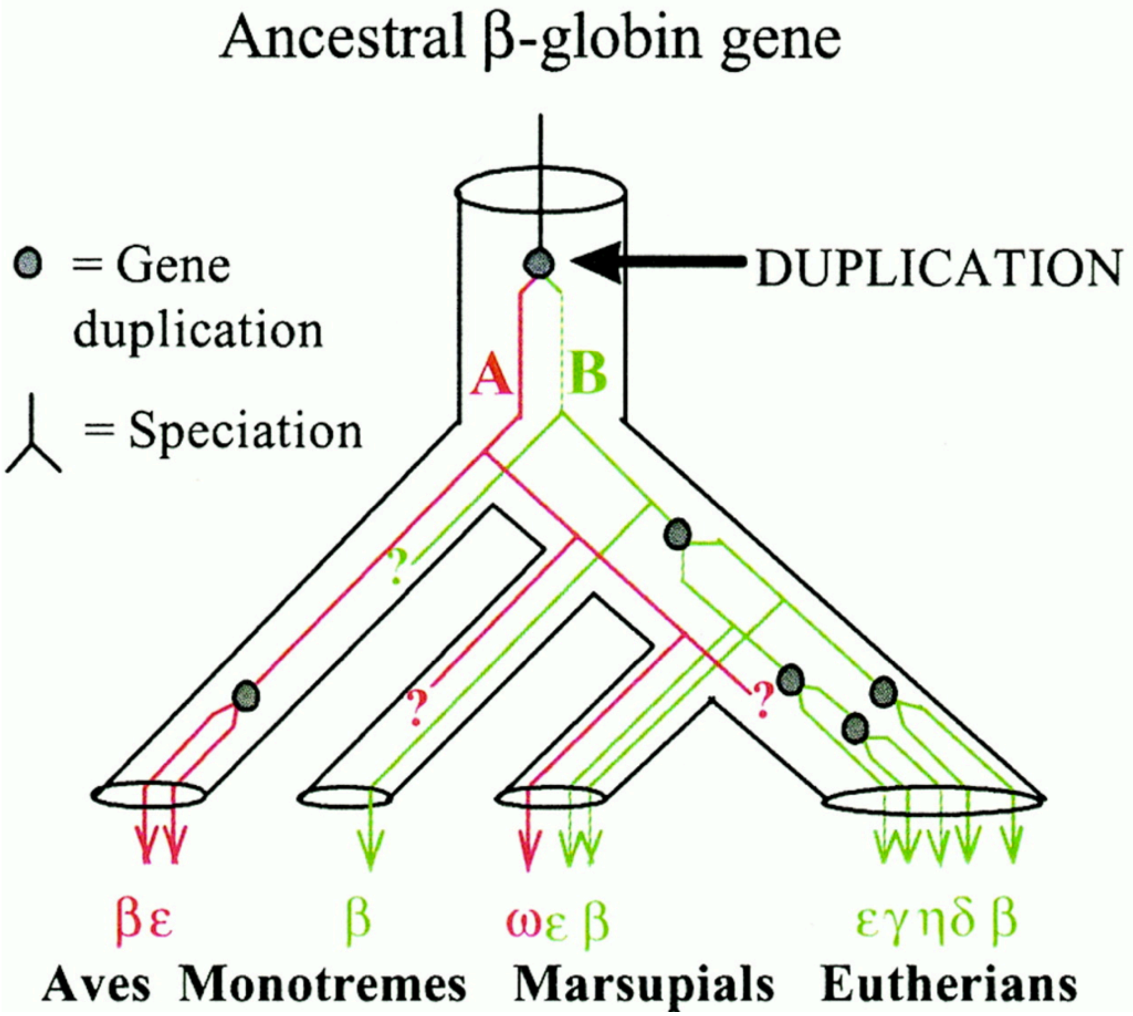


Figure 3.1: A model for the evolution of β-globin genes in mammals. The gene tree is drawn within the constraints of a species tree. The ancient gene duplication event (indicated by an arrow) gave rise to two ancestral genes, gene A (red) and gene B (green). Gene A was the progenitor of marsupial ω-globin and the β-like globin genes of birds. Gene B gave rise to the β-like globin genes of mammals. Genes or pseudogenes that may be expected to occur are indicated by question marks. To simplify the diagram, not all of the avian β-like globin genes (as exemplified by the chicken) are shown, and the eutherian genes shown are typical of humans and some primates. Figure from Wheeler et al. [185].

complications make phylogenetic inference of the species tree more challenging, as such inference typically makes use of many genes, each with its own individual history. Most of the phylogenetic methods mentioned throughout this dissertation assume that the sequences of a particular gene among a set of taxa are orthologous, so it is important to be able to identify orthologs among species and differentiate them from paralogs when selecting data for inclusion in an analysis. Relevant to data analyzed in this dissertation, Figure 3.2 shows phylogenetic relationships and orthologous gene information for 12 species of Insecta and Arachnida, which include some members of Lepidoptera.

The most widely-used method for orthology determination is known as "reciprocal best BLAST hit" (RBH) [188, 189], a criterion stating that a pair of genes, each of which belonging to a different taxon, may be designated orthologous if their protein products are found as the best hit for one another in a reciprocal similarity search of the two proteomes. Other interesting methods, such as reciprocal smallest distance (RSD) [190], have also been developed. These methods are far from perfect, however, and currently much attention is being paid to this area, both by computer scientists (algorithm developers), and genome annotators (users of orthology prediction tools). Combined expertise in these two areas is demonstrated by the sophistication of the Ensembl orthology determination pipeline [46], for example, which we leverage in Chapter 8.

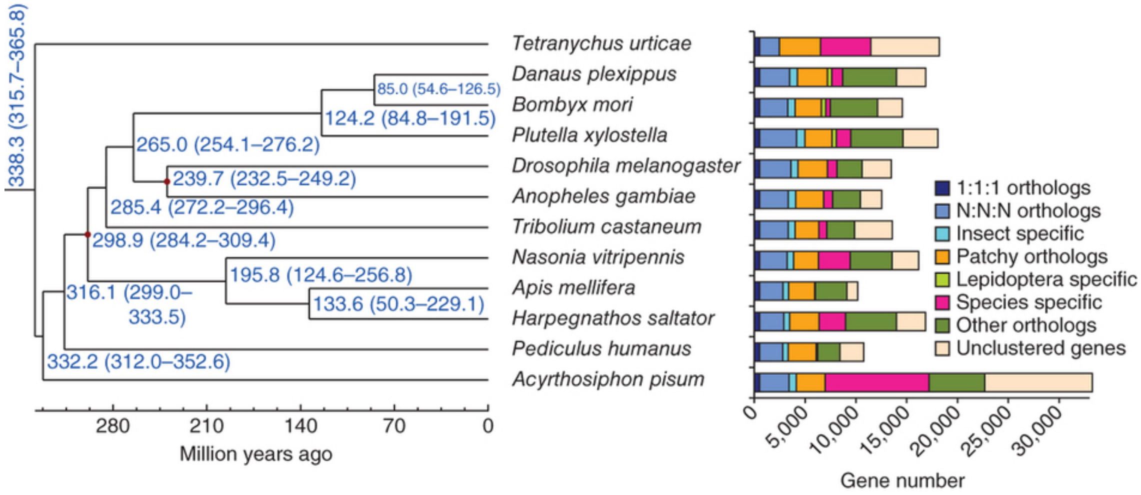


Figure 3.2: Number of orthologs found among arthropods. The red dots (for calibration) represent the divergence time of *Drosophila melanogaster* and Culicidae (295.4-238.5 mya) and the divergence time of *D. melanogaster* and *Apis mellifera* (307.2-238.5 mya), which are based on fossil evidence. The Arachnida, *Tetranychus urticae*, was used as an outgroup. 1:1:1 orthologs include the common orthologs with the same number of copies in different species; N:N:N orthologs include the common orthologs with different copy numbers in the different species; patchy orthologs include the orthologs existing in at least one species of vertebrates and insects; other orthologs include the unclassified orthologs; and unclustered genes include the genes that cannot be clustered into known gene families. Figure from You et al. [187].

3.2 Consensus sequences: an alternative to orthology determination

Orthology determination is fraught with challenges due to the variety and complexity of biological processes that govern the evolution of genes. The size of gene families frequently differs, even in closely-related species, which leads to possibly complex, “one-to-many” or “many-to-many” orthology assignments. Existing methods (such as RBH) attempt to assign orthology status to individual genes chosen from paralogous gene sets, and frequently make errors in doing so. The novel idea developed in this dissertation is to integrate over the uncertainty in orthology determination by representing a set of paralogous genes in a particular individual as a single consensus sequence, one that potentially includes ambiguous bases (using standard IUPAC nucleotide ambiguity codes [191]). Therefore, in a strict sense, we do not determine the orthology status of any individual gene sequence, thus mitigating orthology assignment errors that would result from choosing a single gene sequence. The consensus gene sequences from different individuals may subsequently be compared as if they were orthologs.

The performance of the consensus method can be evaluated using taxa whose true evolutionary relationships are known; in such cases, if phylogenetic analyses using consensus sequences from those taxa recover the correct tree, the consensus method is shown to be effective. Such analyses are described in the following section.

3.3 Validation of the consensus method

We validated the consensus method by applying it to phylogenetic problems in three different eukaryotic kingdoms: fungi, animals, and plants.

3.3.1 Yeast transcriptome analysis

We obtained transcriptome data from three yeast species that were described and analyzed in Scannell et al. [192] (*S. cerevisiae*, *S. castellii* and *C. glabrata*), along with a yeast outgroup. Based on observed patterns of evolution, Scannell et al. divided yeast genes into various classes (C0, C1, C2, and C4); some classes had subclasses denoted by a letter (e.g., C1A, C1B, etc.). In addition, we ran OrthoMCL [193] on the four yeast transcriptomes to create a new class ("C5") consisting of the 46 gene sets that had eight or more genes represented in all four species. For each class (or subclass), we created three types of ortholog sets: (1) a "gold standard" ortholog set using data from the Yeast Gene Order Browser (YGOB) database [194]; (2) an ortholog set created using the RBH implementation in OrthoMCL; and (3) an ortholog set of paralogs represented by a single consensus sequence (abbreviated "CON").

For each gene in each ortholog set, we performed a maximum likelihood search for the best tree using PAUP* [195], and we also performed 2,000 bootstrap replicates using a branch-and-bound search. Since there are four taxa in the data set, there are only three possible bifurcating topologies. The topology that represents the correct species tree for the four yeast species is denoted .**. (*S. cerevisiae* and *C. glabrata* form a monophyletic group).

Table 3.1 shows for each class (or subclass) and for each ortholog set (YGOB, RBH, and CON) the number (and corresponding percentage) of genes for which phylogenetic inference yielded each of the three possible topologies. For each ortholog set, we calculated the average percentage of genes across all classes for which tree inference found the correct topology. We observed that YGOB (our "gold standard") and CON (the consensus method) performed nearly identically, finding the correct topology in $\approx 35\%$ of cases. This was somewhat better than RBH (the most widely-used method for orthology determination), which only found the correct topology in $\approx 27\%$ of cases. However, the percentage of genes for which YGOB and CON recovered the correct topology ($\approx 35\%$) was barely better than random (which would have been $\approx 33\%$), so these results were not especially compelling on their own.

We were not surprised that phylogenetic analysis found the correct tree with only approximately one-third of the genes, as this is known to be a particularly challenging phylogenetic problem [196]. Furthermore, our criterion for correctness was rather strict; if we were to perform an approximately unbiased (AU) test [197], we would expect to find that some of the genes do not have enough signal to significantly reject the two alternative topologies.

Therefore, we decided to test the following hypothesis: as gene length increases, the percentage of genes for which the two orthology determination methods and the consensus method recover the correct topology increases concomitantly. We pooled the genes from classes where there was an equal number of genes in both the YGOB and CON data sets — C1A, C1B, C1C, C2B, C2D, and C2F — a total of 509

Class	Topology	YGOB		RBH		CON	
		Genes	Correct (%)	Genes	Correct (%)	Genes	Correct (%)
C0	.**.	172	36.1	61	25.8	99	41.9
C0	..**	154	32.3	38	16.1	65	27.5
C0	.*.*	151	31.7	40	16.9	72	30.5
C1A	.**.	16	31.4	13	25.5	17	33.3
C1A	..**	17	33.3	3	5.9	14	27.5
C1A	.*.*	18	35.3	13	25.5	20	39.2
C1B	.**.	9	33.3	7	25.9	8	29.6
C1B	..**	11	40.7	8	29.6	10	37
C1B	.*.*	7	25.9	4	14.8	9	33.3
C1C	.**.	45	32.4	22	16.1	49	35.8
C1C	..**	37	26.6	18	13.1	35	25.5
C1C	.*.*	57	41	37	27	53	38.7
C2A	.**.			1	33.3	3	50
C2A	..**			1	33.3	2	33.3
C2A	.*.*			1	33.3	1	16.7
C2B	.**.	64	42.1	49	32.5	69	45.7
C2B	..**	45	29.6	34	22.5	44	29.1
C2B	.*.*	43	28.3	29	19.2	38	25.2
C2C	.*.*			1	100	2	100
C2D	.**.	29	33.3	21	24.1	27	31
C2D	..**	20	23	12	13.8	23	26.4
C2D	.*.*	38	43.7	30	34.5	37	42.5
C2F	.**.	18	32.1	18	32.1	18	32.1
C2F	..**	15	26.8	5	8.9	20	35.7
C2F	.*.*	23	41.1	17	30.4	18	32.1
C4	.**.	1,285	36.7	1,218	34.7	1,285	36.7
C4	..**	995	28.4	946	27	995	28.4
C4	.*.*	1,222	34.9	1,172	33.4	1,222	34.9
C5	.**.					10	23.3
C5	..**					15	34.9
C5	.*.*					16	37.2
Average percent correct			34.7		27.1		35.8

Class	Topology	YGOB		RBH		CON	
		Samples	Correct (%)	Samples	Correct (%)	Samples	Correct (%)
CONCAT10K	.**.	4,080	40.8	4,965	49.7	4,562	45.6
CONCAT10K	..**	2,010	20.1	1,327	13.3	2,060	20.6
CONCAT10K	.*.*	3,910	39.1	3,708	37.1	3,378	33.8

Table 3.1: Percentage of genes that recovered the correct topology for various yeast data sets and orthology determination methods. Classes C0-C4 are from Scannell et al. [192]. In addition, we ran OrthoMCL on the four yeast transcriptomes to create a new class ("C5") consisting of the 46 gene sets that had eight or more genes represented in all four species. The correct topology is shown in the first row of each class, denoted by .**. RBH percentages are corrected (using the total number of genes from the corresponding CON entry) to account for missing observations due to lack of a significant BLAST hit.

genes. Then we binned genes according to the number of parsimony-informative characters they contained. The histogram in Figure 3.3 shows no indication that an increase in gene length is positively correlated with an increased probability of recovering the correct tree.

Thus, we hypothesized that even the longest single genes still did not carry enough phylogenetic signal to recover the correct topology, so we decided to move beyond single gene analysis to concatenated gene analysis. From the pool of 509 genes, we drew 10 genes at random from the YGOB ortholog set and concatenated them to form a single sample alignment. We then took the exact corresponding genes from the RBH and CON ortholog sets to build analogous RBH and CON sample alignments, respectively[1]. We drew 10^4 such samples, and for each sample we performed the same phylogenetic analyses described previously (ML search and bootstrap analysis using PAUP*). The results, shown in Table 3.1, indicated an improvement in all three methods: they all found the correct topology at least 40% of the time. Perhaps surprisingly, CON (45.6% correct) outperformed YGOB (40.8% correct), and even more surprisingly, RBH (49.7% correct) performed the best of all three methods.

The Venn diagram in Figure 3.4 shows the overlap among the YGOB, CON, and RBH concatenated samples that recovered the correct tree. We see, for example, that

[1] In the case of RBH, if the corresponding gene was missing, we simply omitted it. Thus, RBH samples contained, on average, a couple fewer genes. We felt that this was an appropriate "penalty" under the basic assumption that a longer alignment affords a better chance to recover the correct topology.

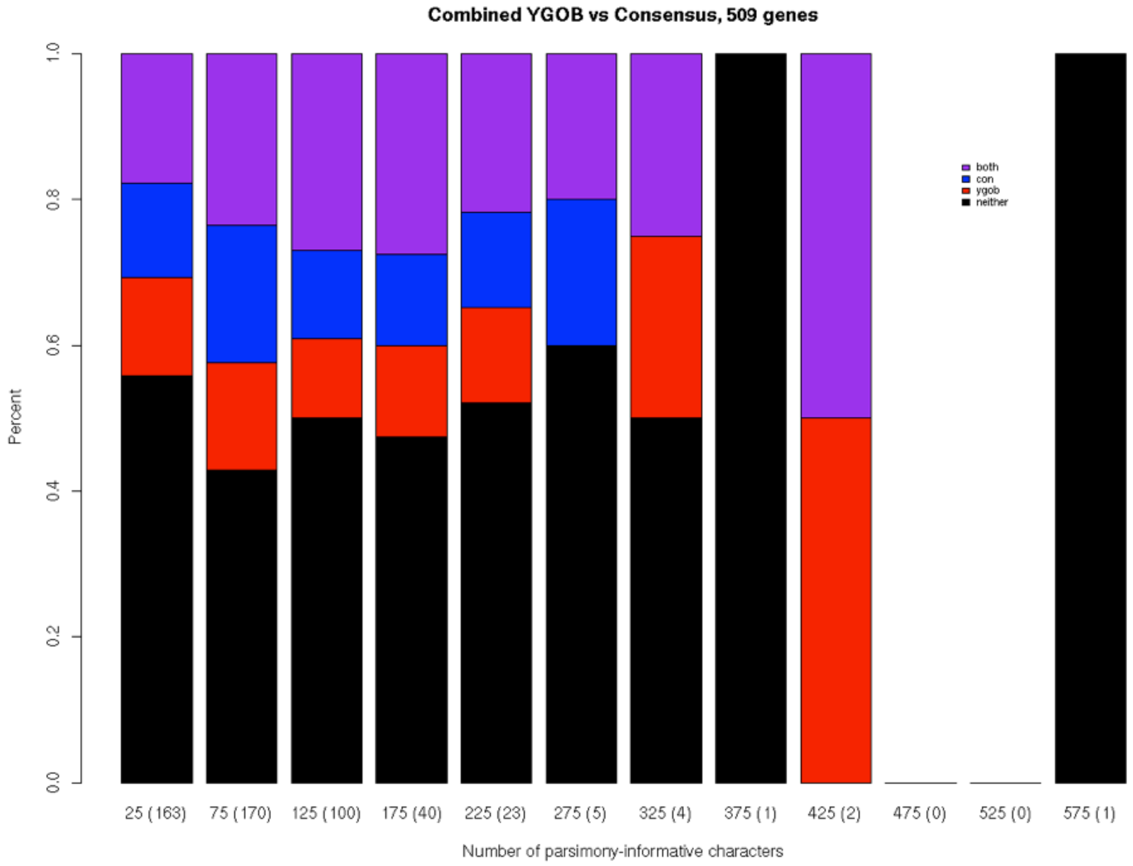


Figure 3.3: Histogram showing the breakdown of the percentage of genes for which YGOB and CON ortholog sets recover the correct tree as gene length increases. Along the x-axis are buckets of size 50. The number of genes in the bucket is shown in parentheses.

for 13.4% of the samples, both orthology determination methods and the consensus method recovered the correct tree, whereas for 21.2% of the samples, all three methods failed to recover the correct tree. Interestingly, CON had the largest percentage of samples that recovered the correct tree when the others did not (17.8%).

Finally, we created three scatterplots (all pairwise combinations of data sets) that show, for a subset of the concatenated samples, the percentage of the 2,000 bootstrap replicates that recovered the correct tree (Figure 3.5). Our objective was to see if any of the three data sets significantly outperformed the other two in terms of tree inference robustness as measured by bootstrapping. However, we found that overall, the percentages were fairly even (Figure 3.5).

From the yeast analysis, we conclude that the consensus method is competitive with the two orthology determination methods we tested.

3.3.2 Vertebrate proteome analysis

We isolated 39 homologous gene groups from seven vertebrate proteomes and applied the consensus method to the 1,563 individual gene sequences so that each taxon/locus combination was represented by a single consensus sequence. We then concatenated these 39 genes to form a CON vertebrate data set.

To construct a competing reciprocal best BLAST hit (RBH) data set, we used OrthoMCL as with the yeast data (Section 3.3.1), which resulted in 2,129 alignment files containing 14,903 gene sequences. We searched these files for occurrences of sequence identifiers that were also present in the CON vertebrate data set, which

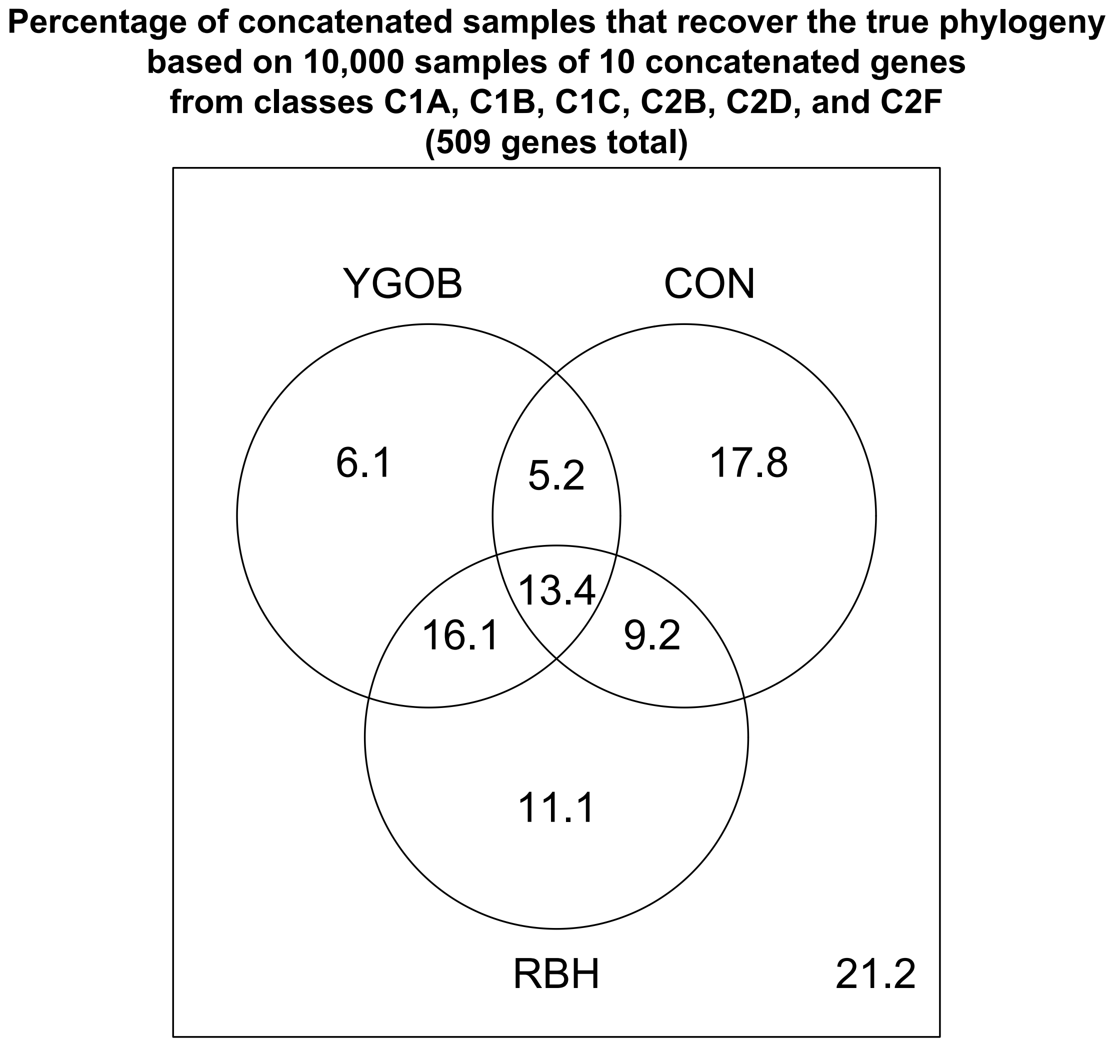


Figure 3.4: Venn diagram showing overlap among the YGOB, CON, and RBH concatenated samples that recovered the correct tree.

returned eight loci. We concatenated these eight genes to form an RBH vertebrate data set.

We performed 100 maximum likelihood searches for the best tree on both the CON and RBH data sets using GARLI 2.0 [198] with a GTR+I+G nucleotide model. The tree with the highest likelihood score from each analysis was selected for comparison to the canonical vertebrate phylogeny, which is as follows:

(((((Human,Mouse),Dog),Chicken),Frog),(Zebrafish,Pufferfish));

We ran TREEDIST from the PHYLIP [199] package to calculate the symmetric distance between the CON/RBH vertebrate trees and the reference phylogeny. For the purpose of comparing tree distance between and among data sets, we also report a normalized distance based on the maximum symmetric tree difference for a pair of trees (2n-6, where n is the number of taxa).

distance between reference and CON = 2 (normalized distance = 2/(2(7)-6) = 0.25)

distance between reference and RBH = 0 (normalized distance = 0/(2(7)-6) = 0.00)

In this analysis, the RBH tree matched the reference phylogeny exactly, and the CON tree was almost correct.

3.3.3 Plant proteome analysis

We selected 11 plant genes and downloaded the corresponding files from the PLAZA [200] database, in which 23 plant species were represented (Figure 3.6). We applied the consensus method to the 618 gene sequences therein, and concatenated these 11 genes to form a CON plant data set.

To construct the reciprocal best BLAST hit (RBH) data set, we used the same methodology as with the yeast and vertebrate data (Sections 3.3.1 and 3.3.2, respectively), which resulted in each taxon being represented by exactly one sequence in each gene file. We concatenated these 11 genes to form a RBH plant data set.

We performed 100 maximum likelihood searches for the best tree on both of these data sets using GARLI 2.0 with a GTR+I+G nucleotide model. The tree with the highest likelihood score from each analysis was selected for comparison to the canonical PLAZA phylogeny, which is shown in Figure 3.6.

We ran TREEDIST from the PHYLIP package to calculate the symmetric distance between the CON/RBH plant trees and the reference phylogeny. For the purpose of comparing tree distance between and among data sets, we also report a normalized distance based on the maximum symmetric tree difference for a pair of trees (2*n*-6, where *n* is the number of taxa).

distance between reference and CON = 8 (normalized distance = 8/(2(23)-6) = 0.2)

distance between reference and RBH = 4 (normalized distance = 4/(2(23)-6) = 0.1)

We found that both trees were quite similar to the reference phylogeny, and were also very similar to one another.

3.4 Summary

In our validation studies, we found that the consensus method performed comparably to the orthology determination methods we tested. Perhaps this comments on the relative performance of existing orthology determination methods, as intuitively one might not have expected the consensus method to perform as well as RBH, for example, which is more algorithmically complex and computationally intensive. Generally speaking, the simplicity of the consensus method may allow it to be applied in situations where orthology status cannot be easily determined by other methods.

In Chapter 7, we apply the consensus method to the analysis of Leptree data in the context of a phylogenomic workflow, and evaluate its utility compared to that of selecting a single representative sequence per locus (Appendix A). We find that the consensus method is competitive with, and even sometimes outperforms the representative sequence selection method, and thus in Chapter 8 we transition to using the consensus method exclusively.

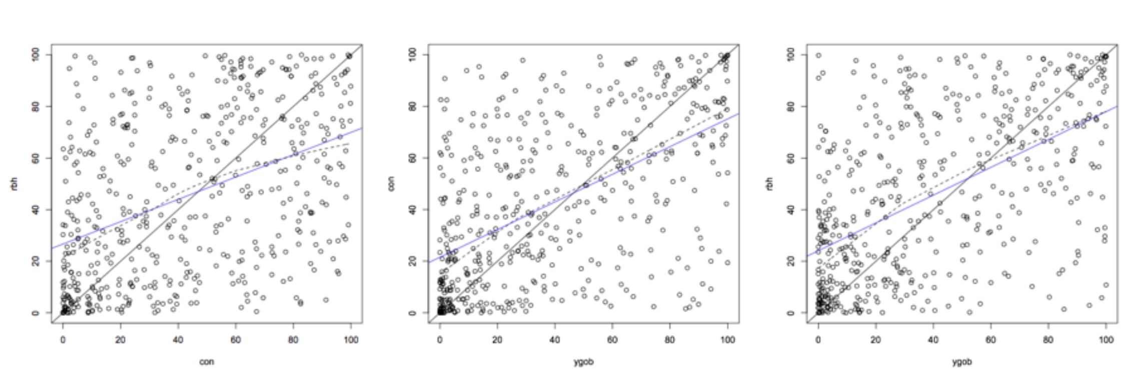


Figure 3.5: Bootstrap scatterplots for all pairwise combinations of YGOB, RBH, and CON for a random subset of the concatenated data sets. The blue line is a least squares fit line, and the dashed line is a LOWESS line.

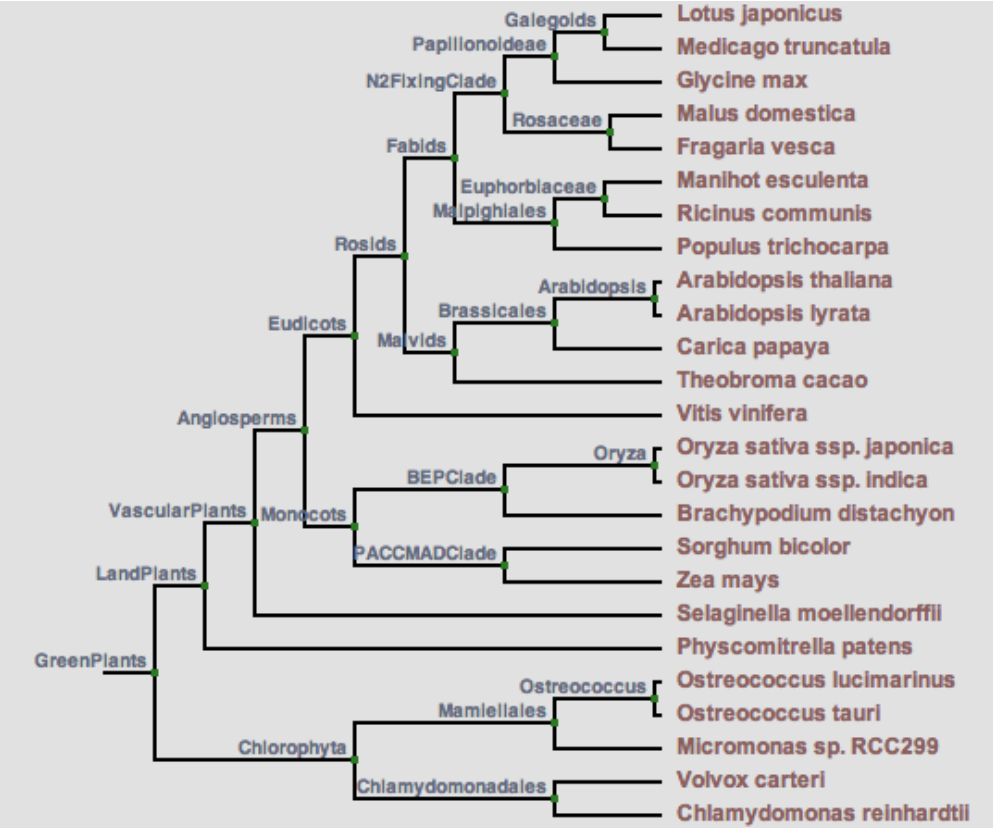


Figure 3.6: The canonical PLAZA phylogeny, taken from `http://bioinformatics.psb.ugent.be/plaza/`. At the time we performed the plant proteome analysis, *Fragaria vesca* and *Theobroma cacao* were not included in the phylogeny.

Chapter 4: The GARLI web service

This chapter is based on the following publication: Adam L. Bazinet, Derrick J. Zwickl, and Michael P. Cummings. A gateway for phylogenetic analysis powered by grid computing featuring GARLI 2.0. *Systematic Biology*, 63(5):812-818, 2014. Corrections included.

4.1 Introduction

The most widely used modern statistical methods of phylogenetic inference fall into two broad classes: maximum likelihood (ML) methods, and Bayesian inference methods. Depending on the number of sequences, the number of characters, and the chosen evolutionary model, both ML and Bayesian tree inference methods can be computationally intensive, thus creating the need for strategies that speed up computation and decrease time to results. One such strategy is parallelization, which distributes a logical unit of computation over multiple processors. Maximum likelihood methods are generally more amenable to parallelization than Bayesian inference methods, since the hundreds or thousands of searches for the ML tree and bootstrap trees that are required for a typical phylogenetic analysis may be run independently of one another. We have developed a grid computing system that features the maximum likelihood-based program GARLI [2] for high-throughput phylogenetic analysis. Here we describe this publicly available system, in particular focusing on the user-friendly GARLI web interface available at `molecularevolution.org`.

GARLI is an open-source phylogenetic inference program that uses the maximum likelihood criterion and a stochastic evolutionary algorithm to search for optimal solutions within the joint space of tree topologies, branch length parameter values, and model parameter values. GARLI was developed with the goal of increasing both the speed of maximum likelihood tree inference and the size of data sets that can be reasonably analyzed. GARLI 2.0 implements models for the analysis of biological sequence data (at the level of nucleotides, amino acids, or codons), as well as morphology and (not officially released) insertion-deletion characters. Version 2.0 introduced support for partitioned models, allowing simultaneous use of different data types or assignment of differing model parameters and rates to individual loci or codon positions. The program design focuses on flexibility of model choice and rigor in parameter estimation.

Searches through phylogenetic tree space may become entrapped in local optima, and therefore it is necessary to perform multiple GARLI searches for the tree with the highest likelihood, which we simply call the *best tree*. This could entail hundreds of searches, depending on the difficulty of the problem. Furthermore, one typically conducts hundreds or thousands of bootstrap replicate searches to assess confidence in the bipartitions found in the best tree. Depending on the number of sequences, the number of unique alignment columns, the evolutionary models employed, various GARLI configuration settings, and the capability of the computational resource, it can take hours or even days to complete a single GARLI search replicate. Thus, running many search replicates in parallel on a grid computing system greatly reduces the amount of time required to complete a set of analyses.

Grid computing is a model of distributed computing that seamlessly links geographically and administratively disparate computational resources, allowing users to access them without having to consider location, operating system, or account administration [48]. The Lattice Project, our grid computing system based on Globus software [49], incorporates volunteer computers running BOINC [50] as well as traditional grid computing resources such as Condor pools [51] and compute clusters. The architecture and functionality of the grid system is described extensively elsewhere [52]; fundamentally, however, The Lattice Project provides access to scientific applications (which we term grid services), as well as the means to distribute the computation required by these services over thousands of processors. In recent years, the system has been enhanced by the development of a web interface to the GARLI grid service [54]. The GARLI grid service has been used in at least 60 published phylogenetic studies, with usage having increased dramatically since the release of the GARLI web interface [5, 6, 6–8, 61–130]. As of 09 June 2015, 1,191 distinct web service users have completed 6,901 analyses comprising 3,235,709 individual GARLI search replicates (Figure 4.1).

Here we compare The Lattice Project to other scientific gateways and describe the features of the GARLI web service. In addition, we provide details about how the grid system efficiently processes computationally-intensive phylogenetic analyses.

4.2 The Lattice Project compared to other scientific gateways

There are a number of other scientific gateways that provide bioinformatics tools and services, including those for phylogenetic analysis. These include the Cyberinfrastructure for Phylogenetic Research (CIPRES) Gateway [201], the University of Oslo Bioportal [202, which has recently closed], the Cornell Computational Biology Service Unit (`cbsuapps.tc.cornell.edu`), Phylemon [203], and Mobyle [204]. Although each of these other systems has proved to be of use in phylogenetic research, our grid system has some distinguishing characteristics.

1. **GARLI version 2.0** — Of the gateways supporting phylogenetic analysis, only The Lattice Project and the CIPRES gateways offer a GARLI 2.0 [198] service.

2. **Unlimited computation** — The GARLI service at `molecularevolution.org` currently allows an unlimited number of submissions, up to 100 best tree search replicates (1,000 search replicates in "adaptive" mode) or 2,000 bootstrap replicates per submission, and no resource or runtime limitations. We are able to offer this level of service due to our implementation of stringent error checking, advanced scheduling mechanisms, and inclusion of several types of grid computing resources.

3. **Facile user interface and resource abstraction** — Fully embracing the grid computing model, the computing resources backing the GARLI service are abstracted from the user, facilitated by an elegant user interface. In contrast,

the CIPRES gateway requires the user to become familiar with their computing resources and to specify their analysis in such a way that it will complete on the allocated resource (usually only a small number of processors) within an allotted period of time.

4. **Sophisticated and relevant post-processing** — The use of stochastic algorithms, multiple search replicates, and bootstrap analyses generates a large number of individual results that must be compiled and processed for evaluation and subsequent use. We perform much of this post-processing automatically, including computation of the best tree found or bootstrap majority rule consensus tree, and the calculation of various summary statistics and graphical representations (Section 4.5).

5. **Large-scale public participation** — The Lattice Project is the only phylogenetic analysis system that provides an easy and meaningful opportunity for public participation in research, which is achieved by using our BOINC project (`boinc.umiacs.umd.edu`). Volunteers simply download a lightweight client to their personal computer, thus enabling it to process GARLI workunits for The Lattice Project. As of 02 April 2014, more than $16,956$ people from 146 countries have participated.

6. **Minimal energy usage** — *Emergy*, the energy embodied in computing components (which includes manufacture and transportation), accounts for the majority of power consumed in computing [205]. Put another way, the "greenest" computer is one that is never built. Apart from a few servers for web,

database, and middleware services, no hardware is purchased specifically for our grid system. The institutional resources we use are comprised largely of desktop systems and clusters purchased for other purposes (e.g., teaching labs and research, respectively), and we use these resources only when they are not being used for their primary purpose. In addition, more than $38,481$ computers from the general public have been volunteered at various stages of the project. For all of these resources, the *emergy* investment has already been made, and our use of these resources amortizes this investment over a greater usage basis. In contrast, phylogenetic analyses through other gateways compete for limited resources on high-capacity clusters, where the jobs often do not take advantage of the high-bandwidth, low-latency interconnects and other special hardware features offered. Furthermore, the widely-distributed, low-density computing model of our grid system results in almost no additional energy use for cooling compared to the substantial energy costs of cooling computer data centers.

No other openly-accessible phylogenetic computing system collectively shares these attributes. Although dedicated high-performance computing resources have their place in scientific research, a substantial share of phylogenetic analyses can be performed very effectively, and more energy efficiently, by means of grid and public computing.

4.3 GARLI web service: user interface and functionality

We have recently upgraded the user interface to our grid system from a Unix command-line interface to a web-based one. This greatly reduces the entry barrier for potential non-technical users. Researchers were previously required to use command-line tools to upload data, submit analyses to a particular grid service (e.g., GARLI), and download subsequent results. Basic utilities were also available to query the status of jobs or cancel them.

Although the command-line interface is still available, the web-based interfaces to our services have generated considerably more interest; the GARLI web service was the first of these to be developed. The following sections describe the modes of use and the basic functionality of the GARLI web service at `molecularevolution.org`.

4.3.1 Modes of use

A GARLI web service user may register an account or choose to remain anonymous. Anonymous users are only required to provide an email address (used to notify them of job status updates) and to fill out a CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) for each job submission (to prevent spam submissions). Anonymous use of the web service is a convenient way to try out the service with minimal effort. However, registration at `molecularevolution.org` confers several advantages: (1) one does not have to fill out a CAPTCHA for each job submission; (2) one gains access to a file repository that can be used to store and reuse input files (Figure 4.2); and (3) one gains the ability to view a list of their jobs and manage them.

4.3.2 Create job page

Submitting a GARLI analysis via the **create job** page (Figure 4.3) consists of the following general steps: (1) specification of a job name, analysis type (best tree or bootstrap search), and number of replicates (up to 2,000); (2) upload or specification of necessary input files (sequence data, starting tree, and/or constraint file); and (3) specification of model parameters and other program settings. Upon job submission, the system uses a special validation mode of the GARLI program to ensure that there are no problems with the user-supplied data file and the parameters specified; for example, very large data sets may require more RAM than the system currently allows (8000 MB). GARLI search replicates are then scheduled to run in parallel on one or more grid system resources that meet the job requirements (e.g., that have enough RAM). The user is notified by email if their job was submitted successfully or if it failed for some reason.

4.3.3 Job status page

The **job status** page (Figure 4.4) allows a registered user to view and manage a list of their jobs. For each job listed, the following attributes are displayed: job id, job name, number of replicates complete, job status, and time the job was created. The dropdown at the top of the page allows one to filter jobs by a particular job status ("idle", "running", "retrieved", "failed", or "removed"). Finally, using the button at the bottom of the page, one may remove jobs that are no longer of interest. If the jobs to be removed are in the process of running, they will be canceled.

4.3.4 Job details page

When a registered user selects a particular job from the **job status** page, or an anonymous user enters a valid e-mail address/job id combination on the same page, the **job details** page is shown (Figure 4.5). This page contains a section for job input files (both user-provided and system-generated) and a section for job output files. The job output files section always includes a ZIP file that contains all of the currently available output associated with the analysis. If all of the replicates for a particular analysis are complete, then the job output files section will also include the results of post-processing (Section 4.5).

4.4 Partitioned analysis specification

Support for partitioned substitution models is the most significant new feature of GARLI 2.0. However, partitioned analysis specification can be a relatively complicated and error-prone process. We have made the specification of modestly-complex partitioned analyses easier by introducing a *guided mode* that allows the user to specify the details of the partitioned analysis with graphical form elements (Figure 4.6), rather than by manually composing a NEXUS sets block and GARLI model blocks. Guided mode is enabled once the user has selected a valid NEXUS data file, which the system processes with the Nexus Class Library [206]. The user then creates one or more character sets (*charsets*), each consisting of a name, a start position, and an end position; charsets may also be specified by codon position using a checkbox. Once the user specifies one or more valid charsets they will be made available to be added to *data subsets*. Each data subset must contain at least one charset, but may contain more than one. The service currently allows the definition of up to ten data subsets in guided mode. For each data subset, a particular substitution model (or particular model parameters) may be specified. When the partitioned analysis is submitted, the service will automatically transform the charset and subset data into a NEXUS sets block and include it in the data file, and will likewise produce the appropriate model blocks and add them to the GARLI configuration file. For users who prefer to provide their own NEXUS sets block and GARLI model blocks, we provide an *expert mode* that allows the user to input them directly.

4.5 Post-processing routines

Due to the difficulty of inferring large phylogenetic trees, multiple searches for the best tree are typically performed with GARLI. This increases the thoroughness of the search for the best tree, but the resulting large number of files and analysis results can be overwhelming. To ease the burden on the end user, our web-based system performs some post-processing routines, which include graphical and quantitative characterizations of the set of trees inferred from multiple search replicates.

Post-processing generates a textual summary for all analyses (Figure 4.7). This file contains the following general information: (1) the data file used; (2) the number of replicates performed; (3) the cumulative GARLI runtime; and (4) suggestions for citing the GARLI web service (omitted from Figure 4.7). The analysis summary for a best tree search also contains summary statistics that characterize the distribution of log-likelihood scores and symmetric tree distances [207] (absolute and normalized), as well as estimates of the number of search replicates required to recover the best tree topology at three probability levels (Section 4.5.1).

In the case of a best tree search, post-processing generates the following files in addition to the analysis summary: (1) a NEXUS tree file containing the single tree with the highest likelihood score; (2) a file containing all of the trees found across search replicates, as well as a file containing only the unique trees found (both files in NEXUS format); (3) a file containing a sorted list of the likelihood scores of the trees found by the analysis and a file containing a sorted list of the likelihood scores of the unique trees found; (4) a PDF file showing the distribution of likelihood scores among trees (Figure 4.8a); and (5) a PDF file showing the distribution of symmetric tree distances (Figure 4.8b).

In the case of a bootstrap analysis, post-processing uses DendroPy [208] to generate the following files in addition to the analysis summary: (1) a NEXUS file containing all of the bootstrap trees from the analysis; (2) a NEXUS file containing the majority rule bootstrap consensus tree with bootstrap probability values embedded; (3) a PDF file showing the 0.90, 0.95, and 0.99 confidence intervals for the bootstrap probabilities observed in the majority rule bootstrap consensus tree, calculated using the formulas given in [209] (Figure 4.9); and (4) a table giving the 0.90, 0.95, and 0.99 confidence intervals for the bootstrap probabilities observed in the majority rule bootstrap consensus tree.

4.5.1 Calculating the required number of GARLI search replicates

Our post-processing routines for a best tree search include the calculation of χ, the number of search replicates necessary to guarantee a particular probability (e.g., 0.95) of recovering the tree topology with the highest observed likelihood score [5]. This statistic, based on properties of the binomial distribution, is calculated using the number of replicates that find the identical best topology (x), where "identical topology" is defined as having symmetric distance from the best topology equal to zero.

For example, if the topology of the best tree is unique among 100 topologies ($x = 1$), $\chi = ln(0.05)/ln(1 - (x/100)) \approx 298$. Thus, 298 replicates are required in order to recover the best topology with a probability of at least 0.95 (Figure 4.10). Of course, it is entirely possible that upon running 298 replicates, χ would be revised upwards; e.g., if the topology of the best tree were still unique among the set of topologies, then yet more replicates would be required.

This statistical estimate of the number of search replicates required to achieve a given probability of obtaining the best tree is intended to inform users about the joint behavior of their data and the GARLI search algorithm, and consequently how many search replicates they should perform. The GARLI web service is now able to automatically and adaptively perform the appropriate number of search replicates on behalf of the user (Section 5.5). This introduces an objective decision process into the analysis design that eliminates guesswork and the need to evaluate intermediate output, thus saving investigator time and improving analytical results. It also reduces waste of grid resources and energy by suggesting that the user run only the number of replicates needed. Eventually, it may also be possible to do something similar for bootstrap replicates, perhaps based on a desired level of precision (Figure 4.9) or other criteria [210].

4.6 System performance

The performance of any distributed computing system depends on how efficiently its resources are used. We have implemented a number of scheduling optimizations that enable efficient use of our grid computing resources [52]. These include a round-robin scheduling algorithm to distribute load evenly among resources (Section 5.1.1); a scheme for benchmarking resources and prioritizing job assignments so that faster resources receive jobs before slower resources (Section 5.4); use of predicted job runtime to ensure that long-running jobs are placed on resources where they are unlikely to be interrupted (Section 5.2); and a mechanism for combining many short-running jobs into a single job with an "optimal" aggregate runtime to maximize system throughput (Section 5.3.1). These last two features depend on a framework we developed for GARLI runtime prediction using random forests [211, 212], a machine learning method. We have improved this framework so that the runtime prediction model is continuously updated as new jobs are run (Section 5.2).

It is important to keep in mind that our grid system is designed for high-throughput computing rather than high-performance computing. As a result, while any one analysis might run more quickly on a dedicated high-performance platform, The Lattice Project allows many such analyses to run concurrently and still complete in a relatively modest amount of time (Figure 4.11). In addition, use of a high-performance system may not necessarily yield decreased time to results once allocation processes, system availability, queue waiting times, scheduling policies, and other considerations commonly associated with the use of high-performance resources are factored in. The high-throughput computing gateway at `molecularevolution.org` is well-matched to the requirements of many typical phylogenetic analyses, and it has already proven useful to many researchers conducting maximum likelihood phylogenetic analyses using GARLI 2.0.

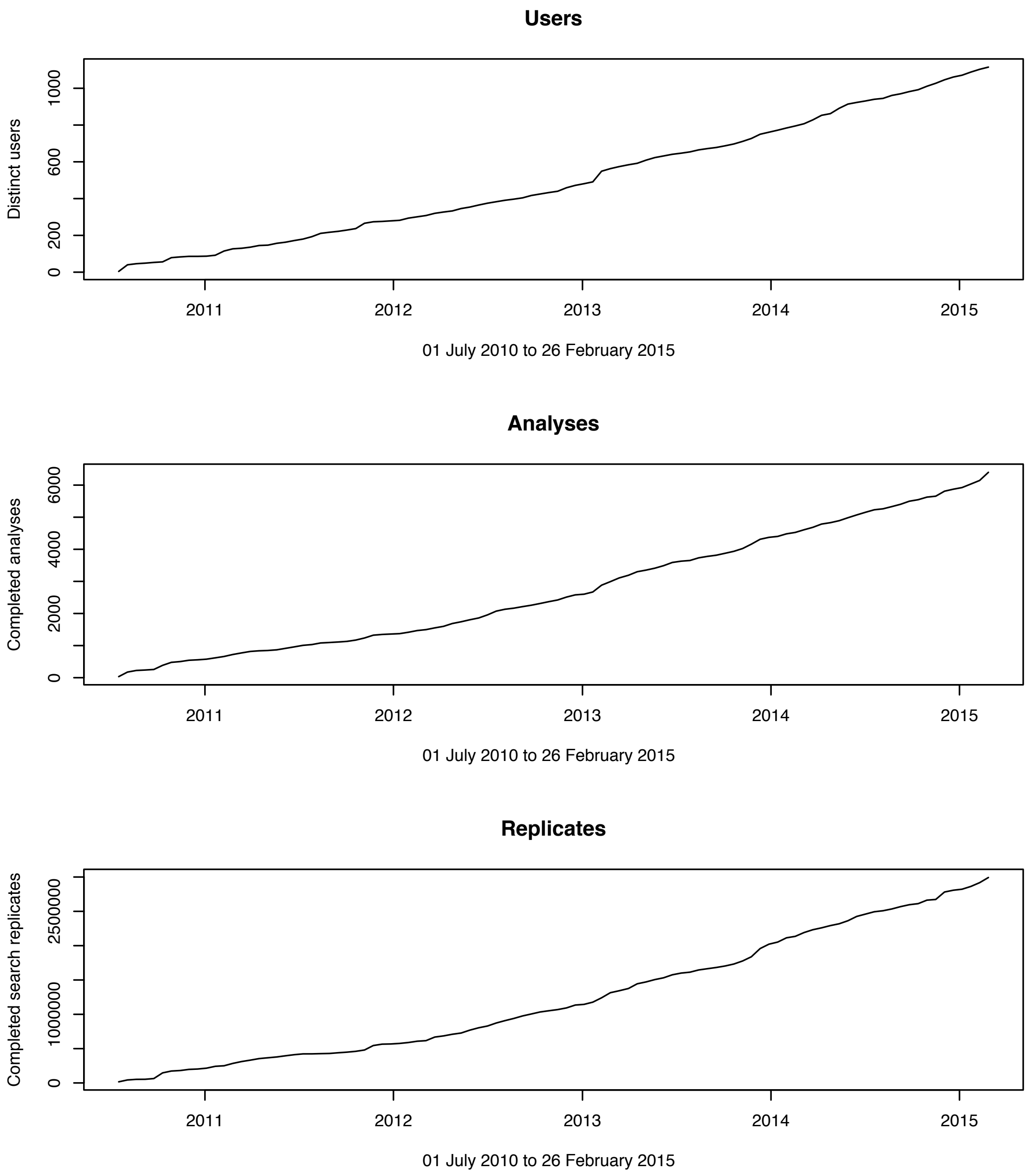
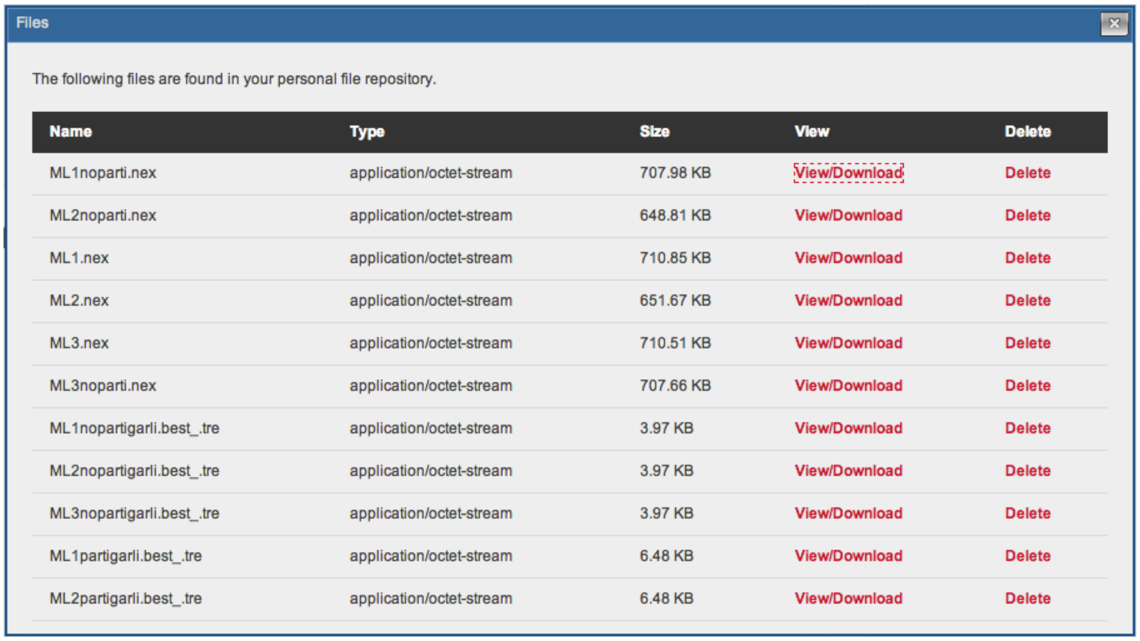


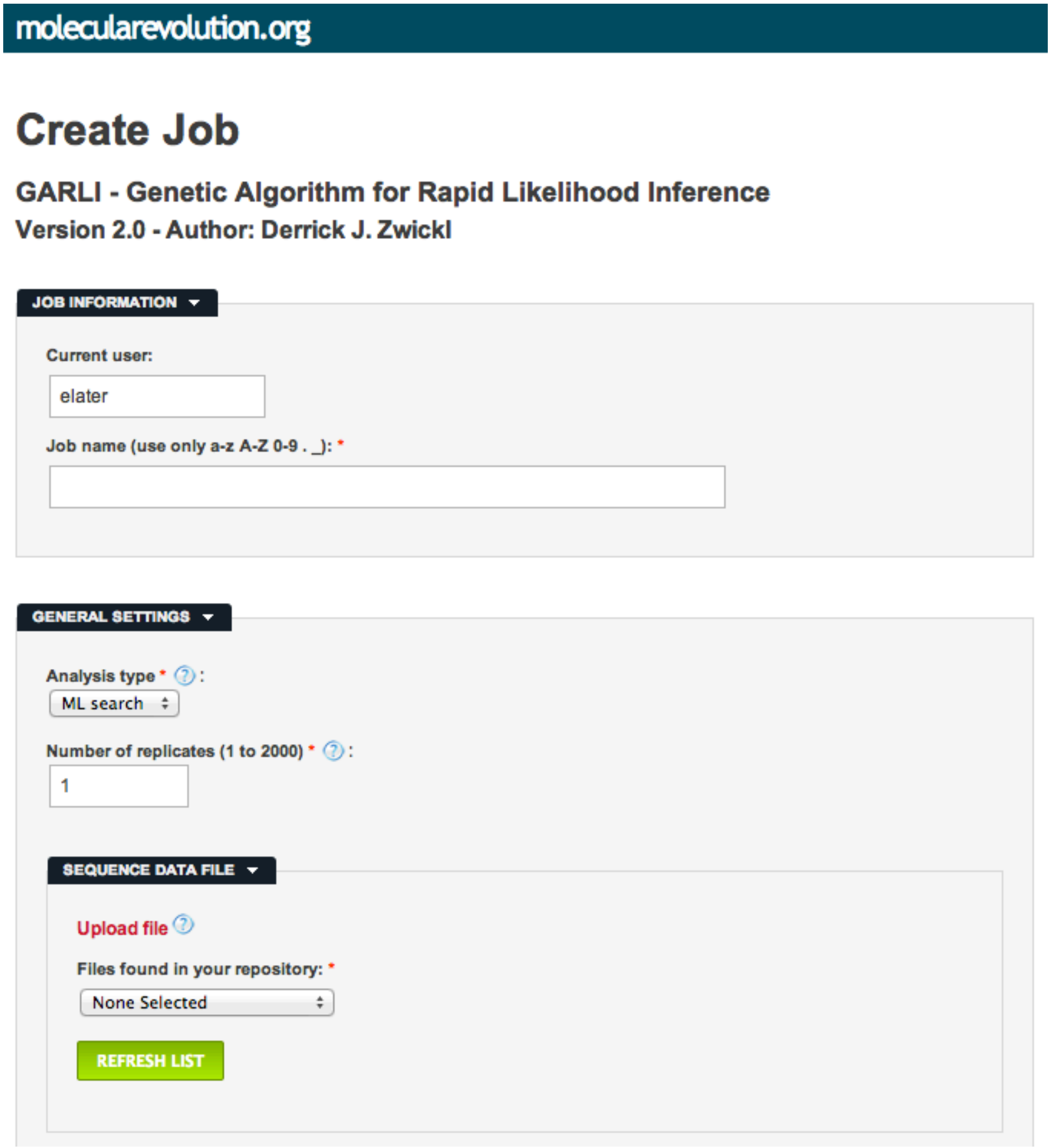
Figure 4.1: GARLI web service statistics showing number of distinct users, completed analyses, and completed search replicates over a five-year time period.



The following files are found in your personal file repository.

Name	Type	Size	View	Delete
ML1noparti.nex	application/octet-stream	707.98 KB	View/Download	Delete
ML2noparti.nex	application/octet-stream	648.81 KB	View/Download	Delete
ML1.nex	application/octet-stream	710.85 KB	View/Download	Delete
ML2.nex	application/octet-stream	651.67 KB	View/Download	Delete
ML3.nex	application/octet-stream	710.51 KB	View/Download	Delete
ML3noparti.nex	application/octet-stream	707.66 KB	View/Download	Delete
ML1nopartigarli.best_.tre	application/octet-stream	3.97 KB	View/Download	Delete
ML2nopartigarli.best_.tre	application/octet-stream	3.97 KB	View/Download	Delete
ML3nopartigarli.best_.tre	application/octet-stream	3.97 KB	View/Download	Delete
ML1partigarli.best_.tre	application/octet-stream	6.48 KB	View/Download	Delete
ML2partigarli.best_.tre	application/octet-stream	6.48 KB	View/Download	Delete

Figure 4.2: The file repository belonging to an example registered user. Registered users may select from among the files found in their repository when specifying input files for a GARLI analysis, in addition to being able to upload new files.



molecularevolution.org

Create Job

GARLI - Genetic Algorithm for Rapid Likelihood Inference
Version 2.0 - Author: Derrick J. Zwickl

Current user: elater

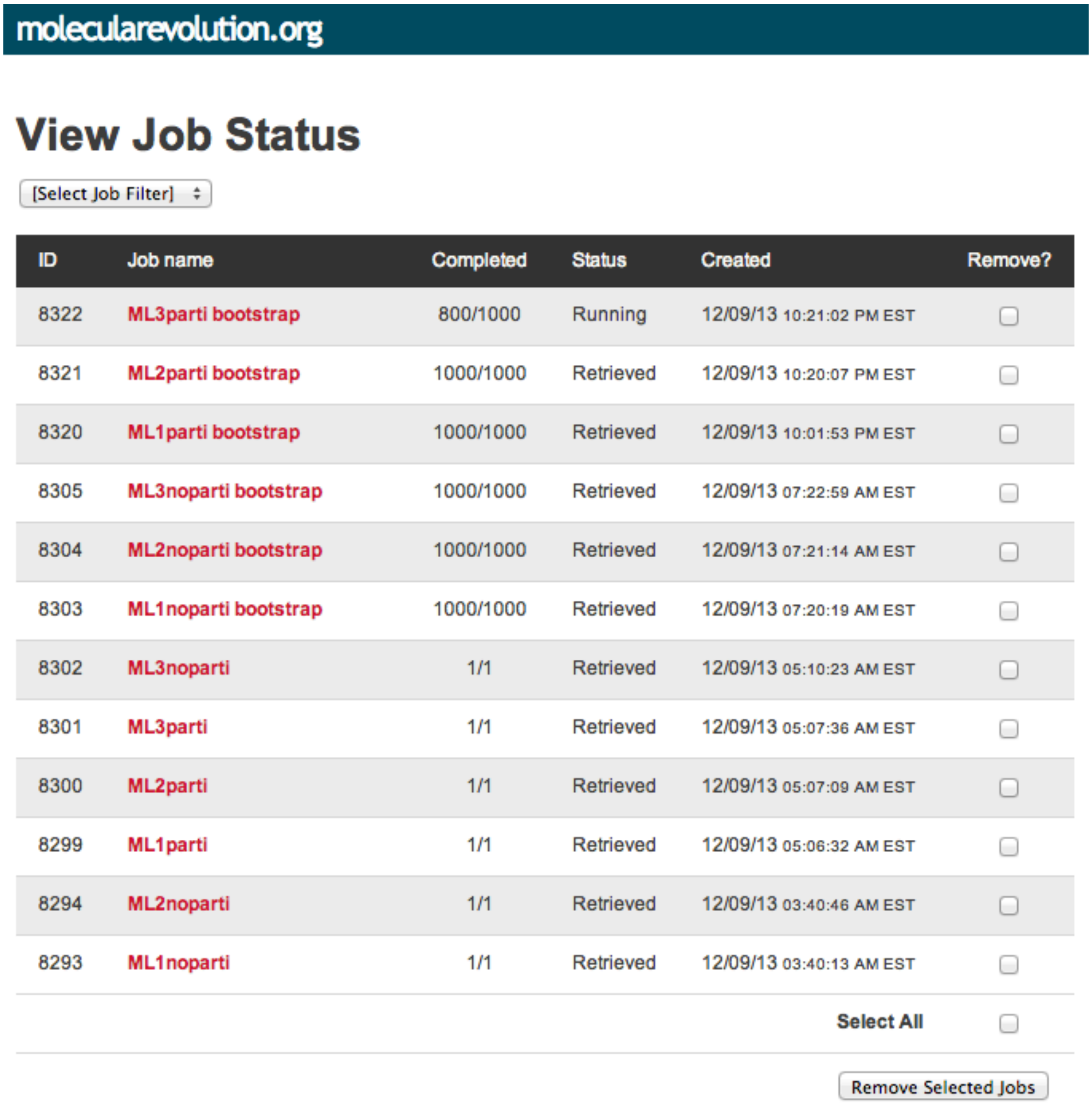
Job name (use only a-z A-Z 0-9 . _): *

Analysis type * ?: ML search

Number of replicates (1 to 2000) * ?: 1

Files found in your repository: * None Selected

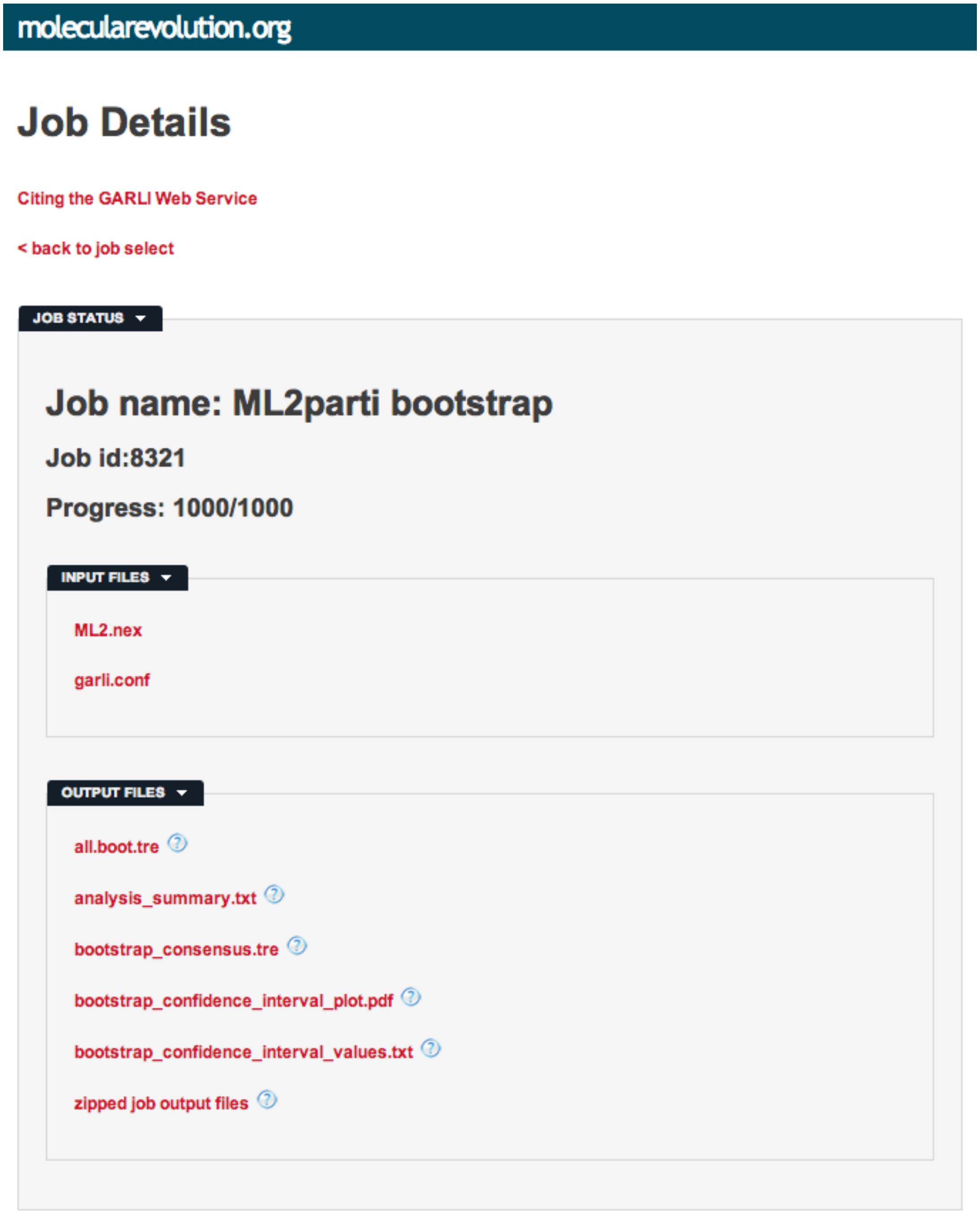
Figure 4.3: The **create job** form found at molecularevolution.org, as viewed by a registered user.



View Job Status

ID	Job name	Completed	Status	Created	Remove?
8322	ML3parti bootstrap	800/1000	Running	12/09/13 10:21:02 PM EST	□
8321	ML2parti bootstrap	1000/1000	Retrieved	12/09/13 10:20:07 PM EST	□
8320	ML1parti bootstrap	1000/1000	Retrieved	12/09/13 10:01:53 PM EST	□
8305	ML3noparti bootstrap	1000/1000	Retrieved	12/09/13 07:22:59 AM EST	□
8304	ML2noparti bootstrap	1000/1000	Retrieved	12/09/13 07:21:14 AM EST	□
8303	ML1noparti bootstrap	1000/1000	Retrieved	12/09/13 07:20:19 AM EST	□
8302	ML3noparti	1/1	Retrieved	12/09/13 05:10:23 AM EST	□
8301	ML3parti	1/1	Retrieved	12/09/13 05:07:36 AM EST	□
8300	ML2parti	1/1	Retrieved	12/09/13 05:07:09 AM EST	□
8299	ML1parti	1/1	Retrieved	12/09/13 05:06:32 AM EST	□
8294	ML2noparti	1/1	Retrieved	12/09/13 03:40:46 AM EST	□
8293	ML1noparti	1/1	Retrieved	12/09/13 03:40:13 AM EST	□
				Select All	□

Figure 4.4: The **job status** page found at molecularevolution.org, as viewed by a registered user.



Job Details

Citing the GARLI Web Service

< back to job select

Job name: ML2parti bootstrap

Job id:8321

Progress: 1000/1000

ML2.nex

garli.conf

all.boot.tre ?

analysis_summary.txt ?

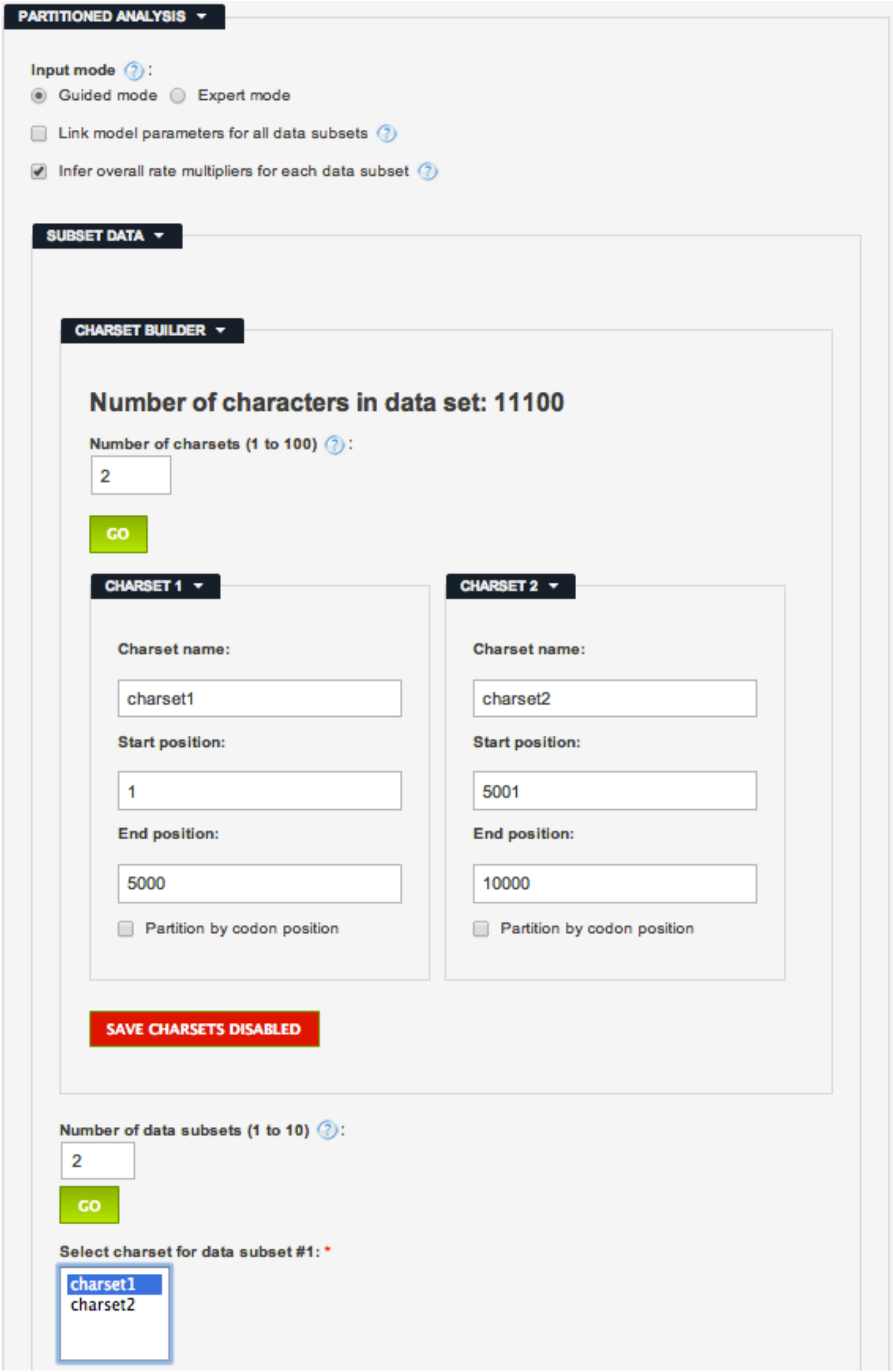
bootstrap_consensus.tre ?

bootstrap_confidence_interval_plot.pdf ?

bootstrap_confidence_interval_values.txt ?

zipped job output files ?

Figure 4.5: The **job details** page found at molecularevolution.org, as viewed by a registered user.



Input mode ?:
● Guided mode ○ Expert mode

□ Link model parameters for all data subsets ?

☑ Infer overall rate multipliers for each data subset ?

Number of characters in data set: 11100

Number of charsets (1 to 100) ?: 2

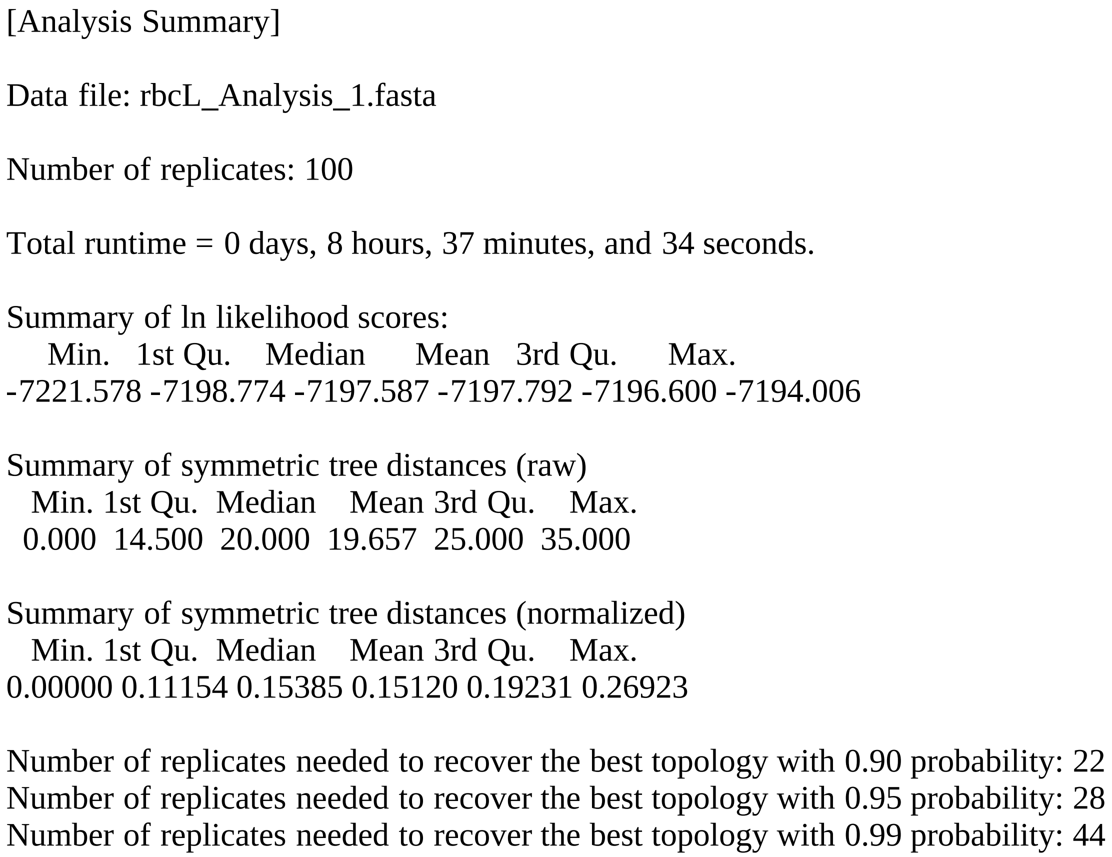
Charset name: charset1
Start position: 1
End position: 5000
□ Partition by codon position

Charset name: charset2
Start position: 5001
End position: 10000
□ Partition by codon position

Number of data subsets (1 to 10) ?: 2

Select charset for data subset #1: *
charset1
charset2

Figure 4.6: The partitioned analysis portion of the **create job** form, currently showing a *guided mode* specification of two character sets and two data subsets.



```
[Analysis Summary]

Data file: rbcL_Analysis_1.fasta

Number of replicates: 100

Total runtime = 0 days, 8 hours, 37 minutes, and 34 seconds.

Summary of ln likelihood scores:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-7221.578 -7198.774 -7197.587 -7197.792 -7196.600 -7194.006

Summary of symmetric tree distances (raw)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000  14.500  20.000  19.657  25.000  35.000

Summary of symmetric tree distances (normalized)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.00000 0.11154 0.15385 0.15120 0.19231 0.26923

Number of replicates needed to recover the best topology with 0.90 probability: 22
Number of replicates needed to recover the best topology with 0.95 probability: 28
Number of replicates needed to recover the best topology with 0.99 probability: 44
```

Figure 4.7: Partial output summarizing the results of 100 best tree search replicates. Among the results of the post-processing displayed here are summary statistics that characterize the distribution of log-likelihood scores, symmetric tree distances (raw and normalized), and estimates of the number of search replicates required to recover the best tree topology at three probability levels.

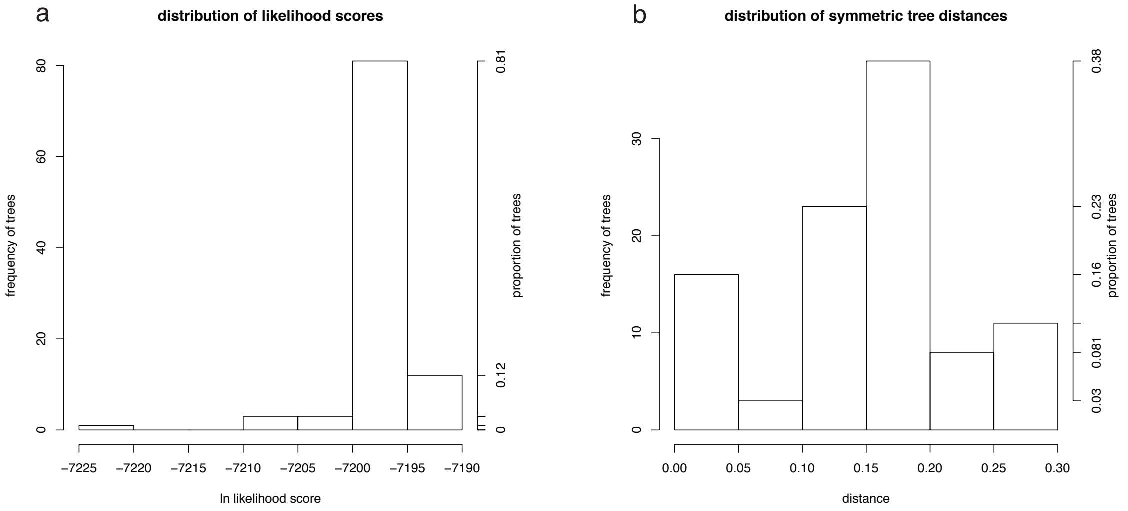


Figure 4.8: Properties of trees from multiple search replicates for a representative GARLI analysis. a) The distribution of likelihood scores. b) The distribution of symmetric tree distances (as a fraction of the maximum possible value for the data set). Both measures are given as frequency and proportion.

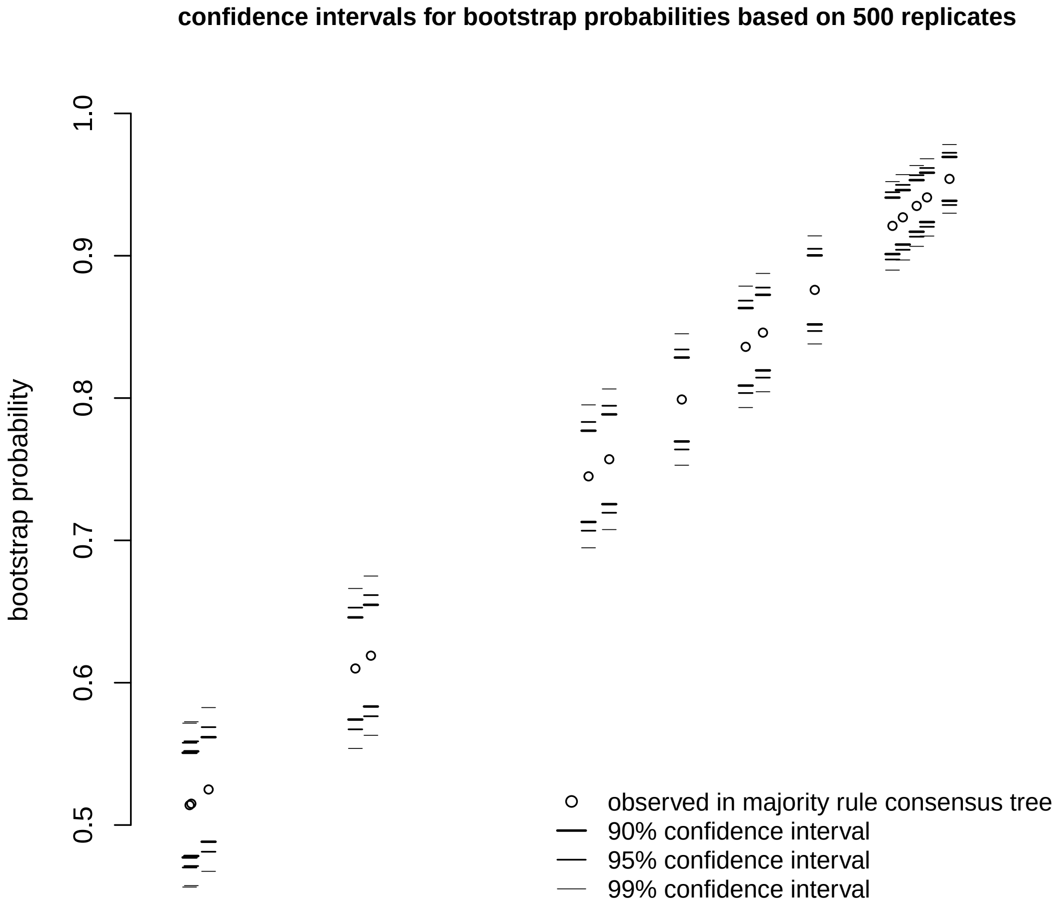


Figure 4.9: Confidence intervals associated with the bootstrap probabilities observed in the majority rule consensus tree computed from 500 GARLI bootstrap replicates. Confidence intervals are given for three probabilities (0.90, 0.95, and 0.99).

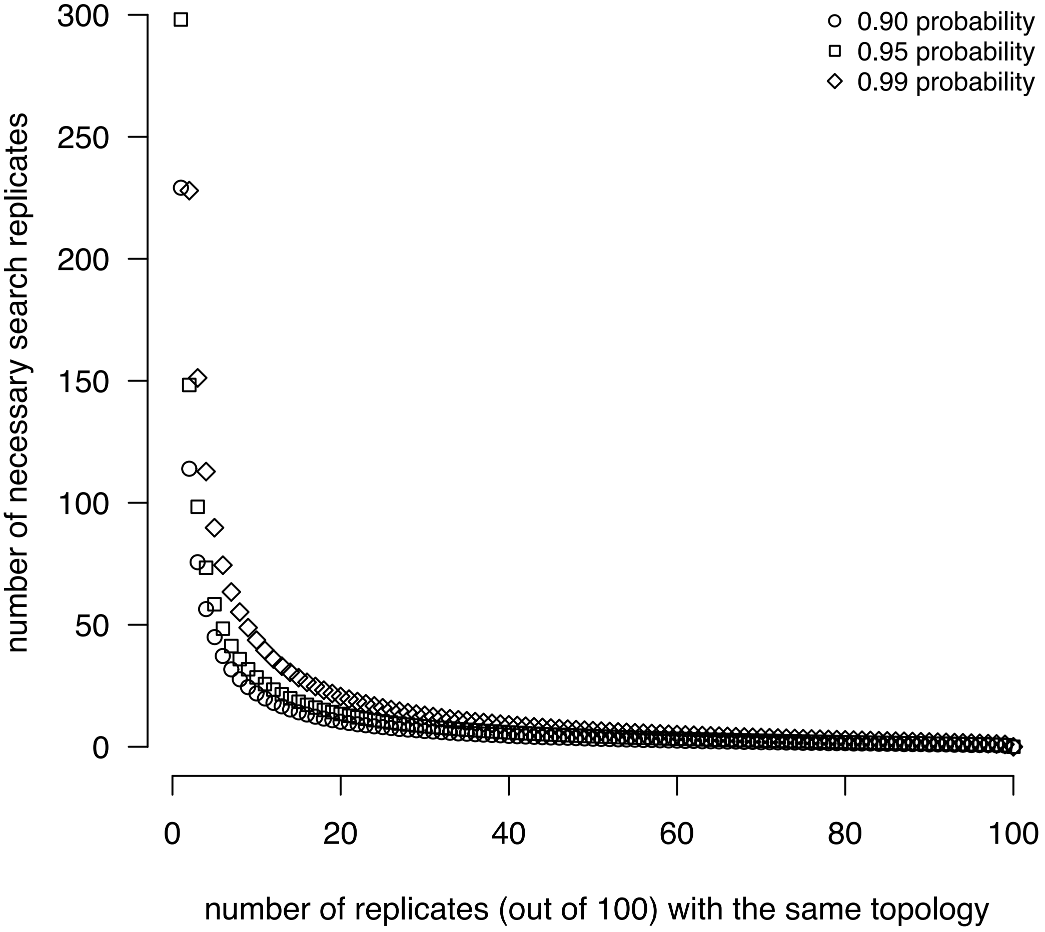


Figure 4.10: Relationship between the number of search replicates (out of 100) returning the identical topology as that of the best tree found (x), and the estimated number of search replicates necessary to achieve a certain probability of recovering that topology (χ). Estimates are given at three probabilities (0.90, 0.95, 0.99).

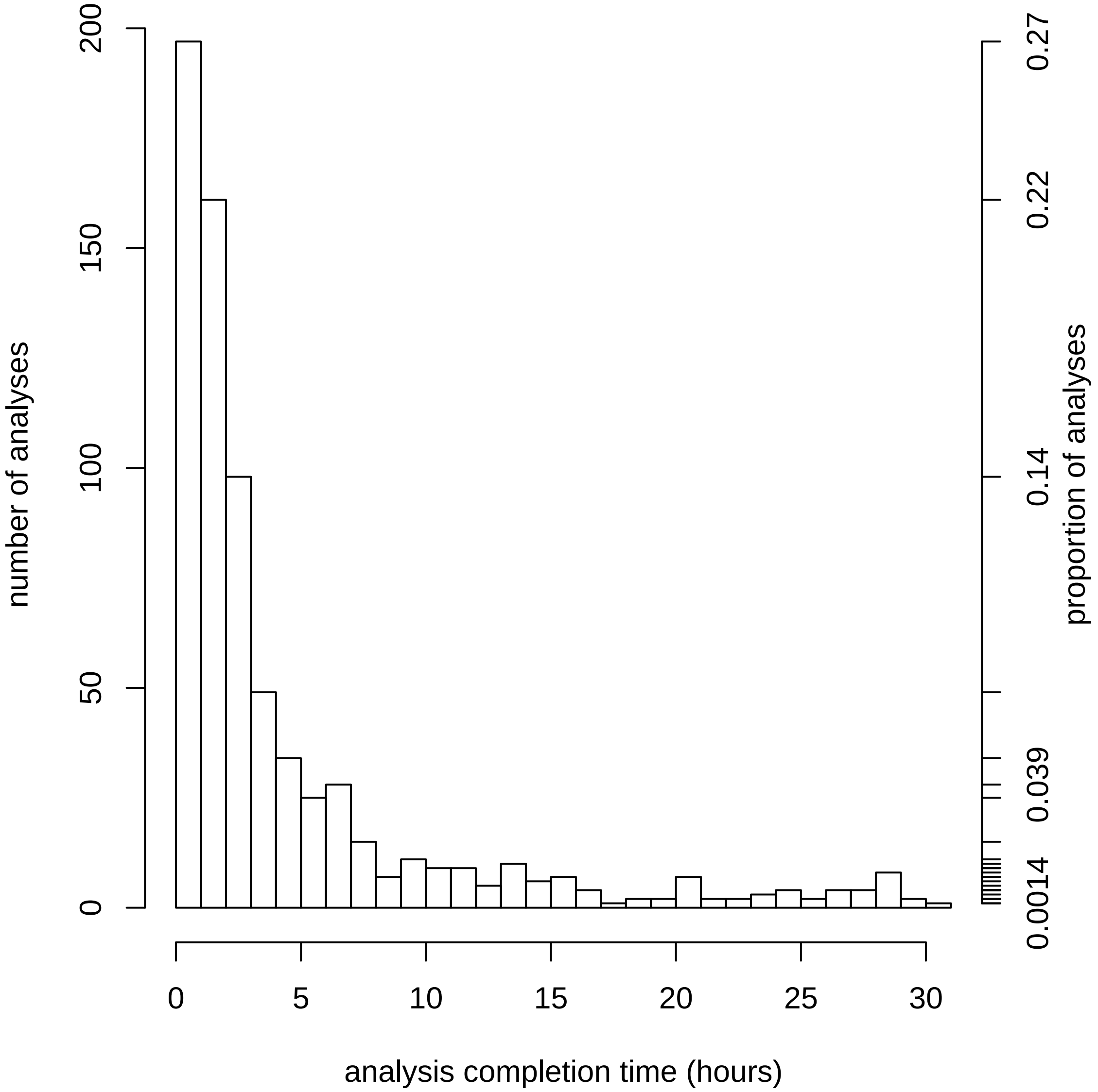


Figure 4.11: Completion times of 719 analyses submitted to the GARLI web service for a recent six-month period (23 July 2013 to 23 January 2014). Despite great variation in analysis parameters (e.g., data matrix size, substitution model used, number of replicates requested), ≈ 97% of analyses were completed in less than 24 hours.

Chapter 5: Improvements to grid computing for phylogenetics

This chapter is based, in part, on the following publication: Adam L. Bazinet and Michael P. Cummings. Computing the tree of life: leveraging the power of desktop and service grids. In *Proceedings of the Fifth Workshop on Desktop Grids and Volunteer Computing Systems (PCGrid)*, 2011. Corrections included.

5.1 Meta-scheduling framework

Many of the improvements to the grid computing system specifically for phylogenetics concern the performance and efficiency of grid-level scheduling (also called *meta-scheduling*). Thus, we first provide some background on the meta-scheduling framework used in our grid system.

The scheduling component of any grid system is likely to be one of the most important and logically complex, because to a large extent it determines the overall efficiency of the system. The grid-level scheduler must decide to submit a job to one of several possible grid resources (i.e., *local resources*; e.g., various Condor pools, clusters, or pools of BOINC clients); after the job is submitted to the local resource, it is usually scheduled *again* to a compute node in the local environment by the scheduler managing the local resource. Thus, the grid-level scheduler is termed a *meta-scheduler* because it performs scheduling one level above that of local resources — i.e., at the grid level. The meta-scheduler must be informed about the current state of local resources, which we achieve using the Monitoring and Discovery Service (MDS), a standard Globus component that requires minimal configuration. Take the example of a Globus installation for which MDS has been configured to report about the status of a Condor pool. In that case, a script called the Condor scheduler provider will periodically parse the output of the `condor_status` command to discover the total number of nodes in the pool, the number of nodes that are free to be used by Condor (not bound to a machine owner or another computational process), as well as other attributes about the Condor pool. This information is stored as XML in the Globus container memory space, and expires after a specified amount of time (in our system, three minutes).

The MDS database can be queried for the information it contains, such as the status of the Condor pool in the preceding example. The information in an MDS database can also be periodically propagated to another MDS database running in a Globus container on a different host. Using this mechanism, we centrally aggregate all of our grid resource data in the MDS database on our central grid server, and query it to assist in scheduling decisions. In the next section we describe our grid meta-scheduling algorithm in detail.

5.1.1 Meta-scheduling algorithm

First of all, the meta-scheduler needs to know which grid resources are reporting. If a grid resource goes offline, any jobs sent there will fail, so we cannot safely assume that our resources are always up and running. Therefore, if we cease to receive MDS information from a particular resource, we mark the resource as "offline", thus ensuring that new jobs will not be scheduled to run there. The meta-scheduler then uses several criteria to choose from among the resources that are reporting. First, not all jobs will run on all resources, so the scheduler matches on various attributes to narrow down the list of possible resources. For example, the system keeps track of which CPU architecture and operating system combinations each grid-enabled application is compiled for (e.g., Intel/Mac OS X), and compares this list to the platforms each grid resource is advertising. From the remaining eligible resources, the scheduler then eliminates resources that do not have sufficient memory (RAM) to run the job. Other resource requirements are also considered if necessary, such as whether or not the resource is MPI-capable, and whether or not it has additional required software installed (e.g., R [178]). One can imagine any number of additional filtering and ranking criteria, especially concerning complex issues such as policy — determining which grid users may access a particular resource, which users have priority over other users, when a particular resource may be used and for how long, and so on. We have not yet placed any such policy restrictions on resource use at the grid level, though it may be necessary to do so in the future. However, it is important to mention that when grid jobs run on a local resource, they are always subject to the local policies that govern use of that resource. From the final set of eligible grid resources, the scheduler chooses the one with the lightest load and submits the job there. If multiple resources are considered equally loaded, the scheduler submits the job to the resource with the highest throughput rating (Section 5.4.3).

5.2 GARLI runtime estimation with random forests

5.2.1 Motivation for a GARLI runtime estimate

Procuring an accurate runtime estimate for a GARLI analysis in advance of job scheduling is useful for a number of reasons. First, it helps prevent long-running jobs from being scheduled to a resource where they do not have a chance of completing. Interruptions can occur because of interference from human users or other computational processes, limits on resource use, technical failures, and a variety of other factors. Thus, we currently prevent GARLI jobs whose estimated runtime is greater than ten hours from being scheduled to Condor resources, where interruptions are especially frequent.

Second, having a runtime estimate allows us to deal with BOINC-specific scheduling issues. For example, we can programmatically specify reasonable workunit deadlines, which are needed on a volunteer computing platform to periodically reissue work if results are not received in a timely manner. Before runtime estimates were available, we had to specify a workunit deadline manually for each batch of work we ran through BOINC, a practice that is not feasible if we wish to use volunteer computing for the wide variety of GARLI jobs routinely submitted through the GARLI web service. In a similar vein, accurate runtime estimates allow the BOINC scheduler to hand out the proper amount of work when a client makes a work request, thus overloading fewer BOINC clients and improving overall system efficiency.

Third, if we find that a grid user has submitted a very short-running analysis — e.g., only a few minutes of runtime per search replicate — we can ratchet up the number of search replicates each individual GARLI invocation will perform (Section 5.3.1). Similarly, long-running jobs can be broken up into homogeneous-length subunits, which is an especially efficacious strategy to use when running on BOINC (Chapter 6). We have found that both of these optimizations greatly improve system efficiency.

Finally, in combination with other data, runtime estimates may eventually help us provide researchers with an estimated runtime for their phylogenetic analysis submissions, which would be helpful for project planning and time management purposes.

5.2.2 Use of random forests for GARLI runtime estimation

GARLI is a particularly challenging program for which to compute runtime estimates. For one, the size of the input data can vary from modest (a few taxa, short sequences) to massive (hundreds or thousands of taxa, sequences thousands or millions of characters in length). Furthermore, the program supports a variety of evolutionary models, some requiring much more computation than others. For example, amino acid and codon models tend to take substantially longer than nucleotide models to analyze the same data set, primarily due to an increased number of possible character states. Finally, because the program is genetic algorithm-like, there are numerous options for controlling the behavior of the genetic algorithm that can have an impact on runtime. In our approach, we first identify the parameters that are most likely to affect runtime, and then we use a machine learning algorithm to produce a runtime estimate for a particular combination of inputs on the basis of training data collected from previous GARLI runs. This approach contrasts with machine learning techniques for runtime prediction that are based solely on historical workload traces [213, 214]. There are many machine learning methods for classification and regression [215]; the particular method we use here is *random forests*, for which we provide additional background in the following section.

5.2.3 Background on random forests

Random forests [211, 212, 216, 217] is a machine learning technique developed by Breiman and Cutler to perform classification and regression using an ensemble of tree-based statistical models (hence, "forest") instead of just one, thus producing more accurate results. Final predictions are obtained by a voting scheme using the ensemble. Bagging [212] is an early example of this technique in which each tree is constructed from a bootstrap sample [218] drawn with replacement from the training data. Bagging reduces prediction error for unstable predictors, such as trees, by reducing the variance through averaging [212, 219]. Minimizing the correlation between the quantities being averaged can favorably enhance this effect, so random forests seek to effect such correlation reduction by a further injection of randomness. Instead of determining the optimal split of a given node of a constituent tree by evaluating all allowable splits on all covariates, as is done with single tree methods or bagging, a subset of the covariates drawn at random is employed. Breiman [211,217] argues that random forests (a) display exceptional prediction accuracy, (b) that this accuracy is attained for a wide range of settings of the single tuning parameter employed, and (c) that overfitting does not arise due to the independent generation of ensemble members.

To estimate GARLI runtimes, we generated random forests made up of 10^4 individual trees constructed by subsampling nine predictor variables at each node. Variable importance was assessed by measuring the increase in group purity when partitioning data based on a variable. We used the R package `randomForest` [178, 220].

5.2.4 Random forests model building

In order to construct a model with random forests, it is necessary to select the analysis parameters that will be included. Based on a combination of our experience using GARLI, program documentation, and correspondence with the program author, we isolated nine parameters that were most likely to affect runtime: data type, proportion of invariant sites, memory used, number of rate categories, number of taxa, number of unique patterns, rate heterogeneity model, rate matrix, and specification of state frequencies (Figure 5.1). Unlike other machine learning methods, random forests does not require variable (attribute) selection. Rather, it allows use of all possible variables, and the importance of each variable is quantified. This is illustrated in our model, in which the most important analysis parameter affecting runtime is use of a substitution rate heterogeneity model, with a difference in mean square error of 89.7%, followed by data type (nucleotide, amino acid, or codon) at 72.4%. In contrast, the number of rate categories turned out to be of very little importance. All analysis parameters and their effect on runtime prediction, as measured by percent increase in mean square error, are shown in Figure 5.1.

Approximately 150 GARLI jobs were initially used as training data; these represented a wide diversity of production jobs that had been previously submitted by grid system users. The values of the nine predictor variables, along with the *response variable* (runtime, measured in seconds) were determined for each job, arranged in a matrix and used to build a model with the `randomForest` R package [178, 220]. The percentage of variance explained by the model was approximately 93%. The model itself, an ensemble of 10^4 individual trees stored as an R object, may subsequently be used to make a runtime estimate for a new set of predictor values. Our cross-validation testing showed that estimated job runtimes matched actual job runtimes closely enough to be used for the purposes stated in Section 5.2.1 (data not shown).

There are several reasons why random forests is a good choice of machine learning algorithm for GARLI runtime estimation: (a) its estimation performance is excellent; (b) it automatically produces a measure of variable importance that

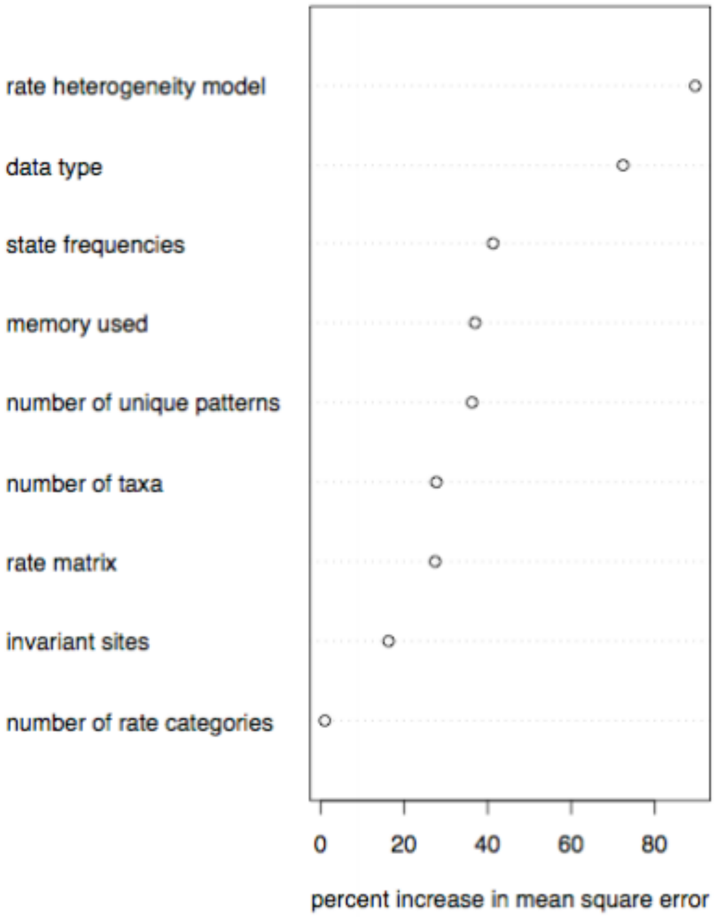


Figure 5.1: Importance of GARLI analysis parameters in predicting analysis runtime as determined by random forests and measured in terms of percent increase in mean square error.

enables better understanding of how analysis parameters affect runtime; (c) it easily incorporates categorical and continuous variables in the analysis; and (d) building and updating the model is computationally tractable.

5.2.5 Integration with the grid meta-scheduler

The grid system uses the random forests model to produce a runtime estimate for each GARLI analysis submitted. More specifically, the system collects the values of the nine predictor variables from each job and produces a runtime estimate via the `predict()` function from the `randomForest` package. This runtime estimate is subsequently used for the purposes previously stated: (a) to decide to send a particular job to a stable or an unstable resource (scaling the runtime estimate used to make this decision by the runtime-only throughput rating of each grid resource; Section 5.4); (b) to provide BOINC with an accurate runtime estimate that is used to estimate the number of floating-point operations associated with the job, as well as to set a reasonable wall clock deadline; (c) to increase the number of search replicates per GARLI invocation in the case of very short-running analyses; and (d) to provide the researcher an estimate of time to results.

As the training data did not cover the entire spectrum of possible values for the nine predictor variables, and because GARLI itself is periodically updated, it is advantageous to continuously update the model based on information collected from incoming jobs. To do this, we simply fork off a single *estimate* job replicate that is representative of a particular submission, which is scheduled like a normal job to execute on a particular grid resource. After it completes, we record the runtime (after scaling it by the runtime-only throughput rating of the resource on which it executed; Section 5.4) along with the values of the predictor variables, and add them to the training data matrix. The model is rebuilt once nightly and is then immediately available to provide runtime estimates for newly-submitted GARLI analyses. In this manner, the accuracy of the random forests model is continually improved (Figure 5.2).

We also added a routine for determining runtime estimates for partitioned GARLI analyses, which treat various subsets of the data matrix in an analysis differently, such as by applying a different instance of an evolutionary model to each subset. The routine calculates a separate runtime estimate for each data subset and adds them together to produce a single runtime estimate for the partitioned analysis as a whole.

5.3 Use of optimal-length GARLI jobs for grid computing

There is significant overhead associated with managing each grid-level job submission due to the negotiation of various layers of middleware, latency associated with file transfers, queue wait times, and so on. Thus, the submission of large numbers of short-running grid jobs leads to reduced efficiency and reduced overall system throughput, as the majority of each job lifetime is composed of various sources of latency rather than actual scientific computation. Conversely, long-running jobs are not ideal either because they are more likely than short-running jobs to be inter-

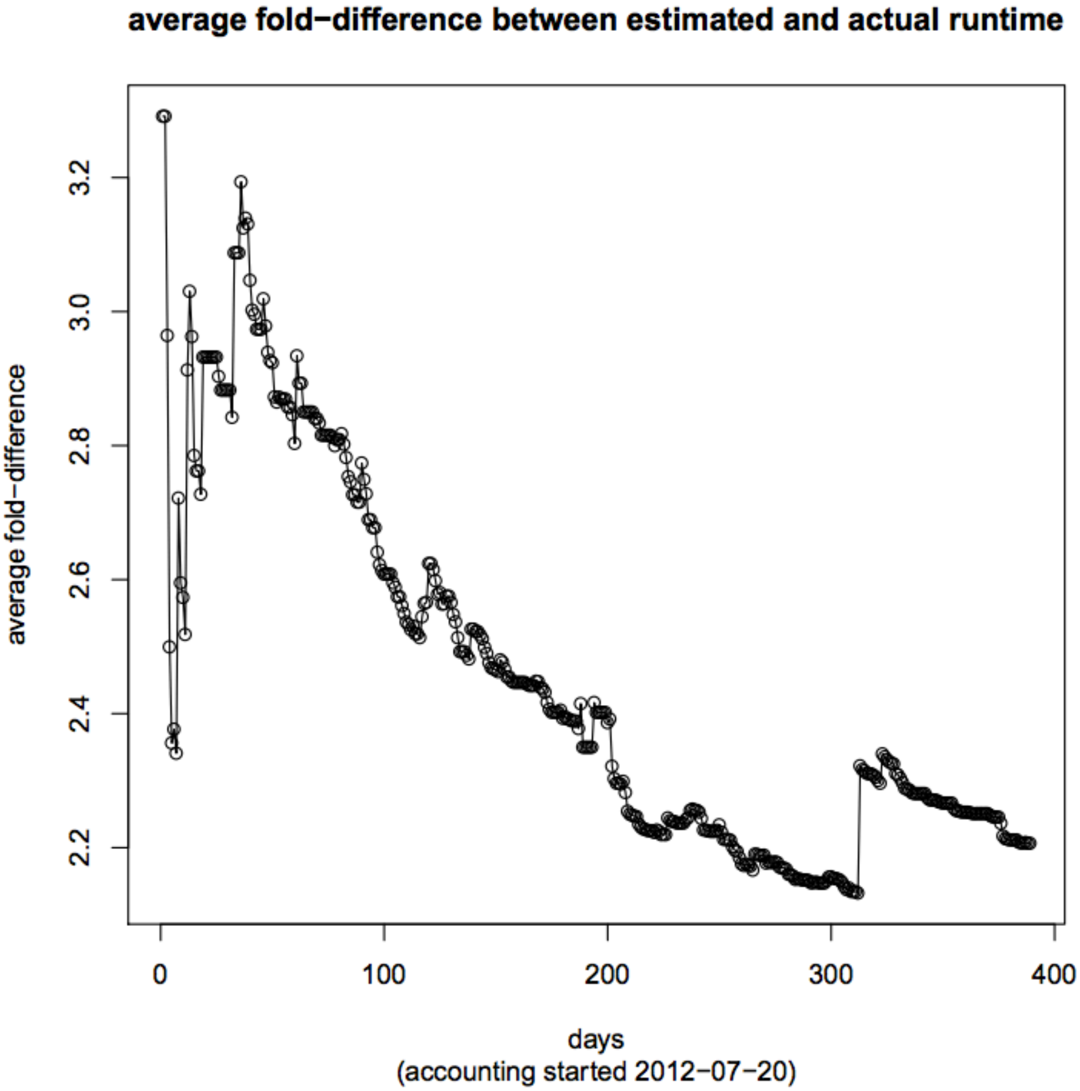


Figure 5.2: Average fold-difference between estimated and actual GARLI runtime, for data spanning approximately 400 days. The average fold-difference gets smaller over time, indicating that the accuracy of the random forests model is improving.

rupted by other processes, computer failures and reboots, and human intervention, all of which lead to wasted computation. In the middle of these two undesirable extremes there exists a theoretically "optimal" job runtime (r), which maximizes scientific computation by minimizing both unnecessary job overhead and potential for interruption. Determining r is a challenging problem when one considers the heterogeneity of jobs that are submitted, the complexity of the grid meta-scheduling algorithm, and the heterogeneity of grid resources. Once r is determined, it is possible to combine short-running GARLI jobs into optimal-length jobs by increasing the number of `searchreps` (or `bootstrapreps`) performed in a single GARLI invocation. The following sections describe the optimizations we have implemented that depend on the optimal runtime value, as well as how we have determined r for non-BOINC resources. (In Chapter 6, we describe a scheme that subdivides long-running GARLI jobs into shorter, fixed-length workunits on the BOINC platform.)

5.3.1 Combining short-running GARLI jobs into optimal-length jobs

The first major optimization combines predicted short-running GARLI jobs into optimal-length jobs. In the event of a GARLI job submission whose estimated runtime is less than $r/2$, we increase the number of search replicates each GARLI invocation will perform, which reduces the number of grid jobs we must manage and simultaneously improves performance gains from parallelization. For example, given $r = 3,600$ seconds (one hour) and a runtime estimate for a particular analysis equal to 300 seconds (five minutes), the system will set `searchreps` $= 12$ (or `bootstrapreps` = 12, in the case of a bootstrap analysis) for each GARLI invocation. Given a reasonable setting of r, we have found that this simple optimization greatly increases overall system efficiency.

To date, however, we have only applied this optimization to jobs submitted to our non-BOINC (dedicated) resources, even though it might also naturally benefit jobs assigned to our BOINC pool. This is primarily because we have avoided scheduling GARLI web service jobs to BOINC, as our dedicated resources have provided a more consistent turnaround time and have generally sufficed to handle the number of submissions we receive. Thus, the following section discusses how we have determined r for dedicated resources.

5.3.2 Determination of r for dedicated resources

The attributes and dynamics of the BOINC pool set it apart from our dedicated resources (i.e., Condor pools and compute clusters); in particular, job completion and overall turnaround times tend to be longer for BOINC due to the need to set generous workunit deadlines for our volunteers (typically a few days at minimum). In general, the scheduling strategies used for BOINC are different from the scheduling strategies used for our dedicated resources. Thus, there is reason to believe that the optimal job runtime (r) will be quite different for BOINC than it will be for dedicated resources. Here, we specifically focus on how we have determined r for dedicated resources.

For any sufficiently complex and dynamic system, it is difficult to determine

r theoretically; it is almost certainly more expedient to determine r empirically (and if necessary, adaptively; r is likely to be a dynamic quantity). Given that performance characteristics among our different Condor pools and clusters vary, each resource likely has its own optimal runtime value. However, the specification of job parameters such as the number of `searchreps` per GARLI invocation (discussed in Section 5.3.1) currently occurs *before* job scheduling; a significant amount of reengineering would be required to invert this order of operations, and it would likely create other inefficiencies to address. Thus, it would be convenient to find a single value of r that works uniformly well for all of our dedicated resources. The value that we have used to date is 3,600 seconds (one hour), which is simply an estimate conditioned on many years of experience running GARLI jobs on the grid. Anecdotally, this value has been working quite well in production, but here we explore ways to optimize r more rigorously.

5.3.2.1 Strategies for empirically determining r

There are two basic strategies for gathering the data necessary to determine r empirically. The first strategy is to use everyday production jobs submitted to the grid system, and the second strategy is to submit special jobs specifically for data collection purposes.

As mentioned previously, r is currently set to 3,600 seconds (and has been for several years, resulting in a large amount of historical data associated with this value). The first strategy, which uses the production system, is to conduct a simple hill-climbing search near the current value of r; i.e., raise or lower r from its current value in increments until performance ceases to improve in whichever direction seems beneficial. There are several drawbacks to this approach, however. The first problem is that we know neither in which direction to move r, nor how much of a “shock” the current system can tolerate. Thus, it is potentially dangerous and costly to conduct such experiments with the production system, but unfortunately there is no convenient alternative: we do not have a grid system simulator at our disposal. Second, it is potentially difficult to evaluate the positive or negative effect of each new setting of r. A possible method of evaluation would be to consider a set of jobs that have a very similar base execution time, constrain evaluation to one grid resource at a time, and then compare the actual turnaround time of batches of such jobs under different values of r. However, an effective calculation would require a relatively large sample size for each set of jobs and each value of r tested, and because we are at the mercy of our grid users to provide these jobs, we could potentially spend a long time searching for the optimal runtime value. In fact, we may never actually be able to cease the search because the true optimal runtime may change significantly during the time it takes to gather data. The major *advantage* of this overall strategy, however, is that it makes the most realistic possible use of the system by modeling all sources of latency and the various idiosyncrasies associated with typical patterns of job submission.

The second strategy is to submit a suite of test jobs specifically earmarked to collect data that will be used to determine r. This strategy has at least two advantages over the previous one: (1) the characteristics of the jobs that are used for testing can be exactly specified, and variability in some aspects of jobs can be eliminated when desired, thus leading to more precise measurements; and (2) because these tests can be explicitly designed and executed, they can be completed relatively quickly, thus mostly eliminating "change in resource characteristics" as a confounding variable for any single test. Drawbacks as compared to the first strategy, however, include the following: (1) a relatively significant amount of computational cost will be incurred, which has the potential to interfere with the execution of normal production jobs; (2) some effort will be needed to simulate "realistic" grid usage, which involves considering the load induced on resources as well as the period of time over which the experiment is conducted; and (3) the procedure will likely have to be repeated periodically to keep up with changing resource characteristics.

Given these considerations, we decided to try out the second strategy. The following section details our implementation of that strategy and the results.

5.3.2.2 Using test jobs to determine r for dedicated resources

To control for as many variables as possible, we restricted ourselves to a single, relatively short-running GARLI job named `rana`. By fixing the random seed, we ensured the execution of `rana` was deterministic. The runtime estimate produced by the system for this job is 633 seconds — a little over 10 minutes.

All six of our dedicated resources were targeted in this experiment. These included four Condor pools (CMNS, Coppin, TerpCondor, and UMIACS) and two clusters (Deepthought and Topaz). To simulate "normal" grid use to a limited extent, we introduced a delay between job submissions to any particular resource equal to 12 hours plus a random number of additional hours (0–12). For each value of r considered, 10 job submissions were made to each resource, each consisting of 100 search replicates executed by 100 or fewer GARLI invocations (simply called *replicates*). Each replicate ran on a separate processor.

The following values of r were considered: 633 seconds (x), $4x$, $10x$, $25x$, and $100x$. These values were chosen to make the arithmetic work out neatly: an optimal runtime value equal to x (633 seconds) would cause each of 100 GARLI replicates to consist of a single search replicate, whereas $r = 100x$ would pack all of the computation into a single GARLI replicate with `searchreps` = 100.

The measurement of interest is *turnaround time* — i.e., the total time a batch of jobs spends on a resource, including time spent waiting in scheduling queues. This quantity is simply the difference between the timestamps recorded when a batch finished and when it was submitted. We expected better values of r to produce shorter average turnaround times.

The results of the experiment are shown in Table 5.1. The "1 rep/100 sr" column represents a single GARLI replicate that ran 100 search replicates on a single processor. At the other extreme, the "100 reps/1 sr" column represents 100 GARLI replicates that each ran a single search replicate on a separate processor.

If each resource were comprised of computers that were homogeneous, always willing to process grid jobs immediately and at the highest priority, and of sufficient quantity to run all 100 `rana` search replicates simultaneously, then one would expect a linear speedup from parallelization. Although each of these assumptions are routinely violated in practice, we nonetheless observe a roughly linear speedup from parallelization (Table 5.1). Thus, this data suggests that we should set r to a value that would cause each `rana` search replicate to run on its own processor, which in this case would be $r < (2 \times 633)$ seconds. If $r > (2 \times 633)$, then the grid would increase the number of search replicates per GARLI replicate, which we can see from the data usually leads to longer turnaround times. This result disagrees with our impression that system performance has improved since we began combining short-running jobs into jobs of approximately 3,600 seconds. The following section describes several factors not considered in this experiment that can account for this discrepancy.

5.3.2.3 Factors that favor a longer optimal runtime value

As of 30 January 2014, there were 833 GARLI web service analyses that received a runtime estimate > 316 seconds but $\leq 1,800$ seconds. Thus, had the optimal runtime value initially been set to 633 seconds instead of 3,600, 833 fewer analyses ($\approx 20\%$ of all analyses) would have been "compressed" via increased values of `searchreps` or `bootstrapreps`.

As mentioned in Section 5.3.2.1, it is difficult for a strategy that uses test jobs to approximate real grid usage without interfering overly much with normal grid operation. The following factors that favor $r = 3,600$ were left unaccounted for in the experiment described in the previous section.

1. The previous experiment did not take into account the extra time needed to

submit multiple batches as part of a GARLI web service submission. Using $r = 633$ instead of $r = 3,600$, for example, would result in $\approx 5.7\times$ as many batches submitted, on average. The time it takes to submit each batch is approximately three minutes. Given that the maximum batch size for a GARLI web service submission is 100 replicates, and the maximum number of search replicates allowed in a single submission is 2,000, up to 20 batches may be submitted if multiple search replicates per job are not used, thus resulting in approximately 60 minutes of submission time. Hence, for short-running jobs, using $r = 3,600$ instead of $r = 633$ could save as much as 50 minutes during a single submission.

2. The grid system periodically iterates through the list of GRAM[1] jobs it is managing (each of which may be a batch of jobs consisting of as many as 100 GARLI replicates), and polls the Globus container on the appropriate remote grid resource to update the status of these jobs. Although we have implemented an exponential backoff scheme that doubles the amount of time between status checks (up to some maximum amount) in the event the status of a job is queried and found not to have changed, the fact remains that checking and updating the status of hundreds or thousands of GRAM jobs is still a source of latency. Compressing multiple short-running GARLI replicates into a single GARLI invocation leads to a reduction in the number of GRAM jobs there are

[1] GRAM is a software component of the Globus Toolkit that can locate, submit, monitor, and cancel jobs on grid computing resources. It provides reliable operation, stateful monitoring, credential management, and file staging [221].

to manage, thus improving system throughput.

3. Using fewer GRAM jobs saves on bandwidth and disk space, as fewer file transfers are needed.

4. Using fewer GRAM jobs simplifies grid system administration. For example, it is easier to monitor and manage the relatively large number of GRAM jobs that job submissions generate when the number of GRAM jobs is minimized, and it takes less time to fix problems that occur.

Item four in the list above would be difficult to study because it involves human-computer interaction; however, the other factors that favor a longer optimal runtime value could be included in an experiment (cf. the first strategy described in Section 5.3.2.1). However, due to the problems associated with performing that experiment, we will only explore that option if we observe a problem with system load that could be hypothetically alleviated by modifying r. Thus, we have not changed our setting of $r = 3,600$ seconds for dedicated resources, which represents a compromise between the parallelization data we have gathered, and our knowledge of other factors that are more difficult to model.

5.4 Automatic measurement of resource throughput

An important attribute of a grid computing resource is its processing capability, or rate of throughput. A throughput rating that is comparable across resources can be used to send jobs to the fastest resources first, to scale runtime estimates by the rating of potential assigned resources, and to evaluate and maximize overall system efficiency. However, devising a single, uniformly-calculated throughput rating is challenging for large, heterogeneous computing resources composed of machines that frequently change their capabilities, architectures, operating systems, and other attributes.

Prior to dissertation-related work, the procedure to measure grid resource throughput was to execute an identical, short-running GARLI job on each individual computer that made up a grid resource, record the runtimes of these jobs, and average them. We would then compare this averaged runtime to the runtime of the same GARLI job on a "reference computer", which was arbitrarily assigned a throughput rating of 1.0. If the reference job ran in half the time on the resource we were testing, that resource was assigned a rating of 2.0 — or in twice the time, a rating of 0.5 — and so on. This procedure was performed via periodic manual submission of test jobs, which limited its effectiveness because resource characteristics were likely to change more frequently than we could measure them, thus leading to throughput ratings that were quickly outdated. Furthermore, this *runtime-only* throughput rating did not take into account the time that jobs spent waiting in queue on a remote resource.

Homogeneous sets of production jobs are often split up to run on multiple different grid resources, so we can use such analyses to assess differential resource performance by measuring job runtime as well as other sources of latency, such as time spent in queue. Thus, our improved throughput rating procedure uses production jobs that continually flow through the system to evaluate resources, instead of manually submitting jobs for this purpose. The improvements we have implemented provide for continuous, automatic updating of resource throughput ratings while simultaneously introducing a *composite* throughput rating for use in scheduling that incorporates both job runtime and resource latency in its calculation.

5.4.1 Introducing the composite throughput rating

A *composite* throughput rating, one made up of both job runtime and other sources of latency, such as time spent in queue, is desirable because it is a more accurate measure of average job throughput on a given resource than the runtime-only rating. For example, resource X may have more capable computers than resource Y, but if Y is more available than X — i.e., there is less competition for Y, or the grid user has a higher priority on Y — then for the purpose of ranking resources by job throughput, Y may deserve a higher ranking than X.

The basic data measurement used for the composite throughput rating calculation is job *turnaround time* — i.e., the time a job actually spends on a resource, including time spent waiting in scheduling queues — which is simply the difference between the timestamps recorded when a job is submitted and when it finishes. To derive the turnaround time, nothing new needed to be added to our job profiling procedures; all of the necessary information was already being stored in our database.

The steps to calculate the composite throughput rating are as follows:

1. For all GARLI analyses that are split up to run on at least two resources,

calculate the turnaround time — i.e., the average time a GARLI job is resident on each resource, which includes runtime and various sources of latency. Also record the number of search replicates from which the turnaround time is derived.

2. An *equivalency factor* for each pair of resources enables one to compare their relative throughput and make statements such as "resource X is 2.5 times faster than resource Y" (or conversely,"resource Y is 0.4 times as fast as resource X"). The calculation of the equivalency factor is as follows: for all analyses involving a particular resource pair (e.g., X and Y), divide the larger turnaround time by the smaller turnaround time and weight the result by the total number of search replicates used in the calculation.

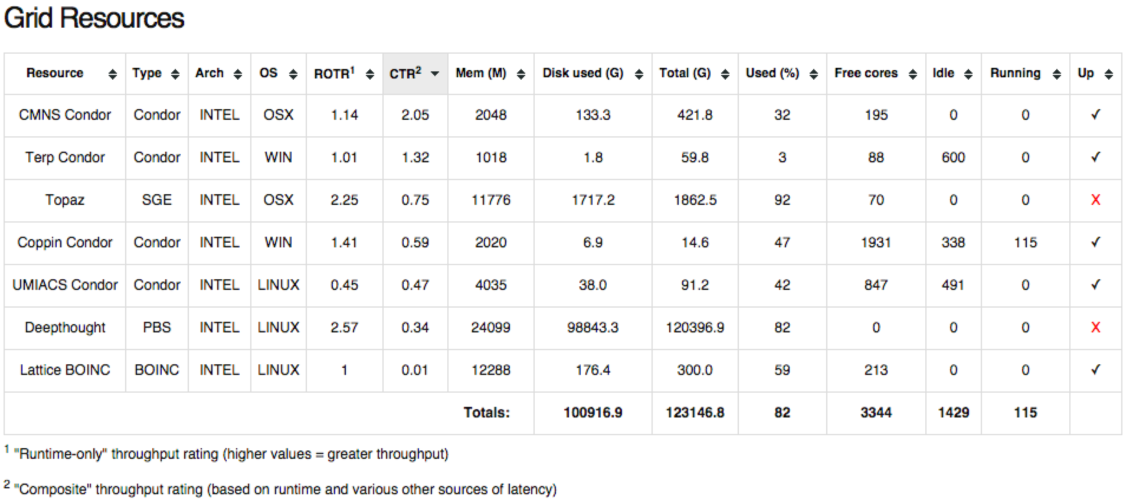
3. Derive the composite throughput rating for a particular resource by iterating through the list of resource pairs of which it is a member and computing an average of the equivalency factor values calculated for the resource, weighted by the number of search replicates associated with each equivalency factor.

The composite throughput rating is used by the grid meta-scheduler to send jobs to the fastest resources first, as well as to allow faster resources to have longer job queues (Section 5.4.3). The previously developed *runtime-only* throughput rating, now computed similarly to the composite throughput rating, is still appropriate for at least two system functions, however: (1) when deciding if it is necessary to send a job to a *stable* resource where it is unlikely to be interrupted, one must scale the estimated job runtime by the runtime-only throughput rating of the proposed resource; and (2) when updating the runtime prediction model (Section 5.2), one must scale the runtime collected from an *estimate* job by the runtime-only throughput rating of the resource on which it ran. Thus, we calculate both ratings and use each in the appropriate context. Figure 5.3 shows both the runtime-only rating and the composite throughput rating for our various grid resources.

5.4.2 Updating throughput ratings automatically

Both the composite and runtime-only throughput ratings are now updated automatically using regular, production jobs that flow through the system. An individual analysis submitted via the GARLI web interface may comprise as many as 2,000 identically-specified search replicates, which are often split up to run on multiple grid resources as a natural consequence of the grid meta-scheduling algorithm. We use this fact to our advantage: as long as the jobs associated with a particular analysis run on at least two different resources, that analysis can be used to update our resource throughput ratings.

Many resources are highly dynamic: computers are added and decommissioned, are upgraded, or change their capability. Furthermore, the intensity of local resource usage by non-grid users fluctuates over time, all of which affects grid job throughput. Thus, intuitively, one would value *recent* performance of a grid resource over *historical* performance, since resource characteristics can change radically and without warning. With this in mind, we recalculate resource throughput ratings once nightly using data from only the past 30 days. If a particular resource did not



Grid Resources

Resource	Type	Arch	OS	ROTR[1]	CTR[2]	Mem (M)	Disk used (G)	Total (G)	Used (%)	Free cores	Idle	Running	Up
CMNS Condor	Condor	INTEL	OSX	1.14	2.05	2048	133.3	421.8	32	195	0	0	✓
Terp Condor	Condor	INTEL	WIN	1.01	1.32	1018	1.8	59.8	3	88	600	0	✓
Topaz	SGE	INTEL	OSX	2.25	0.75	11776	1717.2	1862.5	92	70	0	0	X
Coppin Condor	Condor	INTEL	WIN	1.41	0.59	2020	6.9	14.6	47	1931	338	115	✓
UMIACS Condor	Condor	INTEL	LINUX	0.45	0.47	4035	38.0	91.2	42	847	491	0	✓
Deepthought	PBS	INTEL	LINUX	2.57	0.34	24099	98843.3	120396.9	82	0	0	0	X
Lattice BOINC	BOINC	INTEL	LINUX	1	0.01	12288	176.4	300.0	59	213	0	0	✓
						Totals:	**100916.9**	**123146.8**	**82**	**3344**	**1429**	**115**	

[1] "Runtime-only" throughput rating (higher values = greater throughput)

[2] "Composite" throughput rating (based on runtime and various other sources of latency)

Figure 5.3: A snapshot of information associated with The Lattice Project grid resources. The resources are shown sorted by composite throughput rating. A live version of this information is available at `http://lattice.umiacs.umd.edu/resources`.

complete at least 1,000 jobs during that time period, its ratings are left unchanged.

5.4.3 Faster resources are allowed longer job queues

Allowing faster resources (those with higher composite throughput ratings) longer idle job queues than slower resources is a scheduling optimization that we implemented in conjunction with the introduction of the composite throughput rating. The motivation for this optimization was simple: we wish to complete jobs as quickly as possible using the resources that are available. Before this optimization was implemented, if all eligible grid resources were fully occupied, then additional jobs would be spread evenly among them. This was undesirable because the fastest resource would complete the jobs in its queue and would be unoccupied while slower resources finished their jobs. Thus, allowing faster resources longer idle job queues in proportion to their throughput rating should increase system efficiency by ensuring that all jobs finish at roughly the same time.

In general, the grid meta-scheduler tries to balance various competing priorities, such as spreading jobs around to resources evenly (not overloading any one resource, and attempting to use all resources efficiently), and sending jobs to the fastest resources first for maximum throughput. Thus, when considering how much load is on a resource, it looks at the number of jobs assigned to it that are idle. If the scheduler finds a resource with significantly fewer idle jobs than others, it chooses to send jobs to that resource. However, there needed to be an exact definition of "significantly fewer"; thus, we introduced a variable called *idle job equality* (*IJE*):

resources are considered equally loaded if their idle job counts are within IJE of each other (currently, $IJE = 100$). Among equally loaded resources, then, jobs are sent to the resource with the highest composite throughput rating.

In order to provide faster resources longer idle job queues, the effective number of idle jobs calculated for each resource is scaled by the composite throughput rating for that resource according to the following formula:

effective idle job count = actual idle job count + (actual idle job count × (1 / composite rating))

Hereby, resources are penalized by an amount inversely proportional to their throughput rating.

If the differences in queue lengths resulting from the formula above are too modest (e.g., speedy resource X frequently completes all its jobs and is waiting while slower resources finish theirs), then it would be straightforward to add an inflationary parameter that would exaggerate the differences. The system could monitor the occupancy of various resources over time (Section 5.4.4) and dynamically adjust the value of the inflationary parameter to maximize resource occupancy.

5.4.4 Demonstrating the utility of the composite throughput rating

While it seems obvious that the introduction of the composite throughput rating and adaptively-sized resource queues based on the rating would increase system efficiency and overall throughput, we wanted to demonstrate this definitively. We posit that if scheduling is efficient, then sets of jobs associated with a particular analysis that are assigned to different resources should finish at roughly the same time. For a set of jobs assigned to a particular resource, then, we define *occupancy* as the difference between the time the last job was submitted and the time the last job finished. Thus, for a particular analysis, we have a set of occupancy values (one per resource), each measured in seconds.[2]

A simple measure of the variation in occupancy values is desired; thus, we calculate the coefficient of variation (CV) for each set of occupancy values as CV $= \sigma/\mu$, where σ and μ are the standard deviation and mean of the set of occupancy values, respectively. The CV is a relative measure that allows for comparison of variation in occupancy even when the absolute values of occupancy vary significantly between analyses.

For any analysis submitted through the GARLI web service that ran on two or more resources, it is possible to calculate the CV for that analysis. Thus, to determine if the scheduling changes associated with the composite throughput rating were effective, we calculated the median CV for a set of analyses consisting of approximately 350,000 total GARLI search replicates immediately prior to the scheduling changes, and performed the same calculation for a set of analyses immediately following the scheduling changes. We found that the median CV calculated from the set of analyses performed after the scheduling changes (0.42) was substantially smaller than the median CV calculated from the set of analyses performed prior to the scheduling changes (0.70), thus demonstrating that the changes had the desired,

[2] This framework assumes that all of the jobs are submitted to various resources roughly simultaneously; thus, we only consider the primary jobs associated with the initial analysis submission, and not any make-up jobs that may have been submitted later on due to processing errors, etc.

beneficial effect.

The CV could also potentially be used to adjust the inflationary parameter mentioned in Section 5.4.3. A running average could show the recent trend in CV values, and the inflationary parameter could be dynamically adjusted to minimize the CV values. We save this for future work.

5.4.5 Changes to the random forests runtime prediction model

The random forests model used for GARLI runtime prediction, described in Section 5.2, was eventually updated as a result of the changes to the way the runtime-only throughput rating was computed. On 01 July 2014, a new random forests model was put into production that used as training data only analyses submitted after 30 July 2013 — i.e., the model was only built from jobs that executed after the new procedures for determining and updating resource throughput ratings described in this chapter had been implemented. These new procedures resulted in several important changes: (1) throughput ratings are now updated dynamically (practically continuously), whereas previously they were updated quite infrequently; (2) the new ratings are based on more realistic use of grid resources, since production jobs may only run on a subset of computers that constitute a heterogeneous grid resource, whereas the previous procedure for assigning ratings assumed that jobs were distributed evenly among all computers that constituted a resource; and (3) perhaps most importantly, the new throughput ratings are not directly comparable to previous ones: in the previous scheme, one particular resource (the "SEIL" cluster) was designated as a reference; its rating was fixed at 1.0, and all other ratings were relative to it. In the new system, the "reference", which can be thought of as a centroid, is not associated with any one resource and its value changes dynamically. Thus, the level of throughput represented by a particular rating in the current system — e.g., "1.0" — is not comparable to the level of throughput represented by "1.0" previously.

For these reasons, we hypothesized that we might get better runtime prediction performance if we built a random forests model using only analyses that executed after these changes to throughput rating calculation had been implemented. To test this, we kept updating the "all data" model, and in parallel we created and updated a "recent data" model. After monitoring the diagnostics that computed the difference between estimated and actual runtime (Figure 5.2) for approximately two months, we decided to put the newer "recent data" model into production, as its performance characteristics were indeed somewhat improved (average fold-difference between estimated and actual runtime ≈ 2.4 for the "recent data" model as compared to ≈ 3.1 for the "all data" model as of 01 July 2014). The "recent data" model contained approximately 40% as much training data as the "all data" model, a proportion that will increase over time. Future work might test the efficacy of periodically removing the oldest entries from the random forests model, which would essentially amount to iterating the procedure described here. However, determining the optimal amount of historical data to use is relatively difficult, as both too little data and perhaps, as demonstrated here, too much data could be detrimental. Moreover, testing any particular model requires a significant number of analysis submissions, so as a practical matter, this optimization procedure is quite involved. In this particular case, we tested an alternative model only because we knew we had likely made a significant amount of historical data inaccurate by changing various system attributes; thus, it was logical to consider removing that historical data from the model.

5.5 Adaptive best tree search

The statistical estimate of the number of search replicates required to achieve a specific probability of obtaining the best feasible tree, χ, is intended to inform users about the joint behavior of their data and the GARLI search algorithm, and consequently how many search replicates they should perform (Section 4.5.1). A previous limitation of the system, however, was that if χ indicated that more search replicates were needed, there was no way to "add replicates" to an existing job submission; instead, the user had to submit a separate, larger analysis. Furthermore, and perhaps even more importantly, examination of completed GARLI web service analyses revealed that 73% of best tree searches (1,194/1,630 eligible analyses) performed more search replicates than necessary. For any such analysis, the median absolute difference between the number of replicates performed and the number of replicates recommended at the 0.95 level was 92 replicates. Similarly for these analyses, the service ran, on average, 11 times as many replicates as were needed. The total number of "unnecessary" search replicates performed was 447,177 — a huge amount of probably unnecessary computation. Here we describe our implementation of an *adaptive* best tree search that automatically determines the required number of replicates and performs them on behalf of the user, thus eliminating the need for the user to "guess" how many search replicates to specify.

5.5.1 Implementation

When the GARLI web service user chooses to run an adaptive best tree search, they do not specify the number of replicates to perform. Instead, the service submits a default number of starting replicates (currently 10). When those replicates finish, post-processing is run as usual, and χ is calculated on this initial set of tree topologies. If $\chi \leq 10$ replicates at the 0.95 level, then the user is notified that their analysis is complete. If $\chi > 10$, then the service submits the necessary supplemental replicates. For example, if the best tree topology were unique among the 10 initial replicates, the statistic would recommend that 28 total search replicates be performed, and the service would submit the 18 additional required search replicates. If, for some reason, the best tree topology continued to be unique among all replicates, then the total number of required replicates would continue to increase in an approximately geometric progression: 10, 28, 82, 244, 729, ... In practice, this particular situation occurs quite frequently due to insufficient strength of signal in the data and a very large search space. Thus, because we could continue to add replicates indefinitely, we set an upper bound on the number of allowed search replicates (currently 1,000). In turn, this ensures that the maximum number of "rounds" — the number of times we add replicates to an analysis — is relatively small. If the evaluation conducted after a particular round indicates that a sufficient number of replicates have been performed, the adaptive search is terminated and the user is notified that their analysis is complete. Therefore, an adaptive best tree search will perform a minimum of 10 search replicates, and a maximum of 1,000.

5.5.2 Incentivizing usage

Initially, we did not completely eliminate the standard best tree search as an option for GARLI web service users, but instead we encouraged users to try out the adaptive search, which offers a principled and efficient means of determining search effort as opposed to specifying an arbitrary, often large number of search replicates without any intermediate evaluation of results. In order to incentivize users to choose the adaptive best tree search, we made it the default analysis type, and we limited the number of search replicates allowed in a standard best tree search to 100. Eventually, we eliminated the standard best tree search entirely in favor of the adaptive search.

5.5.3 Discussion

Having the system automatically calculate and perform the required number of search replicates is advantageous for the following reasons.

1. It introduces an objective and automatic decision process into the analysis design, eliminating guesswork and the need to evaluate intermediate results, thus saving investigator time and improving analytical results.

2. It provides reasonable and quantified assurance of adequate search space exploration for a particular phylogenetic analysis.

3. It reduces waste of grid resources and energy by running only the necessary number of search replicates.

The statistic (χ) could potentially be refined to take into account the difference in likelihood score between alternative topologies instead of the simpler "identical topology" criterion currently in use, although this would necessitate a somewhat more sophisticated mathematical and conceptual framework. It should also be noted that using χ to determine search effort is not a panacea in all instances; while we observe that the majority of replicates for "difficult" search problems return varying topologies and likelihood scores, and the majority of replicates for "easy" search problems readily converge on the same topology, one can imagine a case where every search replicate returns a similar, sub-optimal result, thereby disguising a difficult search problem as an easy one. However, as a general means of determining search effort, we find this simple statistic to be highly useful.

Resource	1 rep/100 sr	4 reps/25 sr	10 reps/10 sr	25 reps/4 sr	100 reps/1 sr
CMNS	176,443	15,153	7,865	8,107	1,407
Coppin	170,453	67,166	10,036	62,850	1,351
TerpCondor	176,411	17,888	7,569	8,164	3,032
UMIACS	386,953	25,876	10,021	5,656	1,420
Deepthought	154,273	1,240,725*	20,201	1,170,388*	39,570
Topaz	364,032	692,526*	4,484	696,157*	1,533

Table 5.1: Turnaround times for 100 `rana` search replicates submitted to dedicated resources, measured in seconds. Each value in the table represents an average of 10 submissions that were spaced apart by at least 12 hours. Values marked with an asterisk are anomalously large due to cluster maintenance that occurred over the course of the experiment.

Chapter 6: Subdividing long-running, variable-length analyses into short, fixed-length BOINC workunits

This chapter is based on the following publication: Adam L. Bazinet and Michael P. Cummings. Subdividing long-running, variable-length analyses into short, fixed-length BOINC workunits. *Journal of Grid Computing.* Submitted.

6.1 Summary

We describe a scheme for subdividing long-running, variable-length analyses into short, fixed-length BOINC workunits using phylogenetic analyses as an example. Fixed-length workunits decrease variance in analysis runtime, improve overall system throughput, and make BOINC a more useful resource for analyses that require a relatively fast turnaround time, such as the phylogenetic analyses submitted by users of the GARLI web service at `molecularevolution.org`. Additionally, we explain why these changes should benefit volunteers who contribute their processing power to BOINC projects, such as the Lattice BOINC Project (`boinc.umiacs.umd.edu`). Our results, which demonstrate the advantages of relatively short workunits, should be of general interest to anyone who develops and deploys an application on the BOINC platform.

6.2 Introduction

Computing resources volunteered by members of the general public can greatly benefit scientific research, as demonstrated by high-profile research projects in disparate areas such as radio astronomy (SETI@home; `setiathome.berkeley.edu`), climate modeling (`climateprediction.net`), protein folding (Rosetta@home; `boinc.bakerlab.org/rosetta`), and particle accelerator physics (LHC@home; `lhcathomeclassic.cern.ch/sixtrack`), to name just a few. The most widely-used platform for volunteer computing is, by far, the Berkeley Open Infrastructure for Network Computing, or BOINC [50]. Our research group has made BOINC an addressable computational resource in The Lattice Project [222, 223], a grid computing system built on Globus [49] software. In recent years, our grid system development has been increasingly focused on improving phylogenetic analysis capability [54]. Our primary phylogenetic inference application is GARLI [2, 198], a popular maximum likelihood-based program. Recently, we have made a GARLI web service publicly available at `molecularevolution.org` [53], which executes GARLI analyses on Lattice Project computing resources. The Lattice BOINC Project (`boinc.umiacs.umd.edu`) is an outstanding resource for running GARLI analyses: a significant proportion of volunteer computers have an appreciable amount of memory, which GARLI analyses often require; and GARLI automatically checkpoints its state when running on BOINC, which allows for efficient use of the BOINC platform. Indeed, having the capability to run GARLI analyses on BOINC has been critical to the successful completion of several phylogenetic studies [5, 7, 86]. However, thus far it has not been feasible to run GARLI web service analyses on BOINC because it has been difficult to guarantee complete results from BOINC in a timely manner. Here we address this problem by subdividing long-running GARLI analyses into short, fixed-length BOINC *workunits* (the term used for a unit of work on the BOINC platform). This speeds up analysis completion by reducing the variance in workunit runtimes, thus making BOINC a more attractive resource for analyses that require a relatively fast turnaround time. The remainder of the paper is organized as follows. In Section 6.3, we put the problem in context by providing some background on phylogenetic analysis and our computing systems. In Section 6.4, we provide a more detailed description of the problem and our proposed solution. In Section 6.5, we describe our implementation of the steps required to subdivide GARLI analyses into fixed-length BOINC workunits. In Sections 6.6, 6.7, and 6.8, we demonstrate the efficacy of our implementation with large-scale tests using the Lattice BOINC Project. Finally, in Section 6.9 we make some concluding remarks.

6.3 Background on phylogenetic analysis and computing systems

A very common analysis type in evolutionary biology, and increasingly in other areas of biology, is the reconstruction of the evolutionary history of organisms (e.g., species) or elements of organisms that have evolutionary or genealogical relationships (e.g., members of gene families, or sampled alleles in a population), sometimes simply called *operational taxonomic units.* This phylogenetic inference problem is especially computationally intensive when based on statistical methods that use parameter-rich models, as is commonly done with maximum likelihood and Bayesian inference. The combination of increasingly sophisticated models and rapidly increasing data set sizes has prompted the development of strategies that speed up analysis execution. Our own work has focused on decreasing time to results through parallelization of maximum likelihood phylogenetic inference, which is more amenable to atomization than Bayesian inference because the many searches that typically comprise an analysis can be performed separately and concurrently. Specifically, we have chosen to deploy an open-source program for maximum likelihood-based phylogenetic inference — GARLI (Genetic Algorithm for Rapid Likelihood Inference) [2,198] — in a heterogeneous-resource grid computing environment.

As with all phylogenetic inference programs that analyze more than a small number of operational taxonomic units and use an optimality criterion, GARLI employs a heuristic algorithm to solve the simultaneous optimization problem. Specifically, GARLI uses a stochastic evolutionary algorithm to search for the point of maximum likelihood in the multidimensional space consisting of tree topology, branch lengths, and other model parameters, which we simply call the *best tree*. Because of this stochasticity, it is both usual and recommended to perform multiple searches so as to avoid results that represent local optima, seeking instead to obtain results that more nearly reflect the global optimum. Our system assists with this task by dynamically adjusting the number of search replicates performed so as to be reasonably assured of finding the best tree with a high probability [53]. Furthermore, in addition to searches for the best tree, one typically conducts hundreds or thousands of bootstrap replicate searches to assess confidence in the bipartitions that constitute the best tree. Depending on the size and complexity of the analysis, and the capability of the computational resources used, it may take many hours to complete even a single GARLI search replicate. Thus, running many search replicates in parallel on a grid computing system greatly reduces the time required to complete an analysis.

Grid computing is a model of distributed computing that seamlessly links geographically and administratively disparate computational resources, allowing users to access them without having to consider location, operating system, or account administration [48]. The Lattice Project, our grid computing system based on Globus software, incorporates volunteer computers running BOINC, as well as traditional grid computing resources such as Condor pools [51] and compute clusters. The architecture and functionality of the grid system is described extensively elsewhere [52]; fundamentally, however, The Lattice Project provides access to scientific applications (which we call *grid services*), as well as the means to distribute the computation required by these services over thousands of processing nodes. In recent years, we have enhanced the system by developing a web interface to the GARLI grid service [53], which is currently available at `molecularevolution.org`. The GARLI grid service has been used in over 60 published phylogenetic studies, with usage having increased dramatically since the release of the GARLI web service in July 2010 [5–7, 86, 113, 130] (see `lattice.umiacs.umd.edu/publications` for the full publication list). As of 09 June 2015, 1,191 GARLI web service users have completed 6,900 analyses comprising well over three million individual search replicates.

As mentioned previously, however, we have not yet been able to use our most novel and potentially most valuable computational resource — our pool of BOINC clients — for processing GARLI web service analyses. The reasons for this are expounded upon in the following section.

6.4 Problem description and proposed solution

6.4.1 Optimal-length analyses for grid computing

There is substantial overhead associated with managing each grid-level analysis submission due to the negotiation of various layers of middleware, latency associated with file transfers, queue wait times, and so on. Thus, the submission of a large number of short-running analyses leads to reduced efficiency and reduced overall system throughput, as the majority of each analysis lifetime is composed of various sources of latency rather than actual scientific computation. Conversely, long-running analyses are not ideal either because they are more likely than short-running analyses to be interrupted by other processes, computer failures and reboots, and human intervention, which all lead to wasted computation. In the middle of these two undesirable extremes there exists a conceptually "optimal" analysis runtime, which maximizes scientific computation by minimizing both unnecessary overhead and potential for interruption. Determining a grid-wide optimal runtime is challenging when one considers the heterogeneity of analyses that are submitted, the heterogeneity of our grid resources, and the complexity of the grid meta-scheduling algorithm. Thus, here we specifically consider only GARLI analyses running on our BOINC resource, as we anticipate that such analyses will greatly benefit from runtime optimization.

6.4.2 Optimal-length GARLI analyses for BOINC

There are multiple factors that contribute to variance in GARLI analysis runtimes on BOINC. These include factors specific to the BOINC platform, such as variability as to when *result units* (instances of a workunit) are downloaded by volunteer computers, differences in volunteer computer capabilities and reliability, and variation in the computing preferences expressed by volunteers. In addition, the stochastic nature of the GARLI algorithm leads to variable and indeterminate runtimes for individual GARLI search replicates. These and other factors produce analysis batch completion dynamics with a markedly heavy tail (Figure 6.1). By standardizing the length of GARLI workunits, we aim to improve overall analysis batch turnaround time by decreasing the variance in analysis runtimes. The *optimal workunit runtime* maximizes analysis batch throughput (while not taxing system resources, such as storage space, overly much).

Assuming that a near-optimal runtime for GARLI analyses on BOINC can be determined, it is possible to combine multiple short-running GARLI analyses into a single fixed-length analysis by increasing the number of search replicates (or bootstrap replicates) that a single GARLI invocation performs. This optimization is relatively trivial, and we do not discuss it further here; instead, we focus exclusively on the converse problem of breaking up a single long-running GARLI analysis into multiple short, fixed-length subunits, which GARLI enables by providing a lightweight check-

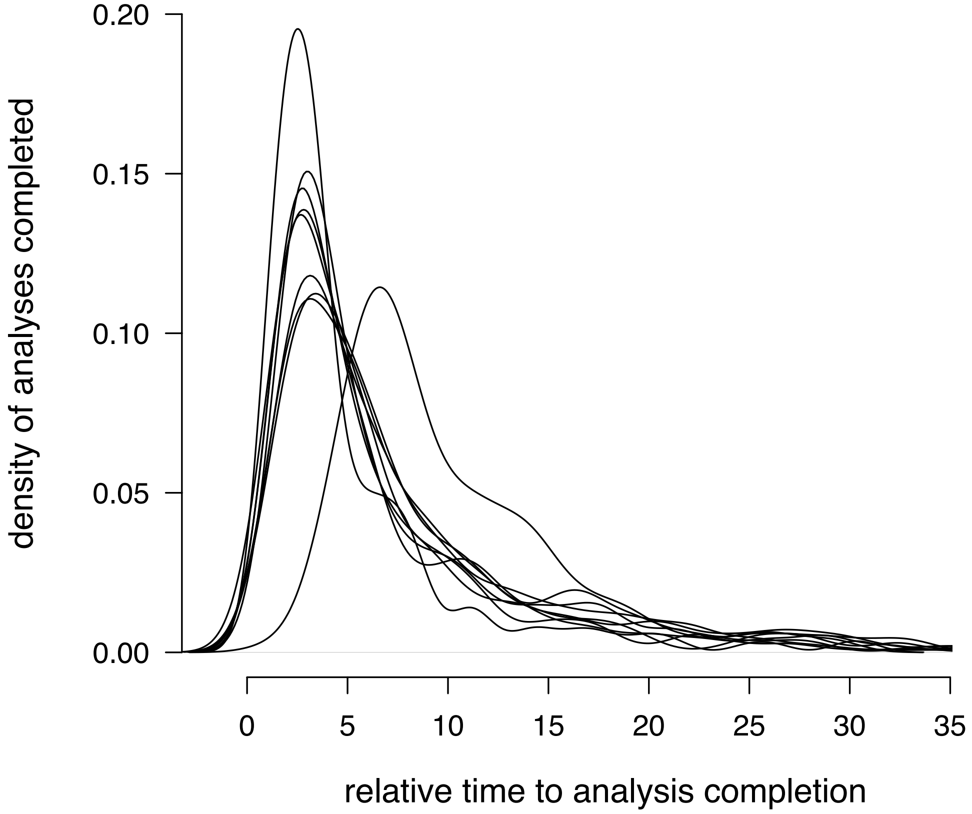


Figure 6.1: BOINC analysis batch completion dynamics for nine typical analysis batches as the density of analyses completed by their relative time to completion. All batches exhibit the typical heavy-tail distribution.

pointing mechanism. If checkpointing is activated, GARLI periodically writes some small text files to disk that contain the information needed to restart a run from that point. This ensures that not much computation is lost if a volunteer computer is rebooted, for example, or if computation is interrupted for some other reason. (Checkpointing is not currently possible during either the initial optimization or final optimization analysis stages of the program, however.) By setting various GARLI parameters appropriately, it is possible to checkpoint the state of a GARLI result unit on a volunteer computer after a fixed length of time has elapsed or amount of computation has been performed (e.g., one hour, or some number of floating point operations, respectively), terminate the analysis, and send the intermediate results back to the BOINC server. The pertinent analysis files and checkpoint files can then be downloaded by another BOINC compute node (simply termed a *host*) to resume computation where it left off, again for the same fixed length of time or amount of computation. Though this type of scheme incurs some additional overhead in terms of required data movement and storage, communication, and record keeping, we expect that these costs will be outweighed by the performance gains, which are potentially substantial and important in several ways.

6.4.3 Benefits of fixed-length analyses

The performance gains that result from the standardization of GARLI analysis runtime on the BOINC platform are realized both for BOINC volunteers, as well as researchers who use the GARLI web service.

BOINC volunteers tend to prefer uniform-length workunits, an expectation derived from participation in BOINC projects that have a practically unlimited supply of homogeneous workunits (e.g., SETI@home; `setiathome.berkeley.edu`). Hence, dividing variable-length, long-running analyses into short, fixed-length workunits better meets the expectations of BOINC volunteers and increases their enthusiasm about running GARLI analyses, which in turn leads to greater volunteer participation and retention. Furthermore, long-running analyses of unknown runtime create many opportunities for failure and interruption, as well as uncertainty and anxiety about when the analyses will finish, all of which causes some BOINC volunteers to abort such analyses prematurely. Thus, by shortening and standardizing the length of GARLI workunits, we make our system much more appealing to volunteers. Finally, standard-length workunits afford the opportunity to grant a fixed amount of credit per workunit, an inherently fair procedure that volunteers tend to favor.

For researchers using the GARLI web service, GARLI analysis runtime optimization yields performance benefits as well. As already mentioned, subdividing long-running analyses into shorter-length workunits increases reliability by decreasing the probability of premature workunit termination. It also provides a natural load-balancing mechanism by affording the most capable BOINC hosts more opportunities to process more workunits, thus lightening the tail in Figure 6.1 by shifting the distribution to the left. This decrease in the variance of analysis completion times should result in increased overall system throughput and decreased time to results for GARLI web service users.

6.4.4 Related work

There has been some research into optimizing BOINC scheduling policies [224–226], often through simulation. However, these studies attempt to solve a more general scheduling problem than we do here, and thus model many different factors: heterogeneity in host capabilities and computing preferences, variation in workunit properties and deadlines, requirements of multiple simultaneously connected BOINC projects, and so on. One such study [227] specifically focuses on optimizing scheduling policies for "medium-grained" tasks (tasks that take minutes or hours), which is relevant to our present work because we are targeting tasks of this length. We do not change or optimize any BOINC scheduling policies ourselves, however, but we would benefit from any such optimizations that already exist, especially ones targeted at relatively short tasks. In this work, we take the current BOINC scheduling policies as a given, and demonstrate how reducing workunit runtimes leads to faster turnaround time for analysis batches.

6.5 Implementation of fixed-length GARLI workunits

To implement this scheme, we divide each GARLI analysis into at least three workunits: the *initial* workunit, which performs the initial optimization phase of the analysis; 1 to *n main* workunits, which perform the bulk of the search; and the *final* workunit, which performs the final optimization phase of the analysis. As mentioned previously, checkpointing is not available during the initial or final optimization phases, so we are unable to precisely control the runtime of the initial or final workunits. However, these program phases are typically short, and do not account for more than 10% of the overall program execution time. The main workunits, on the other hand, comprise the majority of the runtime, and their maximum execution time can be precisely controlled.

To divide program execution into phases, a `workphasedivision` option was added to GARLI version 2.1. When `workphasedivision=1`, GARLI automatically checkpoints and terminates immediately after initial optimization is complete, and immediately before final optimization begins. Additionally, the `stoptime` parameter, which is a positive number of seconds after which an analysis should be terminated, was redefined in GARLI version 2.1 to be relative to the time an analysis was most recently restarted instead of its very beginning. Thus, by setting `stoptime=3600`, for example, one may cause a GARLI main workunit to terminate after one hour of runtime. (Note: `stoptime` is ignored during the initial and final optimization phases.)

6.5.1 GARLI checkpoint files and BOINC homogeneous redundancy

Unfortunately, GARLI checkpoint files are not portable between operating systems, and may not even be portable between 32 bit and 64 bit variants of the same operating system. This presents a major implementation obstacle, as one may not simply mix and match execution hosts indiscriminately. To deal with this issue, we made use of a BOINC feature called *homogeneous redundancy* (HR), which was originally developed to ensure that multiple instances of the same workunit (termed *result units*) would run on the same "class" of host. This guaranteed that the numerical output from multiple result units would match exactly, which was required to use a voting scheme to verify that results were computed correctly. Depending on how a particular application was compiled and what computations it was performing, host classes could be more or less broadly defined. Maximally-inclusive host classes are desirable because having more hosts available to run any particular workunit improves overall system throughput. BOINC currently defines two HR types: a *coarse-grained* type in which there are four host classes (Windows, Linux, Mac-PowerPC, and Mac-Intel), and a *fine-grained* type in which there are 80 host classes (four operating system and 20 CPU types). For our testing, we enabled coarse-grained HR for GARLI, along with a BOINC feature called *homogeneous app version* (HAV) that ensured consistent use of either the 32 bit or 64 bit version of GARLI. These settings did not completely eliminate errors related to checkpoint portability, but allowed testing to proceed with a sufficiently low error rate (less than 2%).

With normal use of HR, each BOINC workunit may have a different HR class; it is the various result units associated with a particular workunit that must have the same HR class. Thus, the HR class for a given workunit is not usually determined until its first result unit is assigned to a particular host. In our scheme, the main and final workunits associated with a particular GARLI analysis must have the same HR class as that of the initial workunit, so we needed to set the HR class of the main and final workunits at the time of their creation. To accomplish this, we used a new argument to the BOINC `create_work` program. In addition, we added the hr_class_static tag to the BOINC configuration file, which suppresses the mechanism that clears the HR class of a workunit if a result unit fails when there are no other result units for that workunit in progress or already completed.

6.5.2 Modifications to grid system components

The implementation of this scheme was relatively complex and involved changes to several different grid system components. The component that was the most heavily modified was the BOINC job manager, the Perl script responsible for transforming a generic Globus job description into appropriate BOINC workunits [222]. The BOINC job manager checks the GARLI configuration file for workphasedivision=1; upon finding it, the script creates the initial workunit and writes three separate workunit templates and assimilator scripts for the analysis, one for each workunit type (initial, main, and final). The workunit templates specify the input and output files for each workunit, which vary depending on the workunit type; they also specify that input and output files associated with initial and main workunits are not allowed to be deleted immediately after such workunits complete, unlike files associated with regular GARLI workunits. The appropriate assimilator script is invoked when a workunit of a particular type completes successfully. The *initial* assimilator script sets restart=1 in the GARLI configuration file, which causes the main and final workunits to restart from checkpoint files. It also moves the checkpoint files and the standard output (associated with the canonical result of the initial workunit) from the BOINC upload directory to the download directory, so these can be used as additional input files to the first main workunit. Finally, the initial assimilator script creates the first main workunit using the correct templates and other parameters, and sets its HR class to that of the initial workunit. The *main* assimilator script parses the GARLI log file to determine if the analysis is ready for final optimization; if so, it creates the final workunit; if not, it creates the next main workunit. The *final* assimilator script copies the final output files to the location where Globus expects them, removes all intermediate output files that may be resident on disk from associated initial or main workunits, and updates the BOINC database.

Numerous changes were made to other grid system components as well; a few examples follow. The BOINC scheduler event generator (SEG), a Globus component that periodically queries the BOINC database for the status of jobs [52], was modified to include final workunits in its queries, but to exclude initial and main workunits from such queries. The BOINC validator, a daemon that verifies that GARLI results returned by BOINC clients include a valid tree file [52], was modified to ignore results from initial or main workunits. The BOINC assimilator, a daemon that processes successfully completed workunits [52], was modified so that the number of the main workunit was passed to our custom assimilator scripts, among other minor changes.

Although not discussed here in detail, we made additional modifications to support *analysis batches*, which allowed multiple initial workunits to be created simultaneously and to be associated with one another as a batch of analyses. Each initial workunit still generates its own main and final workunits that are tracked and updated independently of those associated with other initial workunits in the batch.

This functionality allowed us to quickly and easily submit batches of thousands of workunits, which was the order of magnitude required to properly evaluate the performance of this scheme.

Should we enable this scheme for production GARLI web service analyses in the future, some additional development will be necessary to support GARLI analyses that specify multiple search replicates or bootstrap replicates, and to support analyses that use different numbers and types of input and output files. More robust status updating, workunit tracking, and error handling will also be needed. However, the development described up to this point was sufficient to enable large-scale testing of the fixed-length workunit paradigm, and to compare it to the normal, "full-length" paradigm in which a single BOINC workunit executes an entire GARLI analysis from start to finish. We describe this testing in the following sections.

6.6 Fixed-length vs. full-length GARLI workunit tests

We decided that a comparison of the new "fixed-length workunit" paradigm to the standard "full-length workunit" paradigm was best accomplished with large-scale BOINC testing — i.e., we would assess runtimes and other performance characteristics using thousands of analyses, which would exercise the BOINC client pool in a realistic manner. For these tests, we analyzed an 82-taxon, 13-mitochondrial-gene data set with GARLI using a codon model. If uninterrupted, the runtime of this analysis on an average computer was approximately 10 to 15 hours, and the 1024 MB memory requirement was low enough that the majority of clients could participate. We used the following GARLI settings: `randseed=42`; `availablemem=1024`; and `stopgen=30000`. Additionally, for fixed-length analyses, we set `stoptime=3600`, which caused main workunits to terminate after one hour. BOINC workunit wall clock deadlines were set to two days for initial and final workunits, and six hours for main workunits. The deadline for full-length workunits was one week. Each test began with the submission of 1,000 workunits (either 1,000 initial workunits, or 1,000 full-length workunits). Other test attributes, including the date of the test, the number of result units per workunit, and the status of the hosts in the BOINC pool at the time of submission are given in Table 6.1.

The purpose of this series of tests was, first and foremost, to compare the performance of series of fixed-length workunits to standard, full-length workunits. Secondarily, we also sought to measure the effect of using two result units per workunit instead of just one. For each combination of fixed-length or full-length, and one result unit or two result units, we performed two large-scale tests to increase the overall precision of our assessment; this totaled eight tests (Table 6.1).

For evaluation purposes, we measured *total analysis time* as follows. For a fixed-length workunit series, total analysis time was measured as the time interval beginning when the initial workunit was created, and ending when valid results from the final workunit were returned to our BOINC server. For a full-length workunit, total analysis time was measured simply as the time interval beginning when the workunit was created, and ending when valid results from the workunit were returned to our BOINC server. In Figures 6.2 and 6.3, we compare the total analysis time of fixed-length and full-length analysis batches; Figure 6.2 includes the tests that used one result unit per workunit, and Figure 6.3 includes the tests that used two result units per workunit.

The one-result unit comparison (Figure 6.2) shows the general pattern that we expected to observe: the variance in total analysis time is lower in the fixed-length workunit scheme. Thus, while the fixed-length scheme takes longer to complete $\approx 70\%$ of the analyses, it completes all of its analyses $\approx 2.3\times$ more quickly than the equivalent number of full-length analyses.

The effect of doubling the number of result units per workunit (Figure 6.3) is also apparent: the analysis batches complete more quickly, as faster hosts in the pool are given the opportunity to process more work. The effect is greatest for the full-length analysis batches, which complete $\approx 3.3\times$ more quickly in the two-result unit tests. Comparing the fixed-length scheme to the full-length scheme in the two-result unit case, however, we observe a performance pattern that is similar to the one-result unit case, as the performance of the fixed-length scheme begins to equal or outperform the full-length scheme at a large proportion of analyses completed.

Thus, here we demonstrate two ways of improving performance: 1) using a series of fixed-length workunits instead of a single full-length workunit, which incurs no additional cost in terms of BOINC client resources; and 2) doubling the number of result units per workunit, which incurs twice the cost in BOINC client resources. Supporting summary statistics for these tests are given in Table 6.2.

Date of test	Test type	Result units per workunit	Hosts granted credit[a,b]	Hosts reporting[a,b]	Result units in progress[a]
14 Jan 2015	fixed-length	one	$\approx 1,300$	3,813	37
21 Jan 2015	full-length	one	$\approx 1,325$	3,755	26
31 Jan 2015	fixed-length	one	$\approx 1,300$	3,715	165
02 Feb 2015	full-length	one	$\approx 1,400$	3,724	123
05 Feb 2015	fixed-length	two	$\approx 1,350$	3,698	331
08 Feb 2015	full-length	two	$\approx 1,420$	3,732	145
13 Feb 2015	fixed-length	two	$\approx 1,430$	3,724	172
15 Feb 2015	full-length	two	$\approx 1,480$	3,695	134

[a] Conditions at the time of submission.

[b] Tallied over the previous 30 days.

Table 6.1: Attributes of large-scale BOINC tests of fixed-length vs. full-length GARLI workunits.

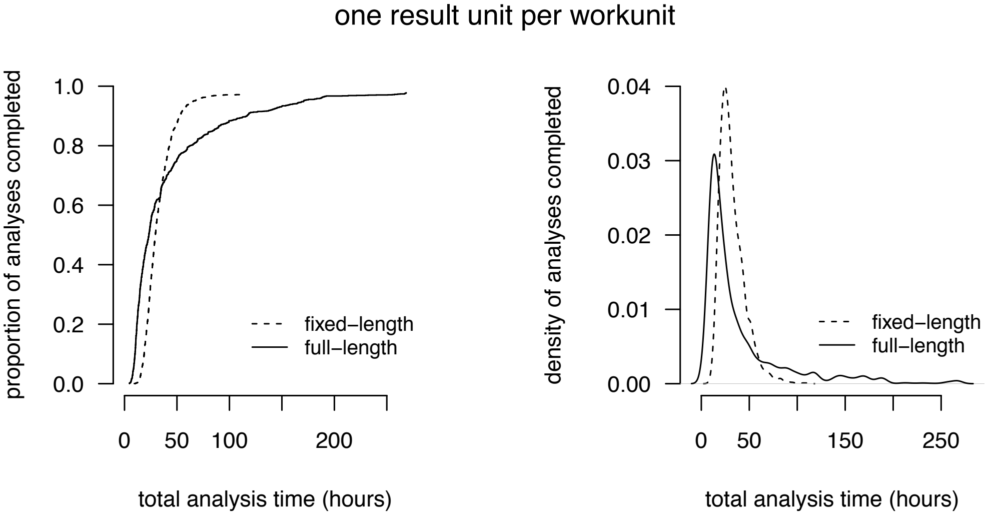


Figure 6.2: Total analysis time, in hours, for fixed-length and full-length analysis batches that used one result unit per workunit. A cumulative distribution plot gives the proportion of analyses completed by total analysis time, and a density plot gives the density of analyses completed by total analysis time. Each line shown is derived from a series of at most 2,000 points (1,000 from each test replication), where each point represents an individual GARLI analysis.

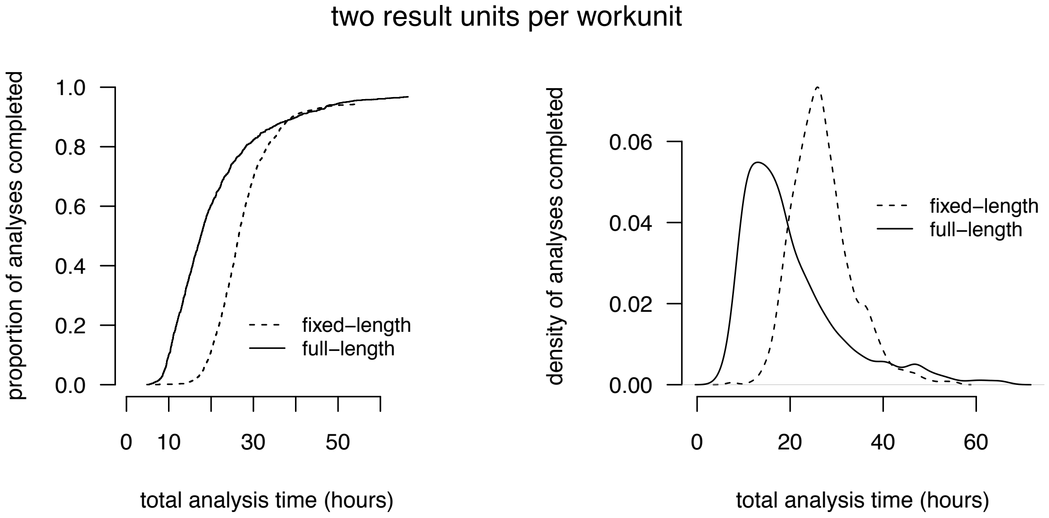


Figure 6.3: Total analysis time, in hours, for fixed-length and full-length analysis batches that used two result units per workunit. A cumulative distribution plot gives the proportion of analyses completed by total analysis time, and a density plot gives the density of analyses completed by total analysis time. Each line shown is derived from a series of at most 2,000 points (1,000 from each test replication), where each point represents an individual GARLI analysis.

Date of test	Test type	Result units per workunit	Analyses included in results	Mean analysis time (hr)	Median analysis time (hr)	Standard deviation (hr)
14 Jan 2015	fixed-length	one	966	37.0	35.2	14.3
21 Jan 2015	full-length	one	962	33.4	24.2	24.6
31 Jan 2015	fixed-length	one	968	26.1	25.5	5.5
02 Feb 2015	full-length	one	959	21.1	17.8	11.5
05 Feb 2015	fixed-length	two	978	27.1	25.2	9.5
08 Feb 2015	full-length	two	994	26.7	18.0	21.7
13 Feb 2015	fixed-length	two	920	28.0	26.8	7.4
15 Feb 2015	full-length	two	977	19.4	16.7	10.0

Table 6.2: Results of large-scale BOINC tests of fixed-length vs. full-length GARLI workunits.

6.7 Optimal-length GARLI workunit tests

The next round of large-scale tests was intended to approximately determine an efficient length for GARLI main workunits. For these tests, all of which were fixed-length, we used the same 82-taxon, 13-mitochondrial-gene data set as before, together with the same GARLI settings, except that we varied `stoptime` so as to test main workunit lengths of 30 minutes, 60 minutes, 120 minutes, and 240 minutes. As before, the wall clock deadline for initial and final workunits was set to two days; for main workunits, the deadline was scaled proportionally to the main workunit length (Table 6.3). Each test began with the submission of 1,000 initial workunits. Other test attributes, including the date of the test, the main workunit length and wall clock deadline, and the status of the hosts in the BOINC pool at the time of submission are given in Table 6.3. For each main workunit length evaluated, we performed two large-scale tests to increase the overall precision of our assessment; this totaled eight tests (Table 6.3).

For our evaluation, we measured total analysis time as in our previous round of testing. Figure 6.4 compares total analysis time of analysis batches of varying main workunit lengths.

We observe, in general, that varying the main workunit length does not impact the performance characteristics of analysis batches especially greatly, at least at the main workunit lengths we tested. As before, we observe less variance in analysis time with shorter main workunit lengths. Once we go as low as 30 minutes, however, we notice some deleterious effects of overhead associated with generating the increased number of workunits and input files that are required. Indeed, each successive halving of main workunit runtime incurs twice as much file system and database storage cost, and doubles the processing load on our servers. Thus, as the 60-minute and 120-minute runtimes performed comparably, we would probably choose a main workunit runtime of 120 minutes (two hours) to minimize overhead costs. A two-hour runtime is certainly in keeping with our *a priori* expectation of a reasonable main workunit length, an expectation based on many years of interaction with our BOINC volunteers. Supporting summary statistics for these tests are given in Table 6.4.

6.8 Final fixed-length vs. full-length GARLI workunit test

The final round of large-scale tests was intended to measure the performance of the fixed-length scheme against the full-length scheme with a longer GARLI analysis. For these tests, we used the same 82-taxon, 13-mitochondrial-gene data set as before, along with the same GARLI settings, except that we set `stopgen=60000` and `enforcetermconditions=0`, which together roughly doubled the length of the analysis. For the fixed-length test, we set `stoptime=7200`, which was the best-performing main workunit runtime determined from the previous round of tests. BOINC wall clock deadlines were set to two days for initial and final workunits, and 12 hours for main workunits. The deadline for full-length workunits was one week. Each test began with the submission of 1,000 workunits (either 1,000 initial workunits, or 1,000 full-length workunits). Other test attributes, including the date of

Date of test	Main WU length (minutes)	Main WU deadline (minutes)	Hosts granted credit[a,b]	Hosts reporting[a,b]	Result units in progress[a]
19 Feb 2015	30	180	$\approx 1,460$	3,687	233
22 Feb 2015	60	360	$\approx 1,520$	3,695	133
24 Feb 2015	120	720	$\approx 1,580$	3,707	123
26 Feb 2015	240	1,440	$\approx 1,545$	3,721	123
02 Mar 2015	240	1,440	$\approx 1,620$	3,776	24
05 Mar 2015	120	720	$\approx 1,630$	3,771	69
07 Mar 2015	60	360	$\approx 1,630$	3,772	45
10 Mar 2015	30	180	$\approx 1,635$	3,794	76

[a] Conditions at the time of submission.

[b] Tallied over the previous 30 days.

Table 6.3: Attributes of large-scale BOINC tests to determine optimal-length GARLI workunits.

Date of test	Main WU length (minutes)	Main WU deadline (minutes)	Analyses included in results	Mean analysis time (hr)	Median analysis time (hr)	Standard deviation (hr)
19 Feb 2015	30	180	896	35.3	34.8	10.2
22 Feb 2015	60	360	945	29.6	28.1	11.6
24 Feb 2015	120	720	898	30.6	29.9	12.3
26 Feb 2015	240	1,440	888	30.3	25.9	16.3
02 Mar 2015	240	1,440	964	32.5	30.4	16.1
05 Mar 2015	120	720	979	30.3	28.1	12.6
07 Mar 2015	60	360	963	33.7	32.1	13.8
10 Mar 2015	30	180	963	38.0	38.1	10.3

Table 6.4: Results of large-scale BOINC tests to determine optimal-length GARLI workunits.

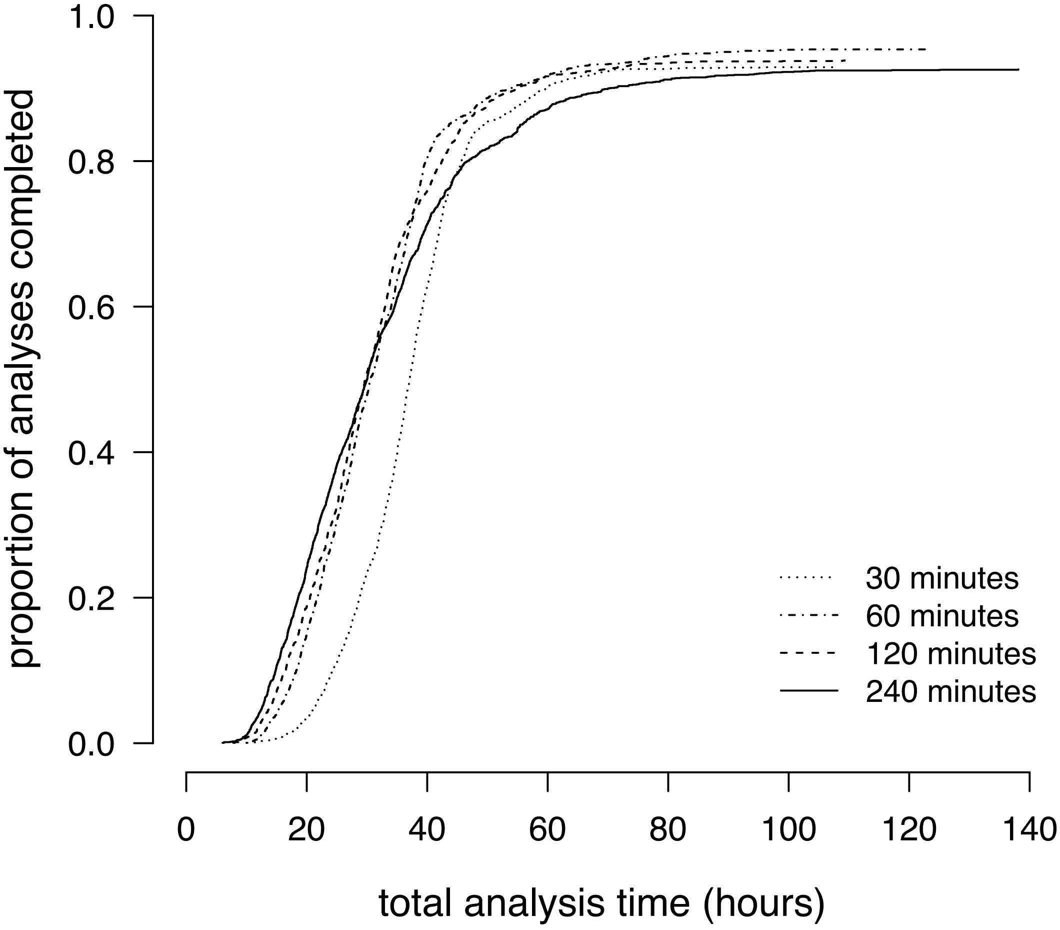


Figure 6.4: Total analysis time, in hours, for analysis batches of different main workunit lengths. A cumulative distribution plot gives the proportion of analyses completed by total analysis time. Each line shown is derived from a series of at most 2,000 points (1,000 from each test replication), where each point represents an individual GARLI analysis.

the test, the number of result units per workunit, and the status of the hosts in the BOINC pool at the time of submission are given in Table 6.5.

For our evaluation, we measured total analysis time the same way as in our previous rounds of testing. Figure 6.5 compares the total analysis time of fixed-length and full-length analysis batches.

We observe the same pattern that we did in previous tests; the variance in total analysis time is significantly reduced in the fixed-length workunit test. Thus, while the fixed-length scheme takes longer to complete ≈ 50% of the analyses, it completes all of its analyses ≈ 1.8× more quickly than the equivalent number of full-length analyses. Thus, we note that the relative performance of the fixed-length paradigm improves as overall analysis length increases. Supporting summary statistics for these tests are given in Table 6.6.

6.9 Conclusion

As the preceding tests demonstrate, the reduction in analysis time variance achieved by subdividing long-running GARLI analyses into short, fixed-length BOINC workunits results in faster completion times for analysis batches. Furthermore, taking a number of factors into consideration, we arrived at a best-performing main workunit length of two hours. We also demonstrated how the relative performance of the fixed-length workunit scheme improves as overall analysis length increases. Although with a highly heterogeneous pool of consumer-grade computers there will always be some degree of variance in analysis completion times, our results suggest

Date of test	Test type	Result units per workunit	Hosts granted credit[a,b]	Hosts reporting[a,b]	Result units in progress[a]
23 Mar 2015	full-length	one	$\approx 1,610$	3,865	3
31 Mar 2015	fixed-length	one	$\approx 1,480$	3,875	270

[a] Conditions at the time of submission.

[b] Tallied over the previous 30 days.

Table 6.5: Attributes of final large-scale tests of fixed-length vs. full-length GARLI workunits.

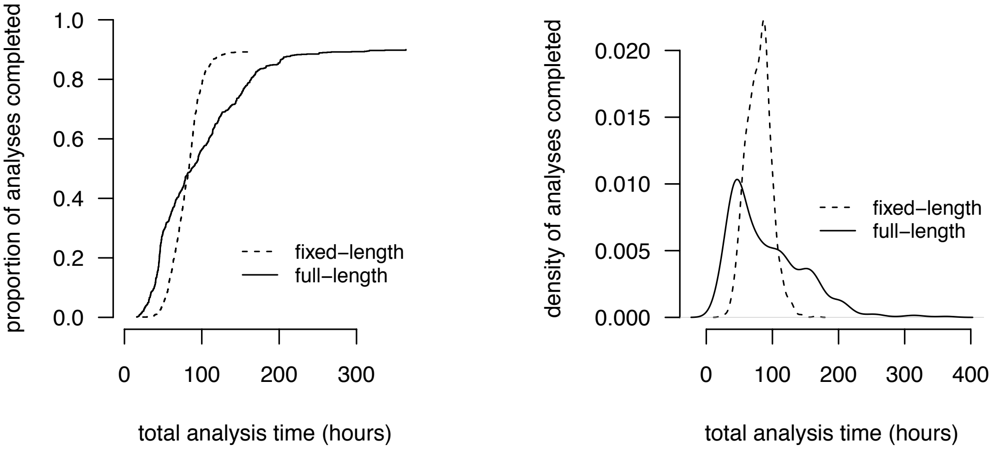


Figure 6.5: Total analysis time, in hours, for fixed-length and full-length analysis batches. A cumulative distribution plot gives the proportion of analyses completed by total analysis time, and a density plot gives the density of analyses completed by total analysis time. Each line shown is derived from a series of at most 1,000 points, where each point represents an individual GARLI analysis.

that the heavy tail on analysis batches (Figure 6.1) can be substantially reduced by subdividing analyses into short workunits. We would expect these results to generalize to other BOINC applications as well. Therefore, other BOINC projects, even those whose applications checkpoint, may be motivated by these results to shorten their workunits. In our case, we are optimistic that this reduction in runtime variance, along with strategies such as submitting more than the required number of analyses and using the first results that become available, will make BOINC a viable and effective resource for processing GARLI web service analyses.

Date of test	Test type	Result units per workunit	Analyses included in results	Mean analysis time (hr)	Median analysis time (hr)	Standard deviation (hr)
23 Mar 2015	full-length	one	900	91.9	77.3	56.1
31 Mar 2015	fixed-length	one	893	79.4	80.1	18.6

Table 6.6: Results of final large-scale tests of fixed-length vs. full-length GARLI workunits.

Chapter 7: Can RNA-Seq resolve the rapid radiation of advanced moths and butterflies (Hexapoda: Lepidoptera: Apoditrysia)? An exploratory study

This chapter is based on the following publication: Adam L. Bazinet, Michael P. Cummings, Kim T. Mitter, and Charles W. Mitter. Can RNA-Seq resolve the rapid radiation of advanced moths and butterflies (Hexapoda: Lepidoptera: Apoditrysia)? An exploratory study. *PLoS ONE* 8(12):e82615, 2013. Corrections included.

7.1 Background and motivation

As mentioned in Section 1.2.3, RNA-Seq enables the acquisition of relatively large amounts of genomic data with lower cost and less effort than complete genome sequencing would require. Moreover, the data obtained is primarily expressed protein-coding sequence that is useful for phylogenetic analysis. Thus, RNA-Seq has become a popular choice as a data generation method for phylogenomic analyses, although other techniques are also being used and developed [228–230].

Following the pioneering methodology of Hittinger et al. [11], many additional phylogenomic studies have been undertaken [15–25, 25–42]. The majority of these studies use the same canonical workflow: specimen collection; RNA or DNA purification, library preparation, and sequencing; quality control and filtering; transcript assembly; orthology determination; multiple sequence alignment; and phylogenetic analysis.

As part of an effort known informally as “Leptree-II”, we have sequenced upwards of 67 lepidopteran transcriptomes with the Illumina HiSeq 1000, generating many gigabases of sequence data. Additionally, we have reassembled various other publicly available lepidopteran transcriptomes to incorporate into our phylogenomic analyses. This work is expected to elucidate previously unresolved aspects of lepidopteran evolution, starting with difficult-to-resolve early divergences known as the lepidopteran “backbone”.

A phylogenomic workflow for analyzing this data has been developed as part of the dissertation research. In addition to the implications for the evolutionary relationships of Lepidoptera, the methodology used to construct the Leptree-II workflow will itself be of interest to researchers pursuing similar goals for other taxonomic groups. In this chapter, we describe the first version of the phylogenomic workflow that we used to analyze 46 lepidopteran taxa. In the following chapter, we describe an updated version of the workflow that we used to analyze 82 taxa.

7.2 Introduction

The insect order Lepidoptera (moths and butterflies; >157,000 spp.; [231]) is arguably the largest single radiation of plant-feeding insects. A prominent element of terrestrial ecosystems, Lepidoptera function as herbivores, pollinators and prey, with substantial impact on humans. Highly destructive as agricultural pests, they have also become icons for environmental conservation, and supply food and fiber to multiple societies [232]. And, they provide important model systems for studies of genetics, physiology, development, and many aspects of ecology and evolutionary biology [233], including the question of why herbivorous insects, 25% of earth's known species, are so species-rich [234–236].

A robust phylogenetic framework is essential for all attempts to understand the diversity, adaptations and ecological roles of Lepidoptera. The past decade has seen tremendous advances in our understanding of lepidopteran phylogeny at all levels. Molecular data have proven especially powerful for defining superfamilies and relationships within them. In a remarkable burst of community progress, robust molecular phylogenies for nearly all of the major superfamilies (those containing hundreds to thousands of species), combined with review of the morphological evidence, have been published in the past few years or will be forthcoming shortly. Recent examples (not an exhaustive list) include studies of Bombycoidea [111], Gelechioidea [237], Geometroidea [238–240], Gracillarioidea [113], Noctuoidea [241, 242], Papilionoidea [243], Pyraloidea [98], Tortricoidea [99], and Yponomeutoidea [6]. In all of these superfamilies, a majority of the major divergences (at least) seem credibly established, though important uncertainties remain. Progress is also now rapid at more subordinate levels.

The past few years have likewise seen the first attempts at "backbone" phylogenies spanning much or all of the order [5, 114, 244]. A recent such study [7], with the largest gene and taxon sampling to date, used 483 exemplars, representing 115 of the approximately 125 families of Lepidoptera [245], sequenced for up to 19 nuclear protein-encoding genes/14.7 kb. It gave a topology quite similar to those of earlier nuclear gene studies, but with stronger bootstrap support. It also agrees with newly-emerging evidence from whole mitochondrial genomes (e.g., [244, 246]; see Discussion). The main conclusions of the Regier et al. study [7] are summarized in Figure 7.1.

The so-called non-ditrysian lineages (Figure 7.1, left side) are mostly species-poor but rich in morphological variation, and often have apparently relictual distributions suggesting great age. Exhaustive comparative-anatomical studies of these groups (e.g., [247–249]), an early application of Hennigian phylogenetics, yielded many synapomorphies and a well-resolved backbone phylogeny. Although important puzzles remain, the molecular data strongly resolve a majority of these early divergences, recovering previously-recognized major clades including Glossata, Heteroneura and Eulepidoptera (Figure 7.1). There is also strong molecular support for several novel proposals, such as apparent non-monophyly of Palaephatidae. The molecular data strongly corroborate the clade Ditrysia, named for the presence in the female Terminalia of separate openings for mating and for oviposition, which contains over 98% of lepidopteran species and 80% of the families.

The superfamilies of Ditrysia, in contrast to the non-ditrysians, tend to be species-rich, cosmopolitan and less distinct morphologically, so that major groupings have been difficult to discern. The authoritative morphological hypothesis synthesized by Kristensen and collaborators [245, 250, 251] postulated only 11 tentative

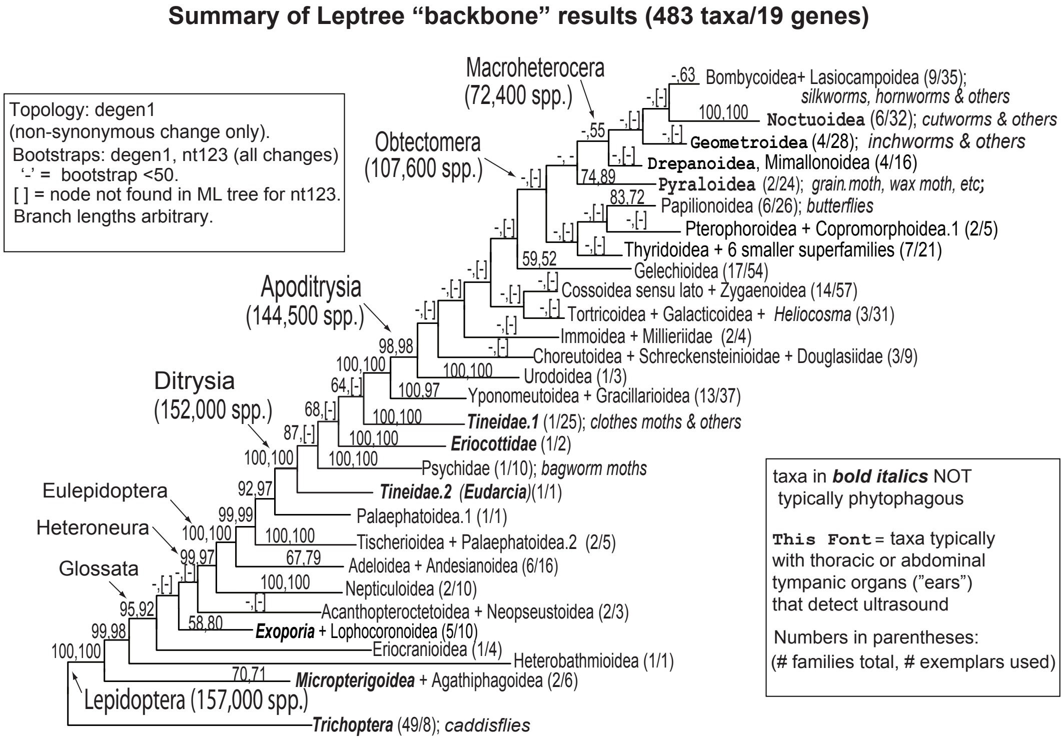

Summary of Leptree “backbone” results (483 taxa/19 genes)
Topology: degen1
(non-synonymous change only).
Bootstraps: degen1, nt123 (all changes)
‘-’ = bootstrap <50.
[ ] = node not found in ML tree for nt123.
Branch lengths arbitrary.
Macroheterocera
(72,400 spp.)
Obtectomera
(107,600 spp.)
Apoditrysia
(144,500 spp.)
Ditrysia
(152,000 spp.)
Eulepidoptera
Heteroneura
Glossata
Lepidoptera (157,000 spp.)
-,63
Bombycoidea+ Lasiocampoidea (9/35); silkworms, hornworms & others
-,[-]
100,100
Noctuoidea (6/32); cutworms & others
-,55
-,[-]
-,[-]
Geometroidea (4/28); inchworms & others
-, -
-,[-]
Drepanoidea, Mimallonoidea (4/16)
-,[-]
74,89
Pyraloidea (2/24); grain.moth, wax moth, etc;
83,72
Papilionoidea (6/26); butterflies
-,[-]
-,[-]
Pterophoroidea + Copromorphoidea.1 (2/5)
-,[-]
-,[-]
Thyridoidea + 6 smaller superfamilies (7/21)
-,[-]
59,52
Gelechioidea (17/54)
-,[-]
-,[-]
Cossoidea sensu lato + Zygaenoidea (14/57)
-,[-]
-,[-]
Tortricoidea + Galacticoidea + Heliocosma (3/31)
-,[-]
-,[-]
Immoidea + Millieriidae (2/4)
-,[-]
Choreutoidea + Schreckensteinioidae + Douglasiidae (3/9)
98,98
100,100
Urodoidea (1/3)
100,100
100,97
Yponomeutoidea + Gracillarioidea (13/37)
64,[-]
100,100
Tineidae.1 (1/25); clothes moths & others
68,[-]
100,100
Eriocottidae (1/2)
87,[-]
100,100
Psychidae (1/10); bagworm moths
100,100
Tineidae.2 (Eudarcia)(1/1)
92,97
Palaephatoidea.1 (1/1)
99,99
100,100
Tischerioidea + Palaephatoidea.2 (2/5)
100,100
67,79
Adeloidea + Andesianoidea (6/16)
99,97
100,100
Nepticuloidea (2/10)
-,[-]
-,[-]
Acanthopteroctetoidea + Neopseustoidea (2/3)
-,[-]
58,80
Exoporia + Lophocoronoidea (5/10)
95,92
Eriocranioidea (1/4)
99,98
Heterobathmioidea (1/1)
100,100
70,71
Micropterigoidea + Agathiphagoidea (2/6)
Trichoptera (49/8); caddisflies
taxa in bold italics NOT typically phytophagous
This Font = taxa typically with thoracic or abdominal tympanic organs (”ears”) that detect ultrasound
Numbers in parentheses:
(# families total, # exemplars used)

Figure 7.1: Summary of previous backbone phylogeny results (483 taxa/19 genes), modified from Regier et al. [7]. ML topology shown for `degen1` (non-synonymous change only) is based on 100 GARLI searches. Bootstrap percentages are `degen1` followed by `nt123` (all nucleotides), based on 1,000 bootstrap replicates with 15 search replicates each. Only values greater than 50% are shown. Branch lengths are arbitrary. The “-” means the node was not found in the ML tree for `nt123`. Numbers in parentheses after taxon names indicate number of families/number of exemplars studied. Names in bold denote clades in which larvae are not typically phytophagous. Names in serif font denote clades in which adults typically bear ultrasound-detecting tympanic organs on the thorax and/or abdomen. Classification follows van Nieukerken et al. [231].

monophyletic groupings among the 33 ditrysian superfamilies recognized. Molecular data markedly strengthen resolution for the initial divergences within Ditrysia. There is now strong molecular support (Figure 7.1) for the morphological inference that all Ditrysia apart from Tineoidea form a monophyletic group. Molecular data also strongly support four new or previously uncertain conclusions: (1) The Tineoidea themselves are paraphyletic with respect to all other Ditrysia; (2) Yponomeutoidea and Gracillarioidea are sister groups; (3) Yponomeutoidea and Gracillarioidea together form the sister group to the remaining Ditrysia; and (4), the remaining ditrysians form a strongly supported group consisting of Apoditrysia in an earlier sense [252, 253] plus Gelechioidea. Apoditrysia *sensu novo* [231], now including Gelechioidea, are also supported by several morphological synapomorphies [237, 254].

In striking contrast to those in earlier-originating clades, "backbone" relationships in the Apoditrysia *sensu lato* are almost entirely lacking in strong support from either molecules or morphology, although rogue taxon removal [255] helps somewhat. Recent large-scale molecular studies consistently recover monophyly of some variant of the huge group Obtectomera (107,551 spp.; [231]), originally proposed for families with relatively immobile pupae [252], but support is very weak (Figure 7.1). Molecular studies also find the large superfamily Gelechioidea to be closely related to Obtectomera, but again with weak support (Figure 7.1). Within Obtectomera, the morphological working hypothesis recognized a group Macrolepidoptera, consisting of the butterflies (Papilionoidea; 18,363 spp. [231]) and the familiar large moths (inchworms, cutworms, silkmoths and relatives; five superfamilies, 72,398 spp.; [231]).

Molecular studies have instead consistently separated the butterflies from the large moths, and found that the latter, termed the Macroheterocera [231], are more closely related to the non-macrolepidopteran superfamily Pyraloidea (15,587 spp.; [231]). These findings too, however, have weak bootstrap support (Figure 7.1). Within Macroheterocera, neither nuclear genes nor morphology provide strong evidence for any relationships at all among superfamilies (Figure 7.1; but see [246, 256]). This phylogenetic uncertainty, in turn, limits the power of analyses of the origins, ages and evolutionary consequences of traits hypothesized to promote the spectacular diversification of Apoditrysia, which include 144,524 species in 93 families and 26 superfamilies according to a recent classification [231].

Low support along the apoditrysian backbone probably reflects rapid diversification, as in other major insect radiations [257, 258]. The alternative explanation, of pervasive strong conflict among gene trees, found little support in our earlier studies [5]. If short branches resulting from rapid radiation are the problem, it may be feasible to strengthen resolution by radically increasing the gene sample. Empirical tests of this proposition, however, have been few. In this chapter we assess the potential of massive gene sampling for resolving the apoditrysian radiation by analyzing 741 gene sequences, obtained through RNA-Seq, in 46 exemplars spanning nearly all major lineages of Apoditrysia. The resulting dramatic but non-uniform increase in bootstrap support illustrates both the power and the complexity of the phylogenomic approach.

7.3 Taxon sampling and taxon set design

The goal of this study was to assess the degree to which RNA-Seq transcriptome data can increase the support for relationships among the superfamilies of Apoditrysia over that found in our previous 19-gene study [7]. Our 46 exemplars include 42 apoditrysians spanning 16 of 26 superfamilies and 34 of 93 families of Apoditrysia in a recent classification [231]. The distribution of our exemplars across that classification is shown in Table 7.1, while the collecting locality, accession number and other details for each specimen are given in Table S1 of Bazinet et al. [86]. The only large apoditrysian superfamily (>1,000 species) not sampled was Papilionoidea. The phylogenetic position of Papilionoidea is the focus of a forthcoming independent RNA-Seq study that is yielding results similar to those we report below [25].

As outgroups we used two non-apoditrysian Ditrysia and two non-ditrysians. For two taxa we used previously published data: for *Bombyx mori*, we used the published genome (SilkDB; [259]), and for *Striacosta albicosta*, we reassembled raw sequences from an earlier study that used older sequencing technology [179]. The purpose of including *S. albicosta* was to gauge how much data can be extracted from such older transcriptome studies, and whether these data can be successfully incorporated into a phylogeny estimate based mainly on newer, larger transcriptome assemblies. For the other 44 taxa we generated transcriptomes *de novo* by RNA-Seq. We matched the taxa included as closely as possible to those in our previous backbone study [7]. Thirty-eight of the 44 species had been included in that study, and for a majority of these we were able to use the same specimen. Four other species

LEPIDOPTERA (43 superfamilies, including all those below)
Hepialoidea: Hepialidae: Phymatopus californicus
Palaephatoidea: Palaephatidae: Palaephatus luteolus
DITRYSIA (29 superfamilies, including all those below)
Tineoidea: Psychidae: Thyridopteryx ephemeraeformis
Yponomeutoidea: Yponomeutidae: Yponomeutinae: Yponomeuta multipunctella
APODITRYSIA (26 superfamilies, including all those below)
Urodoidea: Urodidae: Urodus decens
Zygaenoidea: Epipyropidae: Epipomponia nawai
Lacturidae: Lactura subfervens
Limacodidae: Limacodinae: Euclea delphinii
Megalopygidae: Megalopyginae: Megalopyge crispata
Zygaenidae: Zygaeninae: Zygaena fausta
Cossoidea: Cossidae: Cossinae: Culama sp. 5, Prionoxystus robiniae; Hypoptinae: Givira mucidus; Zeuzerinae: Psychogena personalis; Cossulinae: Spinulata maruga
Dudgeoneidae: Archaeoses polygrapha
Sesiidae: Sesiinae: Podosesia syringae, Vitacea polistiformis
Tortricoidea: Tortricidae: Olethreutinae: Grapholitini: Cydia pomonella; Olethreutini: Phaecasiophora niveiguttana
Immoidea: Immidae: Imma tetrascia
Choreutoidea: Choreutidae: Choreutinae: Hemerophila diva
Pterophoroidea: Pterophoridae: Pterophorinae: Emmelina monodactyla
Gelechioidea: Amphisbatidae: Psilocorsis reflexella
Elachistidae: Antaeotricha schlaegeri
Gelechiidae: Dichomeris punctidiscella
OBTECTOMERA (12 superfamilies, including all those below)
Thyridoidea: Thyrididae: Striglininae: Striglina suzukii
Pyraloidea: Crambidae: Crambinae: Catoptria oregonica
Pyralidae: Galleriinae: Galleria melonella
Mimallonoidea: Mimallonidae: Lacosoma chiridota
MACROHETEROCERA (5 superfamilies)
Lasiocampoidea: Lasiocampidae: Macromphaliinae: Tolype notialis
Bombycoidea: Bombycidae: Bombycinae: Bombyx mori
Drepanoidea: Drepanidae: Cyclidiinae: Cyclidia substigmaria; Thyatirinae: Pseudothyatira cymatophoroide
Cimeliidae: Axia margarita (formerly in its own superfamily; Kristensen, 2003)
Doidae: Doa sp. (formerly in Noctuoidea; Kristensen, 2003)
Geometroidea: Epicopeiidae: Epicopeia hainesii (formerly in Drepanoidea; Kristensen, 2003)
Uraniidae: Epipleminae: Calledapteryx dryopterata
Geometridae: Ennominae: Biston betularia; Geometrinae: Chlorosea margaretaria; Sterrhinae: Idaea sp. 5
Noctuoidea: Erebidae: Lymantriinae: Lymantria dispar; Noctuidae: Heliothinae: Helicoverpa zea, Heliothis virescens; Noctuinae: Striacosta albicosta

Table 7.1: Classification of exemplar species included, following van Nieukerken et al. [231]. See Table S1 of Bazinet et al. [86] for accession number, collecting locality and life stage used.

were congeners of taxa in the earlier study, and an additional two belonged to the same subfamily and tribe (see Table S1 of Bazinet et al. [86]). These substitutions were made because no more material of the same species or genus, respectively, was available. All of the specimens we sequenced came from the ATOLep collection built by the Assembling the Lepidoptera Tree of Life project (Leptree), and had been stored in 100% ethanol at -80° C, some for more than 20 years.

Taxon sampling in this exploratory study expanded in phases, from 16 to 38 to 46 exemplars, each with a separate phylogenetic analysis, as we sought to characterize the data and develop our informatic and analytical workflows. The initial test set focused (14/16 taxa) on one especially problematic tree region, the hypothesized group consisting of Cossoidea + Sesioidea + Zygaenoidea [251, 253]. This assemblage, here termed the "CSZ clade", consists of 5,996 species in 19 families according to van Nieukerken et al. [231], who merged Sesioidea into Cossoidea. It is one of very few groupings among apoditrysian superfamilies that is postulated in the morphology-based working hypothesis [250]. It also presents an exceptionally clear superfamily-level contrast in a major life history feature, internal versus external feeding: Cossoidea and Sesioidea are mostly stem borers, whereas Zygaenoidea are mostly external folivores. In analyses with the 19 Leptree genes (14.7 kb), the CSZ clade is only sometimes monophyletic, and always with very weak support [7]. A core subset of Zygaenoidea is reliably monophyletic, but Sesioidea, Cossoidea and Cossidae never are. Relationships of the sesioid families, the cossoid families and subfamilies, and the two aberrant (parasitic) families of Zygaenoidea (Epipyropidae and Cyclotornidae), to each other and to the "core" Zygaenoidea, are almost completely unsupported (e.g., Figure 7.1). The test data set also included one non-apoditrysian outgroup (*Yponomeuta*) and one putative apoditrysian outgroup, *Bombyx mori*.

After testing and improving our protocols using the 16-taxon test set, we added 22 more exemplars representing most of the other major lineages of Apoditrysia, focusing on the other large superfamilies (those with over 2,000 species). Another eight taxa were then added for a final, 46-taxon analysis. These eight had been held back from the second analysis because we considered them especially likely to complicate tree estimation, either because they had much less data than the rest (*Striacosta albicosta*) or because they were previously identified as difficult-to-place or "rogue" taxa [7]. We wanted to see how much the inclusion/exclusion of such taxa would affect the results based on our very large gene samples.

An additional, related benefit to our stepwise increase in taxon sampling is the evidence it provides as to the effects of taxon sampling density, which has been of special concern in phylogenomics [260–262]. Strong conflicts among phylogenies of 16 and 38 and 46 taxa could suggest the presence of false signal due to taxon undersampling, as could strong support in the RNA-Seq phylogenies for nodes contradicting strongly supported nodes in the much larger Leptree taxon sample (Figure 7.1). Successive expansion of the taxon sample could also identify instances in which weak support is increased by denser taxon sampling.

To provide a controlled assessment of the potential benefits of massively increased gene sampling, we compared topologies and branch supports from RNA-Seq analyses both to those from the 19-gene, 483-taxon "backbone" phylogeny [7], and to new 19-gene analyses of 16-, 38- and 45-taxon data sets. The data sets for the 19-gene analyses were taken from the data matrix of Regier et al. [7]. For each species in the RNA-Seq data set, an associated Leptree exemplar from Regier et al. [7], listed in Table S1 of Bazinet et al. [86], was chosen to match it as closely as possible, and was used in our 19-gene analyses. In 38 cases, exactly the same species was used; a closely related substitute was used in six others. For *Striacosta albicosta*, not included in the "backbone" study, we substituted the con-tribal *Agrotis ipsilon*, included by Regier et al. [7], in the 19-gene analysis. We thought it unnecessary to substitute for *Heliothis virescens*, for which we also lack 19-gene data, because it already had a close relative in the 19-gene data set (*Helicoverpa zea*). Thus, the final 19-gene analysis used 45 exemplars instead of 46.

7.4 RNA-Seq data generation

Total RNA was extracted using Promega SV total RNA isolation mini-kits. The great majority of our specimens were adults; four were larvae (see Table S1 of Bazinet et al. [86]), with species identifications verified by comparison of COI sequences with those in the Barcode of Life Data System [263]. For larger moths we used the thorax and/or anterior part of the abdomen; for a few smaller ones we used the entire body. RNA extracts were submitted to the University of Maryland-Institute for Bioscience and Biotechnology Research Sequencing Core. The quality of total RNA was assessed by capillary electrophoresis on an RNA chip using an Agilent Bioanalyzer 2100 system. RNA preps of sufficient quality were subjected to poly-A selection and indexed library construction for sequencing on an Illumina HiSeq 1000. Following Hittinger et al. [11] our libraries were left unnormalized, so as to favor highly-expressed genes likely to be present in most species and life stages. Libraries were run four per lane, yielding about 110 million 100-bp paired-end reads per taxon.

7.5 Sequence quality control and transcript assembly

We used the default Illumina HiSeq 1000 quality filter, which ensures that at least 24 of the first 25 template cycles has a “Chastity” value greater than 0.6. The Chastity value is a ratio between the highest intensity and the sum of the two highest intensities. We discarded reads that did not pass the Chastity quality filter ($\approx$ 5–20% per sample), as well as reads whose Phred quality score [264] was not greater than 20 at greater than 90% of positions ($\approx$ 5–15% per sample). We observed biases in nucleotide composition at the beginning of our Illumina-generated reads [265], but as our canonical workflow does not depend on accurate quantification of transcript abundance, it was not necessary to correct for this bias. The filtered reads input to assembly (mean = 76M reads per sample) had median Phred scores greater than 35 for over 95% of the bases in each read.

De novo transcriptome assembly was performed using both Trinity (versions r2012-03-17 and r2013-02-25 [44]) and Trans-ABySS (versions 1.3.2 and 1.4.4; ABySS versions 1.3.3 and 1.3.5 [45, 266]), and the results compared (Table S2 of Bazinet et al. [86]) for numbers and length of transcripts using standard assembly metrics such as N50 (the length N for which 50% of all bases are contained in contigs of length

$L < N$). A typical Trinity assembly required greater than 100 GB RAM and finished in 24 to 96 hours using 16 processing cores. A typical Trans-ABySS run required less than 4 GB RAM and a single processor, finishing in 1–2 hours. The same was true for each constituent ABySS run, of which there were 23 per sample (k ranged from 52 to 96 in steps of two). In general, Trinity used more RAM and produced fewer transcripts than Trans-ABySS, but it produced longer transcripts (Table S2 of Bazinet et al. [86]). Combining the Trinity and Trans-ABySS assemblies proved early on to yield a slightly more complete data matrix than either alone, which is why we continued to use both. The added cost of doing so was minimal once assembly workflows were established.

Some modification of these methods was necessary for reassembly of the *Striacosta albicosta* transcriptome [179]. We acquired the original 75-bp single-end Illumina reads, which were based on 16 individuals and normalized cDNA, and were not subjected to a "Chastity" filter. Application of our Phred filter eliminated 61% of the reads. We modified Trans-ABySS to work with single-end data, and optimized its k-mer sweep for 75-bp reads (k ranged from 38 to 74 in steps of two). The original assembly contained 16,850 contigs of median length 173 bp; our combined Trinity and Trans-ABySS assembly yielded 336,829 contigs of median length 114 bp, including over 15,000 contigs of median length 351 bp from the Trinity assembly alone.

7.6 Sequence classification

For a variety of reasons, it can be useful to treat the sequences derived from RNA-Seq as a metagenome (Chapter 2). For example, it is known that many species of Microsporidia infect insects such as butterflies and moths [267]. Thus, we experimented with classifying our transcript fragments into various taxonomic groups (e.g., Insecta, Microsporidia, human and bacterial contaminants, etc.) before proceeding with downstream steps. For this purpose, we used MEGAN [148, 180] and MG-RAST [150]. We did not find much in our samples that was not lepidopteran in origin (and hence worthy of isolating and studying separately). Furthermore, sequence classification programs often leave a significant proportion of sequences "unclassified" because none are a high enough quality match to a database sequence. Because we infer these sequences to be mostly lepidopteran, to discard this data would significantly reduce the amount of useful data for subsequent analysis steps. Thus, to simplify matters, we currently omit the sequence classification step from our workflow and instead rely on the orthology determination step to extract lepidopteran sequences from the unfiltered data (explained in Section 7.7). However, we may still use sequence classification programs at a later stage of the project to search our RNA-Seq data with more scrutiny for the presence of organisms such as Microsporidia, in order to recover their transcriptome sequences for independent study (Chapter 9).

7.7 Orthology determination

A variety of methods and implementations of algorithms for orthology determination are available (Chapter 3). In addition, several public databases of orthologous genes have been compiled using these methods [268–271]. The orthology determination methods used by previous phylogenomic studies were surveyed before developing a strategy for our own workflow.

To infer orthology, we used HaMStR (version 9; [272]), which in turn used BLASTP [13], GeneWise [273], and HMMER [274] to search the combined assembly data for protein sequences matching a set of "known" orthologs. The known orthologs in our case consisted of a database of 1,579 profile hidden Markov models (pHMMs; [275]) of orthologous sequence groups called the "Insecta Hmmer3-2 core-ortholog set", obtained from the HaMStR web site. These models are based on six genomes representing three holometabolous insect orders (Hymenoptera: *Apis*; Coleoptera: *Tribolium*; Lepidoptera: *Bombyx*); a non-insect pancrustacean (Vericrustacea: *Daphnia*); a different arthropod subphylum (Chelicerata: *Ixodes*); and a different phylum (Annelida: *Capitella*). An annotated list of the putative orthologs in the Insecta Hmmer3-2 data set can be found at `http://www.deep-phylogeny.org/hamstr/download/datasets/hmmer3/`.

In the first step of the HaMStR procedure, regions of our transcript assemblies (expressed as amino acid sequences) that matched any one of the 1,579 Insecta core-ortholog pHMMs were provisionally assigned to the corresponding orthologous group. To reduce the number of highly-divergent, potentially paralogous sequences returned by this initial search, we changed the E-value cutoff defining a "hit" to 1e-05, from the HaMStR default of 1.0, and retained only the top-scoring quartile of hits. In the next HaMStR step, the provisional "hits" from the Insecta search were compared to a "reference taxon" (*Bombyx mori*), and retained only if they survived a reciprocal best BLAST hit test with that taxon. Once assigned to orthologous groups, protein sequences from our assemblies were aligned using MAFFT [276]. The resulting protein alignments were then converted to the correct corresponding nucleotide alignments using a custom Perl script that substituted for each amino acid the proper codon from the original coding sequence.

Following initial orthology assignments, we computed "coverage per base" for each orthologous group, defined as read length times the median number of reads mapped to orthologous group sequences divided by the median length of orthologous group sequences. Read mappings used Bowtie (version 0.12.8) [277], and allowed for up to four mismatches.

7.8 Data matrix construction and paralogy filtering

Our orthology determination pipeline often yields multiple sequences for a particular taxon-locus combination, which can reflect the presence of multiple orthologs, heterozygosity, alternatively-spliced transcripts, paralogy (including inparalogs; [182]), and sequencing errors, among other possibilities. One general approach for reducing this variation to a single sequence, as required for phylogenetic analysis, is exemplified by the "REPRESENTATIVE" option in HaMStR [272]. This procedure chooses the single sequence (or concatenation of non-overlapping fragments) with the best pairwise alignment to a chosen reference taxon. We developed an alternative that accommodates the uncertainty in orthology determination by combining the set of sequences into a single consensus sequence, using nucleotide ambiguity codes [191] as necessary (Chapter 3). Consensus sequences were generated by providing the alignment of the nucleotide coding sequences corresponding to the amino acid sequences passing our filtering steps, described above, to the `consensus_iupac` BioPerl subroutine [278]. There are two principal motivations for this "CONSENSUS" approach. The first is a desire to incorporate all information about specific nucleotide states for positions that might reasonably be inferred to be orthologous, including those where orthologous relationships among genes between pairs of taxa are many-to-one, and many-to-many, as well as cases of polymorphism. A second motivation is to mitigate the effects of mistaken orthology determination and other errors, including those resulting from incorrect choice of a single representative sequence, by in effect reducing the weight of positions at which transcription fragments differ. By including more available transcription fragments, moreover, CONSENSUS can potentially yield longer total sequences than REPRESENTATIVE, as has been our experience. However, degenerating nucleotide sites that vary among transcripts could result in dilution of phylogenetic information, if the single best sequence chosen by REPRESENTATIVE were almost always the most phylogenetically-appropriate one. The approach that works best is thus an empirical question, which we addressed by performing both procedures and comparing the results (Appendix A).

Despite the filters described above, inspection of our initial 1,579 alignments revealed obvious paralogs. An extreme example was orthologous group 412460 of the Insecta Hmmer3-2 database, annotated there as acetyl-CoA acetyltransferase, a type of thiolase. In our data, HaMStR search returned two divergent sets of sequences for this ortholog group, which upon BLAST search matched two different members of the thiolase gene family in a noctuid moth. No single E-value threshold can eliminate problems of this kind, so we turned to direct scrutiny of gene trees (e.g. [27, 28, 279]). Using the initial 16 test taxa, a maximum likelihood (ML) gene tree was constructed for each orthologous group using all matching sequences, and provided as input to the program PhyloTreePruner [280]. If the sequences for a particular taxon form a polyphyletic group, the program prunes the gene tree to the maximal subtree in which the non-polyphyly criterion is met for all taxa. For the 16 test taxa, PhyloTreePruner pruned 838 of the 1,579 gene trees to some degree. For this exploratory study we took a very conservative action based on these results, using for all subsequent phylogenetic analyses only the 741 genes in which no evidence of paralogy was found in the test taxa, and completely omitting the remaining genes; alternative possibilities for future studies are considered in Section 7.12. Following application of the paralogy filter, the 741 putative ortholog alignments were concatenated, adding gaps for missing data as necessary using a custom Perl script. For all phylogenetic analyses the nucleotide matrix was subjected to `degen1` coding (version 1.4; [281]), and sites not represented by sequence data in at least four taxa were subsequently removed. "Degen" uses degeneration coding to eliminate all synonymous differences among species from the data set, resulting in phylogeny inference based only on non-synonymous nucleotide change. This procedure was shown in our previous backbone study [7] to generally improve recovery of deep nodes. At deeper levels in the Lepidoptera, inclusion of synonymous change in any form, even as part of a codon model, sometimes introduces conflict and systematic error due to compositional heterogeneity [7, 114]. Analysis under `degen1` can be viewed as a computationally efficient approximation to a purely "mechanistic" amino acid model — i.e., one based on the genetic code but not incorporating empirical transition frequencies between amino acids [114, 282, 283].

Sequences and alignments for the 19-gene analyses were extracted from Table S4 of the Leptree backbone study [7]. Nine of these genes are present in the Insecta Hmmer3-2 database. The PCR amplicon codes of these nine from Regier et al. [7] are: 40fin, 109fin, 192fin, 262fin, 265fin, 268fin, 3007fin, 3070fin, and CAD. Five of these genes were eliminated by our paralogy screen, while the following four, listed by their numbers in the Insecta Hmmer3-2 database, were included among the 741 used in phylogenetic analyses: 413101 (262fin); 412564 (268fin); 412293 (265fin); and 412031 (40fin).

7.9 Phylogenetic analysis

Maximum likelihood phylogenetic analyses used GARLI (Genetic Algorithm for Rapid Likelihood Inference; version 2.0 [2]) and grid computing [48, 223] via a web service at `molecularevolution.org` [53] based on tools developed by Bazinet et al. [284] that include post-processing with DendroPy [208], R [285], and custom Perl scripts. The majority of the phylogenetic analyses were completed using the

BOINC volunteer computing platform [50] (`http://boinc.umiacs.umd.edu`). We used a GTR+I+G nucleotide model together with GARLI default settings, including stepwise addition starting trees, except that we lowered the number of successive generations yielding no improvement in likelihood score that prompts termination (`genthreshfortopoterm=5000`), as we found that this saved time and yielded comparable results. Memory requirements ranged from 800 MB for the 16-taxon, 741-gene analysis to 3500 MB for the 46-taxon, 741-gene analysis; some exploratory analyses with larger data matrices used as much as 9 GB of RAM. Each search replicate might have run, on average, anywhere from one to a few days. Each best tree was selected from 100 GARLI search replicates, while bootstrap analyses consisted of 1,000 replicates. Insufficient search effort during bootstrapping has been shown to artificially depress bootstrap support (BP) values [7]. A rough guide to the effort needed was provided by our initial 100 replicate ML search: if the best tree topology was found only rarely, multiple search replicates per bootstrap replicate may be helpful. We tested each of our data sets for the effect of increased search effort on BP values, at levels of one, five, and ten search replicates per bootstrap replicate. We found a significant increase in BP values for several analyses using five search replicates instead of one, but did not find a significant improvement using ten search replicates instead of five. Thus, all results presented here used five search replicates per bootstrap replicate.

The 741-gene and 19-gene data matrices have been deposited in Dryad (doi:10.5061/dryad.02qv3). The Illumina reads have been deposited in the NCBI Sequence Read Archive, as BioProject PRJNA222254.

7.10 Data matrix properties

The paralogy-filtered matrix of 741 genes contained from 742,017 to 873,036 nucleotide positions and was 80–93%-complete, depending on the number of taxa included and the orthology determination procedure used (Table 7.3). Thus, overall matrix completeness was slightly higher than in the 14.7 kb, 483-taxon Leptree analysis [7]. Completeness was fairly consistent among the 44 newly-sequenced taxa, ranging from, e.g., 67% to 84% for the 46-taxon, 741-gene CONSENSUS matrix (Table S2 of Bazinet et al. [86]). Our reassembly of the previously-published *Striacosta albicosta* sequence reads [179] yielded sequence for 1,138 orthologous groups, whose median sequence length was 147 bp. Thus, in the paralogy-filtered 46-taxon data matrices, for example, *S. albicosta* had approximately half the data of our other taxa (Table S2 of Bazinet et al. [86]). Coverage per base (Table 7.4) averaged 103× for 15 test taxa, with a range of 31× to 334×.

7.11 Phylogenetic results

The tree of maximum likelihood found for both the 46-taxon, 741-gene CONSENSUS data set and its REPRESENTATIVE counterpart is shown in Figure 7.2, together with bootstrap values for the CONSENSUS and REPRESENTATIVE 46-taxon, 741-gene data sets and the 45-taxon, 19-gene data set. A phylogram version of the same tree is given in Figure 7.3. ML cladograms and bootstrap values for all other data sets are given in Figures 7.4-7.7.

Contrast[a]	Node	Bootstrap support value		
		16 taxa	38 taxa	46 taxa
1	Noctuoidea + Drepanidae	NA[b]	54	[-][c]
1a	Noctuoidea + Geometroidea + Bombycoidea + Lasiocampoidea	NA	[-]	100
1b	Drepanidae + Doidae + Cimeliidae	NA	NA	100
2	Cossoidea + Sesioidea + core Zygaenoidea (CSZ clade)	83	[-]	[-]
2b	Cossoidea + core Zygaenoidea + Obtectomera	[-]	43	57
3	CSZ clade + Obtectomera	NA	90	21

[a] 1a and 1b, and 2b, are alternative groupings that conflict with nodes 1 and 2, respectively.

[b] NA = not applicable; node not present because the constituent exemplars are not included in that data set.

[c] [-] = node not present in either ML tree or bootstrap majority rule consensus tree for that data set.

Table 7.2: Notable changes in topology and bootstrap support with change in taxon sample size for 741-gene, consensus analyses.

	741 genes					
	46 taxa		38 taxa		16 taxa	
	consensus	representative	consensus	representative	consensus	representative
number of nucleotide positions	873,036	765,078	764,025	762,252	742,668	742,017
number of non-gap chars in alignment	32,032,914	31,732,542	26,682,024	26,572,950	11,085,843	11,047,875
matrix completeness (nt present ÷ *possible* nt)	80%	90%	92%	92%	93%	93%
percent ambiguous nt (non-gap, non-A/C/G/T chars)	37%	37%	34%	34%	37%	37%

Table 7.3: Size and completeness of aligned data matrices from RNA-Seq.

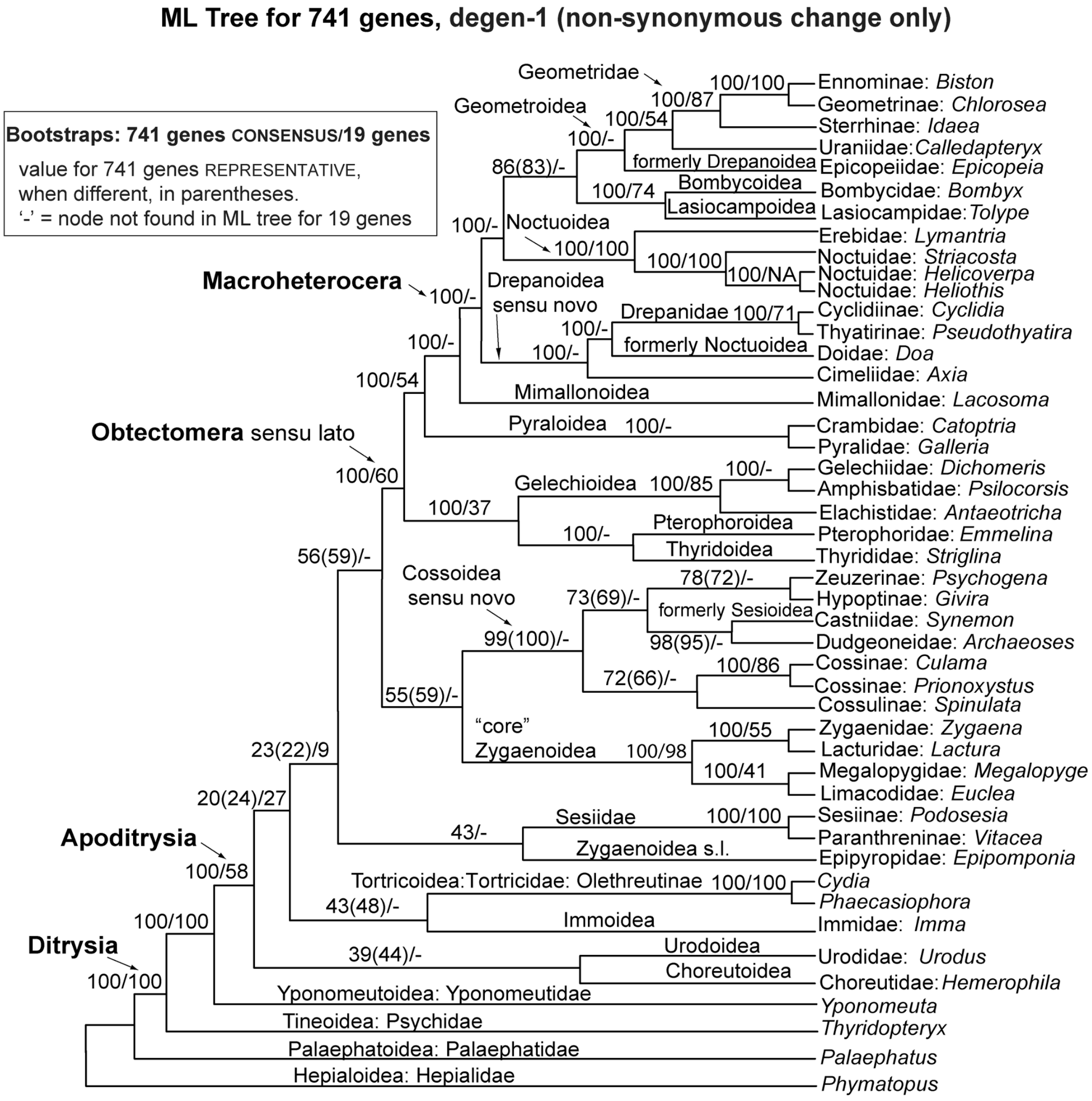


Figure 7.2: ML tree for 46 taxa, 741 paralogy-filtered genes, `degen1` (non-synonymous change only). Bootstrap percentages: 741 genes CONSENSUS method, followed by 741 genes REPRESENTATIVE method in parentheses but only when these two differ, followed by 19 genes, each based on 1,000 bootstrap replicates with 5 search replicates each. The "-" means the node was not found in the ML tree for 19 genes.

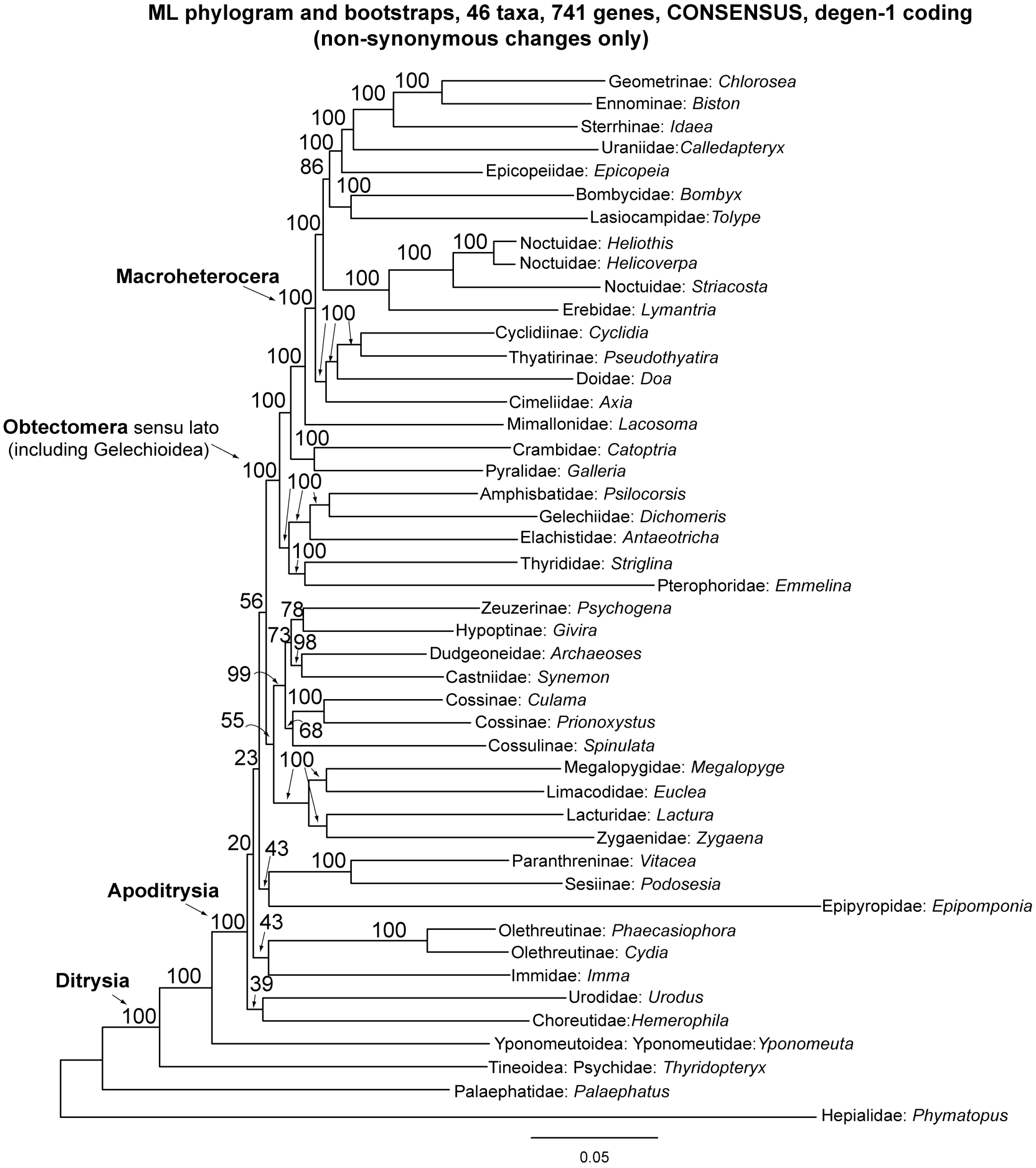


Figure 7.3: ML phylogram and bootstraps for the 46-taxon, 741-gene, CONSENSUS analysis. The topology and CONSENSUS bootstraps are identical to those in Figure 7.2.

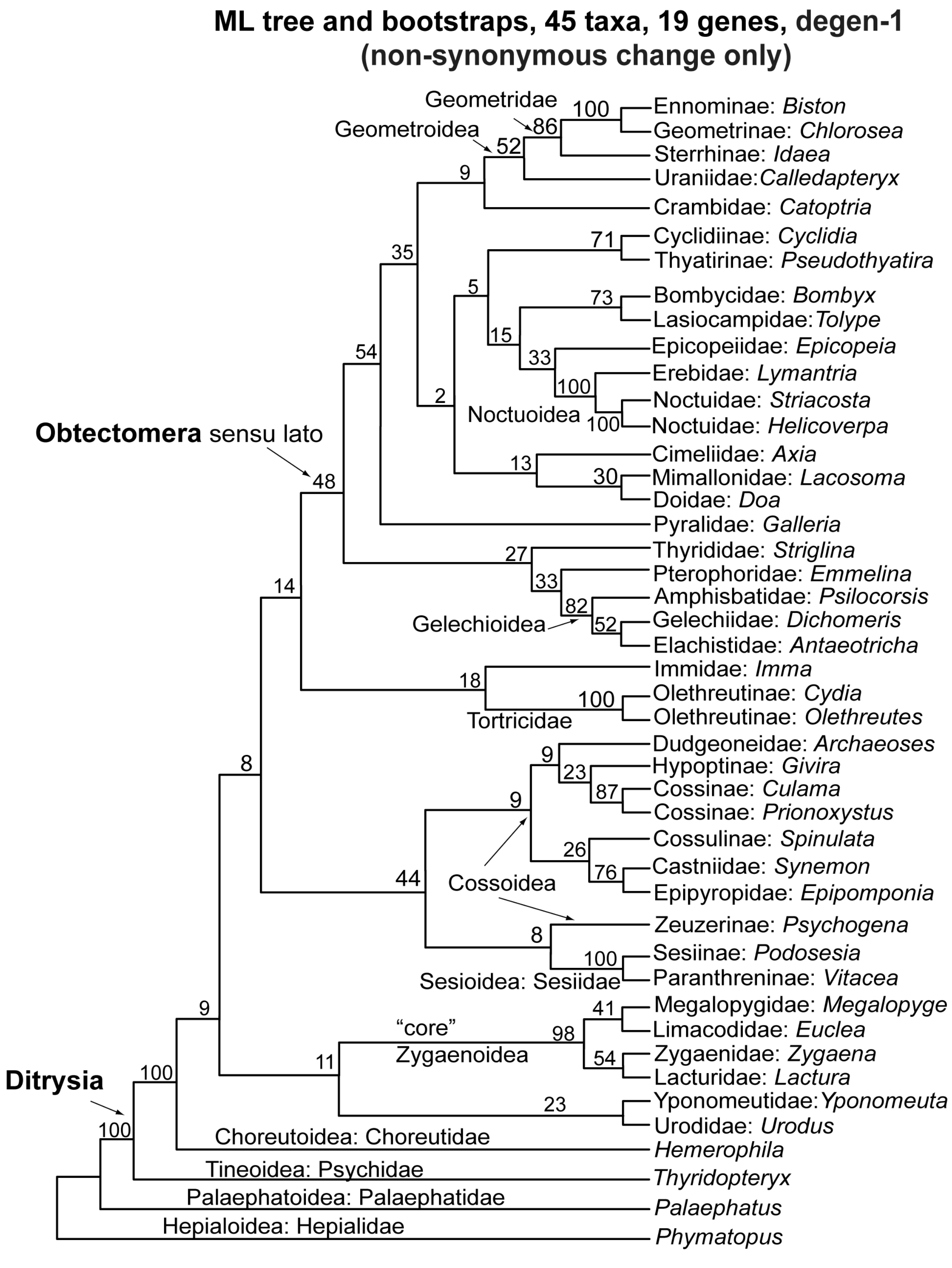


Figure 7.4: ML cladogram and bootstraps for the 45-taxon, 19-gene analysis.

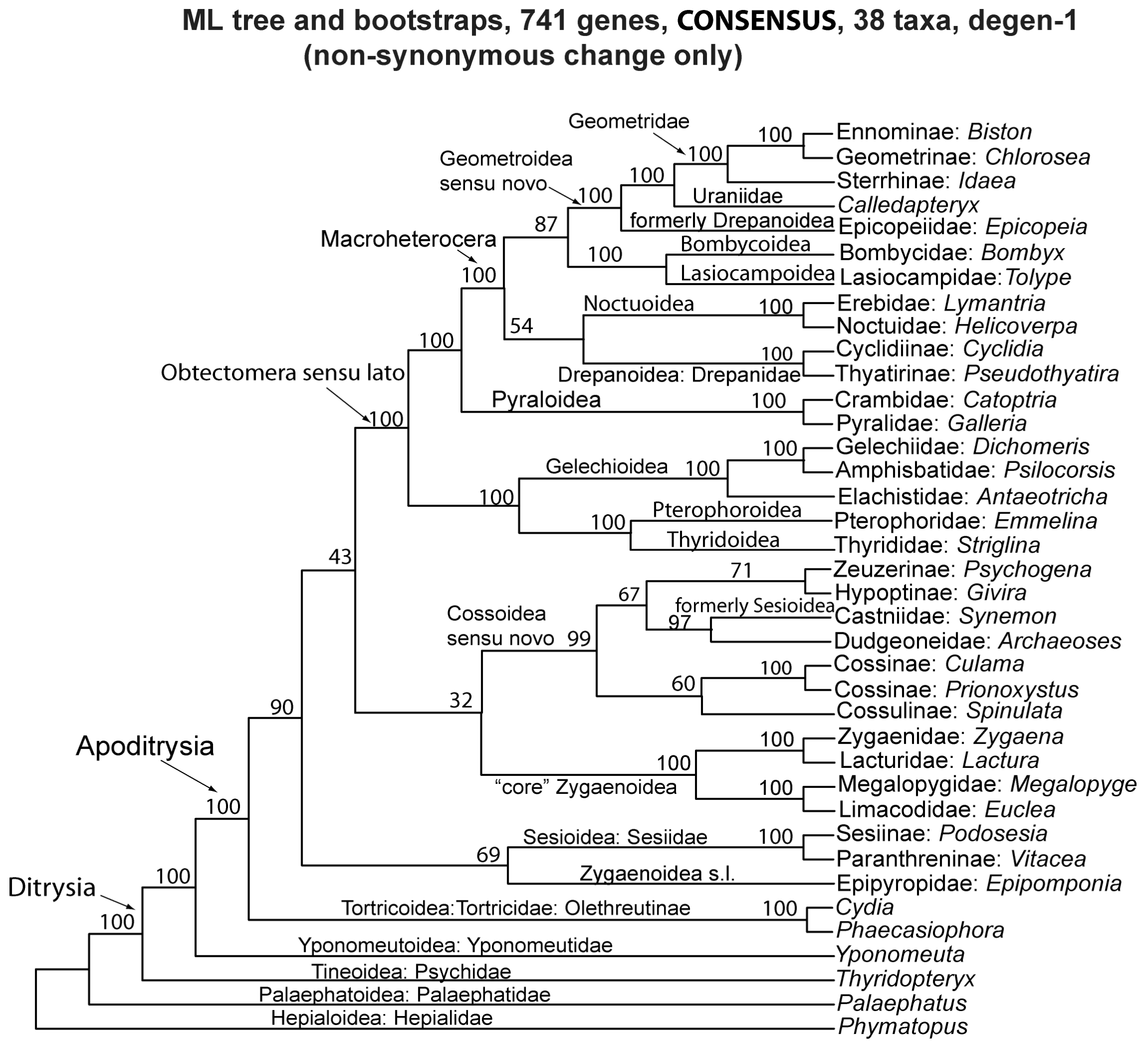


Figure 7.5: ML cladogram and bootstraps for the 38-taxon, 741-gene, CONSENSUS analysis.

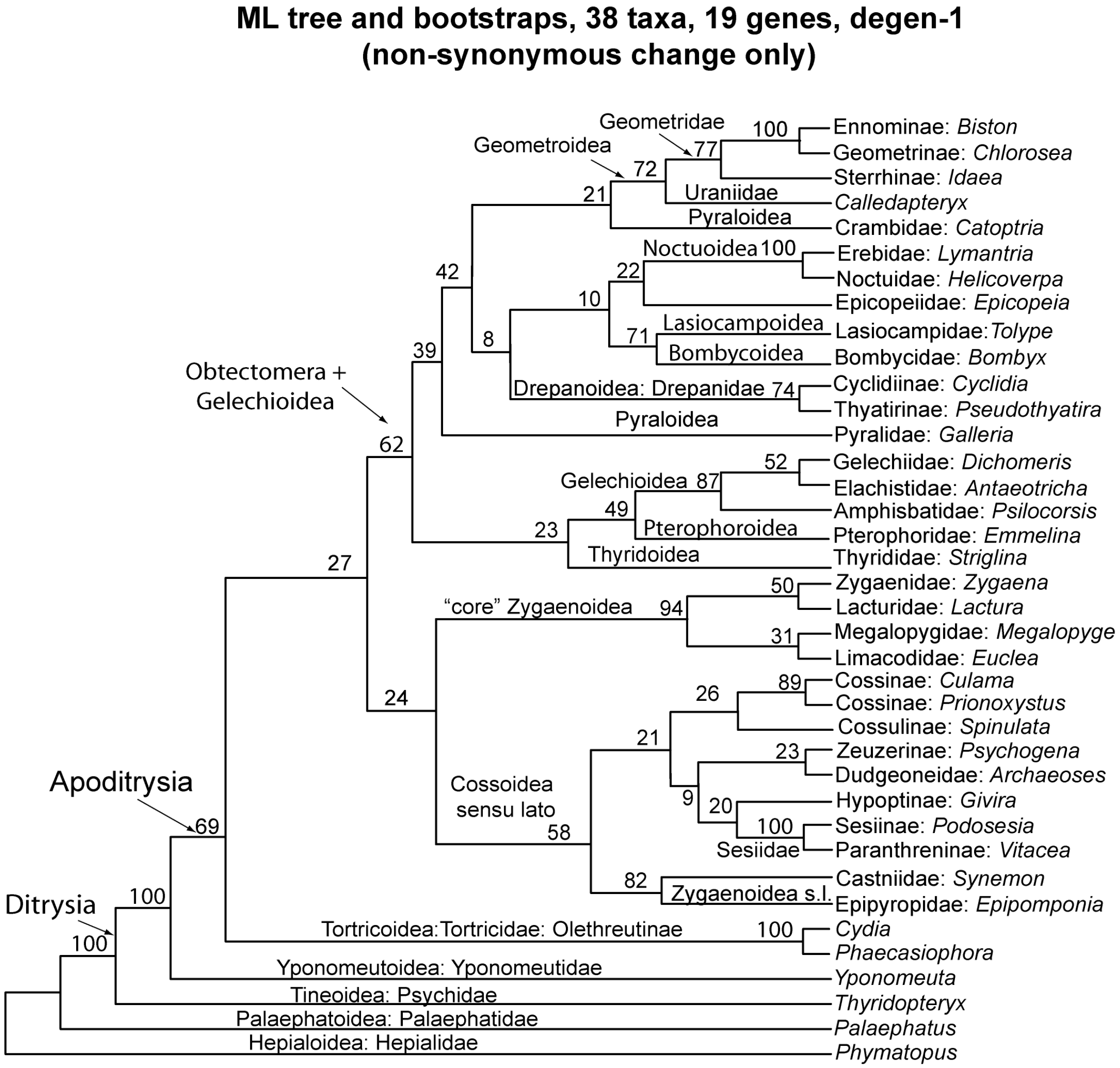


Figure 7.6: ML cladogram and bootstraps for the 38-taxon, 19-gene analysis.

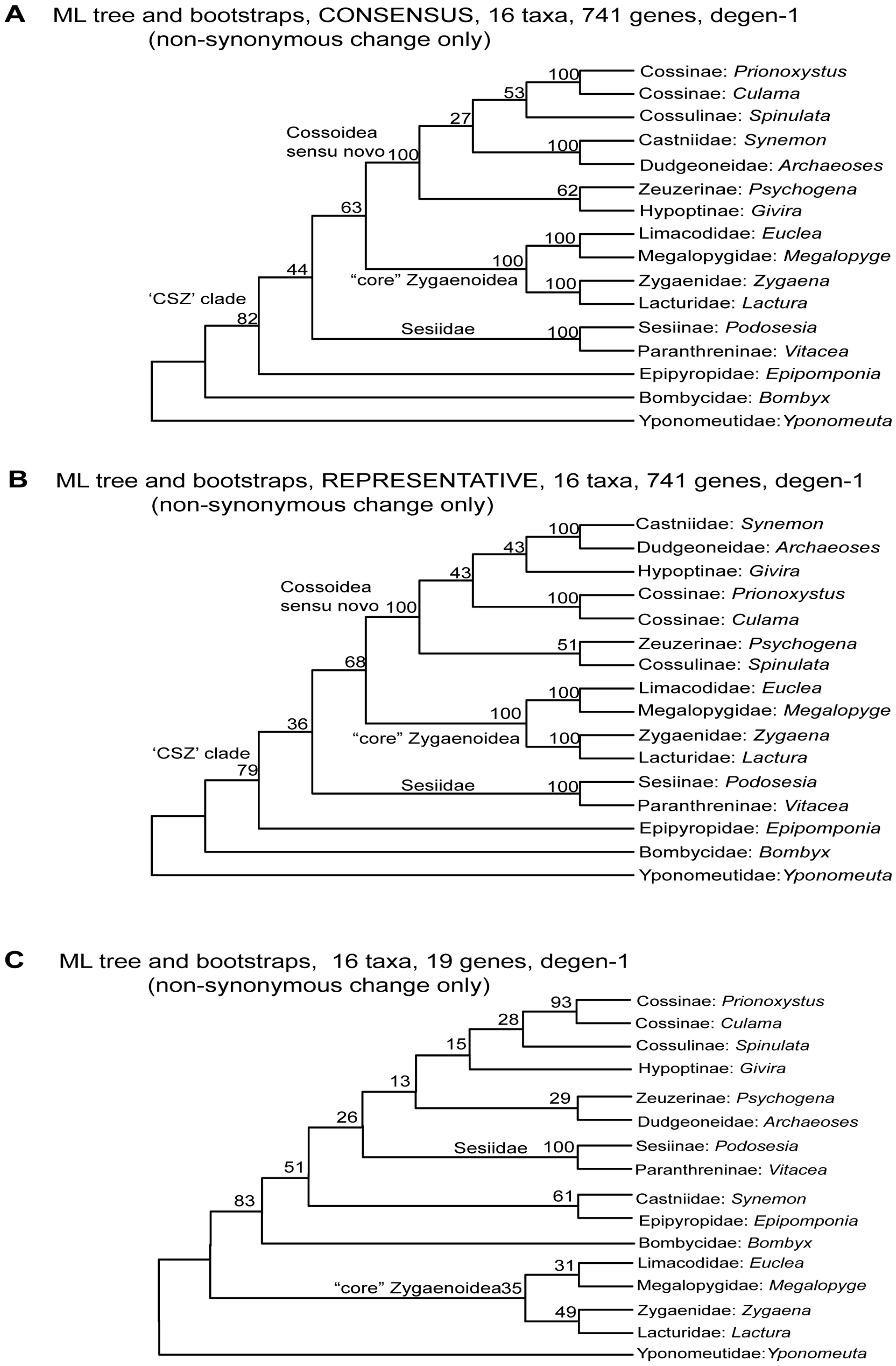


Figure 7.7: ML cladogram and bootstraps for the 16-taxon analyses. (A) the 16-taxon, 741-gene CONSENSUS analysis, (B) the 16-taxon, 741-gene REPRESENTATIVE analysis, and (C) the 16-taxon, 19-gene analysis.

The two alternative procedures for determining a single sequence per taxon-locus combination for phylogenetic inference when orthology search returns multiple "hits" — i.e., REPRESENTATIVE and CONSENSUS — yielded identical ML topologies, and nearly identical bootstrap values (Figure 7.2). A marked difference between the two procedures was observed in the 38-taxon analysis, for which finding the best tree topology took considerably more search effort for REPRESENTATIVE than for CONSENSUS: out of 100 ML searches, the best tree topology was found 25 times for the CONSENSUS matrix, but only once for the REPRESENTATIVE matrix. However, we found no such difference for either the 16- or 46-taxon analyses; in those cases, a comparable amount of search effort for each procedure was required to find the best tree topology. An experiment described in Appendix A suggested that the greater search effort required for REPRESENTATIVE in the 38-taxon case stemmed from conflicting signal in a small proportion of nucleotide positions in that matrix that were left ambiguous in the CONSENSUS matrix.

The most dramatic pattern in the results was the much greater frequency, across all taxon sets, of strong support for nodes subtending multiple superfamilies in the 741-gene analyses than in either the corresponding 19-gene analyses or the 483-taxon "backbone" study. For example, in the 46-taxon, 741-gene ML topology of Figure 7.2, there are 22 nodes within Apoditrysia that subtend taxa assigned to different superfamilies in either the newest classification [231] or its immediate predecessor [245]. Of these, 11 have bootstrap support (BP) of 100%, two additional nodes have BP ≥98%, and one additional node has BP >80%, for a total of 14/22 nodes with "strong" or "very strong" support (Figure 7.2). In contrast, of 23 nodes subtending multiple superfamilies in the ML topology for the 45-taxon, 19-gene matrix (Figure 7.3), none have BP ≥80%; only one has BP >70%, and only three have BP >50% (Figures 7.2, 7.3).

Strong deeper-node support in the 741-gene analyses is not spread evenly across the Apoditrysia, but is restricted almost entirely to a clade consisting of Obtectomera *sensu* van Nieukerken et al. [231] + Gelechioidea + Pterophoroidea (Figure 7.2). Of the 12 nodes within and including this clade that subtend multiple subfamilies in recent classifications, 11 have BP=100% and all have BP >80%. In contrast, of the 11 such nodes elsewhere among the Apoditrysia, none have BP=100% and only two have BP >80%.

Tree topology changed little as taxon sampling expanded for 741 genes. Table 7.2 summarizes the main differences in topology and bootstrap support among the 16-, 38- and 46-taxon analyses. In no comparison among trees for different numbers of taxa were there incompatible nodes that each had strong bootstrap support. Thus, there is little evidence for artifactual strong support resulting from taxon undersampling. The most notable conflict concerns monophyly of the putative CSZ clade. In the 16-taxon analysis, which includes only one apoditrysian (Bombyx) apart from the putative CSZ clade, that clade gets 82% bootstrap support (Figure 7.7). In contrast, the 38- and 46-taxon analyses, which include many other apoditrysian lineages, find the CSZ assemblage to be paraphyletic with respect to the clade Obtectomera + Gelechioidea + Pterophoroidea. Bootstrap support for this conclusion, however, is only 43% and 59% for 38 and 46 taxa, respectively (Figures 7.2, 7.5). The most striking instance of decline in bootstrap support without change in topology involves the grouping of the "CSZ clade" constituents with the Obtectomera, to the exclusion of other apoditrysians. The 90% bootstrap support for this grouping in the 38-taxon analysis falls to 27% in the 46-taxon analysis, which includes three additional non-obtectomeran superfamilies.

The evidence is stronger for a positive effect of taxon sampling density on node support. The clearest examples are the contrasting positions of Noctuoidea and Drepanidae in the 38- versus 46-taxon, 741-gene analyses. In the 38-taxon analysis (Figure 7.5), which is missing several small groups (Cimeliidae, Axiidae, Doidae) that may or may not represent distinct superfamilies of Macroheterocera [231, 245], Noctuoidea are grouped with Drepanidae, but with weak support (BP=54%). When the three missing groups are added, as part of the 46-taxon analysis, Drepanidae and Noctuoidea are no longer paired, but the new positions of these two taxa, together with those of the newly-added families, are all supported by BP=100%. A beneficial effect of denser taxon sampling on node support is also suggested by the generally lower support in our new 19-gene analyses of 16, 38 and 46 taxa than in our previous 19-gene, 483-taxon study [7]. For example, bootstrap support for Apoditrysia, 98% in Regier et al. [7], is only 58% here in the 19-gene, 45-taxon analysis. Moreover, unlike the 483-taxon study, the 19-gene, 45-taxon analysis also fails to support monophyly for Pyraloidea and for Macroheterocera. An interaction between gene and taxon sampling is suggested, finally, by the fact that the 45-taxon, 741-gene analysis supports the monophyly of both Pyraloidea and Macroheterocera with BP=100%.

7.12 Discussion

Our results suggest that the expansive gene sampling yielded by RNA-Seq may be able to strongly resolve inter-superfamily relationships throughout a clade consisting of Obtectomera *sensu* van Nieukerken et al. [231] plus Gelechioidea and Pterophoroidea (at least), comprising over two-thirds of the species of Lepidoptera. But, might these high bootstraps be misleading? Multiple authors have urged caution in the interpretation of bootstrap support in phylogenomic studies (e.g., [100,261]) or even abandonment of bootstraps altogether in favor of other support measures [286]. If random error is sufficiently reduced by massive gene sampling, strong but misleading bootstrap support might arise from even subtle forms of pervasive systematic error, such as minor compositional heterogeneity or slight differences in the relative abundance of strongly-conflicting gene tree topologies, as well as from long-branch attraction due to the typically sparse taxon sampling in phylogenomics.

How could we judge whether the strong support seen in our results is artifactual? That explanation would gain credence if the strongly-supported nodes repeatedly conflicted with groupings that were robustly supported, or at least consistently monophyletic, in previous studies. In fact, however, the topology of the RNA-Seq phylogeny of Figure 7.2 is closely similar, though not identical, to that of the 483-taxon, 19-gene study (Figure 7.1) and to those of earlier molecular studies [5, 114, 244]. It is also consistent, in topology and node support levels, with recent studies using whole mitochondrial genomes [246,256]. All strongly-supported relevant nodes from previous nuclear gene studies are also strongly supported by the RNA-Seq analysis. Nowhere in the tree does a strongly-supported node in the phylogenomic study contradict a strongly-supported node in any earlier study. Moreover, it appears that limited taxon sampling, rather than inducing artifacts, can be better overcome by the RNA-Seq data than by the 19-gene data: in the 38- and 46-taxon analyses, the RNA-Seq data strongly support the monophyly of Pyraloidea, for which previous molecular and morphological evidence is definitive, whereas the 19-gene data fail to group the two pyraloid exemplars.

A second reasonable expectation, if strong support in the phylogenomic results were largely artifactual, is that such support should be distributed across all levels in the tree. Indeed, some of the forces that can produce strong false signal, such as convergence in amino acid composition and long branch attraction, should be more likely for deeper than for shallower divergences. But in fact, within Apoditrysia, strong support from RNA-Seq is concentrated in the subordinate clade Obtectomera, while the deeper divergences have uniformly weak support.

These observations — agreement of strong support with previous groupings, and decreasing signal strength with increasing depth of divergence within Apoditrysia — suggest that such strong support as we find in the RNA-Seq results is real rather than artifactual. They further suggest that even with 741 genes, we are still data-limited: we do not yet have enough characters to fully resolve all stages of the rapid radiation of the Apoditrysia. On the plus side, however, it also appears that, unlike many previous phylogenomic studies, we are not working with levels of divergence at which strong bootstrap support, even from entirely non-synonymous change, is both inevitable and often misleading [100, 261, 286].

If, as we argue, the strong support seen in our 741-gene analyses is real, it appears that further taxon sampling could quickly produce major advances in our understanding of the huge clade Obtectomera. Precise definition of this clade has been difficult, and the placement of multiple superfamilies has been unclear. Our results suggest that there is a sharp discontinuity between superfamilies that are and are not strongly supported as near relatives of the Macroheteroceran moths. If this distinction holds up under further taxon sampling, it would be reasonable to use it to define the Obtectomera, which would then include both Gelechioidea and Pterophoroidea. It appears that RNA-Seq may be able to definitively resolve all or nearly all relationships within Obtectomera so redefined. There is very strong support for monophyly of Macroheterocera *sensu* van Nieukerken et al., and for Mimallonidae as the sister group to these. It might make sense to include Mimallonidae in Macroheterocera. There is also very strong support for a sister group relationship of Mimallonidae + Macroheterocera to Pyraloidea.

All of the superfamilies of Macroheterocera are sampled here, and relationships among them, with one possible exception, are all strongly supported. The basal divergence is between a clade consisting of Cimeliidae + (Doidae + Drepanidae) and one containing the remaining four superfamilies; an identical or similar division, albeit weakly supported, is seen in previous molecular studies. The first grouping corroborates the recent incorporation of all three families into Drepanoidea *sensu novo* [231], and increases the evidence for removal of Doa from Noctuoidea, despite its possession of the two main noctuoid morphological synapomorphies. Within the clade consisting of Noctuoidea, Geometroidea, Bombycoidea and Lasiocampoidea, the latter two are strongly grouped, and only the node uniting these with Geometroidea (BP=84%) has bootstrap support of less than 100%. The position of Epicopeiidae, weakly supported in all previous studies, strongly corroborates their transfer from Drepanoidea to Geometroidea [7, 231]. The close relationship between Geometroidea and Bombycoidea + Lasiocampidae suggested here may explain why Epicopeiidae sometimes grouped (weakly) with the latter in earlier studies [244].

Although Papilionoidea, formerly grouped with the "big moths", were not included in this study, one can confidently predict, from earlier studies (Figure 7.1), that they would fall among the "lower" Obtectomera. In Figure 7.2, this would mean somewhere between the base of Obtectomera and the base of Pyraloidea + Macroheterocera. This prediction has recently been strongly confirmed by studies based on mitochondrial genomes [246, 256] and on RNA-Seq (A. Y. Kawahara, in litt.), although the exact sister group of the butterflies will not be known until sampling of the non-macroheteroceran superfamilies of Obtectomera is complete.

While prospects for resolving the Obtectomera *sensu lato* look promising, the outlook is less bright in the "lower", i.e. non-obtectomeran, Apoditrysia. In this tree region only two nodes subtending multiple current or former superfamilies get bootstrap support approaching conclusive levels (Figure 7.2). There is 99% bootstrap support for a clade consisting of Cossoidea *sensu stricto* [245] plus Castniidae, formerly placed in Sesioidea [179, 245]. If this grouping holds up under further RNA-Seq sampling, it may be useful to redefine Cossoidea to conform to it. Such a definition would re-exclude Sesiidae, included here by van Nieukerken et al. [231], for which no strong placement has been discovered. Within the putative Cossoidea *sensu novo*, only a single inter-family relationship gets notable bootstrap support, namely, the novel pairing of Castniidae with Dudgeonidae (BP=98%). The relationships of the four cossid subfamilies sampled, to each other and to Castniidae + Dudgeonidae, have weaker support (BP=71–81%). Elsewhere in the non-obtectomeran Apoditrysia, no bootstrap value exceeds 59%. Phylogenetic relationships in the Cossoidea-Zygaenoidea-Sesioidea complex will clearly need much further work.

Why are the "lower" Apoditrysia such a difficult phylogenetic problem, in comparison to lepidopteran lineages of both greater and lesser age? Several complementary explanations seem plausible. Cladogenesis might have been particularly rapid at the base of Apoditrysia as compared to later on, resulting in especially short internal branches. Alternatively, the rate of subsequent extinction might have been high, reducing the taxon sample available for reconstructing rapid cladogenesis. Or, these divergences might be harder to reconstruct simply because they are older than those in Obtectomera, leaving more time for synapomorphies to be overwritten by subsequent substitution. Increasing the gene sample might allow us to overcome the first and third effects. To overcome the second effect, we would want to sample taxa as densely as possible, but would face limits set by extinction. Fortunately, as our results so far have shown, gene and taxon sampling are to some degree interchangeable; therefore, more gene sampling might help in this case as well. Thus, further expanding the gene sample may be critical to further resolution of the lower Apoditrysia, no matter why these lineages are so refractory to phylogenetics.

One immediate way to increase our gene sample would be to relax our severe initial interpretation of the PhyloTreePruner results, under which only genes for which no evidence of paralogy was found were considered suitable for phylogenetic analysis. Following Kocot et al. [280], one could recover some of the information thereby lost by estimating bootstrap support for the individual gene trees and avoiding pruning when support is weak. One could also include the partially incomplete pruned gene trees, from which the apparent paralogs have been deleted, in phylogeny calculations. While these measures might be useful, a potentially more profitable approach in the long run would be to address the underlying problem that led us to PhyloTreePruner in the first place. The Insecta Hmmer3-2 database was a highly-useful starting point, but for two reasons it is not ideal for studies within Lepidoptera. First, it contains only the 1,579 genes that were identifiably orthologous across six very divergent arthropod and annelid genomes. Comparisons restricted to Lepidoptera would undoubtedly yield a much higher number of useful genes; for example, the complete proteome of the diamondback moth (Yponomeutoidea: Plutellidae: *Plutella xylostella*) is close to 15,000 genes [187]. Second, presumably because most of the taxa on which the database is built are so divergent from Lepidoptera, many of its putative ortholog groups appear to include sequences that are non-orthologous in Lepidoptera. Therefore, it would be useful to have a new database of Lepidoptera-specific gene models for orthology determination in the Apoditrysia. Such an effort could capitalize on a growing set of annotated lepidopteran genomes and transcriptomes, which now includes multiple apoditrysians as well as a member of the sister group to Apoditrysia [187, 287–289].

7.13 Summary and conclusions

This study explored the potential of next-generation sequencing to conclusively resolve relationships among the superfamilies of advanced ditrysian Lepidoptera (Apoditrysia), which were very weakly supported in previous nuclear gene studies. We used RNA-Seq to generate 1,579 putatively orthologous gene sequences across a taxonomically broad sample of 40 apoditrysians plus four outgroups, to which we added two taxa using previously published data. Phylogenetic analysis of a 46-taxon, 741-gene matrix, resulting from a strict filter that eliminated ortholog groups containing any apparent paralogs, yielded dramatic overall increase in bootstrap support for deeper nodes within Apoditrysia as compared to results from previous and concurrent 19-gene analyses. High support was restricted mainly to the huge apoditrysian subclade Obtectomera broadly defined, in which 11 of 12 nodes subtending multiple superfamilies had bootstrap support of 100%. The strongly-supported nodes showed little conflict with groupings from previous studies, and were little affected by changes in taxon sampling, suggesting that they reflect true signal rather than artifacts of massive gene sampling. Additional taxon sampling has the potential to definitively resolve obtectomeran superfamily relationships. In contrast, strong support was seen at only 2 of 11 deeper nodes among the “lower”, non-obtectomeran apoditrysians. These represent a much harder phylogenetic problem, for which further increase in gene and taxon sampling, together with improved orthology assignments, offers one potential path to resolution. The following chapter continues our lepidopteran systematics work along these lines.

RNA-Seq taxon code	Read length (nt)	Median no. reads mapped to OG seqs	Median length of OG seqs (nt)	Coverage per base
Arc	101	949	1,080	88.7
Cul	101	854	1,080	79.8
Cul2	101	898	1,071	84.7
Giv	101	651	942	69.8
Lag	101	641	1,026	63.1
Mar	101	497	894	56.1
Podo	101	1,532	1,110	139.4
Ppr	101	696	1,008	69.7
Prob	101	1,798	1,113	163.2
Sub	101	3,650	1,104	333.9
Vit	101	516	923	57
YP	101	766	1,047	73.9
Epi	101	653	888	74.3
Euc	101	282	915	31.1
Zyg	101	1,804	1,113	163.7
mean	101	1,079	1,021	103.2

Table 7.4: RNA-Seq “coverage” for 15 test taxa.

Chapter 8: Can RNA-Seq resolve the rapid radiation of advanced moths and butterflies (Hexapoda: Lepidoptera: Apoditrysia)? A follow-up study

This chapter is based on the following publication: Adam L. Bazinet, Michael P. Cummings, Kim T. Mitter, and Charles W. Mitter. Can RNA-Seq resolve the rapid radiation of advanced moths and butterflies (Hexapoda: Lepidoptera: Apoditrysia)? A follow-up study. In preparation.

8.1 Background and motivation

We could have simply continued to use the Insecta core-ortholog set from our previous study (Chapter 7), but based on current estimates of the number of lepidopteran genes (Figure 3.2; [187]), we thought we could recover many more orthologous genes from our transcript data — especially considering that our data matrices, although already very large by current standards, only contained less than 1% of our assembled transcriptome data. Thus, we decided to build a database using the complement of protein-coding genes from several well-annotated lepidopteran genomes that were previously sequenced (*Bombyx mori* [290], *Danaus plexippus* [289], *Heli-*

conius melpomene [287], *Plutella xylostella* [187], and *Manduca sexta* [291]). In this chapter, we discuss the creation of this database and the results of the lepidopteran phylogenomic analyses we conducted that used this newly-created data resource.

8.2 Taxon sampling and taxon set design

The goal of this study was to assess the degree to which increased taxon sampling along with use of a Lepidoptera-specific database could increase the support for relationships among the superfamilies of Apoditrysia over that found in our previous 741-gene study (Chapter 7). Our 81 lepidopteran exemplars and one trichopteran exemplar included five species with previously sequenced genomes, 10 species with publicly-available transcriptomes that we reassembled and incorporated into our analyses, and 67 species for which we sequenced transcriptomes *de novo*.

As the outgroup we used *Philopotamus ludificatus*, a trichopteran. We used previously published genomes for five taxa: *Bombyx mori* [290], *Danaus plexippus* [289], *Heliconius melpomene* [287], *Plutella xylostella* [187], and *Manduca sexta* [291]. Additionally, we reassembled raw transcriptome sequence data from three previously published studies for 10 taxa: *Striacosta albicosta* [179]; *Micropterix calthella* and *Philopotamus ludificatus* [33]; and *Pterodecta felderi*, *Nothus lunus*, *Lantanophaga pusillidactyla*, *Notoplusia minuta*, *Anigraea sp.*, *Manoba major*, and *Lyssa zampa* [25]. For the remaining 67 taxa, we generated transcriptomes *de novo* by RNA sequencing (RNA-Seq). Forty-four of these taxa were used in our previous study (Chapter 7); classification of the 23 newly-sequenced taxa is given in Table

Superfamily	Family	Subfamily	Genus	Species	RNA-Seq taxon code
Alucitoidea	Alucitidae		*Alucita*	*huebneri*	Alu
Choreutoidea	Choreutidae	Brenthiinae	*Brenthia*	*stimulans*	Bren
Choreutoidea	Millieriidae		*Millieria*	*dolosalis*	Mido
Cossoidea	Cossidae	Hypoptinae	*Hypopta*	*sp.*	Hyp
Cossoidea	Cossidae	Metarbelinae	*Lebedodes*	*ianrobertsoni*	Lr3
Cossoidea	Cossidae	Zeuzerinae	*Endoxyla*	*encalypti*	End
Epermenioidea	Epermeniidae		*Epermenia*	*chaerophyllella*	Eper
Galacticoidea	Galacticidae		*Homadaula*	*anisocentra*	Hma
Gracillarioidea	Douglasiidae		*Tinagma*	*gaedikei*	Tgm
Palaephatoidea	Palaephatidae		*Ptyssoptera*	*sp.*	Ptys
Papilionoidea	Pieridae	Coliadinae	*Colias*	*eurytheme*	Pie
Pterophoroidea	Pterophoridae	Agdistinae	*Agdistis*	*americana*	Agd
Sesioidea	Brachodidae		*Miscera*	*basichrysa*	AK142
Tineoidea	Dryadaulidae		*Dryadaula*	*visaliella*	Dry
Tineoidea	Meessiidae		*Eudarcia*	*simulaticella*	Euds
Tineoidea	Tineidae	Tineinae	*Tineola*	*bisselliella*	Tin3
Tischerioidea	Tischeriidae		*gen.*	*sp.*	Ts2
Tortricoidea	Tortricidae	Chlidanotinae	*Auratonota*	*petalocrossa*	Aur
Yponomeutoidea	Yponomeutidae	Attevinae	*Atteva*	*aurea*	Ata
Zygaenoidea	Cyclotornidae		*Cyclotorna*	*sp. ANIC6*	Cycl
Zygaenoidea	Epipyropidae		*Fulgoraecia*	*exigua*	Ful
Zygaenoidea	Epipyropidae		*Heteropsyche*	*sp.*	Het
unplaced	unplaced		*Heliocosma*	*sp. ANIC1*	Hcs

Table 8.1: Exemplars used for RNA-Seq and their distribution across the classification of van Nieukerken et al. [231].

8.1. All of the specimens we sequenced came from the ATOLep collection built by the Assembling the Lepidoptera Tree of Life project (Leptree), and had been stored in 100% ethanol at -80° C, some for more than 20 years.

In this study, some exploratory analyses used the set of 16 test taxa from our previous study (Chapter 7); one comparative analysis used all 46 taxa from our previous study; and the majority of analyses used the complete set of 82 taxa. (One final analysis used a 23-taxon subset of the 82 taxa.) We compare our new results primarily to those of our previous study (Chapter 7).

8.3 RNA-Seq data generation

Total RNA was extracted using Promega SV total RNA isolation mini-kits. The great majority of our specimens were adults; only four taxa were studied as larvae: *Galleria*, *Lymantria*, *Epipomponia*, and *Megalopyge* (see Table S1 of Bazinet et al. [86]). Species identifications were verified by comparison of COI sequences with those in the Barcode of Life Data System [263]. For larger moths we used the thorax and/or anterior part of the abdomen; for a few smaller ones we used the entire body. RNA extracts were submitted to the University of Maryland-Institute for Bioscience and Biotechnology Research Sequencing Core. The quality of total RNA was assessed by capillary electrophoresis on an RNA chip using an Agilent Bioanalyzer 2100 system. RNA preps of sufficient quality were subjected to poly-A selection and indexed library construction for sequencing on an Illumina HiSeq 1000. Following Hittinger et al. [11] our libraries were left unnormalized so as to favor highly-expressed genes likely to be present in most species and life stages. Libraries were either run four per lane, yielding about 110 million 100-bp paired-end reads per taxon (44 taxa; Table S2 of Bazinet et al. [86]), or eight per lane, yielding about 63 million 100-bp paired-end reads per taxon (23 taxa; Table 8.2). Previously-published transcriptome libraries, which were either 100-bp single-end, 100-bp paired-end, or 150-bp paired-end, averaged about 37 million reads per taxon (Table 8.3).

The Illumina reads for the 44 taxa used in the previous study (Chapter 7) are available in the NCBI Sequence Read Archive as BioProject PRJNA222254; the Illumina reads for the 23 newly-sequenced taxa have not yet been made available.

8.4 Sequence quality control and transcript assembly

Quality control of sequence reads and transcript assembly had previously been performed for 44 of our taxa plus *Striacosta* (Chapter 7); thus, we performed quality control and assembly for our 23 newly-generated transcriptomes and nine previously published transcriptomes with the slightly updated methods described here.

We used the default Illumina HiSeq 1000 quality filter, which ensured that at least 24 of the first 25 template cycles had a "Chastity" value greater than 0.6. The Chastity value is a ratio between the highest intensity and the sum of the two highest intensities. We discarded reads that did not pass the Chastity quality filter (≈ 5–7% per sample; Table 8.2). Then we used autoadapt [292] (which in turn calls FastQC [293] and cutadapt [294]) with default settings to detect and remove overrepresented sequences, as well as to trim and remove low-quality reads. About 58 million reads per taxon for our newly-generated transcript data (Table 8.2), and about 33 million reads per taxon for previously-published transcript data (Table 8.3) were used as input to the assembly process.

De novo transcriptome assembly was performed using both Trinity (versions r2014-04-13 and r2014-07-17; [44]) and Trans-ABySS (version 1.4.4; ABySS version 1.5.2; [45, 266]). Assembly statistics such as numbers and length of transcripts, as well as standard assembly metrics such as N50 (the length N for which 50% of all bases are contained in contigs of length $L < N$) are given in Table S2 of Bazinet et al. [86] and in Tables 8.2 and 8.3. A typical Trinity assembly required greater than 100 GB RAM and finished in 24 to 96 hours using 16 processing cores. A typical Trans-ABySS run required less than 4 GB RAM and a single processor, finishing in 1–2 hours. The same was true for each constituent ABySS run, of which there were 23 per sample (k ranged from 52 to 96 in steps of two). In general, Trinity used more RAM and produced fewer transcripts than Trans-ABySS, but it produced longer transcripts (Table S2 of Bazinet et al. [86] and Tables 8.2 and 8.3). Combining the Trinity and Trans-ABySS assemblies yielded a slightly more complete data matrix than using either assembly by itself, so we used the combined assembly throughout the workflow. We also found that recovery of mitochondrial genes (Section 8.5.2) was significantly aided by including the Trans-ABySS assembly.

Taxon	Reads	Failed Chastity	After autoadapt	Transcript fragments		Median TF length (nt)		N50	
				Trinity	Trans-ABySS	Trinity	Trans-ABySS	Trinity	Trans-ABySS
Agd	54,241,534	5.6%	50,652,716	37,649	161,065	439	159	1,438	930
Ata	59,644,204	5.5%	55,637,090	42,306	164,856	515	417	1,817	1,250
Cycl	63,959,982	5.5%	61,687,578	66,195	187,184	428	306	1,182	864
End	59,709,414	5.9%	55,390,312	39,456	168,366	422	186	1,244	725
Ful	59,004,712	6.4%	53,508,538	43,710	204,383	451	176	1,809	1,302
Hcs	61,154,138	5.5%	57,095,220	41,979	168,897	497	287	2,118	1,233
Tin3	60,715,492	5.4%	28,446,636	40,696	174,352	581	322	947	536
Ts2	52,103,346	6.3%	23,982,112	86,328	391,401	385	207	1,357	684
AK142	63,016,778	6.1%	58,456,282	349,543	1,213,686	326	201	500	850
Alu	67,256,716	5.6%	62,986,846	254,153	839,695	335	205	536	650
Aur	72,091,792	7.5%	65,917,352	49,640	253,159	409	162	1,042	509
Bren	78,048,610	7.2%	71,063,808	339,831	608,087	350	207	757	916
Dry	57,540,436	6.4%	53,149,010	100,745	305,698	371	205	1,503	1,741
Eper	58,285,538	7.0%	53,090,988	60,398	227,214	381	210	1,548	1,710
Euds	52,928,190	5.6%	49,465,074	76,663	170,687	429	877	1,676	2,451
Het	74,834,174	6.7%	68,820,760	187,648	711,123	394	228	1,215	1,352
Hma	62,863,690	7.5%	56,790,858	39,839	153,348	386	166	878	544
Hyp	64,960,278	7.5%	59,007,170	36,041	128,598	370	175	708	468
Lr3	70,730,128	7.5%	64,559,468	60,344	283,944	378	179	811	657
Mido	67,985,372	5.8%	63,384,530	588,913	1,608,565	346	203	660	521
Pie	79,784,432	7.1%	73,371,956	44,872	277,674	503	218	1,647	1,106
Ptys	56,743,320	6.3%	52,342,996	196,125	614,087	306	214	739	1,406
Tgm	52,932,672	6.2%	48,849,858	74,784	237,492	409	799	1,817	2,400
mean	63,066,737	6.4%	58,264,605	124,255	402,329	409	274	1,215	1,078

Table 8.2: Summary statistics for RNA-seq reads and assemblies.

Taxon	Reads	Read length	After autoadapt	Transcript fragments		Median TF length (nt)		N50	
				Trinity	Trans-ABySS	Trinity	Trans-ABySS	Trinity	Trans-ABySS
Micropterix calthella	78,539,568	PE100	67,781,972	192,011	737,387	449	284	791	684
Philopotamus ludificatus	68,898,972	PE100	56,235,488	34,088	425,227	591	158	1,031	594
Pterodecta felderi	23,782,033	SE100	22,234,241	38,102	204,909	454	177	1,357	684
Nothus lunus	22,436,454	SE100	20,682,503	39,168	159,384	523	296	1,614	971
Lantanophaga pusillidactyla	23,149,096	SE100	21,506,159	38,438	269,188	483	299	1,370	647
Notoplusia minuta	24,890,722	PE150	24,486,070	64,411	245,092	406	224	1,140	505
Anigraea sp.	26,732,516	PE150	26,406,826	50,199	215,794	495	263	1,586	956
Manoba major	28,162,032	PE150	27,832,930	71,626	238,304	518	260	1,808	1,144
Lyssa zampa	32,486,672	PE150	31,384,338	52,940	204,086	476	404	1,841	1,310
mean	36,564,229	N/A	33,172,281	64,554	299,930	488	263	1,393	833

Table 8.3: Summary statistics for reassembled RNA-seq data from Peters et al. [33] and Kawahara and Breinholt [25].

8.4.1 Reassembling previously-published transcriptomes

Some modifications to our assembly methods were necessary to reassemble the *Striacosta albicosta* transcriptome [179], which are described in Section 7.5, and similar modifications were necessary to reassemble single-end libraries from Kawahara and Breinholt [25].

Comparative assembly statistics for *Micropterix* and *Philopotamus*, given in Table 8.4, indicated that our Trinity assemblies were likely of greater overall quality than the Newbler assemblies used in Peters et al. [33].

To assess the quality of our reassembled data taken from Kawahara and Breinholt [25], we used the *Pterodecta* taxon. Our *Pterodecta* Trinity assembly contained fewer transcripts than the corresponding SOAPdevnovo-Trans [295] assembly from Kawahara and Breinholt [25] (38,102 vs. 83,540, respectively); however, our N50, maximum contig size, and mean contig size were all significantly better (cf. Table 8.3 and Table S1 of Kawahara and Breinholt [25]), suggesting that our assembly was less fragmented and possibly less redundant. To test this more definitively, we used HaMStR (version 13.2.2; [272]) and ran each *Pterodecta* assembly against the `moth+min-one-butterfly` database (Section 8.5.1). The number of hit sequences from the Trinity assembly (4,050) was comparable to the number of hits from the SOAPdevnovo-Trans assembly (4,001), as was the number of hits to unique orthologous groups (Trinity: 3,703; SOAPdevnovo-Trans: 3,561). However, the total size of the sequence hits was significantly better for our Trinity assembly (1059 KB vs. 736 KB), so we reprocessed the sequence data from Kawahara and Breinholt [25] using our quality control and assembly procedures.

8.5 Database construction

In this study, we decided to incorporate analyses that used both nuclear and mitochondrial (mt) genes. The precedent for using nuclear and mt genes, variously, to resolve lepidopteran phylogeny is well established [5,6,126,239,241,246,296]. Our previous study (Chapter 7) only used nuclear genes; however, we thought it likely that we also captured mt gene sequences in the course of RNA sequencing, so we built gene models of the 13 invertebrate protein-coding mt genes so we could search for those sequences in our transcriptome data. We had in mind the fact that Timmermans et al. [296] were unable to resolve the lower Apoditrysia using mitochondrial data alone; however, we reasoned that even if we were similarly unsuccessful, the analyses might at least provide some supporting information; and, due to our more extensive taxon sampling, we would have even more definitively demonstrated the infeasibility of mt data to resolve this problematic region of lepidopteran phylogeny. Here we describe how we constructed both our nuclear and mt gene databases.

8.5.1 The “moth+min-one-butterfly” nuclear gene database

To begin building our Lepidoptera-specific nuclear gene database, we downloaded peptide and coding sequences for *Bombyx mori*, *Heliconius melpomene*, and *Danaus plexippus* from Ensembl Metazoa, release 22 [297,298]. Providing all the peptide, or “gene” sequence identifiers as input, we built up orthologous groups using the one2one, one2many, many2many, within_species_paralog, putative_gene_split, and contiguous_gene_split homology relationships defined in Ensembl that involved any two of these three taxa [299]. We required an orthologous group to contain a *Bombyx* sequence and a minimum of one butterfly sequence (either *Danaus* or *Heliconius*), which resulted in 7,042 orthologous groups. From Ensembl we retrieved the "genetree alignment" corresponding to each orthologous group; from each genetree alignment we extracted only the sequences belonging to the three Lepidoptera species of interest, removed gaps, and realigned the amino acid sequences using the linsi algorithm in MAFFT [276] and our custom LEP62 substitution matrix (Appendix B). We built a preliminary moth+min-one-butterfly database for HaMStR (version 13.2.2; [272]) consisting of 7,042 pHMMs derived from the MAFFT alignments, and a BLAST database that contained the complete proteome of *Bombyx*, our designated reference taxon as required by HaMStR.

8.5.1.1 Alignment filtering

Upon visual inspection, we became concerned that some of the amino acid alignments in the moth+min-one-butterfly database were suboptimal. To avoid including suboptimal alignments in our data matrices, we used T-Coffee [300] to calculate a similarity score for each alignment in the database. The median alignment similarity score was 81.6%; we decided to remove alignments (i.e., orthologous groups) with a similarity score less than 70%, which roughly corresponded to the lowest quartile of alignment similarity scores. This left 5,283 orthologous groups in the `moth+min-one-butterfly` database.

8.5.1.2 Adding *Plutella* and *Manduca*

At the time we performed this study, two additional Lepidoptera genomes were available (*Plutella xylostella* and *Manduca sexta*), although not through Ensembl, and we sought to include these two taxa in our nuclear gene database.

In the case of *Plutella xylostella* (the diamondback moth), two groups were sequencing the genome independently. The Japanese group made their genome sequence available through KONAGAbase [301], and the Chinese group made theirs available through DBM-DB [302]. The data from KONAGAbase consisted of a putative gene set that was the result of combining their genome and transcriptome gene annotations (32,800 sequences) with a putative "unknown" gene set (39,781 sequences). The data from DBM-DB consisted of the coding sequence associated with their genome-based gene predictions (18,073 sequences), together with all "unigenes" from their transcriptome data (171,262 sequences). In order to select one of these data resources, we combined the sequences belonging to each data resource (72,581 sequences for KONAGAbase and 189,335 sequences for DBM-DB) and ran each set of sequences against the `moth+min-one-butterfly` HaMStR database. We found that the "representative" sequences (i.e., the sequences that were the best match to each orthologous group in the database) were longer, on average, in the DBM-DB data than in the KONAGAbase data, and also slightly more numerous; thus, we decided to use only the DBM-DB *Plutella* data going forward.

The *Manduca sexta* (tobacco hornworm) genome data was available from Manduca Base [291] (retrieved late January 2014), and consisted of 27,633 transcripts (CDS regions extracted from the original genome/gff3 file using `gffread`).

To add *Plutella* and *Manduca* to the `moth+min-one-butterfly` database we used HaMStR, setting both the hmmsearch and the BLAST E-value cutoffs to 1e-10. This yielded 9,739 hits in the *Plutella* data (4,809 unique orthologous groups), and 5,593 hits in the *Manduca* data (4,576 unique orthologous groups). We stipulated that in order to add a *Plutella* or *Manduca* hit sequence to an existing `moth+min-one-butterfly` orthologous group, the sequence needed to be at least half as long as the shortest sequence in the existing `moth+min-one-butterfly` orthologous group. Both the relatively stringent E-value and this minimum length criterion were an attempt to keep short, potentially spuriously-matching sequences out of the database.

After adding the *Plutella* and *Manduca* sequences, the orthologous groups in the HaMStR database were realigned *de novo* using MAFFT as before. Following this, we used the T-Coffee similarity statistic to evaluate the new alignments. The median alignment similarity score was 86.2%; once again, we removed alignments with a similarity score less than 70% (131 alignments), leaving 5,152 orthologous groups in the `moth+min-one-butterfly` database.

8.5.2 The Lepidoptera mitochondrial gene database

We selected seven taxa that covered most of the major superfamilies of Apoditrysia (Table 8.5), and used these taxa to build the `lep-mt-gene` database.

We downloaded the mitochondrial genome from GenBank for each of the seven taxa, parsed out the 13 protein-coding mt genes, and created an amino acid alignment for each gene using the `linsi` algorithm in MAFFT [276] and our custom LEP62 substitution matrix (Appendix B). We built a preliminary `lep-mt-gene` database for HaMStR (version 13.2.2; [272]) consisting of 13 pHMMs derived from the MAFFT alignments, and a BLAST database containing the complete proteome of *Bombyx*, our designated reference taxon as required by HaMStR, including its 13 protein-coding mt genes.

8.6 Orthology determination

To infer orthology, we used HaMStR (version 13.2.2; [272]), which in turn used BLASTP [13], GeneWise [273], and HMMER [274] to search the combined assembly transcript data for translated sequences that matched a set of previously-constructed amino acid gene models. The gene models in our study were organized into two databases: the `moth+min-one-butterfly` database of 5,152 nuclear genes, and the `lep-mt-gene` database of 13 mt genes (Section 8.5).

In the first step of the HaMStR procedure, substrings of assembled transcripts (translated nucleotide sequences) that matched one of the gene models in the database were provisionally assigned to the matching orthologous group. To reduce the number of highly-divergent, potentially-paralogous sequences returned by this initial search, we set the E-value cutoff defining a "hit" to 1e-05 (the HaMStR default was 1.0), and retained only the top-scoring quartile of hits. In the second HaMStR step, the provisional hits from the HMM search were compared to a "reference taxon" (*Bombyx mori*), and retained only if they survived a reciprocal best BLAST hit test with the reference taxon. In our implementation, we substituted FASTA [303] for BLAST; we found that using FASTA (specifically, the `fasty` program) with our custom LEP62 substitution matrix provided more discriminatory power than using BLAST with BLOSUM62 (Appendix B). We set the E-value cutoff for the FASTA search to 1e-05 (the HaMStR default was 10.0). Once assigned to orthologous groups, amino acid sequences from our transcripts were aligned using the `addfragments` option to MAFFT [276] and our custom LEP62 substitution matrix (Appendix B), in which procedure the *Bombyx* sequences were considered the reference alignment to which the transcript fragments were added. The resulting amino acid alignments were then converted to the correct corresponding nucleotide alignments using a custom Perl script that substituted for each amino acid the proper codon from the original coding sequence.

When running HaMStR with the `lep-mt-gene` database, it was necessary to change the genetic code to the invertebrate mitochondrial code in several places: (1) in the `translate.pl` HaMStR script; (2) in the call HaMStR made to GeneWise; and (3) in the call HaMStR made to `fasty` ("`-t 5`" or "`-t t5`"; cf. FASTA documentation). In the course of making these modifications, we helped discover and fix two bugs in HaMStR — one having to do with a callout to the Unix `sort` command, and another having to do with sequence translation. These bugfixes are documented in the release notes associated with HaMStR version 13.2.2.

8.7 Data matrix construction and paralogy filtering

Our orthology determination pipeline often yields multiple sequences for a particular taxon-locus combination, which can reflect the presence of multiple orthologs, heterozygosity, alternatively-spliced transcripts, paralogy (including inparalogs [182]), and sequencing errors — among other possibilities. We evaluated two different approaches for reducing this variation to a single sequence, as required for phylogenetic analysis, which we term "REPRESENTATIVE" and "CONSENSUS"; these are described in more detail in Section 7.8 and Appendix A. For the majority of analyses in this study we opted to use only the CONSENSUS procedure; in preliminary analyses we noticed CONSENSUS slightly outperformed REPRESENTATIVE, and if we had continued to use both procedures we would have doubled the already substantial computational requirements of the phylogenetic analyses.

To screen for possible evidence of paralogy in the `moth+min-one-butterfly` database of 5,152 nuclear genes, we constructed a maximum likelihood (ML) gene tree (Section 8.9.1) for each orthologous group using all matching sequences from our 16-taxon test set, and provided the gene trees as input to PhyloTreePruner [280]. If the sequences for a particular taxon formed a polyphyletic group, the program pruned the gene tree to the maximal subtree in which the non-polyphyly criterion was met for all taxa. Gene trees were only constructed for 4,862 of the 5,152 orthologous groups (≈ 94%), as the others had fewer than four sequences. PhyloTreePruner pruned 1,052 of the 4,862 gene trees (≈ 22%) to some extent. As in the previous study (Chapter 7), we took a very conservative action based on these results, using for all subsequent phylogenetic analyses only the 3,810 genes in which no evidence of paralogy was found in the test taxa. Following application of the paralogy filter, the 3,810 putative ortholog alignments were concatenated, adding gaps for missing data as necessary using a custom Perl script. For the majority of phylogenetic analyses, the nucleotide data was recoded with `degen1` (version 1.4; [281]); for mt gene analyses, `degen1` was applied using the invertebrate mitochondrial genetic code. (See Section 7.8 for more information about `degen1` recoding.) Unaltered nucleotide data is referred to with the abbreviation "`nt123`". For the majority of analyses, sites not represented by sequence data in at least four taxa were removed. For two analyses, we used only second codon positions (designated "`nt2`"), and removed sites not represented by sequence data in at least 80% or 90% of taxa. For the final 23-taxon analysis, we filtered each ortholog alignment with PYGOT [304], and then with GUIDANCE2 [305].

8.8 Data matrix properties

Statistics for the paralogy-filtered matrices of 3,810 nuclear genes are given in Tables 8.6 and 8.7, and are given similarly for the matrices of 13 protein-coding mt genes in Table 8.8.

Taxon	Reads	After filtering	Contigs	Large contigs	Average large	Largest contig
			≥ 150 bp	> 500 bp	contig size	
Micropterix calthella (Newbler)	78,539,568	19,182,692	172,391	41,673	752	5,326
Micropterix calthella (Trinity)	78,539,568	67,781,972	192,011	86,304	967	10,772
Philopotamus ludificatus (Newbler)	68,898,972	27,099,685	29,294	10,959	885	3,348
Philopotamus ludificatus (Trinity)	68,898,972	56,235,488	34,088	19,820	1,064	10,796

Table 8.4: Comparative assembly statistics for *Micropterix* and *Philopotamus*, modeled after Table S2 of Peters et al. [33]. Newbler assembly statistics were taken from Table S2 of Peters et al. [33].

Superfamily	Exemplar	GenBank accession
Bombycoidea	*Bombyx mori*	NC_002355
Gracillarioidea	*Leucoptera malifoliella*	NC_018547
Noctuoidea	*Helicoverpa armigera*	NC_014668
Papilionoidea	*Pieris rapae*	NC_015895
Pyraloidea	*Ostrinia furnacalis*	NC_003368
Tortricoidea	*Cydia pomonella*	NC_020003
Yponomeutoidea	*Plutella xylostella*	JF911819

Table 8.5: The seven taxa represented in the Lepidoptera mt gene database.

	82 taxa, 3,810 nuclear genes			
	nt123	degen1	nt2, 80%	nt2, 90%
number of nucleotide positions	7,852,266	4,653,957	671,539	412,265
number of non-gap chars in alignment	254,642,214	250,386,693	50,184,945	32,118,970
matrix completeness (nt present ÷ *possible* nt)	39.5%	65.6%	91.1%	95.0%
percent ambiguous nt (non-gap, non-A/C/G/T chars)	0.6%	37.5%	0.2%	0.2%
GARLI memory requirement (in MB)	83646	64349	8625	4835

Table 8.6: Size and completeness of aligned 82-taxon nuclear gene data matrices from RNA-Seq. All nuclear matrices were constructed using the consensus procedure.

	3,810 nuclear genes			
	46 taxa	23 taxa	16 taxa	
	degen1	degen1	degen1	degen1, 100%
number of nucleotide positions	4,477,332	4,225,176	4,042,908	1,183,539
number of non-gap chars in alignment	148,512,429	72,915,234	50,019,603	18,936,624
matrix completeness (nt present ÷ *possible* nt)	72.1%	75.0%	77.3%	100.0%
percent ambiguous nt (non-gap, non-A/C/G/T chars)	37.5%	39.9%	37.5%	37.3%
GARLI memory requirement (in MB)	27720	10329	4692	829

Table 8.7: Size and completeness of aligned nuclear gene data matrices from RNA-seq. All nuclear matrices were constructed using the CONSENSUS procedure.

	13 protein-coding mt genes				
	82 taxa		16 taxa		
	degen1	nt123[1]	degen1	degen1, rep.	nt123[1], rep.
number of nucleotide positions	11,175	11,967	10,944	10,944	11,328
number of non-gap chars in alignment	829,185	830,220	165,528	161,784	162,246
matrix completeness (nt present ÷ *possible* nt)	90.5%	84.6%	94.5%	92.4%	89.5%
percent ambiguous nt (non-gap, non-A/C/G/T chars)	42.8%	6.7%	44.7%	38.3%	0.04%
GARLI memory requirement (in MB)	180	205	21	16	22

[1] `nt123` matrices were for use with a codon model, so no columns were removed.

Table 8.8: Size and completeness of aligned mt gene data matrices from RNA-seq. Where given, "rep." indicates the matrix was constructed using the REPRESENTATIVE procedure; all other matrices were constructed using the CONSENSUS procedure.

8.9 Phylogenetic analysis

8.9.1 Maximum likelihood phylogenetic analysis

Maximum likelihood phylogenetic analysis used GARLI (Genetic Algorithm for Rapid Likelihood Inference; versions 2.0 and 2.1; [2]) and grid computing [48,223] via a web service at `molecularevolution.org` [53] based on tools developed by Bazinet et al. [284] that include post-processing with DendroPy [208], R [285], and custom Perl scripts. For the nuclear gene analyses we used a GTR+I+G nucleotide model; for the mt gene analyses we used both a GTR+I+G nucleotide model, and a codon model with the following settings: `ratematrix=6rate`; `statefrequencies=f1x4`; `ratehetmodel=none`; `numratecats=1`; `invariantsites=none`; and `geneticcode=invertmito`. We used GARLI default settings, including stepwise addition starting trees, except that we lowered the number of successive generations yielding no improvement in likelihood score that prompts termination (`genthreshfortopoterm=5000`), as we found that this saved time and yielded comparable results. In some cases we used a constraint tree as a starting tree, and for these runs we also enforced this constraint using a constraint file specifically formatted for GARLI. Each best tree was selected from between 10 and 100 GARLI search replicates, while bootstrap analyses consisted of 40 to 2,000 replicates. Insufficient search effort during bootstrapping has been shown to artificially depress bootstrap support (BP) values [7]. A rough guide to the effort needed was provided by our initial ML searches: if the best tree topology was found only rarely, this indicated that multiple search replicates per bootstrap replicate might be helpful. Thus, for data sets that were not too computationally intensive, we ran five search replicates per bootstrap replicate (instead of just one). We used DendroPy [208] to generate a 50% majority-rule bootstrap consensus tree, as well as to plot BP values onto the best tree.

8.9.1.1 Computational requirements and strategies

A substantial amount of computation was required to complete the maximum likelihood analyses in this study. Memory requirements ranged from 16 MB for the smallest concatenated analysis, to 64,349 MB for the largest concatenated analysis (Tables 8.6, 8.7, and 8.8). Each search replicate required anywhere from a few minutes to several days of runtime.

The phylogenetic analyses were completed using a variety of computational resources. For analyses that required a significant amount of runtime, and a small to intermediate amount of memory, we used the BOINC volunteer computing platform [50] through our BOINC project (http://boinc.umiacs.umd.edu). For analyses that required a large amount of memory, we used dedicated computing resources available in the Center for Bioinformatics and Computational Biology at the University of Maryland, College Park. For the largest memory analyses, we used the Deepthought II computing cluster at the University of Maryland, College Park. Analyses that used a large amount of memory frequently occupied an entire multicore compute node; thus, for these analyses we used the OpenMP [306] version of

GARLI so that we could use all of the cores on the compute node simultaneously, which helped the jobs complete significantly more quickly.

As the large-memory analyses required the use of a queuing system, jobs faced specific limits on runtime and memory usage, competition from other users, and general resource consumption limits that made them challenging to complete. Thus, we devised a scheme using features of GARLI version 2.1 that divided long-running analyses into shorter jobs that could be scheduled more quickly and efficiently, and more of which could run in parallel. This scheme, which was similar to the one described in Chapter 6, used the GARLI checkpointing feature along with the configuration settings `workphasedivision=1` and `stoptime` to cause a GARLI analysis to be divided into three parts: initial optimization ($\approx 5\%$ of overall runtime), the main work loop ($\approx 90\%$ of overall runtime), and final optimization ($\approx 5\%$ of overall runtime). The main loop was further divided into equal-length jobs whose maximum length was equal to `stoptime`. In this way, the maximum runtime of most jobs was explicitly controlled, and analyses made progress by restarting from periodic checkpoints. This scheme, which required a relatively complex organizational structure and management of a large number of files, job submissions, and job cancellations was orchestrated with custom Perl scripts that were developed for use with both the TORQUE [307] and SLURM [308] queuing systems.

8.9.2 Gene tree summary methods

Short internodes in the lepidopteran phylogeny, possibly the result of adaptive radiations, may be difficult to resolve with traditional approaches such as data concatenation due to the confounding effect incomplete lineage sorting (ILS) may have on phylogenetic signal. Newer methods, variously termed "gene tree/species tree" or simply "species tree" methods, attempt to adequately account for ILS by incorporating it into their analysis model, and thus may recover robustly-supported species trees in cases where traditional methods would not. Species tree methods may be broadly divided into two categories: "gene tree summary" methods, which reconcile a set of gene trees that have been previously computed independently into a single species tree [309]; and "joint estimation" methods, which estimate gene trees and the species tree simultaneously [310]. Joint estimation methods are known to be computationally prohibitive for the numbers of genes and taxa we examined in our study [311]; thus, we focused our attention exclusively on gene tree summary methods.

In our study, we used the following gene tree summary methods: MP-EST (version 1.4; [312]), NJst (phybase version 1.3; [313]), RTC [314], STAR (phybase version 1.3; [315]), and STEAC (phybase version 1.3; [315]). As input, these methods were provided the 3,810 gene trees previously computed with GARLI (settings for GARLI analyses are given in Section 8.9.1). The gene trees either resulted from a best tree search, in which case two search replicates were performed and the tree with the greater likelihood was used; or they resulted from a bootstrap analysis in which a total of 100 bootstrap replicates were performed. As explained in Mirarab et al. [309], gene tree summary methods can be used with a single maximum likelihood (ML) tree estimate for each gene ("BestML"), or with a set of the ML gene trees estimated for the bootstrap replicates of each gene ("multilocus bootstrap", or "MLBS"). Using the terminology and methodology from Mirarab et al. [309], we calculated both BestML and MLBS trees using each of the gene tree summary methods. With MP-EST, the BestML tree we used was the best tree returned from 10 separate runs of the program; the other methods were deterministic and required only one run. The largest MP-EST analyses took several hours of runtime and required use of TORQUE [307]; all other gene tree summary programs required only minutes to execute. We plotted bootstrap support (BP) values onto the BestML tree using DendroPy [208], and we computed an extended majority-rule (EMR) consensus of the 100 MLBS trees using Rphylip [316].

8.9.3 Quartet methods

We performed a quartets-based analysis using the SVDquartets [317] method. SVDquartets computes a score based on singular value decomposition of a matrix of site pattern frequencies corresponding to a split on a phylogenetic tree. The quartet scores can be used to select the best-supported topology for quartets of taxa, which in turn can be used to infer the species phylogeny using quartet methods. The entire procedure was recently implemented in PAUP* (version 4.0a141; [195]), which we used to conduct this analysis. We evaluated all possible quartets, and ran 100 bootstrap replicates. Due to relatively large memory and computational requirements, we ran this analysis on the Deepthought II computing cluster at the University of Maryland, College Park.

8.10 Phylogenetic results

Our phylogenetic results included seven concatenated nuclear gene analyses (Section 8.10.1.1), five concatenated protein-coding mitochondrial gene analyses (Section 8.10.1.2), seven gene tree summary analyses (Section 8.10.2), and one quartets-based analysis (Section 8.10.3). A final 23-taxon concatenated nuclear gene analysis is presented in Section 8.10.4.

8.10.1 Concatenated-gene phylogenetic results

8.10.1.1 Nuclear gene analyses

16-taxon nuclear gene analyses

We ran two 3,810-nuclear gene GARLI analyses using the 16-taxon test set (see Table 8.7 for data matrix properties); the analyses had the following combinations of attributes: (1) `degen1`, CONSENSUS, nucleotide model; and (2) `degen1`, CONSENSUS, nucleotide model, 100%-complete.

The phylogenetic result for the full data matrix analysis (Figure 8.1) was based on 93 best tree search replicates and 185 bootstrap replicates (five search replicates per bootstrap replicate). To provide an idea of the amount of computation this represented, the 185 bootstrap replicates alone would have taken over three years to compute had they executed sequentially on a single processor. This result was fairly positive: as compared to the 16-taxon, 741-gene result (Figure 7.7), there was a significant increase in bootstrap support at several nodes. The two topologies differed in a few places.

The result for the 100%-complete data matrix analysis (Figure 8.1) was based on 66 best tree search replicates and 210 bootstrap replicates (five search replicates per bootstrap replicate). The memory requirement for this analysis was much reduced as compared to the full matrix analysis, so we obtained the results more quickly. We did not observe a significant change in BP values, though BP values for a couple of deeper nodes decreased a small amount; this drop in support was consistent with the effect of data reduction observed in the Kawahara and Breinholt study [25]. The topology changed somewhat, relative to that of the previous analysis; the placement of Hypoptinae varied, and Epipyropidae grouped with Sesiinae/Paranthreninae, albeit now with BP=72% instead of BP=100%.

46-taxon nuclear gene analysis

We ran one 3,810-nuclear gene GARLI analysis using the 46 taxa from the previous study (Chapter 7) (see Table 8.7 for data matrix properties); the analysis had the following combination of attributes: `degen1`, CONSENSUS, nucleotide model. The purpose of this analysis was to act as an intermediate point of comparison between the 46-taxon, 741-gene analysis (Figure 7.2) and the 82-taxon, 3,810-gene analyses (presented in the following section).

The 46-taxon, 3,810-gene phylogenetic result (Figure 8.2) was based on 10

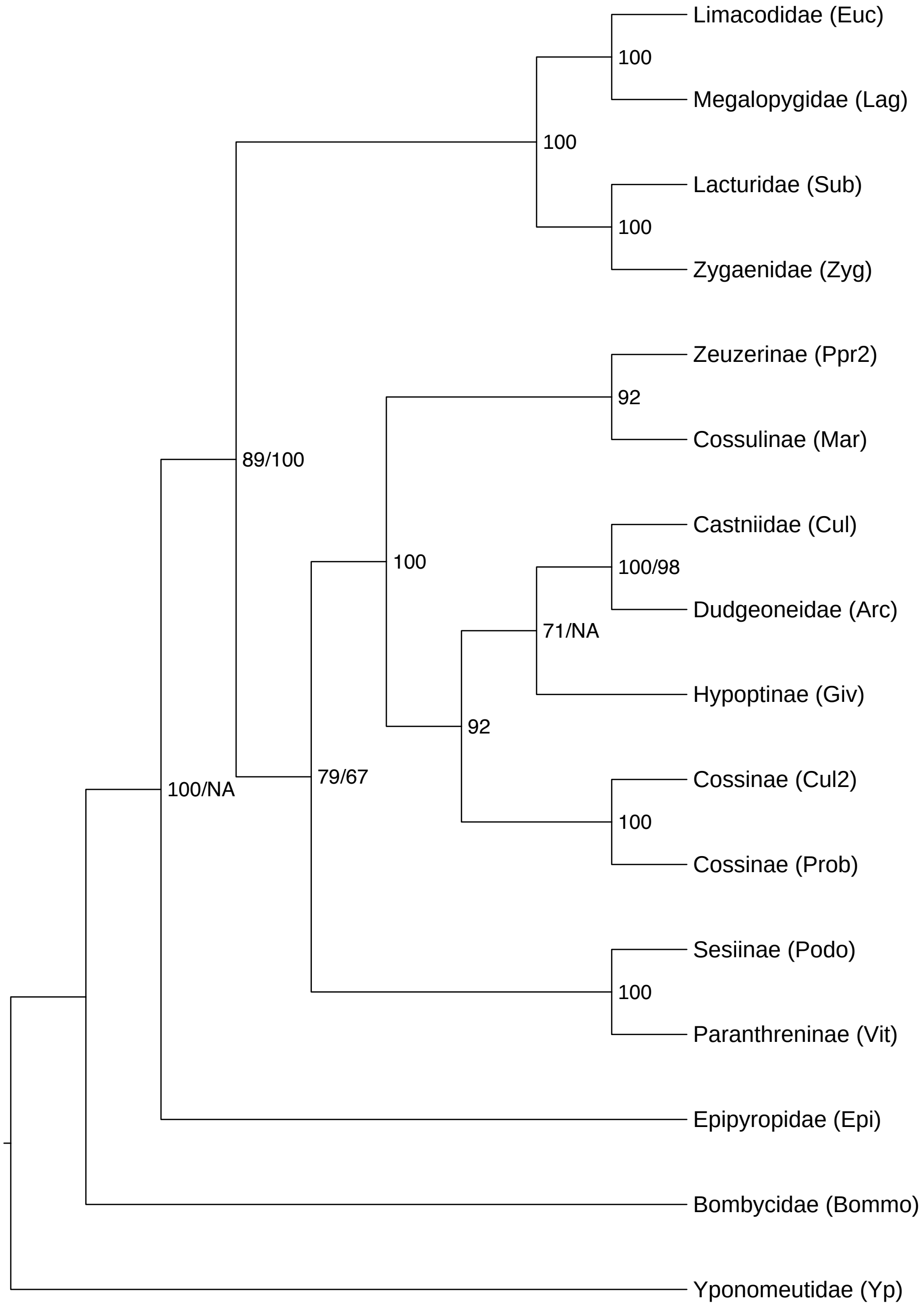


Figure 8.1: Results of the 16-taxon, 3,810 concatenated nuclear gene analyses. Bootstrap support values for the full matrix analysis are shown first, followed by BP values for the 100%-complete matrix analysis if they differed. "NA" indicates the node was not present in the 100%-complete matrix analysis.

best tree search replicates and 100 bootstrap replicates. Comparing the 46-taxon, 741-gene analysis result (Figure 7.2) to the 3,810-gene result (Figure 8.2), we made the following observations: (1) all nodes with BP=100% in the 741-gene analysis were retained in the new analysis; (2) the clade consisting of *Epicopeia hainseii* ("Epc3") through *Bombyx mori* increased to BP=100%, from BP=86%; (3) several nodes increased from weakly supported to less weakly supported; (4) several nodes decreased markedly in support; and (5) six new nodes appeared, with BP ranging from 23% to 80%. Overall, the large increase in gene number yielded only one additional BP=100% node, while producing approximately equal numbers of increases and decreases in more weakly-supported nodes. The total number of nodes with weak bootstrap values (BP=60% and below) was 6/45 for 741 genes, and 10/45 for 3,810 genes; we were surprised that the number of weak nodes was greater in the result based on the larger data matrix, because generally speaking, one hopes that additional information will help to increase support.

82-taxon nuclear gene analyses

We ran four 3,810-nuclear gene GARLI analyses using all 82 taxa (see Table 8.6 for data matrix properties); the analyses had the following combinations of attributes: (1) `degen1`, CONSENSUS, nucleotide model, constrained; (2) `nt2`, CONSENSUS, nucleotide model, 80%-complete, constrained; (3) `nt2`, CONSENSUS, nucleotide model, 90%-complete, constrained; and (4) `nt123`, CONSENSUS, nucleotide model, constrained, partitioned by codon position.

These analyses used a constraint tree (Figure 8.3), as the relationships of

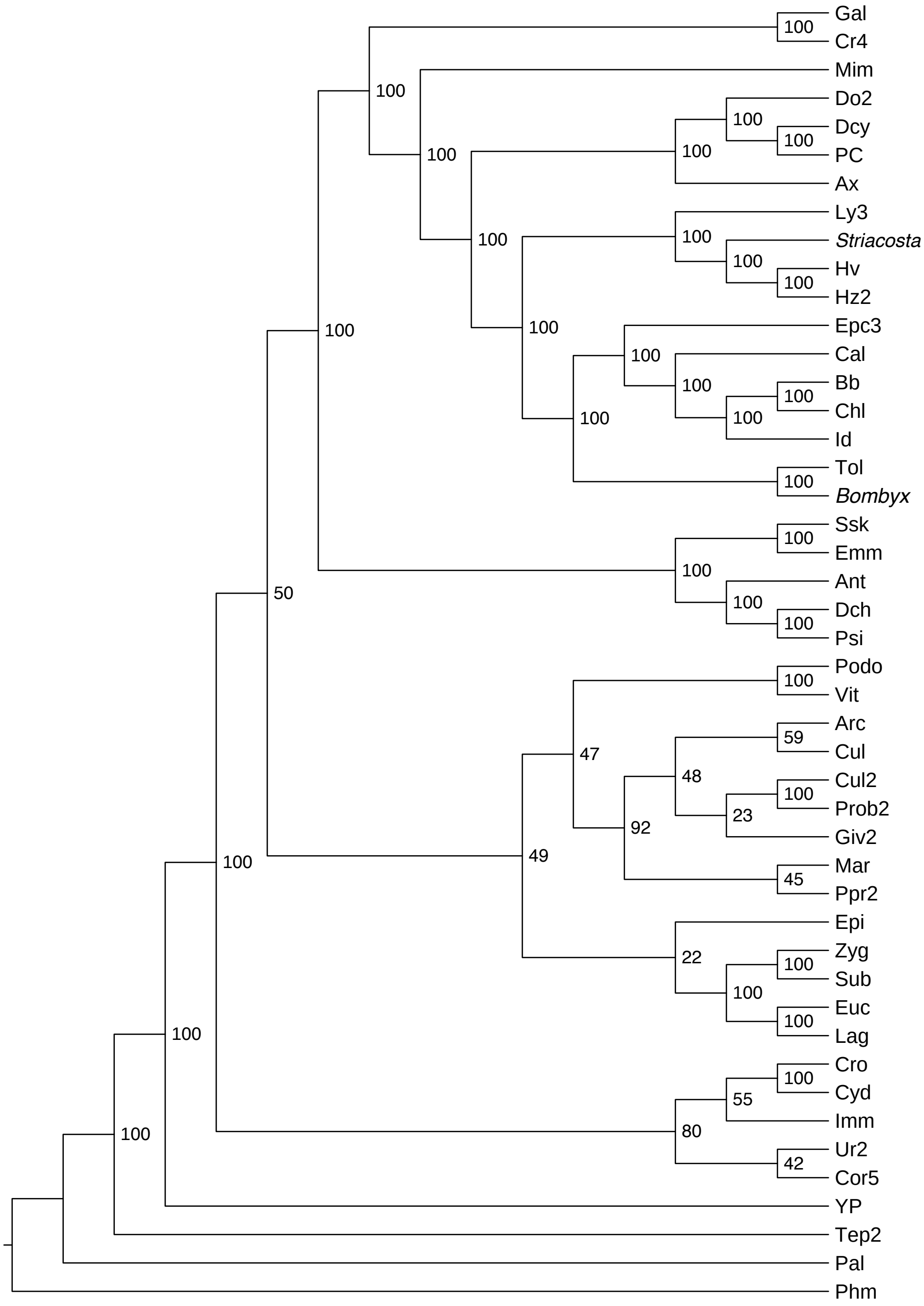


Figure 8.2: Result of the 46-taxon, 3,810 concatenated nuclear gene analysis.

some taxa were already well established and did not need to be inferred again. The constraint tree was used both as a topological constraint and as a starting tree with polytomies randomly resolved.

The phylogenetic result for the `degen1`, full-matrix analysis (Figure 8.4) was based on 10 best tree search replicates and 100 bootstrap replicates. Comparing the previous 46-taxon, 3,810-gene analysis result (Figure 8.2) to the 82-taxon result (Figure 8.4), we made the following observations: (1) overall, the addition of 36 more taxa resulted in lower bootstrap support; (2) for 46 taxa, the fraction of nodes with BP=60% or less was 7/45 (0.16); and (3) for 82 taxa, the fraction of nodes with BP=60% or less was 26/81 (0.32). Thus, by nearly doubling the number of taxa, we doubled the proportion of weakly-supported nodes. A particularly notable tree region showing this effect is the "lower Obtectomera", a clade consisting of *Emmelina monodactyla* ("Emm") through *Antaeotricha schlaegeri* ("Ant") (seven superfamilies including Gelechioidea, Pterophoroidea, Papilionoidea, and others). With 46 taxa, the relationships among the three included superfamilies had BP=100%; with 82 taxa, none of the relationships among the seven included superfamilies reached BP=60%, which was surprising.

Among the always-problematic lower Apoditrysia, we did observe BP=97% for the placement of Brachodidae, a newly-added family represented by *Miscera basichrysa* ("AK142"); we retained, at BP=83%, the group that we called "Cossoidea *sensu novo*" in the previous study (Chapter 7); and we retained the grouping of *Archaeoses polygrapha* ("Arc") and *Synemon plana* ("Cul") at BP=77%. Otherwise in this tree region, however, topology shifted a fair amount from previous results,

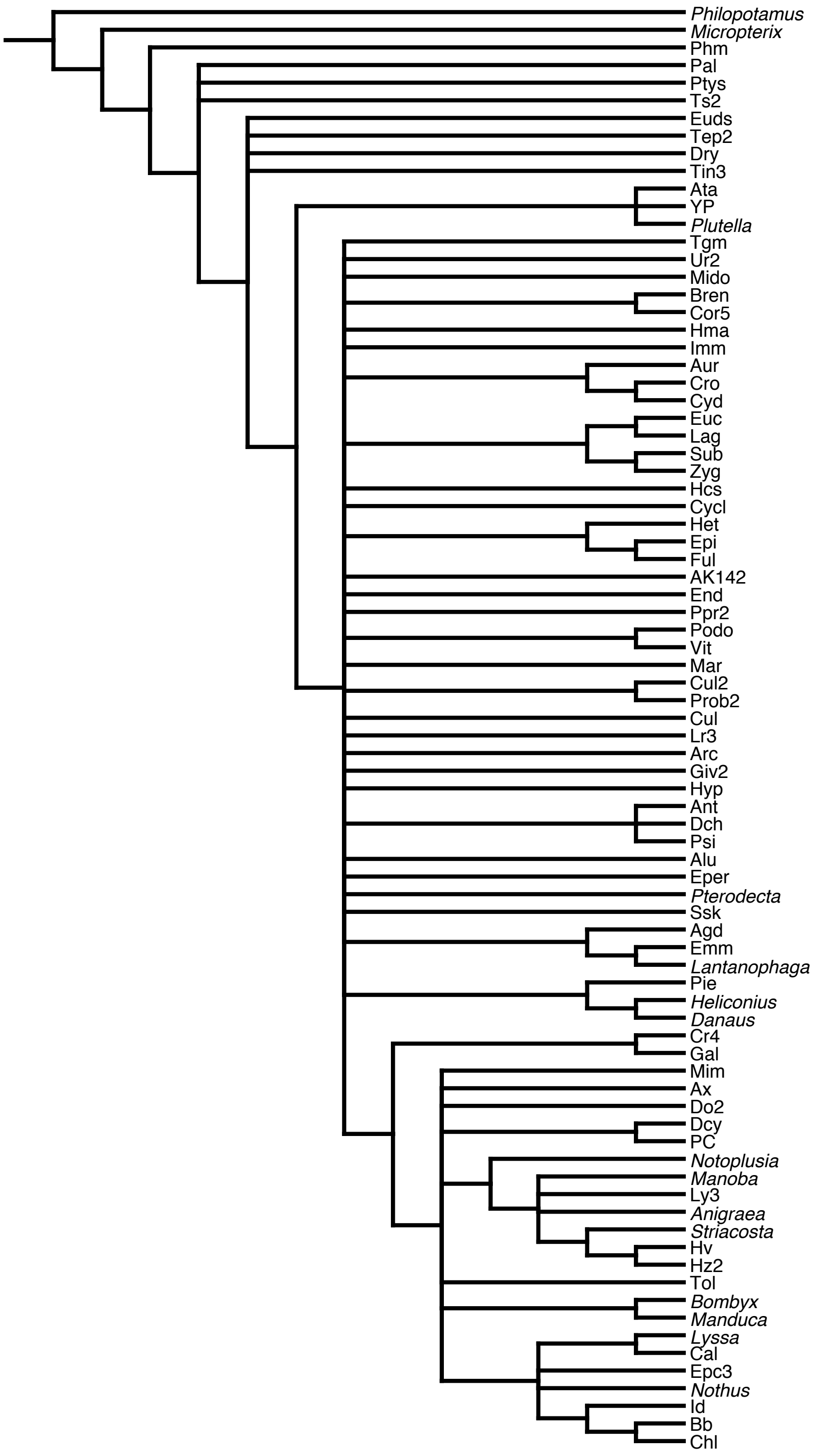


Figure 8.3: Constraint tree used with 82-taxon concatenated nuclear gene analyses.

and bootstrap values were low.

Finding lower support with greater taxon sampling is something one might expect, other things being equal, in smaller data sets, but we did not necessarily expect to find that here. We compared likelihood scores between constrained and unconstrained best tree searches — 10 each — and they overlapped extensively, suggesting that we did not introduce artifacts with our constraint.

The phylogenetic result for the `nt2`, 82-taxon, 3,810-nuclear gene, 80%-complete analysis was based on 15 best tree search replicates and 111 bootstrap replicates; the 90%-complete analysis result was based on eight best tree search replicates and 145 bootstrap replicates. These results were comparable to those from the `degen1` analysis, including the fraction of nodes with weak bootstrap values.

The phylogenetic result for the `nt123`, 82-taxon, 3,810-nuclear gene, partitioned by codon position analysis was based on 10 best tree search replicates and 38 bootstrap replicates. This result had slightly better bootstrap values than the `degen1` and `nt2` analyses, but overall was fairly comparable.

8.10.1.2 Mitochondrial protein-coding gene analyses

16-taxon mt gene analyses

We ran three mt gene GARLI analyses using the 16-taxon test set (see Table 8.8 for data matrix properties); the analyses had the following combinations of attributes: (1) `degen1`, CONSENSUS, nucleotide model; (2) `degen1`, REPRESENTATIVE,

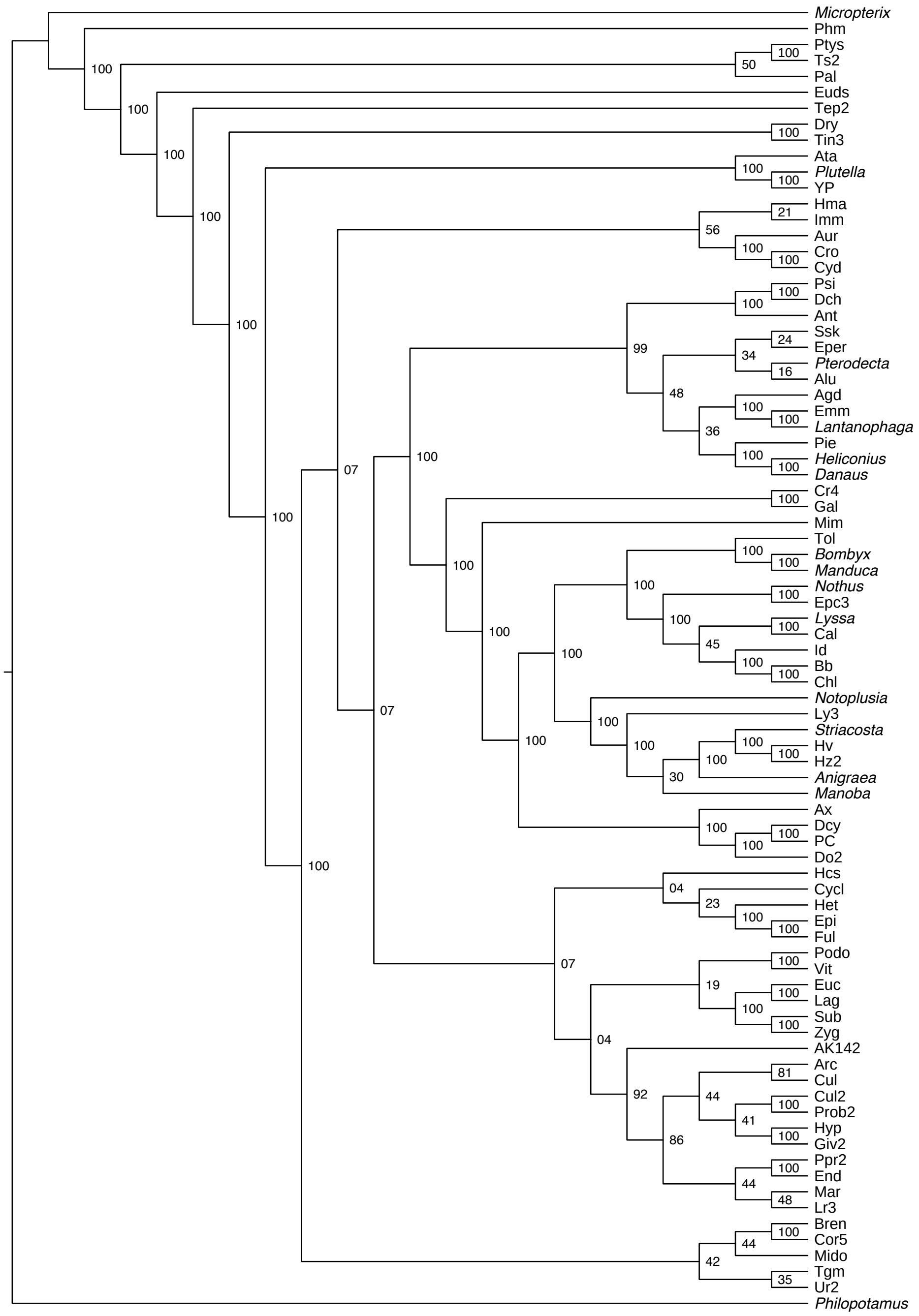


Figure 8.4: Result of the 82-taxon, 3,810 concatenated nuclear gene `degen1` analysis.

nucleotide model; and (3) `nt123`, REPRESENTATIVE, codon model. We ran 100 best tree searches and 1,000 bootstrap replicates. The searches finished quickly and were relatively "easy" according to our search statistic (Section 4.5.1). The groupings returned were not similar to those we had grown accustomed to seeing from nuclear gene analyses, and the bootstrap values were poor (Figure 8.5). We were not surprised that mt data on its own failed to resolve this problematic region of the lepidopteran phylogeny.

82-taxon mt gene analyses

We initially ran two protein-coding mt gene GARLI analyses using all 82 taxa (see Table 8.8 for data matrix properties); the analyses had the following combinations of attributes: (1) `degen1`, CONSENSUS, nucleotide model; and (2) `nt123`, CONSENSUS, codon model. We used the GARLI web service (Chapter 4) to perform an adaptive best tree search and 2,000 bootstrap replicates for each analysis. The results for both the nucleotide and codon model analyses were similar (Figure 8.5): the trees, overall, had fairly weak bootstrap support, although there was good support for a couple of relatively deep nodes and for a handful of relatively shallow groupings. Again, we were not surprised that mt data on its own failed to resolve this problematic region of the lepidopteran phylogeny.

We noticed that two taxa, *Micropterix calthella* and *Philopotamus ludificatus*, had significantly less data than other taxa (the number of non-gap characters in these two sequences was 3,846 and 7,794, respectively, whereas the average number of non-gap characters for other taxa in these alignments was over 10,000 characters). We

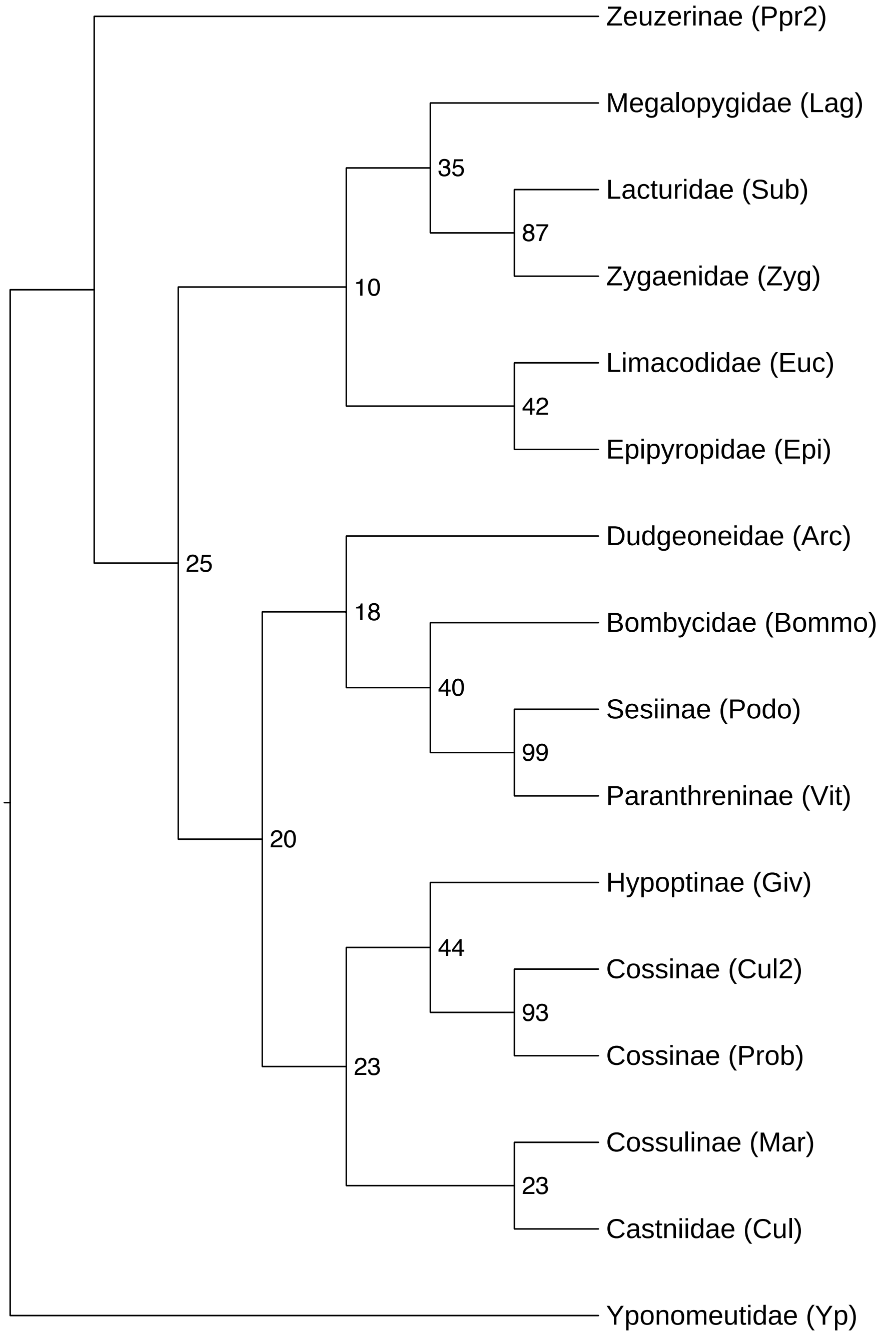


Figure 8.5: Result of the 16-taxon, 13 concatenated protein-coding mt gene, `degen1`, CONSENSUS analysis.

hypothesized that the presence of these taxa might be depressing bootstrap support, so we removed these two taxa and repeated the `degen1` nucleotide model analysis. On the whole, we observed a small but positive effect on BP values; in particular, there was now strong support for one additional relatively deep split. The overall result was not significantly changed, however.

8.10.2 Gene tree summary results

8.10.2.1 16-taxon gene tree summary tests

To test the various gene tree summary methods, we computed gene trees using GARLI for the 3,810 16-taxon nuclear gene, post-CONSENSUS `nt123` alignments; thus, each alignment had at most 16 sequences, which made the downstream gene tree summary analyses computationally tractable. Another option would have been to use the *pre*-CONSENSUS alignments, but aside from the fact that these analyses would have been difficult to complete because of scaling issues associated with large numbers of sequences, the additional sequences would probably not have been helpful: the 3,810 orthologous groups had already passed the PhyloTreePruner filter, and thus multiple sequences per taxon were already guaranteed to be monophyletic. These extra sequences would therefore not have been likely to contribute much, if any, additional information to the gene tree reconciliation process.

Each gene tree was rooted by an outgroup (*Yponomeuta multipunctella*, for the majority of alignments; if *Yponomeuta* did not exist, then *Bombyx mori* was used), and polytomies were arbitrarily resolved. These steps were accomplished

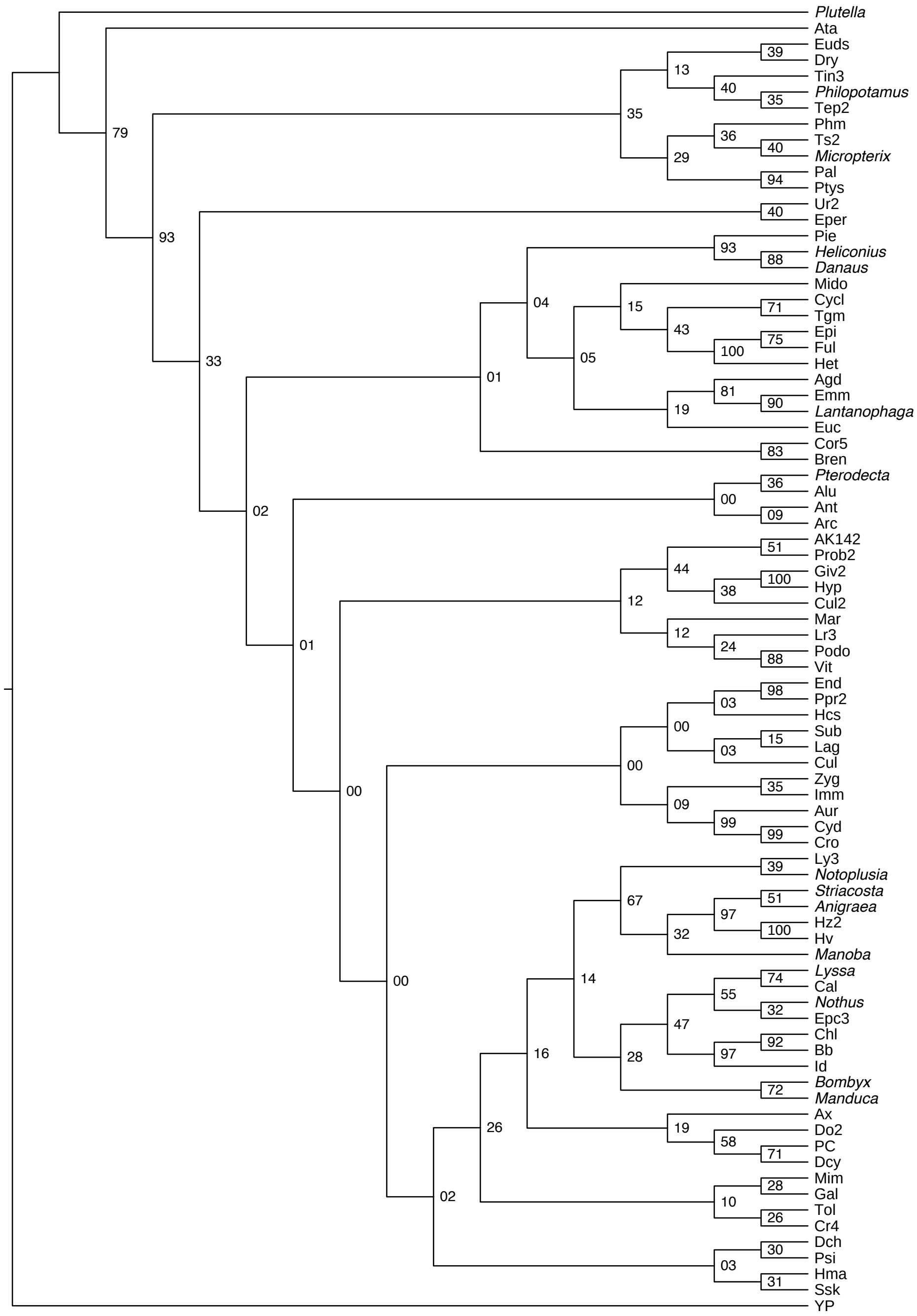


Figure 8.6: Result of the 82-taxon, 13 concatenated protein-coding mt gene, `nt123`, CONSENSUS, codon model analysis.

using DendroPy [208].

We computed a BestML tree using MP-EST [312], NJst [313], RTC [314], STAR [315], and STEAC [315]. The STAR tree matched the topologies from concatenation quite well with only a couple of minor differences. The STEAC tree was also quite similar. The MP-EST, NJst, and RTC tree topologies, on the other hand, were significantly different from the topologies from concatenation.

We also computed 100 MLBS trees using MP-EST, STAR, STEAC, and NJst. Thereafter, we plotted bootstrap support (BP) values onto the BestML tree, and we also computed an extended majority-rule (EMR) consensus of the 100 MLBS trees. We found that MP-EST and STAR produced the highest BP values, so these methods seemed the most promising.

8.10.2.2 82-taxon gene tree summary analyses

As before, we used GARLI to compute gene trees for the 3,810 82-taxon nuclear gene, post-CONSENSUS `nt123` alignments; thus, each alignment had at most 82 sequences.

Each gene tree was rooted by an outgroup in the following order of preference (when the outgroup taxon was present): *Philopotamus ludificatus*, *Micropterix calthella*, *Phymatopus californicus* ("Phm"), *Palaephatus luteolus* ("Pal"), *Ptyssoptera sp.* ("Ptys"), Tischerioidea ("Ts2"), *Eudarcia simulaticella* ("Euds"), *Thyridopteryx ephemeraeformis* ("Tep2"), *Dryadaula sp.* ("Dry"), *Tineola bisselliella* ("Tin3"), *Atteva aurea* ("Ata"), *Yponomeuta multipunctella* ("Yp"), and *Plutella* *xylostella*. Only 14 of the 3,810 gene trees did not contain one of these outgroup taxa; in these cases the trees were midpoint-rooted. In addition, polytomies were arbitrarily resolved for all trees. These steps were accomplished using DendroPy [208].

Based on the exploratory 16-taxon results (Section 8.10.2.1), we decided only to perform mp-est and star analyses. With each method we computed a BestML tree and 100 mlbs trees. We plotted bootstrap support (bp) values onto the BestML tree, and we also computed an extended majority-rule (emr) consensus of the 100 mlbs trees.

The star trees had somewhat stronger bootstrap support than the trees from concatenation, with only 12/81 nodes having bp $< 60\%$ (Figure 8.7). However, some parts of the tree had much weaker support than the trees from concatenation; e.g., the backbone within Obtectomera. A few groupings were very surprising given all evidence to date, particularly the grouping of Gelechioidea with Tortridicidae, Immidae and others thought to be lower Ditrysia.

The mp-est trees resembled those from the star method. As in the star trees, 12/81 nodes were weakly supported, in contrast to > 20 nodes for the trees from concatenation. As in the star trees, some parts of the tree had much weaker support than the trees from concatenation; e.g., the backbone within Obtectomera. Once again there were a few groupings that were very surprising given all evidence to date, particularly the exclusion of Pterophoridae from Obtectomera. In addition, bootstrap support was surprisingly weak for some superfamilies, such as Bombycoidea and Pterophoridae.

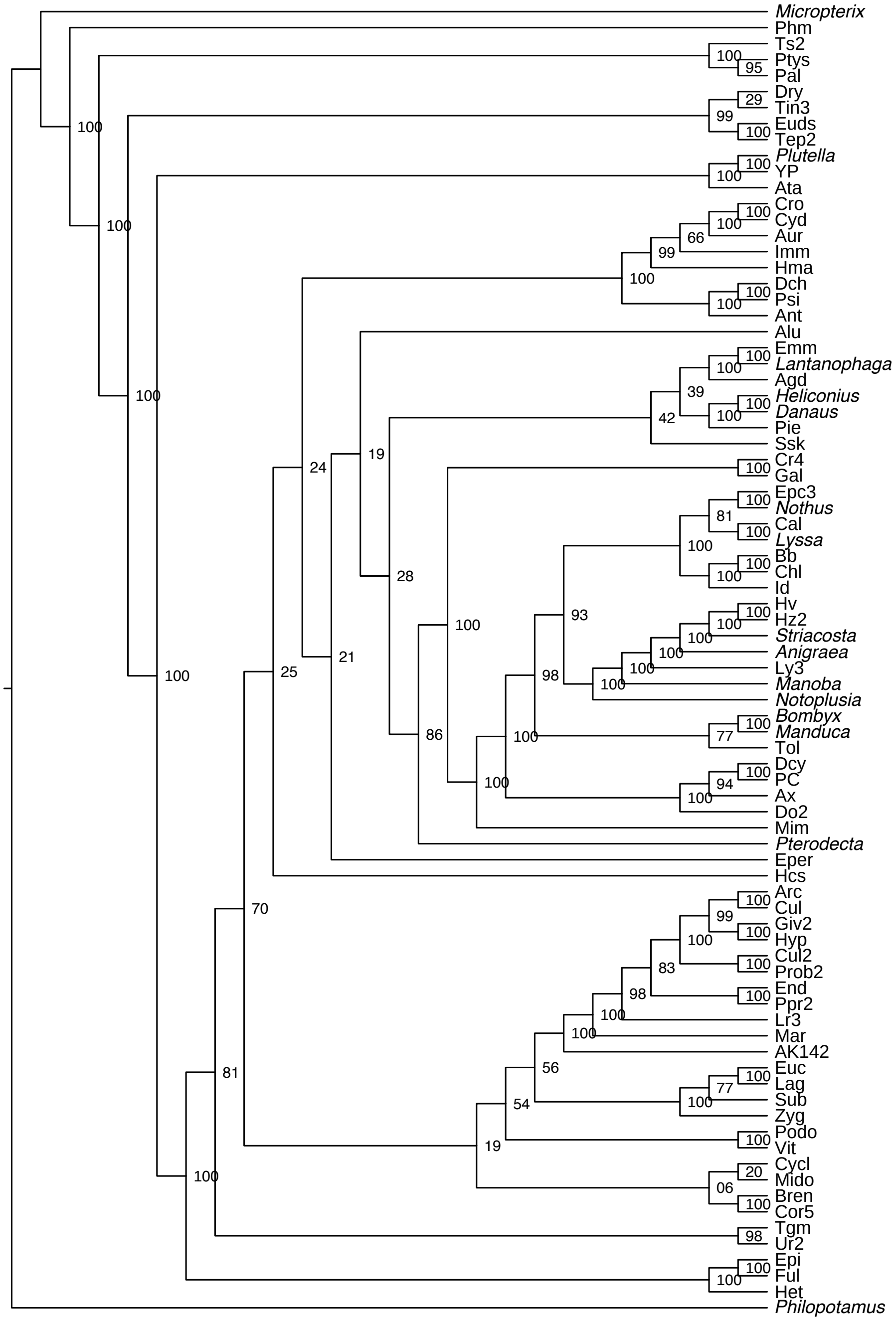


Figure 8.7: Result of the 82-taxon, 3,810 nuclear gene STAR gene tree summary analysis.

8.10.3 SVDquartets results

We analyzed the 82-taxon, 3,810-nuclear gene `nt123` matrix using the implementation of SVDquartets in PAUP* (see Table 8.6 for data matrix properties). We analyzed all possible taxon quartets, and performed 100 bootstrap replicates. The constraint tree was not applied to this analysis because this option was not yet supported in PAUP*.

The overall strength of the SVDquartets tree resolution (Figure 8.8) was intermediate between that of the previous 82-taxon concatenation analyses, and the 82-taxon gene tree summary analyses. The total number of weakly-supported nodes (BP $< 60\%$) was 19 for the SVDquartets tree; the same number for the previous gene tree summary analyses was 12, and for the `degen1` concatenated analysis, it was 24.

As seen with the STAR tree previously (Figure 8.7), some parts of the SVDquartets tree (Figure 8.8) had much weaker support than corresponding parts in the trees built from concatenated data sets (e.g., the backbone within Obtectomera). Furthermore, to an even greater extent than the STAR tree, there were groupings that would be very surprising given all previous evidence to date, including groups that disagreed with the constraint tree that we used for the concatenated analyses. Examples included non-monophyly of the superfamilies Pyraloidea and Geometroidea, and of the major group Obtectomera.

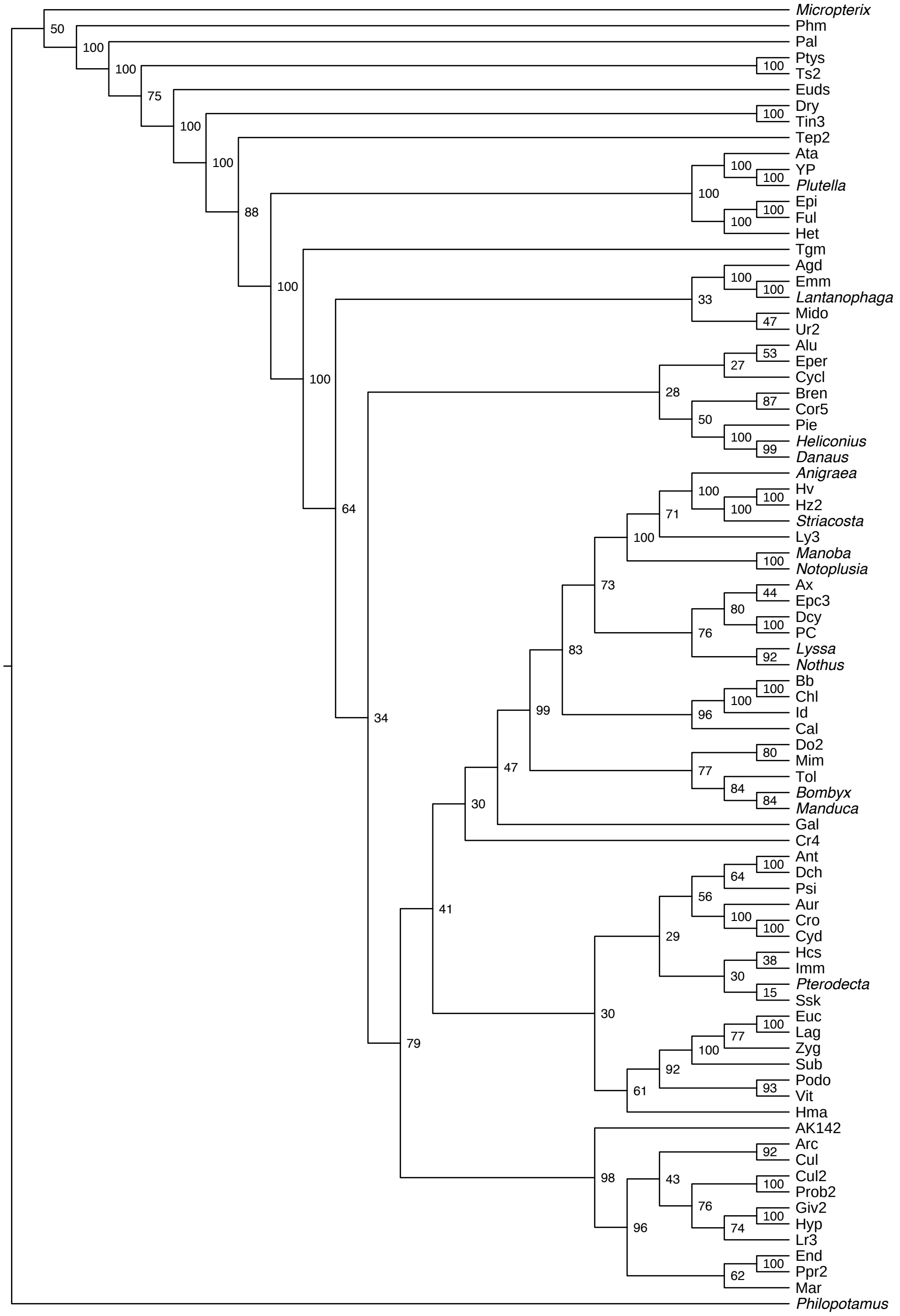


Figure 8.8: Result of the 82-taxon, 3,810 nuclear gene SVDquartets analysis.

8.10.4 23-taxon nuclear gene analysis

Up to this point, none of the analysis results showed a clear improvement over the results from the previous study (Chapter 7). To test new methodologies more rapidly, we created a 23-taxon subset of the 82 taxa that adequately represented the regions of the phylogeny that were still not well resolved. We ran one 3,810-nuclear gene GARLI analysis using the 23-taxon data set (see Table 8.7 for data matrix properties); novel to this analysis, we filtered the individual orthologous group alignments with PYGOT [304] and GUIDANCE2 [305] to remove dubiously-aligned regions. The analysis had the following combination of attributes: `degen1`, CONSENSUS, nucleotide model.

The 23-taxon, 3,810-gene phylogenetic result (Figure 8.9) was based on 10 best tree search replicates and 40 bootstrap replicates.

The majority of the tree was very strongly supported, probably owing to the alignment filtering procedures that were used.

8.11 Summary and conclusions

This study explored the potential of RNA sequencing to conclusively resolve relationships among the superfamilies of advanced ditrysian Lepidoptera (Apoditrysia). The problem remained mostly unresolved despite increased taxon sampling, use of Lepidoptera-specific databases for orthology determination, and application of a variety of phylogenetic analysis methods that made heavy use of our advanced computational infrastructure. One recent result that used a more nimble taxon set

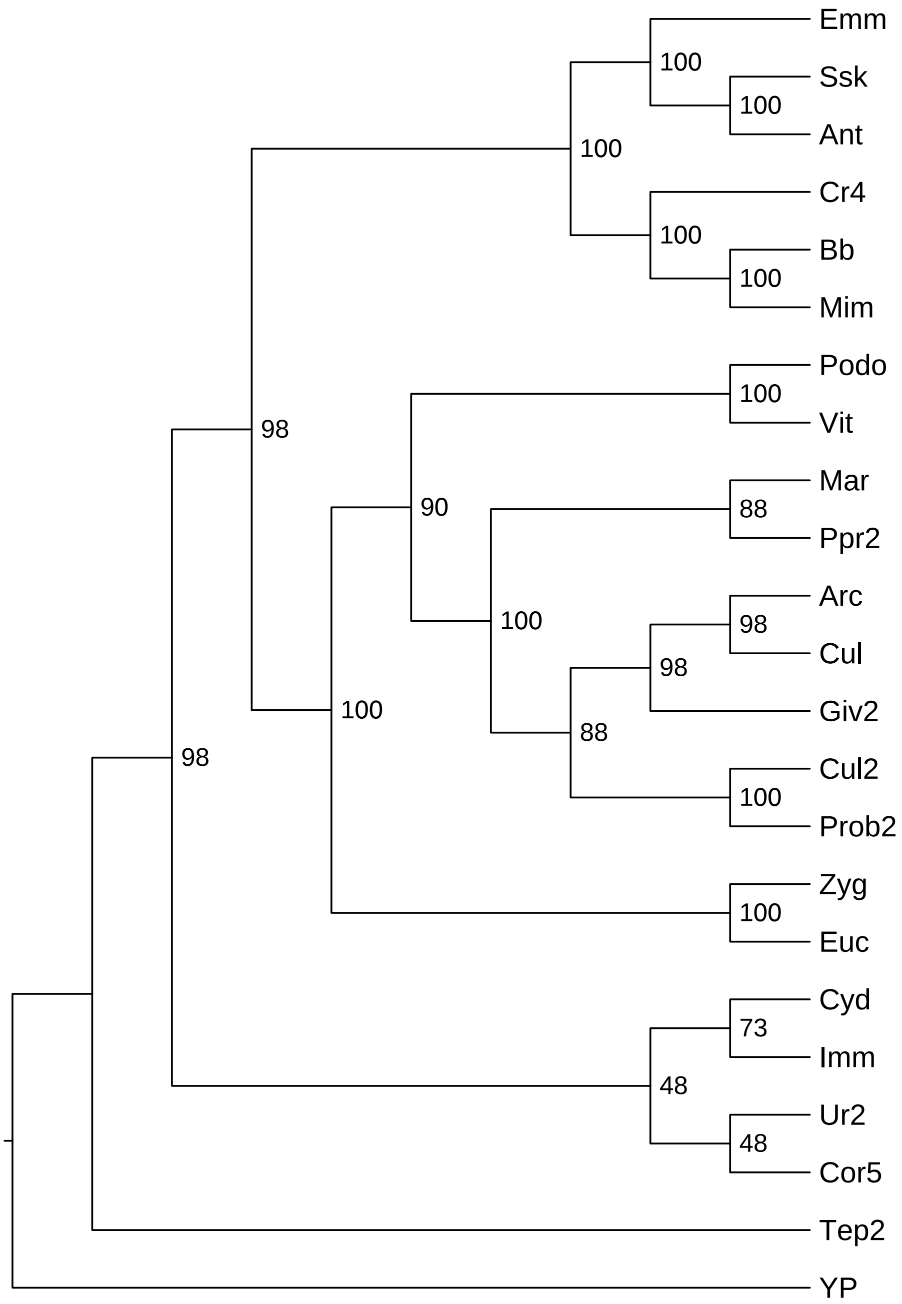


Figure 8.9: Result of the 23-taxon, 3,810 concatenated nuclear gene `degen1` analysis.

and some new alignment filtering procedures did show promise (Section 8.10.4). We are hopeful that this methodology also works well when applied to the full 82-taxon data set. We are also investigating the use of other maximum likelihood phylogenetic inference programs such as RAxML [318].

Chapter 9: Identifying insect endosymbiont sequences in host-derived RNA-Seq data

9.1 Background

The goal of this study was to putatively identify endosymbiont sequences present in RNA-Seq data derived from insect hosts. If successful, one might eventually use such data to analyze patterns of host-parasite coevolution, which might involve co-speciation, host-switching, host range expansion or contraction, or host biogeography more generally. Initially we focused our search on Microsporidia, a fungal parasite, and then expanded our search to include some bacterial endosymbionts.

9.1.1 Microsporidia

Microsporidia are unicellular fungal parasites belonging to the phylum Microspora. They are spore-forming, obligate, intracellular parasites that attack both vertebrates and invertebrates; in particular, they are often found to infect insects, fish, and mammals. Microsporidia are widespread throughout nature with over 1,200 identified species, several of which are known to infect humans with compromised immune systems. Of particular interest to us, however, is that Microsporidia species are known to have an endosymbiotic relationship with Lepidoptera (an insect order comprising moths and butterflies), a group for which we had already acquired and analyzed a substantial amount of RNA-Seq data (Chapters 7 and 8).

9.1.2 Bacterial endosymbionts

A considerable proportion of insect species (and indeed, arthropods more generally) are infected with bacterial secondary symbionts [319]. In this study, we searched transcriptome data from insect hosts for evidence supporting the presence of the following bacterial endosymbionts: *Arsenophonus*, *Blochmannia*, *Buchnera*, *Cardinium*, *Rickettsia*, *Spiroplasma*, and *Wolbachia*.

9.2 Methods

9.2.1 Insect transcriptome data

Our query data consisted of 98 transcriptome data sets generated via RNA sequencing (RNA-Seq). The majority of host taxa were from the order Lepidoptera, but also included 16 additional insect orders (Coleoptera, Diplura, Diptera, Hemiptera, Homoptera, Hymenoptera, Mecoptera, Megaloptera, Neuroptera, Orthoptera, Phasmatodea, Phthiraptera, Raphidioptera, Siphonaptera, Strepsiptera, and Trichoptera). Our group generated many of the RNA-Seq samples *de novo* (Tables 7.1 and 8.1); other samples were obtained from other studies and projects [25,30,33,179] (Tables 8.3, 9.1, and 9.2). All samples were put through our most recent quality control and assembly pipeline (Section 8.4); for this study, we used only the Trinity [44] assembly (except in the case of *Striacosta albicosta*, for which we also used the Trans-ABySS [45, 266] assembly). Summary statistics for publicly-available transcriptome data reassembled specifically for this study are shown in Tables 9.1 and 9.2. We note that many more publicly-available transcriptome data sets remain available to be analyzed.

9.2.2 Sequence database construction

We downloaded all protein records for Microsporidia from UniProt [320], and Microsporidia sequence cluster data from UniRef [321] at the minimum 50% identity level, which resulted in 46,550 microsporidian sequences distributed among 28,436 clusters (June 2014). (There were 20,585 clusters that contained only one microsporidian sequence; we call these “singleton clusters”, or “singletons”.) We removed all non-microsporidian sequences from the clusters. These steps were accomplished using a combination of the UniProt web site (`uniprot.org`), custom Perl scripts, and NCBI E-utilities [322].

We downloaded all protein records for Lepidoptera from UniProt (216,552 sequences; June 2014) and combined these with the 46,550 previously retrieved microsporidian sequences to construct a Lepidoptera/Microsporidia (i.e., host/endosymbiont) sequence database. We used this database with the `fasty` program from the FASTA package [303] to screen transcriptomes from Lepidoptera and other insects for putative Microsporidia sequences. We only considered transcript

Taxon	Reads	Read length	After autoadapt	Transcript fragments	N50
Archaeopsylla erinacei	108,318,120	PE100	107,281,714	59,116	954
Carabus granulatus	122,815,238	PE100	121,679,250	81,072	1,050
Corydalinae sp.	109,799,166	PE100	108,228,826	130,505	606
Mengenilla moldrzyki	213,761,176	PE100	210,807,944	131,504	830
Nannochorista sp.	96,580,760	PE100	95,508,256	84,795	1,081
Nevrorthus apatelios	58,121,176	PE100	54,450,310	32,079	655
Priacma serrata	75,981,168	PE100	71,776,284	41,582	585
Raphidia ariadne	67,920,858	PE100	65,216,538	50,983	774
Sialis lutaria	74,560,668	PE100	70,018,090	24,301	654
Tipula maxima	83,873,478	PE100	79,560,064	42,124	824
Xyela alpigena	67,675,150	PE100	64,234,772	18,378	583

Table 9.1: Summary statistics for RNA-seq data sets from Peters et al. [33] reassembled with Trinity [44].

Taxon	Reads	Read length	After autoadapt	Transcript fragments	N50
Dichochrysa prasina	22,090,140	PE150	22,089,850	119,033	823
Essigella californica	21,640,554	PE150	21,640,212	119,698	780
Meloe violaceus	21,657,976	PE150	21,657,490	40,119	1,246
Menopon gallinae	16,768,868	PE150	16,768,498	52,425	1,917
Occasjapyx japonicus	15,362,662	PE150	15,362,136	61,813	1,394
Okanagana villosa	23,498,120	PE150	23,497,568	113,005	873
Peruphasma schultei	23,480,956	PE150	23,480,362	117,450	752
Platycentropus radiatus	12,712,138	PE150	12,711,986	58,875	698
Prosarthria teretrirostris	23,713,208	PE150	23,712,838	93,573	635
Triodia sylvina	21,885,798	PE150	21,885,482	90,735	935

Table 9.2: Summary statistics for RNA-seq data sets from Misof et al. [30] reassembled with Trinity [44].

sequences for further analysis whose most significant hit was to a microsporidian database sequence with an expectation value less than or equal to 1e-10.

We downloaded the complete UniProt database (Swiss-Prot and trEMBL; 91,408,504 sequences; February 2015), which we used for comprehensive screening of insect transcriptomes for protein-coding sequences from Microsporidia as well as bacterial endosymbionts. Table 9.3 shows the number of sequences in the UniProt database associated with each endosymbiont of interest. Because this database was very large, we searched it with the BLASTX-like algorithm in DIAMOND [323], a fast and sensitive alignment program.

We downloaded the complete RNAcentral database [324] (8,102,559 sequences; February 2015), which we used for comprehensive screening of insect transcriptomes for ribosomal and other types of non-coding RNA from Microsporidia as well as bacterial endosymbionts. Table 9.3 shows the number of sequences in the RNAcentral database associated with each endosymbiont of interest. Because this database was very large, we searched it with the BLASTN-like algorithm in Lambda [325], a fast and accurate alignment program.

9.2.3 Multiple sequence alignments

We aligned all non-singleton Microsporidia clusters with the `einsi` algorithm in MAFFT [276], which yielded 7,851 minimum 50% identity alignments.

9.2.4 The MICRO50 custom amino acid substitution matrix

Following the methods described in Appendix B, we constructed a custom Microsporidia amino acid substitution matrix (MICRO50) using the minimum 50% identity alignments as input. The MICRO50 matrix had an entropy value of 0.7529, and an expected value of -0.5115 (compare to other substitution matrices in Table B.1). We used the MICRO50 matrix with Microsporidia-focused database searches (Sections 9.2.2 and 9.2.5), and with multiple alignments during data matrix construction (Section 9.2.5.1).

9.2.5 Microsporidia HaMStR database

We constructed a HaMStR (version 13.2.2; [272]) database of 4,526 amino acid gene models based on the minimum 50% identity Microsporidia alignments. HaMStR requires the use of reference taxa; for practical as well as theoretical reasons, it is desirable to have a relatively small number of reference taxa that are widely represented among the HaMStR database gene models. Thus, we calculated the distribution of Microsporidia species among gene clusters, and chose eight insect-associated Microsporidia taxa with maximal representation among clusters to use as reference taxa. These included Microsporidia species associated with the silk moth, grasshoppers, mosquitos, and honey bees (Table 9.4). Each gene model in the database needed to contain at least one sequence from a reference taxon, so this only allowed us to use 4,530 of the (originally 7,851) minimum 50% identity alignments. We removed four clusters that might have been associated with transposable elements, as their sequence identifiers contained the strings “transpos”, “Pol protein”, “Pol polyprotein”, or “Gag-pol polyprotein”, which left 4,526 clusters for use. Finally, for each microsporidian reference taxon, we constructed a BLAST [13] database for use with HaMStR that contained all taxon-associated UniProt protein records (Table 9.4).

9.2.5.1 Data matrix construction

Transcriptome data sets selected for further analysis were searched against the Microsporidia HaMStR database (Section 9.2.5), and matching sequences were assigned to one of 4,526 orthologous sequence groups. For each orthologous group, the amino acid sequences of the database reference taxa and the query taxa were aligned using the `linsi` algorithm in MAFFT [276] and the custom MICRO50 substitution matrix. The alignments were concatenated, adding gaps for missing sequences as necessary, and sites with sequence representation in fewer than four taxa were removed.

9.2.6 Maximum likelihood phylogenetic analysis

Maximum likelihood phylogenetic analysis used GARLI (Genetic Algorithm for Rapid Likelihood Inference; version 2.1; [2]) and grid computing [48, 223] via a web service at `molecularevolution.org` [53] based on tools developed by Bazinet et al. [284] that include post-processing with DendroPy [208], R [285], and custom Perl scripts. We used GARLI default settings, including stepwise addition starting trees,

Taxon	NCBI taxon ID	UniProt sequences	RNAcentral sequences
Arsenophonus	637	3,528	358
Blochmannia	203804	2,539	300
Buchnera	32199	10,684	678
Cardinium	273135	1,686	548
Lepidoptera	7088	219,537	58,230
Microsporidia	6029	60,651	4,448
Spiroplasma	2132	12,877	1,147
Rickettsia	780	59,376	1,408
Wolbachia	953	26,718	1,615

Table 9.3: The number of endosymbiont-associated sequences in our UniProt and RNAcentral databases (downloaded February 2015). The number of Lepidoptera sequences is also shown.

Microsporidia species	NCBI taxon ID	Host insect	Sequences in HaMStR DB	UniProt sequences in BLAST DB
Edhazardia aedis USNM 41457	1003232	mosquito	341	4,208
Encephalitozoon romaleae SJ-2008	1178016	grasshopper	1,746	1,826
Nosema apis BRL 01	1037528	honey bee	852	2,727
Nosema bombycis	27978	*Bombyx mori* (moth)	197	216
Nosema bombycis CQ1	578461	*Bombyx mori* (moth)	1,968	4,399
Nosema ceranae	40302	honey bee	193	196
Nosema ceranae BRL01	578460	honey bee	726	2,060
Vavraia culicis subsp. floridensis	948595	mosquito	1,564	2,768

Table 9.4: The eight reference taxa in the Microsporidia HaMStR database.

except that we lowered the number of successive generations yielding no improvement in likelihood score that prompts termination (`genthreshfortopoterm=5000`), as we found that this saved time and yielded comparable results. The Microsporidia analyses used the WAG amino acid model; the *Rickettsia* analysis (Section 9.3.4) used the GTR nucleotide model. The best tree for the Microsporidia gene tree analyses (Section 9.3.1.3) was chosen from the better of two search replicates, midpoint-rooted, and visually inspected. The best tree for the Microsporidia concatenated-gene analysis (Section 9.3.1.3) and the *Rickettsia* gene tree analysis (Section 9.3.4) was found using an adaptive search [53]; for these analyses we also performed 1,000 bootstrap replicates. We used DendroPy [208] to generate 50% majority-rule bootstrap consensus trees.

9.3 Results

9.3.1 Microsporidia analyses

9.3.1.1 Interrogating transcriptome data for Microsporidia

We used `fasty` to search all 98 insect transcriptome data sets against the Lepidoptera/Microsporidia database (described in Section 9.2.2). On average, only a very small proportion of sequences (≈ 0.006) were significant hits to Microsporidia (E-value $\leq$ 1e-10). Upon examining the distribution of these proportions (Figure 9.1), we decided empirically to call a sample "positive" for Microsporidia if its proportion of significant hits assigned to Microsporidia was ≥ 0.01. In total, 12 out of 98 samples tested positive for Microsporidia (Table 9.5).

9.3.1.2 HaMStR results

We ran the putative Microsporidia sequences from 9 of the 12 positive samples[1] (Section 9.3.1.1) against our Microsporidia HaMStR database, which organized matching sequences into orthologous sequence groups. Statistics on sequence matches determined by HaMStR are given in Table 9.6.

9.3.1.3 Phylogenetic results

Gene tree analyses

We performed some preliminary gene tree analyses using Microsporidia sequences from five Microsporidia-positive transcriptomes ("query taxa"), together with sequences from the eight microsporidian reference database taxa. We selected for analysis the 40 orthologous groups in which all five query taxa were represented. The results were encouraging; we observed sensible groupings and "dispersion" of query taxa among reference taxa (Figure 9.2). In contrast, if our query sequences were actually lepidopteran in origin, they would have most likely formed their own clade when analyzed with a diversity of true Microsporidia sequences (which we indeed observed in earlier work in which we did not select microsporidian sequences carefully enough).

One interesting relationship we observed was that sequences of Microsporidia from *Striacosta albicosta* (a query taxon) grouped closely with sequences of Mi-

[1] The other three positive samples were unavailable at this point.

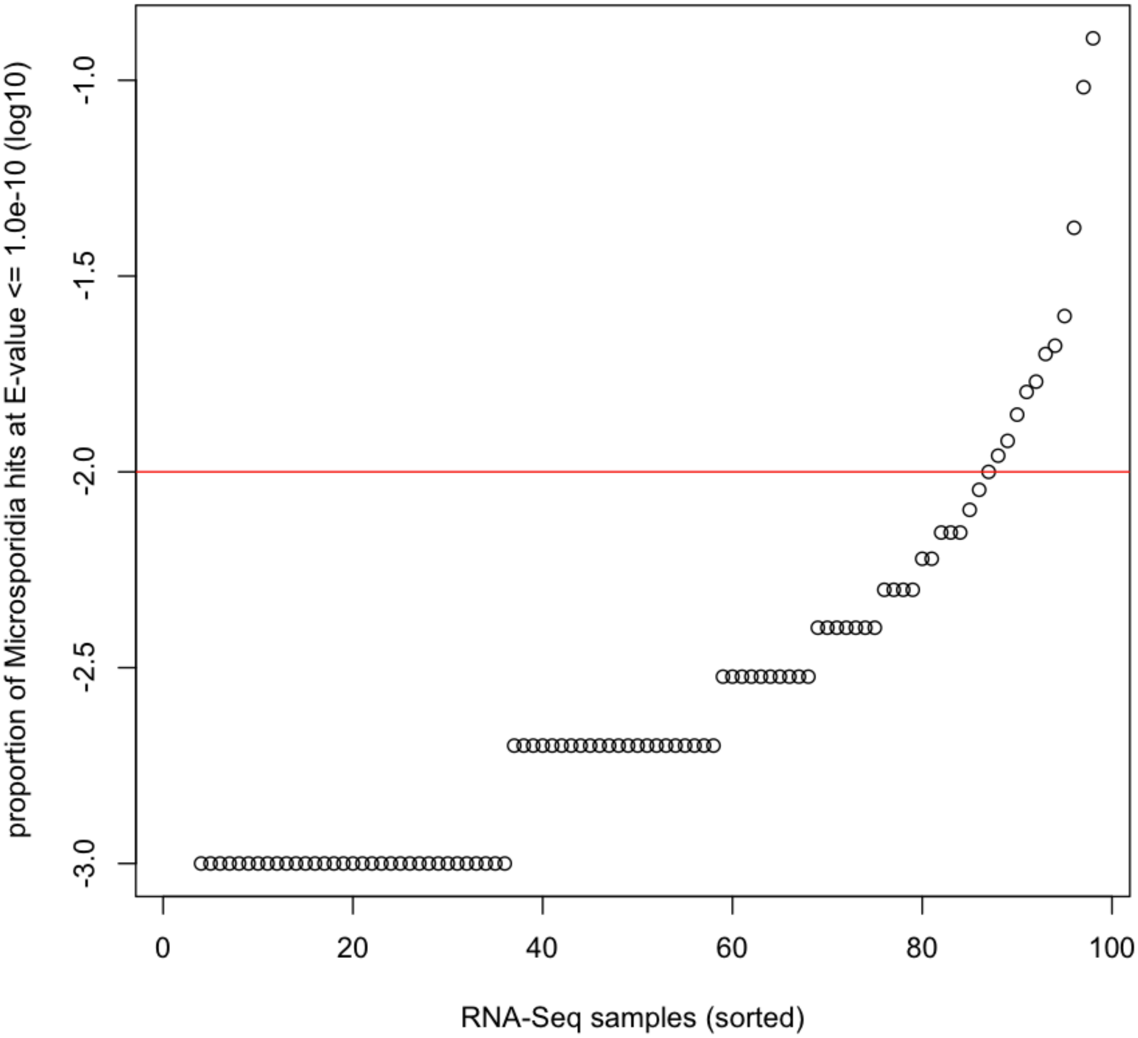


Figure 9.1: Scatterplot of the proportion of Microsporidia hits in 98 RNA-Seq samples. The red line at $y = \log_{10}(0.01)$ represents our chosen cutoff for calling a sample "positive" for Microsporidia.

RNA-Seq taxon	Transcript fragments	Microsporidia hits	Lepidoptera hits	Proportion of Microsporidia hits
Striacosta albicosta	336,829	14,925	101,471	0.128
Philopotamus ludificatus	34,088	1,316	12,350	0.096
Tineola bisselliella	40,696	865	19,639	0.042
Eudarcia simulatricella	76,663	525	20,600	0.025
Mengenilla moldrzyki	131,504	563	26,338	0.021
Corydalinae sp.	130,505	671	32,989	0.020
Micropterix calthella	192,011	549	32,177	0.017
Okanagana villosa	113,005	360	22,376	0.016
Xyela alpigena	59,116	66	4,642	0.014
Peruphasma schultei	117,450	214	17,105	0.012
Dichochrysa prasina	119,033	317	29,513	0.011
Carabus granulatus	81,072	236	24,479	0.010

Table 9.5: RNA-Seq samples that tested positive for Microsporidia, sorted by proportion of Microsporidia hits.

RNA-Seq taxon	Matches identified by HaMStR	Matches to unique loci	Average length of sequence match[1]
Striacosta albicosta	4,536	1,587	120
Philopotamus ludificatus	1,502	1,462	171
Tineola bisselliella	1,152	1,143	138
Eudarcia simulatricella	567	560	136
Mengenilla moldrzyki	160	134	170
Corydalinae sp.	340	303	140
Micropterix calthella	343	305	190
Xyela alpigena	37	37	109
Carabus granulatus	87	80	156

[1] Length is given in amino acids.

Table 9.6: HaMStR statistics for samples that tested positive for Microsporidia.

crosporidia from *Bombyx mori* (a database reference taxon); *Striacosta* and *Bombyx* are closely related species of Lepidoptera. This might suggest a case of host-endosymbiont co-speciation, or at least that the Microsporidia species inhabiting *Striacosta* and *Bombyx* are very closely related.

Concatenated-gene phylogenetic analysis

We performed a concatenated-gene phylogenetic analysis using Microsporidia sequences from nine Microsporidia-positive transcriptomes (“query taxa”), together with sequences from the eight microsporidian database reference taxa. The 4,526-gene data matrix was constructed according to the methods described in Section 9.2.5.1. The final matrix was 170,055 amino acids in length, and was 30.3% complete (69.7% missing data).

The concatenated-gene analysis yielded a tree with uniformly high bootstrap support (Figure 9.3). The species of Microsporidia inhabiting *Tineola* and *Striacosta* (query taxa) fell within a clade of *Nosema*, a genus of Microsporidia. Interestingly, some sister taxa in the tree belonged to entirely different insect orders. The phylogenetic patterns in these strongly-supported results merit further investigation.

9.3.2 Comprehensive database searches

9.3.2.1 Searching UniProt with DIAMOND

We used DIAMOND [323] with default settings to search all 98 insect transcriptome data sets against the UniProt database described in Section 9.2.2. We only reported one alignment per query sequence (the best hit with E-value ≤ 0.001).

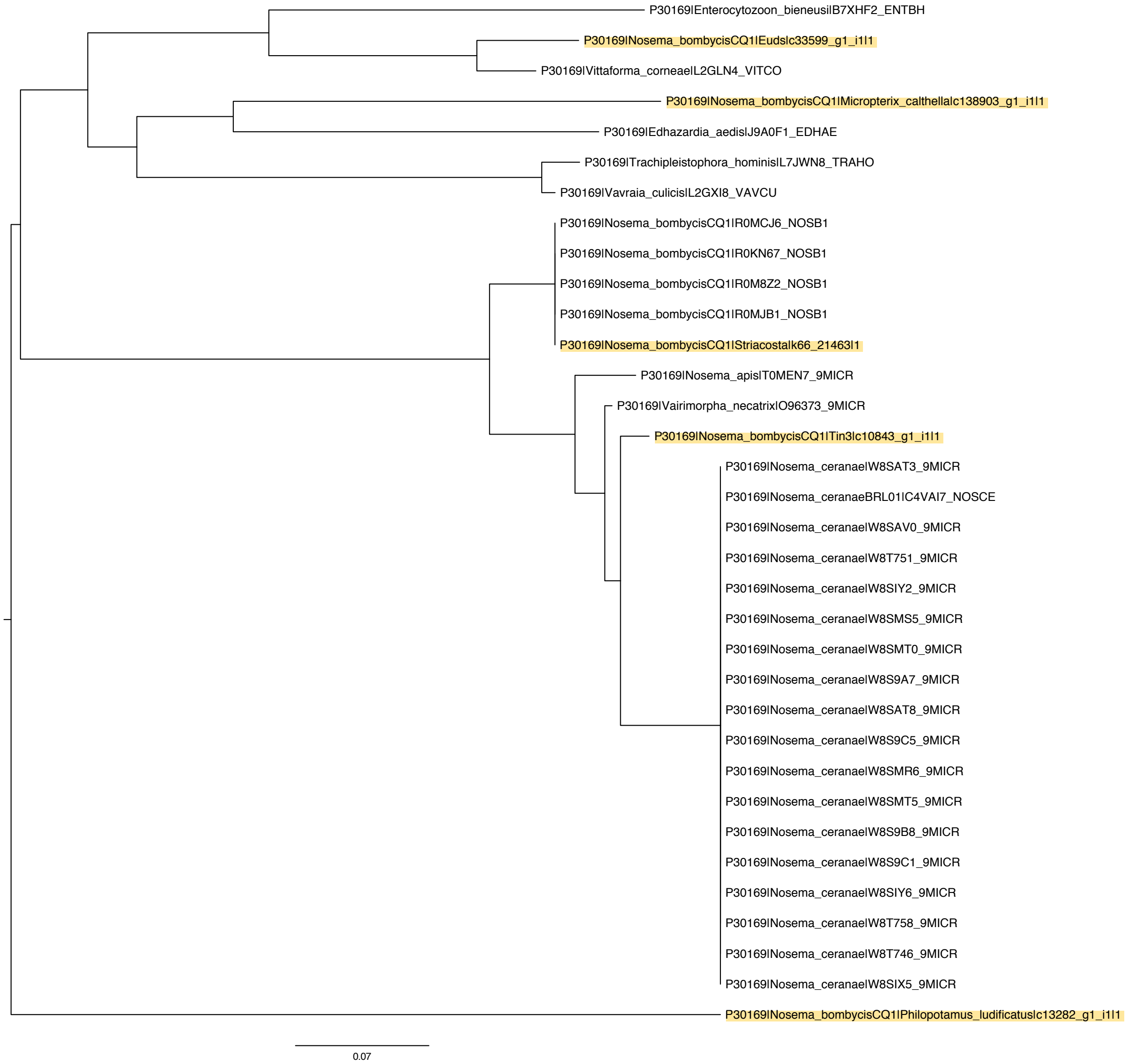

P30169|Enterocytozoon_bieneusi|B7XHF2_ENTBH
P30169|Nosema_bombycisCQ1|Euds|c33599_g1_i1|1
P30169|Vittaforma_corneae|L2GLN4_VITCO
P30169|Nosema_bombycisCQ1|Micropterix_calthella|c138903_g1_i1|1
P30169|Edhazardia_aedis|J9A0F1_EDHAE
P30169|Trachipleistophora_hominis|L7JWN8_TRAHO
P30169|Vavraia_culicis|L2GXI8_VAVCU
P30169|Nosema_bombycisCQ1|R0MCJ6_NOSB1
P30169|Nosema_bombycisCQ1|R0KN67_NOSB1
P30169|Nosema_bombycisCQ1|R0M8Z2_NOSB1
P30169|Nosema_bombycisCQ1|R0MJB1_NOSB1
P30169|Nosema_bombycisCQ1|Striacosta|k66_21463|1
P30169|Nosema_apis|T0MEN7_9MICR
P30169|Vairimorpha_necatrix|O96373_9MICR
P30169|Nosema_bombycisCQ1|Tin3|c10843_g1_i1|1
P30169|Nosema_ceranae|W8SAT3_9MICR
P30169|Nosema_ceranaeBRL01|C4VAI7_NOSCE
P30169|Nosema_ceranae|W8SAV0_9MICR
P30169|Nosema_ceranae|W8T751_9MICR
P30169|Nosema_ceranae|W8SIY2_9MICR
P30169|Nosema_ceranae|W8SMS5_9MICR
P30169|Nosema_ceranae|W8SMT0_9MICR
P30169|Nosema_ceranae|W8S9A7_9MICR
P30169|Nosema_ceranae|W8SAT8_9MICR
P30169|Nosema_ceranae|W8S9C5_9MICR
P30169|Nosema_ceranae|W8SMR6_9MICR
P30169|Nosema_ceranae|W8SMT5_9MICR
P30169|Nosema_ceranae|W8S9B8_9MICR
P30169|Nosema_ceranae|W8S9C1_9MICR
P30169|Nosema_ceranae|W8SIY6_9MICR
P30169|Nosema_ceranae|W8T758_9MICR
P30169|Nosema_ceranae|W8T746_9MICR
P30169|Nosema_ceranae|W8SIX5_9MICR
P30169|Nosema_bombycisCQ1|Philopotamus_ludificatus|c13282_g1_i1|1
0.07

Figure 9.2: Gene tree computed for Microsporidia sequences belonging to orthologous group P30169. Sequences derived from RNA-Seq are highlighted. Scale is the expected number of amino acid replacements per site.

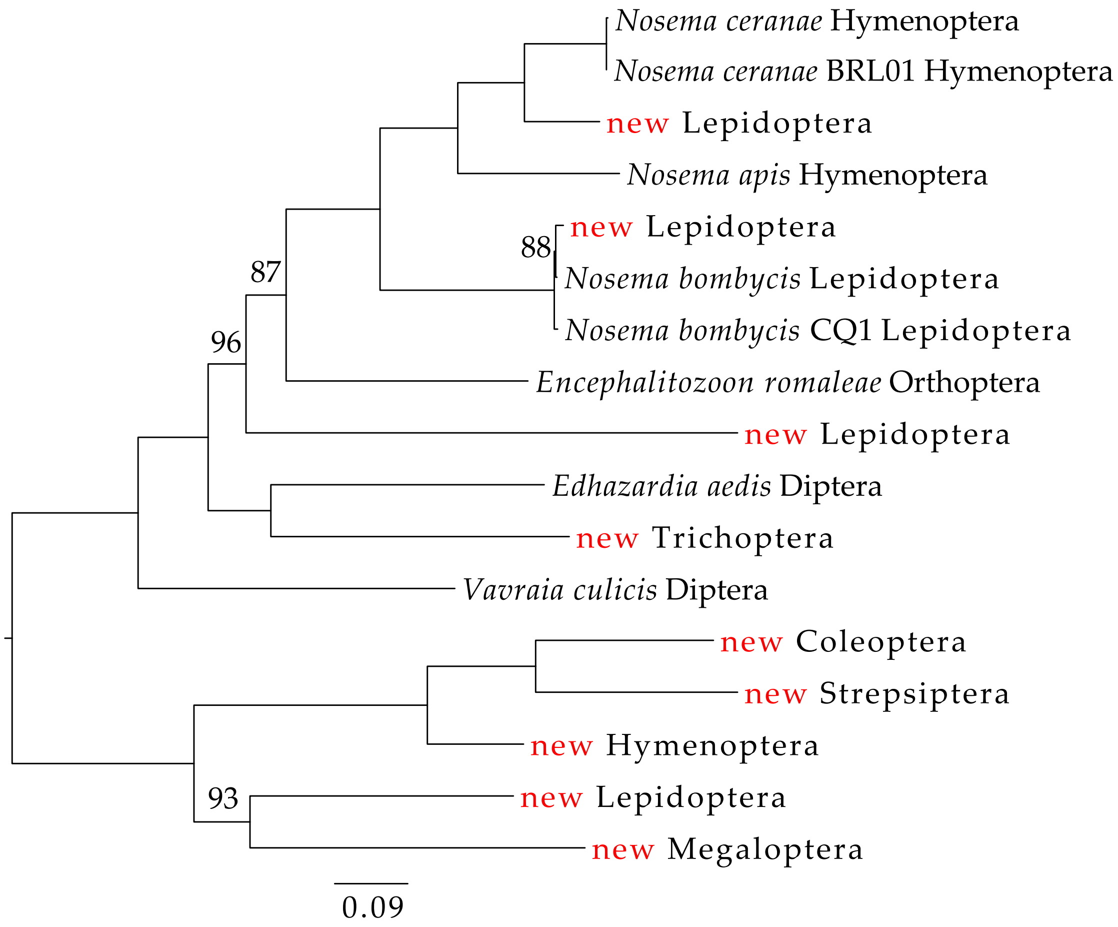


Figure 9.3: Phylogeny of Microsporidia in insect hosts based on maximum likelihood analyses of 4,526 protein-coding genes. Bootstrap values were plotted onto the best tree, which was midpoint-rooted. Tip labels: genus and species for previously recognized taxa (or new if the taxon was first found in this study), followed by the insect order to which the host belongs. All bootstrap values are 100%, except where noted. Scale is the expected number of amino acid replacements per site.

We counted the number of hits to each endosymbiont, and also calculated the proportion of query sequences that each count represented. The UniProt counts and proportions are shown for all 98 RNA-Seq samples in Figures 9.4 and 9.5, respectively. We observed that *Wolbachia* and Microsporidia were well represented in our transcriptome data, followed by *Rickettsia* and *Spiroplasma*. There were relatively few hits to other bacterial endosymbionts.

9.3.2.2 Searching RNAcentral with Lambda

We used Lambda [325] with default settings to search all 98 insect transcriptome data sets against the RNAcentral database described in Section 9.2.2. We only reported one alignment per query sequence (the best hit with E-value ≤ 0.1). We counted the number of hits to each endosymbiont, and also calculated the proportion of query sequences that each count represented. The RNAcentral counts and proportions are shown for all 98 samples in Figures 9.6 and 9.7, respectively. As with the UniProt search (Section 9.3.2.1), Microsporidia was well represented in our transcriptome data; on the other hand, *Wolbachia* was not found quite as regularly. *Spiroplasma* and *Rickettsia* occurred relatively frequently, as did *Buchnera*.

9.3.3 Assessing high-level taxonomic composition of RNA-Seq data

In order to guide future experiments, we sought to obtain a high-level assessment of the taxonomic composition of our transcriptome data. The RNAcentral database search results were likely to be more appropriate than the UniProt results

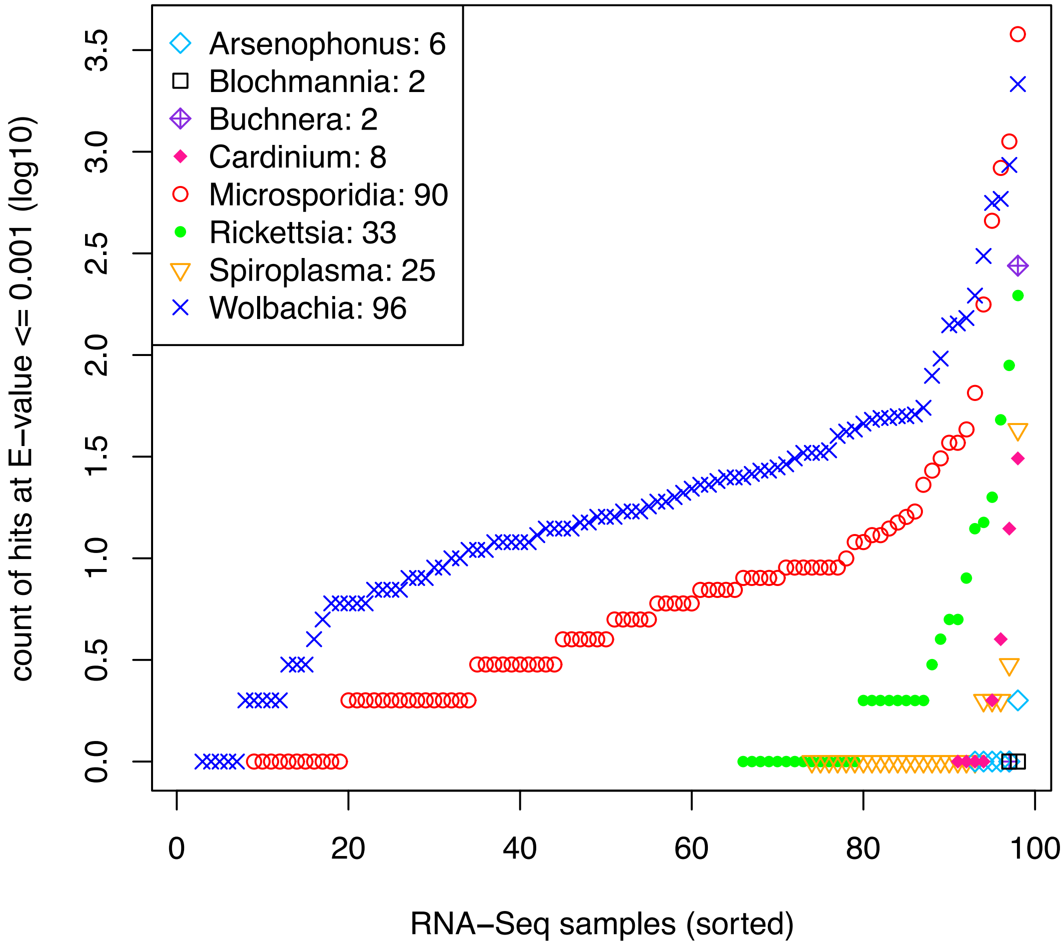


Figure 9.4: Counts of significant best hits to various endosymbiont protein sequences for all 98 RNA-Seq samples as determined by DIAMOND, shown on a log scale. The number of samples for which the count was greater than zero is given for each endosymbiont in the legend.

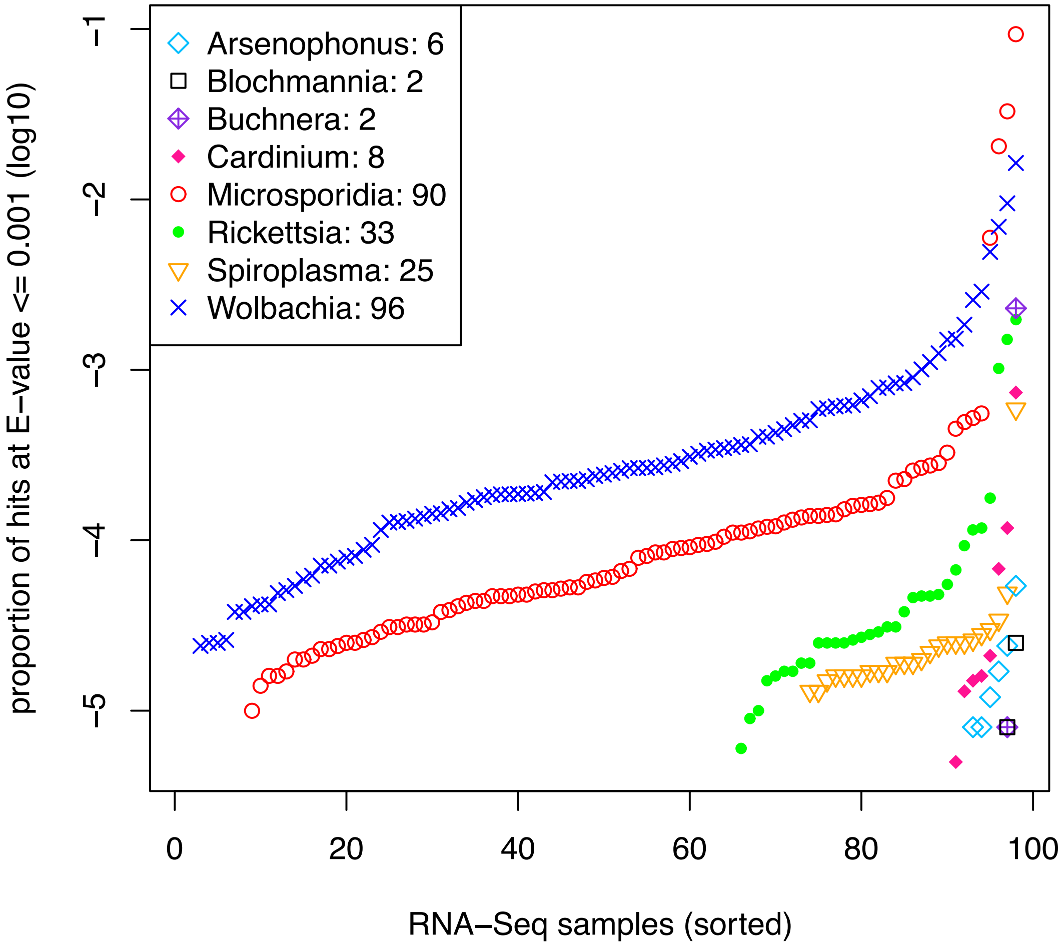


Figure 9.5: Proportions of significant best hits to various endosymbiont protein sequences for all 98 RNA-Seq samples as determined by DIAMOND, shown on a log scale. The number of samples for which the proportion was greater than zero is given for each endosymbiont in the legend.

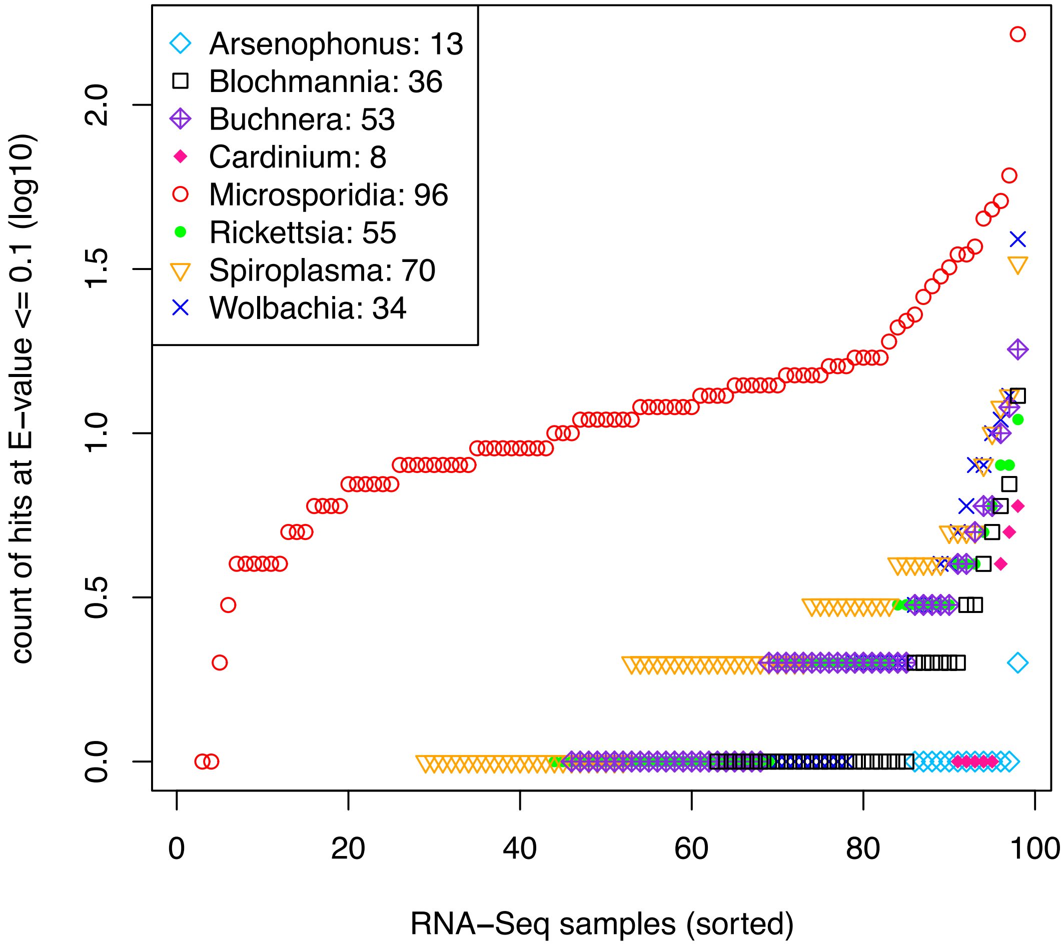


Figure 9.6: Counts of significant best hits to various endosymbiont non-coding RNA sequences for all 98 RNA-Seq samples as determined by Lambda, shown on a log scale. The number of samples for which the count was greater than zero is given for each endosymbiont in the legend.

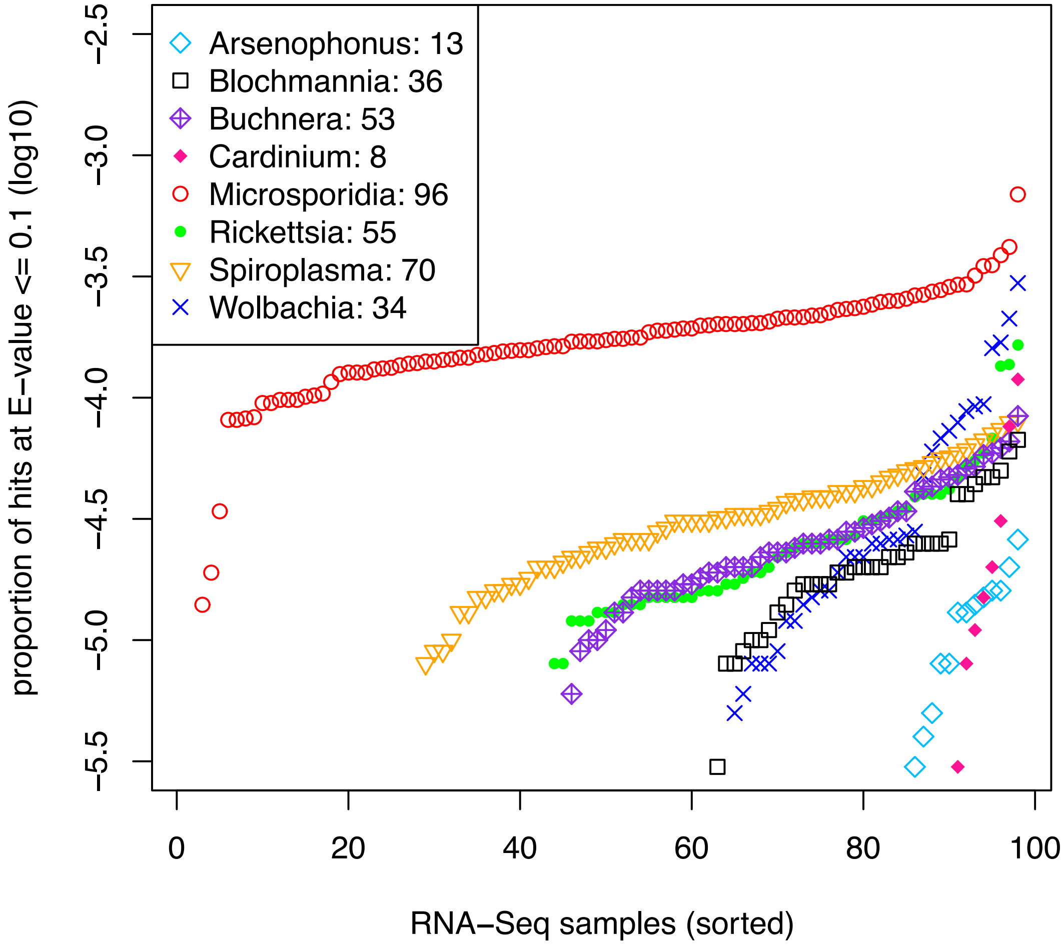


Figure 9.7: Proportions of significant best hits to various endosymbiont non-coding RNA sequences for all 98 RNA-Seq samples as determined by Lambda, shown on a log scale. The number of samples for which the proportion was greater than zero is given for each endosymbiont in the legend.

for this purpose, as the RNAcentral database is largely composed of rRNA sequences — i.e., a relatively few genes for a large number of organisms — thus reducing the likelihood of false positives resulting from a database search. Still, to characterize taxonomic composition, we required exceptionally good matches (E-value $\leq$ 1e-200). Approximately 350,000 hits (pooled across all 98 transcriptome data sets) met this level of significance. We used MEGAN [148,180] to classify these hits according to the NCBI taxonomy, and we generated a breakdown of these hits into taxonomic classes (Figure 9.8). We observed that the majority of hits were to insect sequences, as expected. We also found a relatively large number of bacterial classes represented, as well as some classes of fungi. Both of these findings were in keeping with expectations. Even at this high level of significance, there were a small fraction of assignments to taxa that were unexpected, such as Mammalia. This could either indicate contamination, or possibly a "tie-breaker" situation where the wrong taxon was arbitrarily chosen. To address this latter case, one could report more than just the best hit per transcript fragment (instead report the top 10 or top 100 hits, for example); subsequently, the LCA algorithm in MEGAN could use this additional information to make more conservative assignments (possibly to higher-level taxa).

9.3.4 Rickettsia 16S rRNA analysis

We sought to demonstrate that we could recover rRNA data from our transcriptome data sets, and use such data in phylogenetic analyses. Since we had already focused much of our attention on Microsporidia, we wanted to make this demon-

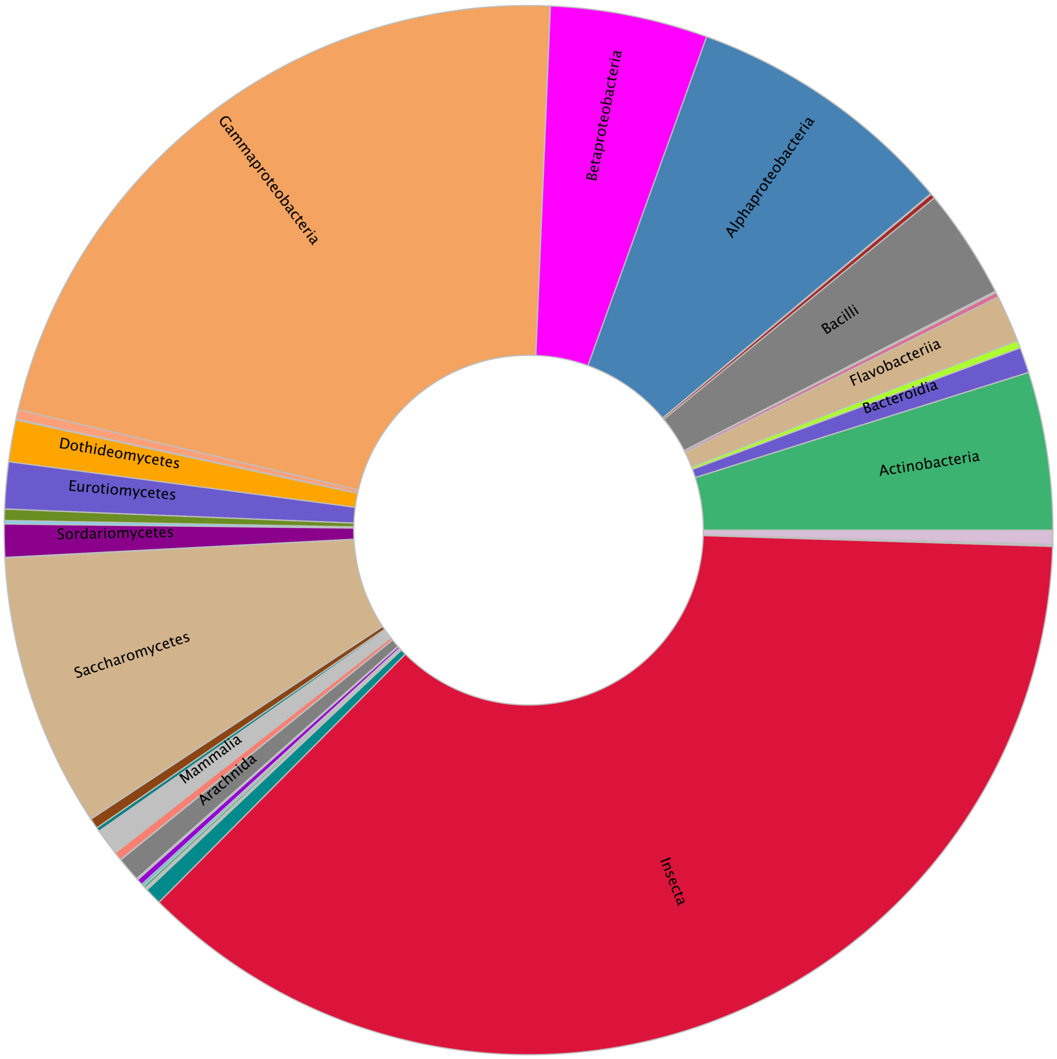


Figure 9.8: We used MEGAN to assign approximately 350,000 Lambda/RNAcentral transcript fragment hits to various taxonomic classes according to the NCBI taxonomy. Each hit used for classification purposes achieved an E-value ≤ 1e-200. We observed that insect, bacterial, and fungal sequences were all present in relative abundance, which was in keeping with expectations.

stration with an understudied bacterial endosymbiont, so we chose *Rickettsia*. Of the 125 Lambda/RNAcentral hits to *Rickettsia* across all 98 transcriptome data sets, 31 hits had an E-value ≤ 1e-05. Of these, there were seven hits to five distinct 16S RNAcentral database sequences, the majority of which seemed to cover most, if not all, of the 16S gene. We combined these seven query sequences and five database sequences, and provided them as input to SINA [326], an alignment service offered by the SILVA [327] database project. (SINA classified nine of the sequences as *Rickettsia*, and the other three sequences — all query sequences — as "unknown", suggesting that perhaps these three sequences were not actually *Rickettsia*.) In this analysis, we also included 41 *Rickettsia* sequences from the SILVA database that SINA identified to be ≥ 95% similar to the 12 sequences provided initially. Thus, our 16S rRNA alignment consisted of a total of 53 sequences.

Phylogenetic analysis with GARLI yielded the results shown in Figure 9.9. The query sequences (derived from RNA-Seq) of "unknown identity" were relatively easy to identify, as they were the taxa with the longest branch lengths: "Ful" (found at the top of the tree), and the two *Archaeopsylla erinacei* sequences (found at the bottom of the tree). There was relatively high bootstrap support for some groupings, although many of these were polytomies.

In summary, this analysis demonstrated that rRNA sequences such as the 16S gene, typically used for bacterial fingerprinting, can be recovered (albeit at low abundance) from transcriptome data sets and used in phylogenetic analyses, although one should be wary of sequence identifications made on the basis of an E-value alone. To this end, it may be helpful to add sequences of confirmed identity from existing databases such as SILVA.

9.4 Conclusion

We demonstrated that if done carefully, one can recover endosymbiont sequences from host-associated transcriptome data sets with relatively high confidence, albeit at relatively low abundance. We also showed that in addition to recovering endosymbiont protein-coding sequences from RNA-Seq data, one can also recover their rRNA sequences. We produced well-supported phylogenies that included endosymbiont and host sequences, as well as sequences from public databases.

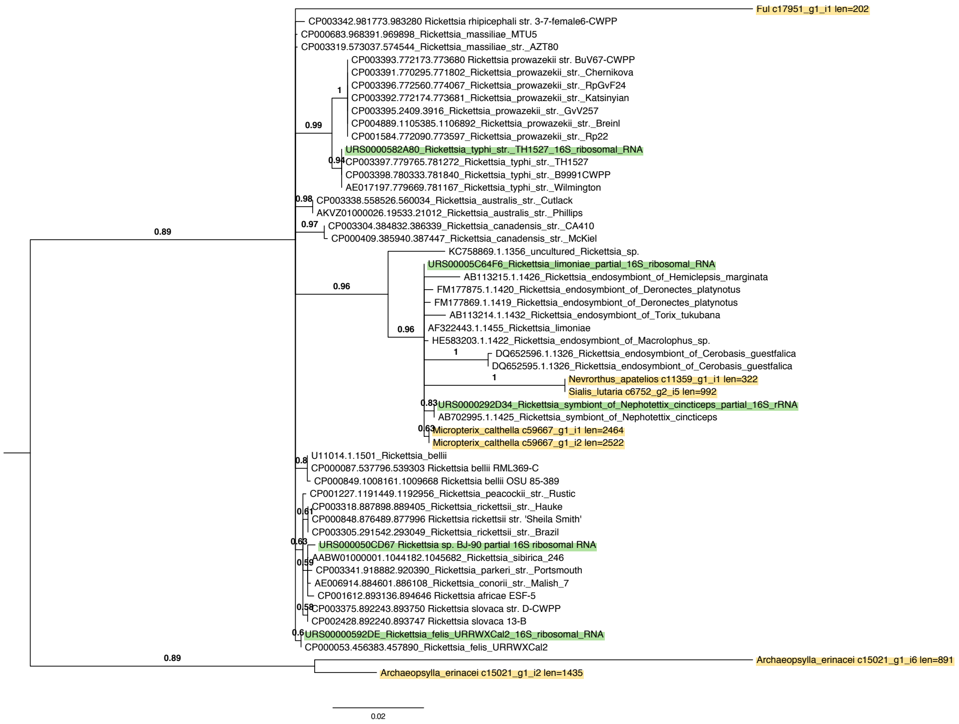

Ful c17951_g1_i1 len=202
CP003342.981773.983280 Rickettsia rhipicephali str. 3-7-female6-CWPP
CP000683.968391.969898_Rickettsia_massiliae_MTU5
CP003319.573037.574544_Rickettsia_massiliae_str._AZT80
CP003393.772173.773680 Rickettsia prowazekii str. BuV67-CWPP
CP003391.770295.771802_Rickettsia_prowazekii_str._Chernikova
CP003396.772560.774067_Rickettsia_prowazekii_str._RpGvF24
1
CP003392.772174.773681_Rickettsia_prowazekii_str._Katsinyian
CP003395.2409.3916_Rickettsia_prowazekii_str._GvV257
0.99
CP004889.1105385.1106892_Rickettsia_prowazekii_str._Breinl
CP001584.772090.773597_Rickettsia_prowazekii_str._Rp22
URS0000582A80_Rickettsia_typhi_str._TH1527_16S_ribosomal_RNA
0.94
CP003397.779765.781272_Rickettsia_typhi_str._TH1527
CP003398.780333.781840_Rickettsia_typhi_str._B9991CWPP
AE017197.779669.781167_Rickettsia_typhi_str._Wilmington
0.98
CP003338.558526.560034_Rickettsia_australis_str._Cutlack
AKVZ01000026.19533.21012_Rickettsia_australis_str._Phillips
0.97
CP003304.384832.386339_Rickettsia_canadensis_str._CA410
0.89
CP000409.385940.387447_Rickettsia_canadensis_str._McKiel
KC758869.1.1356_uncultured_Rickettsia_sp.
URS00005C64F6_Rickettsia_limoniae_partial_16S_ribosomal_RNA
AB113215.1.1426_Rickettsia_endosymbiont_of_Hemiclepsis_marginata
0.96
FM177875.1.1420_Rickettsia_endosymbiont_of_Deronectes_platynotus
FM177869.1.1419_Rickettsia_endosymbiont_of_Deronectes_platynotus
AB113214.1.1432_Rickettsia_endosymbiont_of_Torix_tukubana
AF322443.1.1455_Rickettsia_limoniae
0.96
HE583203.1.1422_Rickettsia_endosymbiont_of_Macrolophus_sp.
1
DQ652596.1.1326_Rickettsia_endosymbiont_of_Cerobasis_guestfalica
DQ652595.1.1326_Rickettsia_endosymbiont_of_Cerobasis_guestfalica
1
Nevrorthus_apatelios c11359_g1_i1 len=322
Sialis_lutaria c6752_g2_i5 len=992
0.83
URS0000292D34_Rickettsia_symbiont_of_Nephotettix_cincticeps_partial_16S_rRNA
AB702995.1.1425_Rickettsia_symbiont_of_Nephotettix_cincticeps
0.63
Micropterix_calthella c59667_g1_i1 len=2464
Micropterix_calthella c59667_g1_i2 len=2522
U11014.1.1501_Rickettsia_bellii
0.8
CP000087.537796.539303 Rickettsia bellii RML369-C
CP000849.1008161.1009668 Rickettsia bellii OSU 85-389
CP001227.1191449.1192956_Rickettsia_peacockii_str._Rustic
CP003318.887898.889405_Rickettsia_rickettsii_str._Hauke
0.61
CP000848.876489.877996 Rickettsia rickettsii str. 'Sheila Smith'
CP003305.291542.293049_Rickettsia_rickettsii_str._Brazil
0.63
URS000050CD67 Rickettsia sp. BJ-90 partial 16S ribosomal RNA
AABW01000001.1044182.1045682_Rickettsia_sibirica_246
0.59
CP003341.918882.920390_Rickettsia_parkeri_str._Portsmouth
AE006914.884601.886108_Rickettsia_conorii_str._Malish_7
CP001612.893136.894646 Rickettsia africae ESF-5
0.58
CP003375.892243.893750 Rickettsia slovaca str. D-CWPP
CP002428.892240.893747 Rickettsia slovaca 13-B
0.6
URS00000592DE_Rickettsia_felis_URRWXCal2_16S_ribosomal_RNA
CP000053.456383.457890_Rickettsia_felis_URRWXCal2
0.89
Archaeopsylla_erinacei c15021_g1_i6 len=891
Archaeopsylla_erinacei c15021_g1_i2 len=1435
0.02

Figure 9.9: 50% majority-rule bootstrap consensus tree for *Rickettsia* 16S sequences. RNA-Seq sequences are highlighted in yellow, RNAcentral sequences are highlighted in green, and SILVA sequences are unhighlighted. Scale is the expected number of nucleotide substitutions per site.

Chapter 10: Summary of contributions

Our contributions to the advancement of phylogenomic workflows included the following: (1) improvements to specific phylogenomic workflow steps, including sequence classification, orthology determination, and phylogenetic analysis; (2) development of a computational system optimized for performing genome-scale phylogenetic analyses; and (3) empirical studies that used complete phylogenomic workflows to analyze transcriptome data from moths and butterflies (Lepidoptera), as well as various insect endosymbionts (e.g., Microsporidia).

We compared a number of programs and methods for sequence classification, most of which were developed with metagenomics applications in mind. Because we treat our RNA-Seq samples as if they were metagenomes, it was important for us to be able to choose from among the vast number of programs available to perform the sequence classification task. Ultimately, we used HaMStR to extract sequences from our transcriptome data that matched gene models in our databases.

We developed an alternative to strict orthology determination procedures called the consensus method, which creates a single consensus sequence from a set of paralogous gene sequences, one that potentially includes ambiguous nucleotides. We initially validated this method using yeast transcriptome data, vertebrate proteomes, and plant proteomes. We then applied it in our phylogenomic workflows and found that it performed as well or better than the primary alternative approach, that of selecting a single sequence to serve as a putative ortholog. The consensus method is now a standard part of our phylogenomic workflows.

We described the GARLI web service, which is freely available at `molecularevolution.org`. The service makes it easy to perform computationally-intensive maximum likelihood phylogenetic analyses, as the analyses run on a powerful computational grid system known as The Lattice Project. We made a number of improvements to the grid computing system specifically to enhance phylogenetic analysis capability. These included the implementation of a computation-saving adaptive best tree search, which runs a relatively small number of searches initially and analyzes the similarity of the topologies that are returned. In the event that the topologies are mostly very similar, no additional searches need to be performed. We described a random forests model that we use to procure GARLI analysis runtime estimates, which in turn are used for job scheduling decisions and optimizations. One such optimization combines multiple predicted short-running analyses into a longer, optimal-length analysis for greater efficiency. We also described our implementation of the converse optimization, which subdivides long-running analyses into short, fixed-length BOINC workunits. This optimization reduces variance in analysis completion times, thus improving the turnaround time of analysis batches submitted to BOINC; this will enable BOINC to be used to compute GARLI web service analyses. Finally, we devised new throughput ratings for grid resources that are computed automatically using ordinary production jobs. The resource throughput ratings are used to send jobs to the fastest resources first, as well as to allow faster resources longer job queues.

We used complete phylogenomic workflows to analyze transcriptome data generated by the Leptree project. The pilot study, which used 46 taxa and 741 genes, resolved many Apoditrysian nodes that were previously uncertain, particularly within Obtectomera. However, although some "lower Apoditrysian" nodes were improved, many were still weakly-supported. In our follow-up analyses we increased our taxon sampling to 82 taxa, and increased our gene sample to 3,810 genes by creating a Lepidoptera-specific HaMStR database. We also created a Lepidoptera-specific mitochondrial gene database, as well as a Lepidoptera-specific amino acid substitution matrix (LEP62). We performed many different maximum likelihood analyses of concatenated-gene data sets, and we also performed a number of gene tree summary analyses. Although most of these failed to produce a substantial improvement over the results from our pilot study, a recent 23-taxon analysis that used PYGOT and GUIDANCE2 to remove dubiously-aligned regions from the data matrix did show promise.

Finally, we demonstrated that we could recover insect endosymbiont sequences from host-derived RNA-seq data with high confidence (e.g., sequences of Microsporidia, or bacterial sequences such as *Wolbachia*), although at relatively low abundance. These included both endosymbiont protein-coding sequences and ribosomal RNA sequences, which we used to produce well-supported phylogenies that also included host sequences and public database sequences. One could use such phylogenies with other data to formulate hypotheses about host-parasite coevolution.

Phylogenomics is still in its infancy, and our work is only the beginning; continuous innovation will be necessary to keep pace with the ever-increasing amount of available genome data and the increasingly complex analytical models needed to accurately reconstruct the tree of life.

Appendix A: Exploration of differences between CONSENSUS and REPRESENTATIVE options.

The two alternative procedures we used for producing a single sequence per taxon-locus combination for phylogenetic inference when orthology search returned multiple “hits” — i.e., REPRESENTATIVE and CONSENSUS — yielded identical ML topologies, and nearly identical bootstrap values in all analyses (Figure 7.2). A marked difference between the two procedures was observed in the 38-taxon analysis, however, for which finding the best tree topology took considerably more search effort for REPRESENTATIVE than for CONSENSUS: out of 100 ML searches, the best tree topology was found 25 times for the CONSENSUS matrix, but only once for the REPRESENTATIVE matrix. (We found no such difference for either the 16- or 46-taxon analyses.) This appendix describes our efforts to find the cause of the different behaviors of the REPRESENTATIVE and CONSENSUS options in the 38-taxon case.

Comparison of the REPRESENTATIVE and CONSENSUS matrices for 38 taxa showed two essential differences. First, CONSENSUS had more ambiguous sites. That is, some nucleotides were ambiguous in CONSENSUS but not in REPRESENTATIVE (though the number of these was less than one percent of the total). Second, CONSENSUS had more total sequence. That is, some nucleotide positions were present in consensus but not in representative. Again, the size of the difference was small; there were less than 2,000 such positions. Although both differences seem quantitatively minor, we felt that one or the other was likely to underlie the contrast in search difficulty. Therefore, we sought to isolate each of the two variables in turn — level of codon ambiguity, and total number of codons — and test the effect of each variable on search efficacy.

To test the effect of increased codon ambiguity, we built a modified representative matrix that differed from the original only in having a level of ambiguity comparable to that of the consensus matrix. We first aligned the consensus and representative amino acid sequences to each other, then converted each matrix back to nucleotide coding sequence. For each comparison of the same taxon between the two aligned matrices, we identified all of the amino acid positions at which there were multiple amino acids in the consensus matrix prior to reduction for phylogenetic analysis, which would result in ambiguity upon conversion to a phylogenetic data matrix by the consensus procedure. In those cases, we replaced the corresponding codon in the representative coding sequence with the consensus codon. The result was a matrix the same size as the original representative data matrix, with mostly the same sequence content, but with some ambiguity added. When this matrix was subjected to 100 ml searches, it returned the best topology far more often than did the unmodified representative matrix, and about as often as the consensus matrix. Thus, the increased ambiguity seemed to make tree search considerably easier.

To determine whether the greater total amount of sequence in the consensus matrix also affected search efficacy, we built a second modified REPRESENTATIVE matrix, intended to differ from the original in size but not level of ambiguity. To do this, we identified all sites at which there was a gap character in the REPRESENTATIVE matrix and a non-gap character in the CONSENSUS matrix. We then added all of these additional non-gap characters to the REPRESENTATIVE matrix. Under ML search, this matrix behaved very much like the original REPRESENTATIVE matrix, finding the best tree only one or a few times in each of 100 searches.

In summary, replacing non-ambiguous REPRESENTATIVE codons with ambiguous CONSENSUS codons essentially reproduced the CONSENSUS result, whereas adding the additional CONSENSUS data to the original REPRESENTATIVE matrix did not have the same transformative effect. We hypothesize that the small number of nucleotides that are ambiguous under CONSENSUS coding but not so under REPRESENTATIVE coding introduce conflicting signal in the latter matrix that makes finding the best tree topology more difficult than if these nucleotides are degenerated. The effect is idiosyncratic, in that we saw it only in the 38-taxon data set. We felt it was nonetheless worth investigating because essentially nothing was known about the comparative performance of these two procedures.

Appendix B: The LEP62 custom amino acid substitution matrix.

A recent study showed the utility of using clade-specific amino acid substitution matrices in *de novo* orthology prediction involving Mollicutes genomes [328]. Using a substitution matrix designed specifically for the group of organisms one is working with makes sense from a theoretical standpoint. As we perform amino acid alignments many places in our phylogenomic workflow, and these rely on a well-calibrated amino acid substitution matrix (usually BLOSUM50 or BLOSUM62 by default), we hypothesized that these alignments would be improved if we used a substitution matrix derived from Lepidoptera-specific protein alignments.

B.1 Creating the LEP62 matrix

We had initially constructed 7,042 orthologous groups from Ensembl genome data (Section 8.5.1), thus providing a fairly large corpus of Lepidoptera-specific amino acid alignment data from which to build a custom substitution matrix. As part of our initial investigation, we calculated (using T-Coffee [300] and custom Perl scripts) that the average sequence identity of the aligned orthologous groups was 61.997%. To build the LEP62 matrix, we retrieved and ran the scripts made available by Lemaitre et al. [328]; this package also included the BLOSUM program [329]. When we compared the LEP62 matrix to BLOSUM62, we found that 86/200 matrix entries were different. Other comparative statistics are given in Table B.1.

Next, we wanted to demonstrate the usefulness of the LEP62 matrix in similarity searches. As with Lemaitre et al. [328], we found no straightforward way to have BLAST use a custom substitution matrix, so instead we used the FASTA package [303], which readily accepted custom substitution matrices. Using a sample protein sequence taken from our transcriptome data, we performed five searches against the NCBI NR database with different combinations of alignment program and substitution matrix: `blast`+BLOSUM62; `fasta`+BLOSUM62; `fasta`+LEP62; `ssearch`+BLOSUM62; and `ssearch`+LEP62. Inspecting the search results, we made the following observations: (1) the top two hits were the same in each search (*Bombyx* and *Danaus* sequences, respectively); (2) `ssearch` produced better E-values than `fasta`; and (3) LEP62 produced better E-values than BLOSUM62.

In another test, we performed the same five searches, this time using the *Bombyx* proteome as the database. The top hit was the same in each search, and had a much lower E-value than any of the other hits; this was a case where the top hit was probably the only "good" hit in the database. Once again, we found that our discriminatory power was highest with `ssearch` and the LEP62 matrix. After conducting these tests, we were reasonably confident that using programs from the FASTA package in conjunction with the LEP62 matrix had the potential to improve workflow performance.

B.2 Modifications to HaMStR

In order to test the efficacy of using the LEP62 matrix in our workflow, we made incremental modifications to HaMStR (version 13.1; [272]) and ran the *Antaeotricha schlaegeri* ("Ant") RNA-Seq sample against the 7,042-gene `moth+min-one-butterfly` database after each modification. Table B.2 gives these modifications and the corresponding statistics generated from each run of HaMStR. Note that with HaMStR we used the `fasty` program from the FASTA package instead of `ssearch`, despite the fact that in our previous tests `ssearch` displayed the best performance; this is because `ssearch` only supports DNA:DNA or protein:protein comparisons, whereas we needed to search a protein database with a translated nucleotide query. The only modification not strictly having to do with the LEP62 matrix involved running `pseg` [330] on the *Bombyx* BLAST database to mask low-complexity regions. The statistics in Table B.2 show a slight increase in the number of hits, generally, when LEP62 was used, as we would expect. A fairly substantial *decrease* in the number of hits was observed after running `pseg`; presumably, we lost hits to low-complexity regions that were not desirable in the first place. At a later date we discovered another place to use LEP62 — namely, in the call HaMStR makes to GeneWise [273] — so this particular modification is not shown in Table B.2; in our tests, this change had only a minor effect on HaMStR search statistics.

Matrix	Entropy	Expected value	Mean mismatch	Mean match
LEP62	0.8120	-0.5624	-1.63	6.3
MOLLI60	0.7126	-0.582	-1.56	6.1
BLOSUM62	0.6979	-0.5209	-1.42	5.8
BLOSUM45	0.3795	-0.2789	-1.34	7.05

Table B.1: Comparative statistics for the LEP62 substitution matrix, the MOLLI60 matrix [328], and two commonly used BLOSUM matrices.

Number of hits	Hits to unique loci	HaMStR modification
19,192	6,192	HaMStR 13.1, unmodified
19,375	6,259	used `fasty` with the LEP62 matrix instead of `blastx`
19,289	6,235	set `blast_eval` to 1e-05 instead of the default value of 10.0
17,205	5,742	used the `-S` flag to `fasty` after running `pseg` on the *Bombyx* BLAST database
17,222	5,740	used LEP62 with MAFFT for co-ortholog assignment
17,227	5,746	used LEP62 with MAFFT for co-ortholog assignment using reference sequences

Table B.2: Modifications to HaMStR that allowed for use of a custom substitution matrix, and their corresponding effect on the number of hits to the `moth+min-one-butterfly` database for the "Ant" RNA-Seq sample.

B.3 Phylogenetic analysis results

Ultimately we wanted to characterize the impact that using the LEP62 matrix throughout the workflow would have on the outcome of the phylogenetic analyses. Previously we had analyzed a 16-taxon, 2,884-gene data matrix to test an early version of the `moth+min-one-butterfly` database; this matrix was constructed with an older version of HaMStR (version 9) that had none of the modifications mentioned in Section B.2. We rebuilt this matrix using the modified version of HaMStR, and repeated the phylogenetic analyses. The new 16-taxon, 2,884-gene matrix had about 10% fewer residues than the previous one, and was slightly more complete. This correlated with the statistics for the Ant sample, for which the final total number of sequences was about 10% less than the starting total (Table B.2). Comparing the new phylogenetic results (110 best tree search replicates; 279 bootstrap replicates; five search replicates per bootstrap replicate) to the previous phylogenetic results, we found only minor differences: the topology was the same; bootstrap support for a couple of internal nodes increased somewhat, and decreased for a couple of others. Thus, while we could not conclude that our modifications had produced a substantial improvement in phylogenetic results, they had also not worsened them. Standing by the logic that using an amino acid substitution matrix specific to the group of organisms one is working with makes sense *a priori*, we continued to use the LEP62 matrix for the remainder of the analyses presented in Chapter 8.

Bibliography

[1] F Sanger and A R Coulson. A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase. *J Mol Biol*, 94(3):441–8, May 1975.

[2] D. J. Zwickl. *Genetic algorithm approaches for the phylogenetic analysis of large biological sequence datasets under the maximum likelihood criterion.* PhD thesis, The University of Texas at Austin., 2006.

[3] J P Huelsenbeck and F Ronquist. MRBAYES: Bayesian inference of phylogenetic trees. *Bioinformatics*, 17(8):754–5, Aug 2001.

[4] Fredrik Ronquist and John P Huelsenbeck. MrBayes 3: Bayesian phylogenetic inference under mixed models. *Bioinformatics*, 19(12):1572–4, Aug 2003.

[5] Jerome C Regier, Andreas Zwick, Michael P Cummings, Akito Y Kawahara, Soowon Cho, Susan Weller, Amanda Roe, Joaquin Baixeras, John W Brown, Cynthia Parr, Donald R Davis, Marc Epstein, Winifred Hallwachs, Axel Hausmann, Daniel H Janzen, Ian J Kitching, M Alma Solis, Shen-Horn Yen, Adam L Bazinet, and Charles Mitter. Toward reconstructing the evolution of advanced moths and butterflies (Lepidoptera: Ditrysia): an initial molecular study. *BMC Evol Biol*, 9:280, 2009.

[6] Jae-Cheon Sohn, Jerome C Regier, Charles Mitter, Donald Davis, Jean-François Landry, Andreas Zwick, and Michael P Cummings. A molecular phylogeny for Yponomeutoidea (Insecta, Lepidoptera, Ditrysia) and its implications for classification, biogeography and the evolution of host plant use. *PLoS ONE*, 8(1):e55066, 2013.

[7] Jerome C. Regier, Charles Mitter, Andreas Zwick, Adam L. Bazinet, Michael P. Cummings, Akito Y. Kawahara, Jae-Cheon Sohn, Derrick J. Zwickl, Soowon Cho, Donald R. Davis, Joaquin Baixeras, John Brown, Cynthia Parr, Susan Weller, David C. Lees, and Kim T. Mitter. A large-scale, higher-level, molecular phylogenetic study of the insect order Lepidoptera (moths and butterflies). *PLoS ONE*, 8(3):e58568, 03 2013.

[8] Jerome C. Regier, Charles Mitter, Niels P. Kristensen, Donald R. Davis, Erik J. Van Nieukerken, Jadranka Rota, Thomas J. Simonsen, Kim T. Mitter, Akito Y. Kawahara, Shen-Horn Yen, Michael P. Cummings, and Andreas Zwick. A molecular phylogeny for the oldest (nonditrysian) lineages of extant Lepidoptera, with implications for classification, comparative morphology and life-history evolution. *Systematic Entomology*, 05/2015 2015.

[9] Stephan C Schuster. Next-generation sequencing transforms today's biology. *Nat Methods*, 5(1):16–8, Jan 2008.

[10] Zhong Wang, Mark Gerstein, and Michael Snyder. RNA-Seq: a revolutionary tool for transcriptomics. *Nat Rev Genet*, 10(1):57–63, Jan 2009.

[11] Chris Todd Hittinger, Mark Johnston, John T Tossberg, and Antonis Rokas. Leveraging skewed transcript abundance by RNA-Seq to increase the genomic depth of the tree of life. *Proc Natl Acad Sci U S A*, 107(4):1476–81, Jan 2010.

[12] Daniel R Zerbino and Ewan Birney. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res*, 18(5):821–9, May 2008.

[13] S F Altschul, W Gish, W Miller, E W Myers, and D J Lipman. Basic local alignment search tool. *J Mol Biol*, 215(3):403–10, Oct 1990.

[14] B Morgenstern. DIALIGN 2: improvement of the segment-to-segment approach to multiple sequence alignment. *Bioinformatics*, 15(3):211–8, Mar 1999.

[15] Jesse W Breinholt and Akito Y Kawahara. Phylotranscriptomics: saturated third codon positions radically influence the estimation of trees based on next-gen data. *Genome Biol Evol*, 5(11):2082–92, 2013.

[16] Rongfeng Cui, Molly Schumer, Karla Kruesi, Ronald Walter, Peter Andolfatto, and Gil G Rosenthal. Phylogenomics reveals extensive reticulate evolution in Xiphophorus fishes. *Evolution*, 67(8):2166–79, Aug 2013.

[17] Ingo Ebersberger, Ricardo de Matos Simoes, Anne Kupczok, Matthias Gube, Erika Kothe, Kerstin Voigt, and Arndt von Haeseler. A consistent phylogenetic backbone for the fungi. *Mol Biol Evol*, 29(5):1319–34, May 2012.

[18] Rosa Fernández, Gustavo Hormiga, and Gonzalo Giribet. Phylogenomic analysis of spiders reveals nonmonophyly of orb weavers. *Curr Biol*, 24(15):1772–7, Aug 2014.

[19] Stefanie Hartmann, Conrad Helm, Birgit Nickel, Matthias Meyer, Torsten H Struck, Ralph Tiedemann, Joachim Selbig, and Christoph Bleidorn. Exploiting gene families for phylogenomic analysis of myzostomid transcriptome data. *PLoS One*, 7(1):e29843, 2012.

[20] Marshal Hedin, James Starrett, Sajia Akhter, Axel L Schönhofer, and Jeffrey W Shultz. Phylogenomic resolution of paleozoic divergences in harvestmen (Arachnida, Opiliones) via analysis of next-generation transcriptome data. *PLoS One*, 7(8):e42888, 2012.

[21] Erich D Jarvis, Siavash Mirarab, Andre J Aberer, Bo Li, Peter Houde, Cai Li, Simon Y W Ho, Brant C Faircloth, Benoit Nabholz, Jason T Howard, Alexander Suh, Claudia C Weber, Rute R da Fonseca, Jianwen Li, Fang Zhang, Hui Li, Long Zhou, Nitish Narula, Liang Liu, Ganesh Ganapathy, Bastien Boussau, Md Shamsuzzoha Bayzid, Volodymyr Zavidovych, Sankar Subramanian, Toni Gabaldón, Salvador Capella-Gutiérrez, Jaime Huerta-Cepas, Bhanu Rekepalli, Kasper Munch, Mikkel Schierup, Bent Lindow, Wesley C Warren, David Ray, Richard E Green, Michael W Bruford, Xiangjiang Zhan, Andrew Dixon, Shengbin Li, Ning Li, Yinhua Huang, Elizabeth P Derryberry, Mads Frost Bertelsen, Frederick H Sheldon, Robb T Brumfield, Claudio V Mello, Peter V Lovell, Morgan Wirthlin, Maria Paula Cruz Schneider, Francisco Prosdocimi, José Alfredo Samaniego, Amhed Missael Vargas Velazquez, Alonzo Alfaro-Núñez, Paula F Campos, Bent Petersen, Thomas Sicheritz-Ponten, An Pas, Tom Bailey, Paul Scofield, Michael Bunce, David M Lambert, Qi Zhou, Polina Perelman, Amy C Driskell, Beth Shapiro, Zijun Xiong, Yongli Zeng, Shiping Liu, Zhenyu Li, Binghang Liu, Kui Wu, Jin Xiao, Xiong Yinqi, Qiuemei Zheng, Yong Zhang, Huanming Yang, Jian Wang, Linnea Smeds, Frank E Rheindt, Michael Braun, Jon Fjeldsa, Ludovic Orlando, F Keith Barker, Knud Andreas Jønsson, Warren Johnson, Klaus-Peter Koepfli, Stephen O'Brien, David Haussler, Oliver A Ryder, Carsten Rahbek, Eske Willerslev, Gary R Graves, Travis C Glenn, John McCormack, Dave Burt, Hans Ellegren, Per Alström, Scott V Edwards, Alexandros Stamatakis, David P Mindell, Joel Cracraft, Edward L Braun, Tandy Warnow, Wang Jun, M Thomas P Gilbert, and Guojie Zhang. Whole-genome analyses resolve early branches in the tree of life of modern birds. *Science*, 346(6215):1320–31, Dec 2014.

[22] Brian R Johnson, Marek L Borowiec, Joanna C Chiu, Ernest K Lee, Joel Atallah, and Philip S Ward. Phylogenomics resolves evolutionary relationships among ants, bees, and wasps. *Curr Biol*, Oct 2013.

[23] Eva Jiménez-Guri, Jaime Huerta-Cepas, Luca Cozzuto, Karl R Wotton, Hui Kang, Heinz Himmelbauer, Guglielmo Roma, Toni Gabaldón, and Johannes Jaeger. Comparative transcriptomics of early dipteran development. *BMC Genomics*, 14:123, 2013.

[24] Laura A Katz and Jessica R Grant. Taxon-rich phylogenomic analyses resolve the eukaryotic tree of life and reveal the power of subsampling by sites. *Syst Biol*, 64(3):406–15, May 2015.

[25] Akito Y Kawahara and Jesse W Breinholt. Phylogenomics provides strong evidence for relationships of butterflies and moths. *Proc Biol Sci*, 281(1788):20140970, Aug 2014.

[26] Kord M Kober and Giacomo Bernardi. Phylogenomics of strongylocentrotid sea urchins. *BMC Evol Biol*, 13:88, 2013.

[27] Kevin M Kocot, Johanna T Cannon, Christiane Todt, Mathew R Citarella, Andrea B Kohn, Achim Meyer, Scott R Santos, Christoffer Schander, Leonid L Moroz, Bernhard Lieb, and Kenneth M Halanych. Phylogenomics reveals deep molluscan relationships. *Nature*, 477(7365):452–6, Sep 2011.

[28] Kevin M Kocot, Kenneth M Halanych, and Patrick J Krug. Phylogenomics supports panpulmonata: Opisthobranch paraphyly and key evolutionary steps in a major radiation of gastropod molluscs. *Mol Phylogenet Evol*, Jul 2013.

[29] Harald O Letsch, Karen Meusemann, Benjamin Wipfler, Kai Schütte, Rolf Beutel, and Bernhard Misof. Insect phylogenomics: results, problems and the impact of matrix composition. *Proc Biol Sci*, 279(1741):3282–90, Aug 2012.

[30] Bernhard Misof, Shanlin Liu, Karen Meusemann, Ralph S Peters, Alexander Donath, Christoph Mayer, Paul B Frandsen, Jessica Ware, Tomáš Flouri, Rolf G Beutel, Oliver Niehuis, Malte Petersen, Fernando Izquierdo-Carrasco, Torsten Wappler, Jes Rust, Andre J Aberer, Ulrike Aspöck, Horst Aspöck, Daniela Bartel, Alexander Blanke, Simon Berger, Alexander Böhm, Thomas R Buckley, Brett Calcott, Junqing Chen, Frank Friedrich, Makiko Fukui, Mari Fujita, Carola Greve, Peter Grobe, Shengchang Gu, Ying Huang, Lars S Jermiin, Akito Y Kawahara, Lars Krogmann, Martin Kubiak, Robert Lanfear, Harald Letsch, Yiyuan Li, Zhenyu Li, Jiguang Li, Haorong Lu, Ryuichiro Machida, Yuta Mashimo, Pashalia Kapli, Duane D McKenna, Guanliang Meng, Yasutaka Nakagaki, José Luis Navarrete-Heredia, Michael Ott, Yanxiang Ou, Günther Pass, Lars Podsiadlowski, Hans Pohl, Björn M von Reumont, Kai Schütte, Kaoru Sekiya, Shota Shimizu, Adam Slipinski, Alexandros Stamatakis, Wenhui Song, Xu Su, Nikolaus U Szucsich, Meihua Tan, Xuemei Tan, Min Tang, Jingbo Tang, Gerald Timelthaler, Shigekazu Tomizuka, Michelle Trautwein, Xiaoli Tong, Toshiki Uchifune, Manfred G Walzl, Brian M Wiegmann, Jeanne Wilbrandt, Benjamin Wipfler, Thomas K F Wong, Qiong Wu, Gengxiong Wu, Yinlong Xie, Shenzhou Yang, Qing Yang, David K Yeates, Kazunori Yoshizawa, Qing Zhang, Rui Zhang, Wenwei Zhang, Yunhui Zhang, Jing Zhao, Chengran Zhou, Lili Zhou, Tanja Ziesmann, Shijie Zou, Yingrui Li, Xun Xu, Yong Zhang, Huanming Yang, Jian Wang, Jun Wang, Karl M Kjer, and Xin Zhou. Phylogenomics resolves the timing and pattern of insect evolution. *Science*, 346(6210):763–7, Nov 2014.

[31] Thomas J Near, Alex Dornburg, Ron I Eytan, Benjamin P Keck, W Leo Smith, Kristen L Kuhn, Jon A Moore, Samantha A Price, Frank T Burbrink, Matt Friedman, and Peter C Wainwright. Phylogeny and tempo of diversification in the superradiation of spiny-rayed fishes. *Proc Natl Acad Sci U S A*, 110(31):12738–43, Jul 2013.

[32] Todd H Oakley, Joanna M Wolfe, Annie R Lindgren, and Alexander K Zaharoff. Phylotranscriptomics to bring the understudied into the fold: monophyletic ostracoda, fossil placement, and pancrustacean phylogeny. *Mol Biol Evol*, 30(1):215–33, Jan 2013.

[33] Ralph S Peters, Karen Meusemann, Malte Petersen, Christoph Mayer, Jeanne Wilbrandt, Tanja Ziesmann, Alexander Donath, Karl M Kjer, Ulrike Aspöck, Horst Aspöck, Andre Aberer, Alexandros Stamatakis, Frank Friedrich, Frank Hünefeld, Oliver Niehuis, Rolf G Beutel, and Bernhard Misof. The evolutionary history of holometabolous insects inferred from transcriptome-based phylogeny and comprehensive morphological data. *BMC Evol Biol*, 14(1):52, 2014.

[34] Ana Riesgo, Sónia C S Andrade, Prashant P Sharma, Marta Novo, Alicia R Pérez-Porro, Varpu Vahtera, Vanessa L González, Gisele Y Kawauchi, and Gonzalo Giribet. Comparative description of ten transcriptomes of newly sequenced invertebrates and efficiency estimation of genomic sampling in non-model taxa. *Front Zool*, 9(1):33, 2012.

[35] Alexander G Shanku, Mark A McPeek, and Andrew D Kern. Functional annotation and comparative analysis of a zygopteran transcriptome. *G3 (Bethesda)*, Mar 2013.

[36] Sabrina Simon, Apurva Narechania, Rob Desalle, and Heike Hadrys. Insect phylogenomics: exploring the source of incongruence using new transcriptomic data. *Genome Biol Evol*, 4(12):1295–309, Jan 2012.

[37] Stephen A Smith, Nerida G Wilson, Freya E Goetz, Caitlin Feehery, Sónia C S Andrade, Greg W Rouse, Gonzalo Giribet, and Casey W Dunn. Resolving the evolutionary relationships of molluscs with phylogenomic tools. *Nature*, 480(7377):364–7, Dec 2011.

[38] Sen Song, Liang Liu, Scott V Edwards, and Shaoyuan Wu. Resolving conflict in eutherian mammal phylogeny using phylogenomics and the multispecies coalescent model. *Proc Natl Acad Sci U S A*, 109(37):14942–7, Sep 2012.

[39] Ruth E Timme, Tsvetan R Bachvaroff, and Charles F Delwiche. Broad phylogenomic sampling and the sister lineage of land plants. *PLoS One*, 7(1):e29696, 2012.

[40] Jun Wen, Zhiqiang Xiong, Ze-Long Nie, Likai Mao, Yabing Zhu, Xian-Zhao Kan, Stefanie M Ickert-Bond, Jean Gerrath, Elizabeth A Zimmer, and Xiao-Dong Fang. Transcriptome sequences resolve deep relationships of the grape family. *PLoS One*, 8(9):e74394, 2013.

[41] Ya Yang, Michael J Moore, Samuel F Brockington, Douglas E Soltis, Gane Ka-Shu Wong, Eric J Carpenter, Yong Zhang, Li Chen, Zhixiang Yan, Yinlong Xie, Rowan F Sage, Sarah Covshoff, Julian M Hibberd, Matthew N Nelson,

and Stephen A Smith. Dissecting molecular evolution in the highly diverse plant clade Caryophyllales using transcriptome sequencing. *Mol Biol Evol*, Apr 2015.

[42] Ming Zou, Baocheng Guo, Wenjing Tao, Gloria Arratia, and Shunping He. Integrating multi-origin expression data improves the resolution of deep phylogeny of ray-finned fish (Actinopterygii). *Sci Rep*, 2:665, 2012.

[43] Emily Moriarty Lemmon and Alan R. Lemmon. High-throughput genomic data in systematics and phylogenetics. *Annual Review of Ecology, Evolution, and Systematics*, 44(1):99–121, 2013.

[44] Manfred G Grabherr, Brian J Haas, Moran Yassour, Joshua Z Levin, Dawn A Thompson, Ido Amit, Xian Adiconis, Lin Fan, Raktima Raychowdhury, Qiandong Zeng, Zehua Chen, Evan Mauceli, Nir Hacohen, Andreas Gnirke, Nicholas Rhind, Federica di Palma, Bruce W Birren, Chad Nusbaum, Kerstin Lindblad-Toh, Nir Friedman, and Aviv Regev. Full-length transcriptome assembly from RNA-Seq data without a reference genome. *Nat Biotechnol*, 29(7):644–52, Jul 2011.

[45] Gordon Robertson, Jacqueline Schein, Readman Chiu, Richard Corbett, Matthew Field, Shaun D Jackman, Karen Mungall, Sam Lee, Hisanaga Mark Okada, Jenny Q Qian, Malachi Griffith, Anthony Raymond, Nina Thiessen, Timothee Cezard, Yaron S Butterfield, Richard Newsome, Simon K Chan, Rong She, Richard Varhol, Baljit Kamoh, Anna-Liisa Prabhu, Angela Tam, YongJun Zhao, Richard A Moore, Martin Hirst, Marco A Marra, Steven J M Jones, Pamela A Hoodless, and Inanc Birol. *De novo* assembly and analysis of RNA-seq data. *Nat Methods*, 7(11):909–12, Nov 2010.

[46] Albert J Vilella, Jessica Severin, Abel Ureta-Vidal, Li Heng, Richard Durbin, and Ewan Birney. EnsemblCompara GeneTrees: Complete, duplication-aware phylogenetic trees in vertebrates. *Genome Res*, 19(2):327–35, Feb 2009.

[47] Julie D Thompson, Benjamin Linard, Odile Lecompte, and Olivier Poch. A comprehensive benchmark study of multiple sequence alignment methods: current challenges and future perspectives. *PLoS One*, 6(3):e18093, 2011.

[48] MP Cummings and JC Huskamp. Grid computing. *EDUCAUSE Review*, 40:116–117, 2005.

[49] I. Foster and C. Kesselman. Globus: a toolkit-based grid architecture. In I. Foster and C. Kesselman, editors, *The Grid: Blueprint for a New Computing Infrastructure*, pages pp. 259–278. Morgan-Kaufmann, 1999.

[50] David P. Anderson. BOINC: A system for public-resource computing and storage. In *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, GRID '04, pages 4–10, Washington, DC, USA, 2004. IEEE Computer Society.

[51] M.J. Litzkow, M. Livny, and M.W. Mutka. Condor–a hunter of idle workstations. In *Distributed Computing Systems, 1988., 8th International Conference on*, pages 104 –111, Jun 1988.

[52] Adam L. Bazinet. The Lattice Project: A multi-model grid computing system. Master's thesis, University of Maryland, College Park, December 2009.

[53] Adam L. Bazinet, Derrick J. Zwickl, and Michael P. Cummings. A gateway for phylogenetic analysis powered by grid computing featuring GARLI 2.0. *Systematic Biology*, 2014.

[54] Adam L. Bazinet and Michael P. Cummings. Computing the tree of life: Leveraging the power of desktop and service grids. *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum*, pages 1896–1902, 2011.

[55] Daniel L Ayres, Aaron Darling, Derrick J Zwickl, Peter Beerli, Mark T Holder, Paul O Lewis, John P Huelsenbeck, Fredrik Ronquist, David L Swofford, Michael P Cummings, Andrew Rambaut, and Marc A Suchard. BEAGLE: an application programming interface and high-performance computing library for statistical phylogenetics. *Syst Biol*, 61(1):170–3, Jan 2012.

[56] J Grand, MP Cummings, TG Rebelo, TH Ricketts, and MC Neel. Biased data reduce efficiency and effectiveness of conservation reserve networks. *Ecol Lett*, 10(5):364–374, May 2007.

[57] Catherine Dibble, Stephen Wendel, and Kristofor Carle. Simulating pandemic influenza risks of US cities. In *Winter Simulation Conference*, pages 1548–1550, 2007.

[58] Sarah A Tishkoff, Mary Katherine Gonder, Brenna M Henn, Holly Mortensen, Alec Knight, Christopher Gignoux, Neil Fernandopulle, Godfrey Lema, Thomas B Nyambo, Uma Ramakrishnan, Floyd A Reed, and Joanna L Mountain. History of click-speaking populations of Africa inferred from mtDNA and Y chromosome genetic variation. *Mol Biol Evol*, 24(10):2180–95, Oct 2007.

[59] R. Varadan, M. Assfalg, S. Raasi, C. Pickart, and D. Fushman. Structural determinants for selective recognition of a Lys48-linked polyubiquitin chain by a UBA domain. *Molecular Cell*, 60:687–698, 2005.

[60] MP Cummings, MC Neel, and KL Shaw. A genealogical approach to quantifying lineage divergence. *Evolution*, 62(9):2411–2422, Sep 2008.

[61] Kirstern L F Haseyama, Brian M Wiegmann, Eduardo A B Almeida, and Claudio J B de Carvalho. Say goodbye to tribes in the new house fly classification: A new molecular phylogenetic analysis and an updated biogeographical narrative for the Muscidae (Diptera). *Mol Phylogenet Evol*, 89:1–12, Aug 2015.

[62] Courtney A Hofman, Torben C Rick, Melissa T R Hawkins, W Chris Funk, Katherine Ralls, Christina L Boser, Paul W Collins, Tim Coonan, Julie L King, Scott A Morrison, Seth D Newsome, T Scott Sillett, Robert C Fleischer, and Jesus E Maldonado. Mitochondrial genomes suggest rapid evolution of dwarf California Channel Islands foxes (*Urocyon littoralis*). *PLoS One*, 10(2):e0118240, 2015.

[63] Shiny Cathlynne S. Yu and Jonas P. Quilang. Molecular phylogeny of catfishes (Teleostei: Siluriformes) in the Philippines using the mitochondrial genes COI, Cyt b, 16S rRNA, and the nuclear genes Rag1 and Rag2. *Philippine Journal of Science*, 143:187–198, 12/2014 2015.

[64] Giulia Fassio, Maria Vittoria Modica, Maria Chiara Alvaro, Stefano Schiaparelli, and Marco Oliverio. Developmental trade-offs in southern ocean mollusc kleptoparasitic species. *Hydrobiologia*, 06/2015 2015.

[65] B. Gehesquière, J. A. Crouch, R. E. Marra, K. Van Poucke, F. Rys, M. Maes, B. Gobin, M. Höfte, and K. Heungens. Characterization and taxonomic reassessment of the box blight pathogen *Calonectria pseudonaviculata*, introducing *Calonectria henricotiae* sp. nov. *Plant Pathology*, 05/2015 2015.

[66] K M Stucker, S A Schobel, R J Olsen, H L Hodges, X Lin, R A Halpin, N Fedorova, T B Stockwell, A Tovchigrechko, S R Das, D E Wentworth, and J M Musser. Haemagglutinin mutations and glycosylation changes shaped the 2012/13 influenza A(H3N2) epidemic, Houston, Texas. *Euro Surveill*, 20(18), 2015.

[67] Pedro L.V. Peloso, Darrel R. Frost, Stephen J. Richards, Miguel T. Rodrigues, Stephen Donnellan, Masafumi Matsui, Cristopher J. Raxworthy, S.D. Biju, Emily Moriarty Lemmon, Alan R. Lemmon, and Ward C. Wheeler. The impact of anchored phylogenomics and taxon sampling on phylogenetic inference in narrow-mouthed frogs (Anura, Microhylidae). *Cladistics*, 03/2015 2015.

[68] Jessica Goodheart, Yolanda Camacho-García, Vinicius Padula, Michael Schrödl, Juan L. Cervera, Terrence M. Gosliner, and Ángel Valdés. Systematics and biogeography of *Pleurobranchus* Cuvier, 1804, sea slugs (Heterobranchia: Nudipleura: Pleurobranchidae). *Zoological Journal of the Linnean Society*, 03/2015 2015.

[69] Garrett L Beier, Stan C Hokanson, Scott T Bates, and Robert A Blanchette. Aurantioporthe corni gen. et comb. nov., an endophyte and pathogen of *Cornus alternifolia*. *Mycologia*, 107(1):66–79, 2015.

[70] Michelle M. Risi and Angus H. H. Macdonald. Molecular examination of rocky shore brachycnemic zoantharians (Anthozoa: Hexacorallia) and their Symbiodinium symbionts (Dinophyceae) in the southwest Indian Ocean. *Marine Biodiversity*, 2015.

[71] Serena Zaccara, Stefania Trasforini, Caterina M. Antognazza, Cesare Puzzi, J. Robert Britton, and Giuseppe Crosa. Morphological and genetic characterization of Sardinian trout *Salmo cettii* Rafinesque, 1810 and their conservation implications. *Hydrobiologia*, 2015.

[72] Kari Roesch Goodman, Neal L Evenhuis, Pavla Bartošová-Sojková, and Patrick M O'Grady. Diversification in Hawaiian long-legged flies (Diptera: Dolichopodidae: Campsicnemus): biogeographic isolation and ecological adaptation. *Mol Phylogenet Evol*, 81:232–41, Dec 2014.

[73] Marc S. Appelhans, Jun Wen, and Warren L. Wagner. A molecular phylogeny of Acronychia, Euodia, Melicope and relatives (Rutaceae) reveals polyphyletic genera and key innovations for species richness. *Molecular Phylogenetics and Evolution*, 6/2014 2014.

[74] William P Haines, Patrick Schmitz, and Daniel Rubinoff. Ancient diversification of Hyposmocoma moths in Hawaii. *Nat Commun*, 5:3502, 2014.

[75] Thomas D. Burger, Renfu Shao, and Stephen C. Barker. Phylogenetic analysis of mitochondrial genome sequences indicates that the cattle tick, Rhipicephalus (Boophilus) microplus, contains a cryptic species. *Molecular Phylogenetics and Evolution*, 3/2014 2014.

[76] Kathryn B Walters-Conte, Diana LE Johnson, Warren E Johnson, Stephen J O'Brien, and Jill Pecon-Slattery. The dynamic proliferation of CanSINEs mirrors the complex evolution of Feliforms. *BMC Evolutionary Biology*, 14:137, 2014 2014.

[77] James B. Pettengill, Ruth E. Timme, Rodolphe Barrangou, Magaly Toro, Marc W. Allard, Errol Strain, Steven M. Musser, and Eric W. Brown. The evolutionary history and diagnostic utility of the CRISPR-Cas system within *Salmonella enterica* ssp. *enterica*. *PeerJ*, 2:e340, 2014 2014.

[78] Seán G Brady, Brian L Fisher, Ted R Schultz, and Philip S Ward. The rise of army ants and their relatives: diversification of specialized predatory doryline ants. *BMC Evolutionary Biology*, 14:93, 2014 2014.

[79] J. Zheng, J. Pettengill, E. Strain, M. W. Allard, R. Ahmed, S. Zhao, and E. W. Brown. Genetic diversity and evolution of *Salmonella enterica* serovar Enteritidis strains with different phage types. *Journal of Clinical Microbiology*, 52:1490 – 1500, 05/2014 2014.

[80] M. Hoffmann, S. Zhao, J. Pettengill, Y. Luo, S. R. Monday, J. Abbott, S. L. Ayers, H. N. Cinar, T. Muruvanda, C. Li, M. W. Allard, J. Whichard, J. Meng, E. W. Brown, and P. F. McDermott. Comparative genomic analysis and virulence differences in closely related *Salmonella enterica* serotype Heidelberg isolates from humans, retail meats and animals. *Genome Biology and Evolution*, 2014.

[81] Donald M. Walker, Brandy R. Lawrence, Jessica A. Wooten, Amy Y. Rossman, and Lisa A. Castlebury. Five new species of the highly diverse genus Plagiostoma (Gnomoniaceae, Diaporthales) from Japan. *Mycological Progress*, 2014.

[82] M.C. Neel, K McKelvey, N Ryman, M W Lloyd, R Short Bull, F W Allendorf, M K Schwartz, and R S Waples. Estimation of effective population size in continuously distributed populations: there goes the neighborhood. *Heredity*, 111:189–199, 9/2013 2013.

[83] Michael W. Lloyd, Lesley Campbell, and Maile C. Neel. The power to detect recent fragmentation events using genetic differentiation methods. *PLoS ONE*, 8:e63981, 5/2013 2013.

[84] EA Pettengill, JB Pettengill, and GD Coleman. Elucidating the evolutionary history and expression patterns of nucleoside phosphorylase paralogs (vegetative storage proteins) in *Populus* and the plant kingdom. *BMC Plant Biology*, 13:118, 2013 2013.

[85] N Hobson and MK Deyholos. Genomic and expression analysis of the flax (*Linum usitatissimum*) family of glycosyl hydrolase 35 genes. *BMC Genomics*, 14:344, 2013 2013.

[86] Adam L. Bazinet, Michael P. Cummings, Kim T. Mitter, and Charles W. Mitter. Can RNA-Seq resolve the rapid radiation of advanced moths and butterflies (Hexapoda: Lepidoptera: Apoditrysia)? An exploratory study. *PLoS ONE*, 8(12):e82615, 12 2013.

[87] CS Herrera, AY Rossman, GJ Samuels, and P Chaverri. *Pseudocosmospora*, a new genus to accommodate *Cosmospora vilior* and related species. *Mycologia*, 105:1287–1305, 09/2013 2013.

[88] DV Wasonga and A Channing. Identification of sand frogs (Anura: Pyxicephalidae: *Tomopterna*) from Kenya with the description of two new species. *Zootaxa*, 3734:221, 11/2013 2013.

[89] JA Wooten, CD Camp, JR Combs, E Dulka, A Reist, and DM Walker. Reevaluating niche conservatism versus divergence in the woodland salamander genus *Plethodon*: a case study of the parapatric members of the *Plethodon glutinosus* species complex. *Canadian Journal of Zoology*, pages 883 – 892, 10/2013 2013.

[90] P Chaverri and GJ Samuels. Evolution of habitat preference and nutrition mode in a cosmopolitan fungal genus with evidence of interkingdom host jumps and major shifts in ecology. *Evolution*, pages 2823–2837, 06/2013 2013.

[91] E. Ortiz-Acevedo and K. R. Willmott. Molecular systematics of the butterfly tribe Preponini (Nymphalidae: Charaxinae). *Systematic Entomology*, 38:440–449, 04/2013 2013.

[92] F Zapata. A multilocus phylogenetic analysis of *Escallonia* (Escalloniaceae): Diversification in montane South America. *American Journal of Botany*, 100:526–545, 03/2013 2013.

[93] TD Burger, R Shao, Marcelo BL, and SC Barker. Molecular phylogeny of soft ticks (Ixodida: Argasidae) inferred from mitochondrial genome and nuclear rRNA sequences. *Ticks and Tick-borne Diseases*, 2013.

[94] B Somogyi, T Felföldi, K Solymosi, K Flieger, K Márialigeti, B Böddi, and L Vörös. One step closer to eliminating the nomenclatural problems of minute coccoid green algae: *Pseudochloris wilhelmii*, gen. et sp. nov. (Trebouxiophyceae, Chlorophyta). *European Journal of Phycology*, 48(4):427–436, 2013.

[95] TD Burger, R Shao, and SC Barker. Phylogenetic analysis of the mitochondrial genomes and nuclear rRNA genes of ticks reveals a deep phylogenetic structure within the genus *Haemaphysalis* and further elucidates the polyphyly of the genus *Amblyomma*. *Ticks and Tick-borne Diseases*, 4(4):265 – 274, 2013.

[96] CS Herrera, AY Rossman, GJ Samuels, C Lechat, and P Chaverri. Revision of the genus *Corallomycetella* with *Corallonectria* gen. nov. for *C. jatrophae* (Nectriaceae, Hypocreales). *Mycosystema*, 32:518–544, 2013.

[97] Y Hirooka, AY Rossman, W-Y Zhuang, C Salgado, and P Chaverri. Species delimitation for *Neonectria coccinea* group including the causal agents of beech bark disease (BBD) in Asia, Europe, and North America. *Mycosystema*, 32:485–517, 2013.

[98] JC Regier, C Mitter, MA Solis, JE Hayden, B Landry, M Nuss, TJ Simonsen, S-H Yen, A Zwick, and MP Cummings. A molecular phylogeny for the pyraloid moths (Lepidoptera: Pyraloidea) and its implications for higher-level classification. *Syst Entomol*, 2012.

[99] JC Regier, JW Brown, C Mitter, J Baixeras, S Cho, MP Cummings, and A Zwick. A molecular phylogeny for the leaf-roller moths (Lepidoptera: Tortricidae) and its implications for classification and life history evolution. *PLoS ONE*, 7(4):e35574, 2012.

[100] JC Regier and A Zwick. Sources of signal in 62 protein-coding nuclear genes for higher-level phylogenetics of arthropods. *PLoS ONE*, 6(8):e23408, 2011.

[101] JC Regier, JW Shultz, A Zwick, A Hussey, B Ball, R Wetzer, JW Martin, and CW Cunningham. Arthropod relationships revealed by phylogenomic analysis of nuclear protein-coding sequences. *Nature*, 463(7284):1079–83, Feb 2010.

[102] JC Regier, JW Shultz, ARD Ganley, A Hussey, D Shi, B Ball, A Zwick, JE Stajich, MP Cummings, JW Martin, and CW Cunningham. Resolving arthropod phylogeny: exploring phylogenetic signal within 41 kb of protein-coding nuclear gene sequence. *Syst Biol*, 57(6):920–938, Dec 2008.

[103] NT Joseph, A Aquilina-Beck, C MacDonald, WA Decatur, JA Hall, SI Kavanaugh, and SA Sower. Molecular cloning and pharmacological characterization of two novel GnRH receptors in the lamprey (*Petromyzon marinus*). *Endocrinology*, 153(7):3345–56, Jul 2012.

[104] D Rubinoff, M San Jose, and AY Kawahara. Phylogenetics and species status of Hawai‘i's endangered Blackburn's Sphinx Moth, *Manduca blackburni* (Lepidoptera: Sphingidae). *Pacific Science*, 66:31 – 41, 01/2012 2012.

[105] AY Kawahara and D Rubinoff. Three new species of Fancy Case caterpillars from threatened forests of Hawaii (Lepidoptera, Cosmopterigidae, Hyposmocoma). *Zookeys*, (170):1–20, 2012.

[106] TD Burger, R Shao, L Beati, H Miller, and SC Barker. Phylogenetic analysis of ticks (acari: Ixodida) using mitochondrial genomes and nuclear rRNA genes indicates that the genus *Amblyomma* is polyphyletic. *Mol Phylogenet Evol*, 64(1):45–55, Jul 2012.

[107] CM Hofmann, N Marshall, K Abdilleh, Z Patel, UE Siebeck, and KL Carleton. Opsin evolution in damselfish: Convergence, reversal, and parallel evolution across tuning sites. *Journal of Molecular Evolution*, 75:79–91, 10/2012 2012.

[108] JB Pettengill and DA Moeller. Tempo and mode of mating system evolution between incipient *Clarkia* species. *Evolution*, 66:1210–1225, 04/2012 2012.

[109] DM Walker, LA Castlebury, AY Rossman, and JF White, Jr. New molecular markers for fungal phylogenetics: two genes for species-level systematics in the Sordariomycetes (Ascomycota). *Mol Phylogenet Evol*, 64(3):500–12, Sep 2012.

[110] WP Haines and D Rubinoff. Molecular phylogenetics of the moth genus *Omiodes* Guenée (Crambidae: Spilomelinae), and the origins of the Hawaiian lineage. *Mol Phylogenet Evol*, 65(1):305–16, Oct 2012.

[111] A Zwick, JC Regier, C Mitter, and MP Cummings. Increased gene sampling yields robust support for higher-level clades within Bombycoidea (Lepidoptera). *Syst Entomol*, 36(1):31–43, 2011.

[112] KT Mitter, TB Larsen, W De Prins, S Collins, G. Vande Weghe, S Sáfián, EV Zakharov, DJ Hawthorne, AY Kawahara, and JC Regier. The butterfly subfamily Pseudopontiinae is not monobasic: Marked genetic diversity and morphology reveal three new species of *Pseudopontia* (Lepidoptera: Pieridae). *Syst Entomol*, 36(1):139–163, 2011.

[113] AY Kawahara, I Ohshima, A Kawakita, JC Regier, C Mitter, MP Cummings, DR Davis, DL Wagner, J De Prinis, and C Lopez-Vaamonde. Increased gene sampling provides stronger support for higher-level groups within gracillariid leaf mining moths and relatives (Lepidoptera: Gracillariidae). *BMC Evol Biol*, 11:182, 2011.

[114] S Cho, A Zwick, JC Regier, C Mitter, MP Cummings, J Yao, Z Du, H Zhao, AY Kawahara, S Weller, DR Davis, J Baixeras, JW Brown, and C Parr. Can deliberately incomplete gene sample augmentation improve a phylogeny estimate for the advanced moths and butterflies (Hexapoda: Lepidoptera)? *Syst Biol*, 60:782–796, 2011.

[115] Y Hirooka, A Y Rossman, and P Chaverri. A morphological and phylogenetic revision of the *Nectria cinnabarina* species complex. *Stud Mycol*, 68:35–56, 2011.

[116] Y. Hirooka, A. Y. Rossman, G. J. Samuels, C. Lechat, and P. Chaverri. A monograph of Allantonectria, Nectria, and Pleonectria (Nectriaceae, Hypocreales, Ascomycota) and their pycnidial, sporodochial, and synnematous anamorphs. *Studies in Mycology*, 71:1–210, 03/2012 2012.

[117] P Chaverri, C Salgado, Y Hirooka, A Y Rossman, and G J Samuels. Delimitation of *Neonectria* and *Cylindrocarpon* (Nectriaceae, Hypocreales, Ascomycota) and related genera with Cylindrocarpon-like anamorphs. *Stud Mycol*, 68:57–78, 2011.

[118] P Cárdenas, JR Xavier, J Reveillaud, C Schander, and HT Rapp. Molecular phylogeny of the Astrophorida (Porifera, Demospongiae) reveals an unexpected high level of spicule homoplasy. *PLoS ONE*, 6(4):e18318, 2011.

[119] JB Pettengill and MC Neel. A sequential approach using genetic and morphological analyses to test species status: the case of United States federally endangered *Agalinis acuta* (Orobanchaceae). *Am J Bot*, 98(5):859–71, May 2011.

[120] DRG Price, RP Duncan, S Shigenobu, and ACC Wilson. Genome expansion and differential expression of amino acid transporters at the aphid *Buchnera* symbiotic interface. *Mol Biol Evol*, 28(11):3113–26, Nov 2011.

[121] M Tatián, C Lagger, M Demarchi, and C Mattoni. Molecular phylogeny endorses the relationship between carnivorous and filter-feeding tunicates (Tunicata, Ascidiacea). *Zool Scr*, 40:603 – 612, 11/2011 2011.

[122] MP Lesser, KL Carleton, SA Böttger, TM Barry, and CW Walker. Sea urchin tube feet are photosensory organs that express a rhabdomeric-like opsin and PAX6. *Proc Biol Sci*, 278(1723):3371–9, Nov 2011.

[123] S Martén-Rodríguez, CB Fenster, I Agnarsson, LE Skog, and EA Zimmer. Evolutionary breakdown of pollination specialization in a Caribbean plant radiation. *New Phytol*, 188(2):403–17, Oct 2010.

[124] ME Reyna-Fabian, JP Laclette, MP Cummings, and M García-Varela. Validating the systematic position of *Plationus* Segers, Murugan & Dumont, 1993 (Rotifera: Brachionidae) using sequences of the large subunit of the nuclear

ribosomal DNA and of cytochrome C oxidase. *Hydrobiologia*, 644(1):361–370, May 2010.

[125] K Uchida, S Moriyama, H Chiba, T Shimotani, K Honda, M Miki, A Takahashi, SA Sower, and M Nozaki. Evolutionary origin of a functional gonadotropin in the pituitary of the most primitive vertebrate, hagfish. *Proc Natl Acad Sci*, 107(36):15832–15837, 2010.

[126] AY Kawahara, AA Mignault, JC Regier, IJ Kitching, and C Mitter. Phylogeny and biogeography of hawkmoths (Lepidoptera: Sphingidae): Evidence from five nuclear genes. *PloS ONE*, 4(5), 2009.

[127] JB Pettengill and MC Neel. Phylogenetic patterns and conservation among North American members of the genus *Agalinis* (Orobanchaceae). *BMC Evol Biol*, 8, 2008.

[128] MC Neel and MP Cummings. Section-level relationships of North American *Agalinis* (Orobanchaceae) based on DNA sequence analysis of three chloroplast gene regions. *BMC Evol Biol*, 4:15, Jun 2004.

[129] MP Cummings, SA Handley, DS Myers, DL Reed, A Rokas, and K Winka. Comparing bootstrap and posterior probability values in the four-taxon case. *Syst Biol*, 52(4):477–487, Aug 2003.

[130] DS Myers and MP Cummings. Necessity is the mother of invention: a simple grid computing system using commodity tools. *J Parallel Distr Com*, 63(5):578–589, May 2003.

[131] Dennis A Benson, Ilene Karsch-Mizrachi, Karen Clark, David J Lipman, James Ostell, and Eric W Sayers. Genbank. *Nucleic Acids Res*, 40(Database issue):D48–53, Jan 2012.

[132] Robert D. Finn, Jaina Mistry, John Tate, Penny Coggill, Andreas Heger, Joanne E. Pollington, O. Luke Gavin, Prasad Gunasekaran, Goran Ceric, Kristoffer Forslund, Liisa Holm, Erik L. L. Sonnhammer, Sean R. Eddy, and Alex Bateman. The Pfam protein families database. *Nucleic Acids Research*, 38(suppl 1):D211–D222, 2010.

[133] Andrey Kislyuk, Srijak Bhatnagar, Jonathan Dushoff, and Joshua S Weitz. Unsupervised statistical clustering of environmental shotgun sequences. *BMC Bioinformatics*, 10:316, 2009.

[134] Sourav Chatterji, Ichitaro Yamazaki, Zhaojun Bai, and Jonathan A. Eisen. CompostBin: a DNA composition-based algorithm for binning environmental shotgun reads. In *Proceedings of the 12th annual international conference on Research in computational molecular biology*, RECOMB'08, pages 17–28, Berlin, Heidelberg, 2008. Springer-Verlag.

[135] David Kelley and Steven Salzberg. Clustering metagenomic sequences with interpolated Markov models. *BMC Bioinformatics*, 11(1):544, 2010.

[136] T Z DeSantis, P Hugenholtz, N Larsen, M Rojas, E L Brodie, K Keller, T Huber, D Dalevi, P Hu, and G L Andersen. Greengenes, a chimera-checked 16S rRNA gene database and workbench compatible with ARB. *Appl Environ Microbiol*, 72(7):5069–72, Jul 2006.

[137] J R Cole, Q Wang, E Cardenas, J Fish, B Chai, R J Farris, A S Kulam-Syed-Mohideen, D M McGarrell, T Marsh, G M Garrity, and J M Tiedje. The Ribosomal Database Project: improved alignments and new tools for rRNA analysis. *Nucleic Acids Res*, 37(Database issue):D141–5, Jan 2009.

[138] Xiang Jia Min and Donal A Hickey. DNA barcodes provide a quick preview of mitochondrial genome composition. *PLoS One*, 2(3):e325, 2007.

[139] CBOL Plant Working Group. A DNA barcode for land plants. *Proc Natl Acad Sci U S A*, 106(31):12794–7, Aug 2009.

[140] Francesca D Ciccarelli, Tobias Doerks, Christian von Mering, Christopher J Creevey, Berend Snel, and Peer Bork. Toward automatic reconstruction of a highly resolved tree of life. *Science*, 311(5765):1283–7, Mar 2006.

[141] Les Dethlefsen, Sue Huse, Mitchell L Sogin, and David A Relman. The pervasive effects of an antibiotic on the human gut microbiota, as revealed by deep 16S rRNA sequencing. *PLoS Biol*, 6(11):e280, Nov 2008.

[142] S F Altschul, T L Madden, A A Schäffer, J Zhang, Z Zhang, W Miller, and D J Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res*, 25(17):3389–402, Sep 1997.

[143] Lutz Krause, Naryttza N Diaz, Alexander Goesmann, Scott Kelley, Tim W Nattkemper, Forest Rohwer, Robert A Edwards, and Jens Stoye. Phylogenetic classification of short environmental DNA fragments. *Nucleic Acids Res*, 36(7):2230–9, Apr 2008.

[144] Wolfgang Gerlach and Jens Stoye. Taxonomic classification of metagenomic shotgun sequences with CARMA3. *Nucleic Acids Res*, 39(14):e91, Aug 2011.

[145] Henrik Stranneheim, Max Kaller, Tobias Allander, Bjorn Andersson, Lars Arvestad, and Joakim Lundeberg. Classification of DNA sequences using Bloom filters. *Bioinformatics*, 26(13):1595–1600, July 2010.

[146] Martin Jones, Anisah Ghoorah, and Mark Blaxter. jMOTU and Taxonerator: turning DNA barcode sequences into annotated operational taxonomic units. *PLoS One*, 6(4):e19259, 2011.

[147] Matthew Horton, Natacha Bodenhausen, and Joy Bergelson. MARTA: a suite of Java-based tools for assigning taxonomic status to DNA sequences. *Bioinformatics*, 26(4):568–9, Feb 2010.

[148] Daniel H Huson, Alexander F Auch, Ji Qi, and Stephan C Schuster. MEGAN: analysis of metagenomic data. *Genome Res*, 17(3):377–86, Mar 2007.

[149] Bo 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, dec. 2010.

[150] Elizabeth M Glass, Jared Wilkening, Andreas Wilke, Dionysios Antonopoulos, and Folker Meyer. Using the metagenomics RAST server (MG-RAST) for analyzing shotgun metagenomes. *Cold Spring Harb Protoc*, 2010(1):pdb.prot5368, Jan 2010.

[151] Fabio Gori, Gianluigi Folino, Mike S M Jetten, and Elena Marchiori. MTR: taxonomic annotation of short metagenomic reads using clustering at multiple taxonomic ranks. *Bioinformatics*, 27(2):196–203, Jan 2011.

[152] M Monzoorul Haque, Tarini Shankar Ghosh, Dinakar Komanduri, and Sharmila S Mande. SOrt-ITEMS: Sequence orthology based approach for improved taxonomic estimation of metagenomic sequences. *Bioinformatics*, 25(14):1722–30, Jul 2009.

[153] Gail Rosen, Elaine Garbarine, Diamantino Caseiro, Robi Polikar, and Bahrad Sokhansanj. Metagenome fragment classification using N-mer frequency profiles. *Adv Bioinformatics*, 2008:205969, 2008.

[154] Gail L Rosen, Erin R Reichenberger, and Aaron M Rosenfeld. NBC: the Naive Bayes Classification tool webserver for taxonomic classification of metagenomic reads. *Bioinformatics*, 27(1):127–9, Jan 2011.

[155] Alice Carolyn McHardy, Héctor García Martín, Aristotelis Tsirigos, Philip Hugenholtz, and Isidore Rigoutsos. Accurate phylogenetic classification of variable-length DNA fragments. *Nat Methods*, 4(1):63–72, Jan 2007.

[156] Kaustubh R Patil, Peter Haider, Phillip B Pope, Peter J Turnbaugh, Mark Morrison, Tobias Scheffer, and Alice C McHardy. Taxonomic metagenome sequence assignment with structured output models. *Nat Methods*, 8(3):191–2, Mar 2011.

[157] Arthur Brady and Steven L. Salzberg. Phymm and PhymmBL: metagenomic phylogenetic classification with interpolated Markov models. *Nature Methods*, 6(9):673–U68, September 2009.

[158] Ozkan U Nalbantoglu, Samuel F Way, Steven H Hinrichs, and Khalid Sayood. RAIphy: phylogenetic classification of metagenomics samples using iterative

refinement of relative abundance index profiles. *BMC Bioinformatics*, 12:41, 2011.

[159] Qiong Wang, George M Garrity, James M Tiedje, and James R Cole. Naive Bayesian classifier for rapid assignment of rRNA sequences into the new bacterial taxonomy. *Appl Environ Microbiol*, 73(16):5261–7, Aug 2007.

[160] Monzoorul Haque Mohammed, Tarini Shankar Ghosh, Nitin Kumar Singh, and Sharmila S Mande. SPHINX–an algorithm for taxonomic binning of metagenomic sequences. *Bioinformatics*, 27(1):22–30, Jan 2011.

[161] Naryttza N Diaz, Lutz Krause, Alexander Goesmann, Karsten Niehaus, and Tim W Nattkemper. TACOA: taxonomic classification of environmental genomic fragments using a kernelized nearest neighbor approach. *BMC Bioinformatics*, 10:56, 2009.

[162] Simon A Berger, Denis Krompass, and Alexandros Stamatakis. Performance, accuracy, and web server for evolutionary placement of short sequence reads under maximum likelihood. *Syst Biol*, 60(3):291–302, May 2011.

[163] Morgan N Price, Paramvir S Dehal, and Adam P Arkin. FastTree: computing large minimum evolution trees with profiles instead of a distance matrix. *Mol Biol Evol*, 26(7):1641–50, Jul 2009.

[164] Frederick A Matsen, Robin B Kodner, and E Virginia Armbrust. pplacer: linear time maximum-likelihood and Bayesian phylogenetic placement of sequences onto a fixed reference tree. *BMC Bioinformatics*, 11:538, 2010.

[165] Martin Wu and Jonathan A Eisen. A simple, fast, and accurate method of phylogenomic inference. *Genome Biol*, 9(10):R151, 2008.

[166] Manuel Stark, Simon A Berger, Alexandros Stamatakis, and Christian von Mering. MLTreeMap–accurate maximum likelihood placement of environmental DNA sequences into taxonomic and functional reference phylogenies. *BMC Genomics*, 11:461, 2010.

[167] Fabian Schreiber, Peter Gumrich, Rolf Daniel, and Peter Meinicke. Treephyler: fast taxonomic profiling of metagenomes. *Bioinformatics*, 26(7):960–1, Apr 2010.

[168] Kasper Munch, Wouter Boomsma, John P Huelsenbeck, Eske Willerslev, and Rasmus Nielsen. Statistical assignment of DNA sequences using Bayesian phylogenetics. *Syst Biol*, 57(5):750–7, Oct 2008.

[169] Kaustubh R. Patil, Peter Haider, Phillip B. Pope, Peter J. Turnbaugh, Mark Morrison, Tobias Scheffer, and Alice C. McHardy. Taxonomic metagenome sequence assignment with structured output models. *Nature Methods*, 8(3):191–192, March 2011.

[170] Konstantinos Mavromatis, Natalia Ivanova, Kerrie Barry, Harris Shapiro, Eugene Goltsman, Alice C McHardy, Isidore Rigoutsos, Asaf Salamov, Frank Korzeniewski, Miriam Land, Alla Lapidus, Igor Grigoriev, Paul Richardson, Philip Hugenholtz, and Nikos C Kyrpides. Use of simulated data sets to evaluate the fidelity of metagenomic processing methods. *Nat Methods*, 4(6):495–500, Jun 2007.

[171] Kim D Pruitt, Tatiana Tatusova, William Klimke, and Donna R Maglott. NCBI Reference Sequences: current status, policy and new initiatives. *Nucleic Acids Res*, 37(Database issue):D32–6, Jan 2009.

[172] Gerard Talavera and Jose Castresana. Improvement of phylogenies after removing divergent and ambiguously aligned blocks from protein sequence alignments. *Syst Biol*, 56(4):564–77, Aug 2007.

[173] Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* Cambridge University Press, July 1999.

[174] Alexandros Stamatakis. RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. *Bioinformatics*, 22(21):2688–90, Nov 2006.

[175] Peter Meinicke. UFO: a web server for ultra-fast functional profiling of whole genome protein sequences. *BMC Genomics*, 10:409, 2009.

[176] Joseph Felsenstein. PHYLIP - Phylogeny Inference Package (Version 3.2). *Cladistics*, 5:164–166, 1989.

[177] A Rambaut. http://tree.bio.ed.ac.uk/software/figtree/.

[178] The R project for statistical computing. http://www.r-project.org/, 2015.

[179] Nicholas J. Miller, Jing Sun, and Thomas W. Sappington. High-throughput transcriptome sequencing for SNP and gene discovery in a moth. *Environ Entomol*, 41(4):997–1007, 2013/02/07 2012.

[180] Daniel H Huson, Suparna Mitra, Hans-Joachim Ruscheweyh, Nico Weber, and Stephan C Schuster. Integrative analysis of environmental sequences using MEGAN4. *Genome Res*, 21(9):1552–60, Sep 2011.

[181] W M Fitch. Homology: a personal view on some of the problems. *Trends Genet*, 16(5):227–31, May 2000.

[182] Eugene V Koonin. Orthologs, paralogs, and evolutionary genomics. *Annu Rev Genet*, 39:309–38, 2005.

[183] Susumu Ohno. *Evolution by Gene Duplication.* Berlin: Springer-Verlag, 1970.

[184] M Pilar Francino. An adaptive radiation model for the origin of new gene functions. *Nat Genet*, 37(6):573–7, Jun 2005.

[185] D Wheeler, R Hope, S B Cooper, G Dolman, G C Webb, C D Bottema, A A Gooley, M Goodman, and R A Holland. An orphaned mammalian beta-globin gene of ancient evolutionary origin. *Proc Natl Acad Sci U S A*, 98(3):1101–6, Jan 2001.

[186] Wayne P. Maddison. Gene trees in species trees. *Syst Biol*, 46(3):523–536, 1997.

[187] Minsheng You, Zhen Yue, Weiyi He, Xinhua Yang, Guang Yang, Miao Xie, Dongliang Zhan, Simon W Baxter, Liette Vasseur, Geoff M Gurr, Carl J Douglas, Jianlin Bai, Ping Wang, Kai Cui, Shiguo Huang, Xianchun Li, Qing Zhou, Zhangyan Wu, Qilin Chen, Chunhui Liu, Bo Wang, Xiaojing Li, Xiufeng Xu, Changxin Lu, Min Hu, John W Davey, Sandy M Smith, Mingshun Chen, Xiaofeng Xia, Weiqi Tang, Fushi Ke, Dandan Zheng, Yulan Hu, Fengqin Song, Yanchun You, Xiaoli Ma, Lu Peng, Yunkai Zheng, Yong Liang, Yaqiong Chen, Liying Yu, Younan Zhang, Yuanyuan Liu, Guoqing Li, Lin Fang, Jingxiang Li, Xin Zhou, Yadan Luo, Caiyun Gou, Junyi Wang, Jian Wang, Huanming Yang, and Jun Wang. A heterozygous moth genome provides insights into herbivory and detoxification. *Nat Genet*, 45(2):220–5, Feb 2013.

[188] P Bork, T Dandekar, Y Diaz-Lazcoz, F Eisenhaber, M Huynen, and Y Yuan. Predicting function: from genes to genomes and back. *J Mol Biol*, 283(4):707–25, Nov 1998.

[189] R L Tatusov, E V Koonin, and D J Lipman. A genomic perspective on protein families. *Science*, 278(5338):631–7, Oct 1997.

[190] D P Wall, H B Fraser, and A E Hirsh. Detecting putative orthologs. *Bioinformatics*, 19(13):1710–1, Sep 2003.

[191] A Cornish-Bowden. Nomenclature for incompletely specified bases in nucleic acid sequences: recommendations 1984. *Nucleic Acids Res*, 13(9):3021–30, May 1985.

[192] Devin R Scannell, Kevin P Byrne, Jonathan L Gordon, Simon Wong, and Kenneth H Wolfe. Multiple rounds of speciation associated with reciprocal gene loss in polyploid yeasts. *Nature*, 440(7082):341–5, Mar 2006.

[193] Li Li, Christian J Stoeckert, Jr, and David S Roos. OrthoMCL: identification of ortholog groups for eukaryotic genomes. *Genome Res*, 13(9):2178–89, Sep 2003.

[194] Kevin P Byrne and Kenneth H Wolfe. The Yeast Gene Order Browser: combining curated homology and syntenic context reveals gene fate in polyploid species. *Genome Res*, 15(10):1456–61, Oct 2005.

[195] D. L. Swofford. Phylogenetic analysis using parsimony (*and other methods). Sinauer Associates, Sunderland, Massachusetts., 2003.

[196] Antonis Rokas, Barry L Williams, Nicole King, and Sean B Carroll. Genome-scale approaches to resolving incongruence in molecular phylogenies. *Nature*, 425(6960):798–804, Oct 2003.

[197] Hidetoshi Shimodaira. An approximately unbiased test of phylogenetic tree selection. *Syst Biol*, 51(3):492–508, Jun 2002.

[198] Derrick J. Zwickl. GARLI 2.0 https://www.nescent.org/wg_garli/main_page, April 2011.

[199] Joseph Felsenstein. PHYLIP (phylogeny inference package) version 3.6. Distributed by the author., 2005.

[200] Sebastian Proost, Michiel Van Bel, Lieven Sterck, Kenny Billiau, Thomas Van Parys, Yves Van de Peer, and Klaas Vandepoele. PLAZA: a comparative genomics resource to study gene and genome evolution in plants. *Plant Cell*, 21(12):3718–31, Dec 2009.

[201] M.A. Miller, W. Pfeiffer, and T. Schwartz. Creating the CIPRES science gateway for inference of large phylogenetic trees. In *Gateway Computing Environments Workshop (GCE), 2010*, pages 1–8, nov. 2010.

[202] Surendra Kumar, Asmund Skjaeveland, Russell J S Orr, Pål Enger, Torgeir Ruden, Bjørn-Helge Mevik, Fabien Burki, Andreas Botnen, and Kamran Shalchian-Tabrizi. AIR: A batch-oriented web program package for construction of supermatrices ready for phylogenomic analyses. *BMC Bioinformatics*, 10:357, 2009.

[203] Rubén Sánchez, François Serra, Joaquín Tárraga, Ignacio Medina, José Carbonell, Luis Pulido, Alejandro de María, Salvador Capella-Gutíerrez, Jaime Huerta-Cepas, Toni Gabaldón, Joaquín Dopazo, and Hernán Dopazo. Phylemon 2.0: a suite of web-tools for molecular evolution, phylogenetics, phylogenomics and hypotheses testing. *Nucleic Acids Res*, 39(Web Server issue):W470–4, Jul 2011.

[204] Bertrand Néron, Hervé Ménager, Corinne Maufrais, Nicolas Joly, Julien Maupetit, Sébastien Letort, Sébastien Carrere, Pierre Tuffery, and Catherine Letondal. Mobyle: a new full web bioinformatics framework. *Bioinformatics*, 25(22):3005–11, Nov 2009.

[205] Barath Raghavan and Justin Ma. The energy and emergy of the Internet. In *HotNets*, page 9, 2011.

[206] Paul O. Lewis. NCL: a C++ class library for interpreting data files in NEXUS format. *Bioinformatics*, 19(17):2330–2331, 2003.

[207] D. R. Robinson and L. R. Foulds. Comparison of phylogenetic trees. *Math Biosci*, 53:131–147, 1981.

[208] Jeet Sukumaran and Mark T Holder. DendroPy: a Python library for phylogenetic computing. *Bioinformatics*, 26(12):1569–71, Jun 2010.

[209] S B Hedges. The number of replications needed for accurate estimation of the bootstrap P value in phylogenetic studies. *Mol Biol Evol*, 9(2):366–9, Mar 1992.

[210] Nicholas D Pattengale, Masoud Alipour, Olaf R P Bininda-Emonds, Bernard M E Moret, and Alexandros Stamatakis. How many bootstrap replicates are necessary? *J Comput Biol*, 17(3):337–54, Mar 2010.

[211] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[212] Leo Breiman and Leo Breiman. Bagging predictors. In *Machine Learning*, pages 123–140, 1996.

[213] C. Glasner and J. Volkert. An architecture for an adaptive run-time prediction system. In *Proceedings of the 7th International Symposium on Parallel and Distributed Computing (ISPDC'08)*, 2008.

[214] H. Li, D. Groep, and L. Wolters. An evaluation of learning and heuristic techniques for application run time predictions. In *Proceedings of 11th Annual Conference of the Advance School for Computing and Imaging (ASCI)*, 2005.

[215] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. In *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, pages 3–24, Amsterdam, The Netherlands, The Netherlands, 2007. IOS Press.

[216] Leo Breiman. Arcing classifier (with discussion and a rejoinder by the author). *The Annals of Statistics*, 26(3):801–849, 06 1998.

[217] Leo Breiman. Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical Science*, 16(3):199–231, 08 2001.

[218] Bradley Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 1993.

[219] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer Series in Statistics. Springer, 2009.

[220] L Breiman, A Cutler, A Liaw, and M Wiener. randomForest. http://cran.r-project.org/web/packages/randomForest/, 2015.

[221] GT 4.0 component fact sheet: Web Service Grid Resource Allocation and Management (WS GRAM). `http://globus.org/toolkit/docs/4.0/execution/wsgram/WSGRAMFacts.html`. [Online; accessed 04-August-2014].

[222] DS Myers, AL Bazinet, and MP Cummings. Expanding the reach of Grid computing: combining Globus- and BOINC-based systems. In E-G Talbi and AY Zomaya, editors, *Grids for Bioinformatics and Computational Biology*, Wiley Book Series on Bioinformatics: Computational Techniques and Engineering, chapter 4, pages 71–85. Wiley-Interscience, Hoboken, 2008.

[223] AL Bazinet and MP Cummings. The Lattice Project: a Grid research and production environment combining multiple Grid computing models. In MHW Weber, editor, *Distributed & Grid Computing — Science Made Transparent for Everyone. Principles, Applications and Supporting Communities*, chapter 1, pages 2–13. Rechenkraft.net, Marburg, 2008.

[224] D.P. Anderson. Emulating volunteer computing scheduling policies. In *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, pages 1839–1846, May 2011.

[225] Derrick Kondo, David P. Anderson, and John Mcleod Vii. Performance evaluation of scheduling policies for volunteer computing.

[226] T. Estrada, D.A. Flores, M. Taufer, P.J. Teller, A. Kerstens, and D.P. Anderson. The effectiveness of threshold-based scheduling policies in boinc projects. In *e-Science and Grid Computing, 2006. e-Science '06. Second IEEE International Conference on*, pages 88–88, Dec 2006.

[227] E.M. Heien, N. Fujimoto, and K. Hagihara. Computing low latency batches with unreliable workers in volunteer computing environments. In *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pages 1–8, April 2008.

[228] Brant C Faircloth, John E McCormack, Nicholas G Crawford, Michael G Harvey, Robb T Brumfield, and Travis C Glenn. Ultraconserved elements anchor thousands of genetic markers spanning multiple evolutionary timescales. *Syst Biol*, 61(5):717–26, Oct 2012.

[229] John E McCormack, Brant C Faircloth, Nicholas G Crawford, Patricia Adair Gowaty, Robb T Brumfield, and Travis C Glenn. Ultraconserved elements are novel phylogenomic markers that resolve placental mammal phylogeny when combined with species-tree analysis. *Genome Res*, 22(4):746–54, Apr 2012.

[230] John E McCormack, Sarah M Hird, Amanda J Zellmer, Bryan C Carstens, and Robb T Brumfield. Applications of next-generation sequencing to phylogeography and phylogenetics. *Mol Phylogenet Evol*, 66(2):526–38, Feb 2013.

[231] van Nieukerken EJ, Kaila L, Kitching IJ, Kristensen NP, Lees DC, and et al. *Animal Biodiversity: An outline of higher level classification and survey of taxonomic richness.*, volume 3148, chapter Order Lepidoptera Linnaeus, 1758., pages 212–221. Zootaxa, 2011.

[232] Wagner DL. *Encyclopedia of Biodiversity.*, chapter Moths., pages 249–270. Academic Press, San Diego, CA., 2001.

[233] Roe A, Weller S, Baixeras J, Brown JW, Cummings MP, Davis DR, Horak M, Kawahara AY, Mitter C, Parr CS, Regier JC, Rubinoff D, Simonsen TJ, Wahlberg N, and Zwick A. *Genetics and Molecular Biology of Lepidoptera*, chapter Evolutionary framework for Lepidoptera model systems, pages 1–24. Taylor & Francis, Boca Raton, 2010.

[234] Paul R. Ehrlich and Peter H. Raven. Butterflies and plants: A study in coevolution. *Evolution*, 18(4):pp. 586–608, 1964.

[235] Charles Mitter, Brian Farrell, and Brian Wiegmann. The phylogenetic study of adaptive zones: Has phytophagy promoted insect diversification? *The American Naturalist*, 132(1):pp. 107–128, 1988.

[236] Isaac S. Winkler and Charles Mitter. The phylogenetic dimension of insect-plant interactions: A review of recent evidence. In *Specialization, Speciation, and Radiation*, pages –. University of California Press, 2008.

[237] Lauri Kaila, Marko Mutanen, and Tommi Nyman. Phylogeny of the mega-diverse Gelechioidea (Lepidoptera): adaptations and determinants of success. *Mol Phylogenet Evol*, 61(3):801–9, Dec 2011.

[238] Young CJ. Molecular relationships of the Australian Ennominae (Lepidoptera: Geometridae) and implications for the phylogeny of the Geometridae from molecular and morphological data. *Zootaxa*, 1264:1–147, 2006.

[239] Satoshi Yamamoto and Teiji Sota. Phylogeny of the Geometridae and the evolution of winter moths inferred from a simultaneous analysis of mitochondrial and nuclear genes. *Mol Phylogenet Evol*, 44(2):711–23, Aug 2007.

[240] Pasi Sihvonen, Marko Mutanen, Lauri Kaila, Gunnar Brehm, Axel Hausmann, and Hermann S Staude. Comprehensive molecular sampling yields a robust phylogeny for geometrid moths (Lepidoptera: Geometridae). *PLoS One*, 6(6):e20356, 2011.

[241] Andrew Mitchell, Charles Mitter, and Jerome C. Regier. Systematics and evolution of the cutworm moths (Lepidoptera: Noctuidae): evidence from two protein-coding nuclear genes. *Systematic Entomology*, 31(1):21–46, 2006.

[242] Reza Zahiri, Ian J. Kitching, J. Donald Lafontaine, Marko Mutanen, Lauri Kaila, Jeremy D. Holloway, and Niklas Wahlberg. A new molecular phylogeny

offers hope for a stable family level classification of the Noctuoidea (Lepidoptera). *Zoologica Scripta*, 40(2):158–173, 2011.

[243] Maria Heikkilä, Lauri Kaila, Marko Mutanen, Carlos Peña, and Niklas Wahlberg. Cretaceous origin and repeated tertiary diversification of the redefined butterflies. *Proc Biol Sci*, 279(1731):1093–9, Mar 2012.

[244] Marko Mutanen, Niklas Wahlberg, and Lauri Kaila. Comprehensive gene and taxon coverage elucidates radiation patterns in moths and butterflies. *Proceedings of the Royal Society B: Biological Sciences*, 277(1695):2839–2848, 2010.

[245] N. P. Kristensen. Lepidoptera, moths and butterflies. vol. 2: Morphology, physiology, and development. In M. Fischer., editor, *Handbook of Zoology 4. Handbook of Zoology 4. Arthropoda: Insecta, part 36.* Walter de Gruyter, Berlin & New York., 2003.

[246] Hui-Fen Lu, Tian-Juan Su, A-Rong Luo, Chao-Dong Zhu, and Chun-Sheng Wu. Characterization of the complete mitochondrion genome of diurnal moth *Amata emma* (Butler) (Lepidoptera: Erebidae) and its phylogenetic implications. *PLoS One*, 8(9):e72410, 2013.

[247] N. P. Kristensen and E. S. Nielsen. The Heterobathmia life history elucidated: Immature stages contradict assignment to suborder Zeugloptera (Insecta, Lepidoptera). *Journal of Zoological Systematics and Evolutionary Research*, 21(2):101–124, 1983.

[248] N.P. Kristensen. *Studies on the morphology and systematics of primitive Lepidoptera (Insecta).* Zoological Museum, 1984.

[249] Davis DR. A new family of monotrysian moths from austral South America (Lepidoptera: Palaephatidae), with a phylogenetic review of the Monotrysia. *Smithsonian Contributions to Zoology*, 434:1–202, 1986.

[250] Kristensen NP, editor. *Handbook of Zoology 4. Lepidoptera, moths and butterflies.* Walter de Gruyter, Berlin & New York., 1998.

[251] Kristensen NP and Skalski AW. Phylogeny and palaeontology. In Kristensen NP, editor, *Handbook of Zoology 4. Lepidoptera, moths and butterflies.*, pages 7–25. Walter de Gruyter, Berlin & New York., 1998.

[252] Minet J. Ebauche d'une classification modern de l'ordre des Lepidopteres. *Alexanor*, 14:291–313, 1986.

[253] Joel Minet. Tentative reconstruction of the ditrysian phylogeny (Lepidoptera: Glossata). *Insect Systematics & Evolution*, 22(1):69–95, 1991.

[254] Lauri Kaila. Phylogeny of the superfamily Gelechioidea (Lepidoptera: Ditrysia): an exemplar approach. *Cladistics*, 20(4):303–340, 2004.

[255] A.J. Aberer and A. Stamatakis. A simple and accurate method for rogue taxon identification. In *Bioinformatics and Biomedicine (BIBM), 2011 IEEE International Conference on*, pages 118–122, Nov 2011.

[256] Min Jee Kim, Ah Rang Kang, Heon Cheon Jeong, Ki-Gyoung Kim, and Iksoo Kim. Reconstructing intraordinal relationships in Lepidoptera using mitochondrial genome data with the description of two newly sequenced lycaenids, *Spindasis takanonis* and *Protantigius superans* (Lepidoptera: Lycaenidae). *Mol Phylogenet Evol*, 61(2):436–45, Nov 2011.

[257] James B Whitfield and Peter J Lockhart. Deciphering ancient rapid radiations. *Trends Ecol Evol*, 22(5):258–65, May 2007.

[258] Brian M Wiegmann, Michelle D Trautwein, Isaac S Winkler, Norman B Barr, Jung-Wook Kim, Christine Lambkin, Matthew A Bertone, Brian K Cassel, Keith M Bayless, Alysha M Heimberg, Benjamin M Wheeler, Kevin J Peterson, Thomas Pape, Bradley J Sinclair, Jeffrey H Skevington, Vladimir Blagoderov, Jason Caravas, Sujatha Narayanan Kutty, Urs Schmidt-Ott, Gail E Kampmeier, F Christian Thompson, David A Grimaldi, Andrew T Beckenbach, Gregory W Courtney, Markus Friedrich, Rudolf Meier, and David K Yeates. Episodic radiations in the fly tree of life. *Proc Natl Acad Sci U S A*, 108(14):5690–5, Apr 2011.

[259] Jun Duan, Ruiqiang Li, Daojun Cheng, Wei Fan, Xingfu Zha, Tingcai Cheng, Yuqian Wu, Jun Wang, Kazuei Mita, Zhonghuai Xiang, and Qingyou Xia. SilkDB v2.0: a platform for silkworm (*Bombyx mori*) genome biology. *Nucleic Acids Res*, 38(Database issue):D453–6, Jan 2010.

[260] K S Pick, H Philippe, F Schreiber, D Erpenbeck, D J Jackson, P Wrede, M Wiens, A Alié, B Morgenstern, M Manuel, and G Wörheide. Improved phylogenomic taxon sampling noticeably affects nonbilaterian relationships. *Mol Biol Evol*, 27(9):1983–7, Sep 2010.

[261] Hervé Philippe, Henner Brinkmann, Dennis V Lavrov, D Timothy J Littlewood, Michael Manuel, Gert Wörheide, and Denis Baurain. Resolving difficult phylogenetic questions: why more sequences are not enough. *PLoS Biol*, 9(3):e1000602, Mar 2011.

[262] John J Wiens and Jonathan Tiu. Highly incomplete taxa can rescue phylogenetic analyses from the negative impacts of limited taxon sampling. *PLoS One*, 7(8):e42925, 2012.

[263] Sujeevan Ratnasingham and Paul D N Hebert. bold: The Barcode of Life Data System (http://www.barcodinglife.org). *Mol Ecol Notes*, 7(3):355–364, May 2007.

[264] Brent Ewing and Phil Green. Base-calling of automated sequencer traces using Phred. II. Error probabilities. *Genome Research*, 8(3):186–194, 1998.

[265] Kasper D Hansen, Steven E Brenner, and Sandrine Dudoit. Biases in Illumina transcriptome sequencing caused by random hexamer priming. *Nucleic Acids Res*, 38(12):e131, Jul 2010.

[266] Inanç Birol, Shaun D Jackman, Cydney B Nielsen, Jenny Q Qian, Richard Varhol, Greg Stazyk, Ryan D Morin, Yongjun Zhao, Martin Hirst, Jacqueline E Schein, Doug E Horsman, Joseph M Connors, Randy D Gascoyne, Marco A Marra, and Steven J M Jones. *De novo* transcriptome assembly with ABySS. *Bioinformatics*, 25(21):2872–7, Nov 2009.

[267] Nadia Kermani, Zainal-Abidin Abu-Hassan, Hamady Dieng, Noor Farehan Ismail, Mansour Attia, and Idris Abd Ghani. Pathogenicity of *Nosema sp.* (Microsporidia) in the diamondback moth, *Plutella xylostella* (Lepidoptera: Plutellidae). *PLoS One*, 8(5):e62884, 2013.

[268] Ann-Charlotte Berglund, Erik Sjölund, Gabriel Ostlund, and Erik L L Sonnhammer. InParanoid 6: eukaryotic ortholog clusters with inparalogs. *Nucleic Acids Res*, 36(Database issue):D263–6, Jan 2008.

[269] Feng Chen, Aaron J Mackey, Christian J Stoeckert, Jr, and David S Roos. OrthoMCL-DB: querying a comprehensive multi-species collection of ortholog groups. *Nucleic Acids Res*, 34(Database issue):D363–8, Jan 2006.

[270] Gabriel Ostlund, Thomas Schmitt, Kristoffer Forslund, Tina Köstler, David N Messina, Sanjit Roopra, Oliver Frings, and Erik L L Sonnhammer. InParanoid 7: new algorithms and tools for eukaryotic orthology analysis. *Nucleic Acids Res*, 38(Database issue):D196–203, Jan 2010.

[271] Robert M Waterhouse, Evgeny M Zdobnov, Fredrik Tegenfeldt, Jia Li, and Evgenia V Kriventseva. OrthoDB: the hierarchical catalog of eukaryotic orthologs in 2011. *Nucleic Acids Res*, 39(Database issue):D283–8, Jan 2011.

[272] Ingo Ebersberger, Sascha Strauss, and Arndt von Haeseler. HaMStR: profile hidden markov model based search for orthologs in ESTs. *BMC Evol Biol*, 9:157, 2009.

[273] Ewan Birney, Michele Clamp, and Richard Durbin. GeneWise and Genomewise. *Genome Res*, 14(5):988–95, May 2004.

[274] Sean R Eddy. Accelerated profile HMM searches. *PLoS Comput Biol*, 7(10):e1002195, Oct 2011.

[275] S R Eddy. Profile hidden Markov models. *Bioinformatics*, 14(9):755–63, 1998.

[276] Kazutaka Katoh and Martin C Frith. Adding unaligned sequences into an existing alignment using MAFFT and LAST. *Bioinformatics*, 28(23):3144–6, Dec 2012.

[277] Ben Langmead, Cole Trapnell, Mihai Pop, and Steven L Salzberg. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol*, 10(3):R25, 2009.

[278] Jason E Stajich, David Block, Kris Boulez, Steven E Brenner, Stephen A Chervitz, Chris Dagdigian, Georg Fuellen, James G R Gilbert, Ian Korf, Hilmar Lapp, Heikki Lehväslaiho, Chad Matsalla, Chris J Mungall, Brian I Osborne, Matthew R Pocock, Peter Schattner, Martin Senger, Lincoln D Stein, Elia Stupka, Mark D Wilkinson, and Ewan Birney. The Bioperl toolkit: Perl modules for the life sciences. *Genome Res*, 12(10):1611–8, Oct 2002.

[279] Casey W Dunn, Andreas Hejnol, David Q Matus, Kevin Pang, William E Browne, Stephen A Smith, Elaine Seaver, Greg W Rouse, Matthias Obst, Gregory D Edgecombe, Martin V Sørensen, Steven H D Haddock, Andreas Schmidt-Rhaesa, Akiko Okusu, Reinhardt Møbjerg Kristensen, Ward C Wheeler, Mark Q Martindale, and Gonzalo Giribet. Broad phylogenomic sampling improves resolution of the animal tree of life. *Nature*, 452(7188):745–9, Apr 2008.

[280] Kevin M Kocot, Mathew R Citarella, Leonid L Moroz, and Kenneth M Halanych. PhyloTreePruner: A phylogenetic tree-based approach for selection of orthologous sequences for phylogenomics. *Evol Bioinform Online*, 9:429–35, 2013.

[281] Andreas Zwick, Jerome C Regier, and Derrick J Zwickl. Resolving discrepancy between nucleotides and amino acids in deep-level arthropod phylogenomics: differentiating serine codons in 21-amino-acid models. *PLoS One*, 7(11):e47450, 2012.

[282] Tae-Kun Seo and Hirohisa Kishino. Statistical comparison of nucleotide, amino acid, and codon substitution models for evolutionary analysis of protein-coding sequences. *Syst Biol*, 58(2):199–210, Apr 2009.

[283] Z Yang, R Nielsen, N Goldman, and A M Pedersen. Codon-substitution models for heterogeneous selection pressure at amino acid sites. *Genetics*, 155(1):431–49, May 2000.

[284] AL Bazinet, DS Myers, J Fuetsch, and MP Cummings. Grid Services Base Library: A high-level, procedural application programming interface for writing Globus-based Grid services. *Future Generation Comp Syst*, 23(3):517–522, 2007.

[285] *R: A language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria., 2011.

[286] Leonidas Salichos and Antonis Rokas. Inferring ancient divergences requires genes with strong phylogenetic signals. *Nature*, 497(7449):327–31, May 2013.

[287] Heliconius Genome Consortium. Butterfly genome reveals promiscuous exchange of mimicry adaptations among species. *Nature*, 487(7405):94–8, Jul 2012.

[288] Yinü Li, Guozeng Wang, Jian Tian, Huifen Liu, Huipeng Yang, Yongzhu Yi, Jinhui Wang, Xiaofeng Shi, Feng Jiang, Bin Yao, and Zhifang Zhang. Transcriptome analysis of the silkworm (*Bombyx mori*) by high-throughput RNA sequencing. *PLoS One*, 7(8):e43713, 2012.

[289] Shuai Zhan, Christine Merlin, Jeffrey L Boore, and Steven M Reppert. The monarch butterfly genome yields insights into long-distance migration. *Cell*, 147(5):1171–85, Nov 2011.

[290] International Silkworm Genome Consortium. The genome of a lepidopteran model insect, the silkworm *Bombyx mori*. *Insect Biochem Mol Biol*, 38(12):1036–45, Dec 2008.

[291] Manduca Base. http://agripestbase.org/manduca/, 2014.

[292] autoadapt. https://github.com/optimuscoprime/autoadapt, 2014.

[293] FastQC. http://www.bioinformatics.babraham.ac.uk/projects/fastqc/, 2014.

[294] Marcel Martin. Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet.journal*, 17(1):10–12, May 2011.

[295] Yinlong Xie, Gengxiong Wu, Jingbo Tang, Ruibang Luo, Jordan Patterson, Shanlin Liu, Weihua Huang, Guangzhu He, Shengchang Gu, Shengkang Li, Xin Zhou, Tak-Wah Lam, Yingrui Li, Xun Xu, Gane Ka-Shu Wong, and Jun Wang. SOAPdenovo-Trans: *de novo* transcriptome assembly with short RNA-Seq reads. *Bioinformatics*, 30(12):1660–6, Jun 2014.

[296] Martijn J T N Timmermans, David C Lees, and Thomas J Simonsen. Towards a mitogenomic phylogeny of Lepidoptera. *Mol Phylogenet Evol*, Jun 2014.

[297] Paul Flicek, M Ridwan Amode, Daniel Barrell, Kathryn Beal, Konstantinos Billis, Simon Brent, Denise Carvalho-Silva, Peter Clapham, Guy Coates, Stephen Fitzgerald, Laurent Gil, Carlos García Girón, Leo Gordon, Thibaut Hourlier, Sarah Hunt, Nathan Johnson, Thomas Juettemann, Andreas K Kähäri, Stephen Keenan, Eugene Kulesha, Fergal J Martin, Thomas Maurel, William M McLaren, Daniel N Murphy, Rishi Nag, Bert Overduin, Miguel Pignatelli, Bethan Pritchard, Emily Pritchard, Harpreet S Riat, Magali Ruffier, Daniel Sheppard, Kieron Taylor, Anja Thormann, Stephen J Trevanion, Alessandro Vullo, Steven P Wilder, Mark Wilson, Amonida Zadissa, Bronwen L Aken, Ewan Birney, Fiona Cunningham, Jennifer Harrow, Javier Herrero, Tim J P Hubbard, Rhoda Kinsella, Matthieu Muffato, Anne Parker, Giulietta Spudich, Andy Yates, Daniel R Zerbino, and Stephen M J Searle. Ensembl 2014. *Nucleic Acids Res*, 42(Database issue):D749–55, Jan 2014.

[298] Fiona Cunningham, M Ridwan Amode, Daniel Barrell, Kathryn Beal, Konstantinos Billis, Simon Brent, Denise Carvalho-Silva, Peter Clapham, Guy Coates, Stephen Fitzgerald, Laurent Gil, Carlos García Girón, Leo Gordon, Thibaut Hourlier, Sarah E Hunt, Sophie H Janacek, Nathan Johnson, Thomas Juettemann, Andreas K Kähäri, Stephen Keenan, Fergal J Martin, Thomas Maurel, William McLaren, Daniel N Murphy, Rishi Nag, Bert Overduin, Anne Parker, Mateus Patricio, Emily Perry, Miguel Pignatelli, Harpreet Singh Riat, Daniel Sheppard, Kieron Taylor, Anja Thormann, Alessandro Vullo, Steven P Wilder, Amonida Zadissa, Bronwen L Aken, Ewan Birney, Jennifer Harrow, Rhoda Kinsella, Matthieu Muffato, Magali Ruffier, Stephen M J Searle, Giulietta Spudich, Stephen J Trevanion, Andy Yates, Daniel R Zerbino, and Paul Flicek. Ensembl 2015. *Nucleic Acids Res*, Oct 2014.

[299] Protein trees and orthologies. `http://Feb2014.archive.ensembl.org/info/genome/compara/homology_method.html`, February 2014. [Online; accessed 31-July-2014].

[300] C Notredame, D G Higgins, and J Heringa. T-Coffee: A novel method for fast and accurate multiple sequence alignment. *J Mol Biol*, 302(1):205–17, Sep 2000.

[301] Akiya Jouraku, Kimiko Yamamoto, Seigo Kuwazaki, Masahiro Urio, Yoshitaka Suetsugu, Junko Narukawa, Kazuhisa Miyamoto, Kanako Kurita, Hiroyuki Kanamori, Yuichi Katayose, Takashi Matsumoto, and Hiroaki Noda. KONAGAbase: a genomic and transcriptomic database for the diamondback moth, *Plutella xylostella*. *BMC Genomics*, 14:464, 2013.

[302] Weiqi Tang, Liying Yu, Weiyi He, Guang Yang, Fushi Ke, Simon W Baxter, Shijun You, Carl J Douglas, and Minsheng You. DBM-DB: the diamondback moth genome database. *Database (Oxford)*, 2014:bat087, 2014.

[303] W R Pearson and D J Lipman. Improved tools for biological sequence comparison. *Proc Natl Acad Sci U S A*, 85(8):2444–8, Apr 1988.

[304] Derrick J Zwickl, Joshua C Stein, Rod A Wing, Doreen Ware, and Michael J Sanderson. Disentangling methodological and biological sources of gene tree discordance on oryza (poaceae) chromosome 3. *Syst Biol*, Apr 2014.

[305] Itamar Sela, Haim Ashkenazy, Kazutaka Katoh, and Tal Pupko. GUIDANCE2: accurate detection of unreliable alignment regions accounting for the uncertainty of multiple parameters. *Nucleic Acids Res*, Apr 2015.

[306] L. Dagum and R. Menon. OpenMP: an industry standard API for shared-memory programming. *Computational Science Engineering, IEEE*, 5(1):46–55, Jan 1998.

[307] Garrick Staples. Torque resource manager. In *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, SC '06, New York, NY, USA, 2006. ACM.

[308] Morris A. Jette, Andy B. Yoo, and Mark Grondona. SLURM: Simple Linux Utility for Resource Management. In *In Lecture Notes in Computer Science: Proceedings of Job Scheduling Strategies for Parallel Processing (JSSPP) 2003*, pages 44–60. Springer-Verlag, 2002.

[309] Siavash Mirarab, Md Shamsuzzoha Bayzid, and Tandy Warnow. Evaluating summary methods for multilocus species tree estimation in the presence of incomplete lineage sorting. *Syst Biol*, Aug 2014.

[310] Liang Liu. BEST: Bayesian estimation of species trees under the coalescent model. *Bioinformatics*, 24(21):2542–3, Nov 2008.

[311] Gergely J Szöllősi, Eric Tannier, Vincent Daubin, and Bastien Boussau. The inference of gene trees with species trees. *Syst Biol*, Jul 2014.

[312] Liang Liu, Lili Yu, and Scott V Edwards. A maximum pseudo-likelihood approach for estimating species trees under the coalescent model. *BMC Evol Biol*, 10:302, 2010.

[313] Liang Liu and Lili Yu. Estimating species trees from unrooted gene trees. *Syst Biol*, 60(5):661–7, Oct 2011.

[314] Gregory B Ewing, Ingo Ebersberger, Heiko A Schmidt, and Arndt von Haeseler. Rooted triple consensus and anomalous gene trees. *BMC Evol Biol*, 8:118, 2008.

[315] Liang Liu, Lili Yu, Dennis K Pearl, and Scott V Edwards. Estimating species phylogenies using coalescence times among sequences. *Syst Biol*, 58(5):468–77, Oct 2009.

[316] Liam J. Revell and Scott A. Chamberlain. Rphylip: an R interface for PHYLIP. *Methods in Ecology and Evolution*, 5(9):976–981, 2014.

[317] Julia Chifman and Laura Kubatko. Quartet inference from SNP data under the coalescent model. *Bioinformatics*, 30(23):3317–24, Dec 2014.

[318] Alexandros Stamatakis. RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics*, 30(9):1312–3, May 2014.

[319] Lucy A Weinert, Eli V Araujo-Jnr, Muhammad Z Ahmed, and John J Welch. The incidence of bacterial endosymbionts in terrestrial arthropods. *Proc Biol Sci*, 282(1807), May 2015.

[320] UniProt Consortium. Activities at the Universal Protein Resource (UniProt). *Nucleic Acids Res*, 42(Database issue):D191–8, Jan 2014.

[321] Baris E Suzek, Hongzhan Huang, Peter McGarvey, Raja Mazumder, and Cathy H Wu. UniRef: comprehensive and non-redundant UniProt reference clusters. *Bioinformatics*, 23(10):1282–8, May 2007.

[322] Sayers E. *The E-utilities In-Depth: Parameters, Syntax and More.* Entrez Programming Utilities Help [Internet]. May 2009.

[323] Benjamin Buchfink, Chao Xie, and Daniel H Huson. Fast and sensitive protein alignment using DIAMOND. *Nat Methods*, 12(1):59–60, Jan 2015.

[324] RNAcentral Consortium. RNAcentral: an international database of ncRNA sequences. *Nucleic Acids Res*, 43(Database issue):D123–9, Jan 2015.

[325] Hannes Hauswedell, Jochen Singer, and Knut Reinert. Lambda: the local aligner for massive biological data. *Bioinformatics*, 30(17):i349–55, Sep 2014.

[326] Elmar Pruesse, Jörg Peplies, and Frank Oliver Glöckner. SINA: accurate high-throughput multiple sequence alignment of ribosomal RNA genes. *Bioinformatics*, 28(14):1823–9, Jul 2012.

[327] Christian Quast, Elmar Pruesse, Pelin Yilmaz, Jan Gerken, Timmy Schweer, Pablo Yarza, Jörg Peplies, and Frank Oliver Glöckner. The SILVA ribosomal RNA gene database project: improved data processing and web-based tools. *Nucleic Acids Res*, 41(Database issue):D590–6, Jan 2013.

[328] Claire Lemaitre, Aurélien Barré, Christine Citti, Florence Tardy, François Thiaucourt, Pascal Sirand-Pugnet, and Patricia Thébault. A novel substitution matrix fitted to the compositional bias in Mollicutes improves the prediction of homologous relationships. *BMC Bioinformatics*, 12:457, 2011.

[329] S Henikoff and J G Henikoff. Amino acid substitution matrices from protein blocks. *Proc Natl Acad Sci U S A*, 89(22):10915–9, Nov 1992.

[330] J C Wootton and S Federhen. Analysis of compositionally biased regions in sequence databases. *Methods Enzymol*, 266:554–71, 1996.