Washington University in St. Louis
Washington University Open Scholarship

All Theses and Dissertations (ETDs)

January 2009

Structural RNA Homology Search and Alignment Using Covariance Models

Eric Nawrocki
Washington University in St. Louis

Follow this and additional works at: https://openscholarship.wustl.edu/etd

Recommended Citation

Nawrocki, Eric, "Structural RNA Homology Search and Alignment Using Covariance Models" (2009). *All Theses and Dissertations (ETDs)*. 256.
https://openscholarship.wustl.edu/etd/256

This Dissertation is brought to you for free and open access by Washington University Open Scholarship. It has been accepted for inclusion in All Theses and Dissertations (ETDs) by an authorized administrator of Washington University Open Scholarship. For more information, please contact digital@wumail.wustl.edu.

WASHINGTON UNIVERSITY IN ST. LOUIS

Division of Biology and Biomedical Sciences

(Computational Biology)

Dissertation Examination Committee:
Sean Eddy, Chair
Michael Brent
Jeremy Buhler
Justin Fay
Jeff Gordon
Rob Mitra
Gary Stormo

STRUCTURAL RNA HOMOLOGY SEARCH AND ALIGNMENT USING COVARIANCE MODELS

by

Eric Paul Nawrocki

A dissertation presented to the
Graduate School of Arts and Sciences
of Washington University in
partial fulfillment of the
requirements for the degree
of Doctor of Philosophy

December 2009

Saint Louis, Missouri

copyright by
Eric Paul Nawrocki
2009

ABSTRACT OF THE DISSERTATION

Structural RNA
Homology Search and Alignment
Using Covariance Models

by

Eric Paul Nawrocki

Doctor of Philosophy in Biology and Biomedical Sciences

(Computational Biology)

Washington University in St. Louis, 2009

Sean R. Eddy, Chairman

Functional RNA elements do not encode proteins, but rather function directly as RNAs. Many different types of RNAs play important roles in a wide range of cellular processes, including protein synthesis, gene regulation, protein transport, splicing, and more. Because important sequence and structural features tend to be evolutionarily conserved, one way to learn about functional RNAs is through comparative sequence analysis - by collecting and aligning examples of homologous RNAs and comparing them.

Covariance models (CMs) are powerful computational tools for homology search and alignment that score both the conserved sequence and secondary structure of an RNA family. However, due to the high computational complexity of their search and alignment algorithms, searches against large databases and alignment of large RNAs like small subunit ribosomal RNA (SSU rRNA) are prohibitively slow. Large-scale alignment of SSU rRNA is of particular utility for environmental survey studies of microbial diversity which often use the rRNA as a phylogenetic marker of microorganisms.

In this work, we improve CM methods by making them faster and more sensitive to remote homology. To accelerate searches, we introduce a query-dependent banding (QDB) technique that makes scoring sequences more efficient by restricting the possible lengths of structural elements based on their probability given the model. We combine QDB with a complementary filtering method that quickly prunes away database subsequences deemed unlikely to receive high CM scores based on sequence conservation alone. To increase search sensitivity, we apply two model parameterization strategies from protein homology search tools to CMs. As judged by our benchmark, these combined approaches yield about a 250-fold speedup and significant increase in search sensitivity compared with previous implementations. To accelerate alignment, we apply a method that uses a fast sequence-based alignment of a target sequence to determine constraints for the more expensive CM sequence- and structure-based alignment. This technique reduces the time required to align one SSU rRNA sequence from about 15 minutes to 1 second with a negligible effect on alignment accuracy. Collectively, these improvements make CMs more powerful and practical tools for RNA homology search and alignment.

Acknowledgements

I gratefully acknowledge financial support from the Howard Hughes Medical Institute (HHMI) and a National Institutes of Health National Human Genome Research Initiative (NIH NHGRI) Institutional Training Grant in Genomic Science, T32-HG000045. Sean Eddy's lab in Washington University at St. Louis, where this work took place from 2004 to 2006, was supported by NIH NHGRI grant RO1-HG01363, HHMI, and Alvin Goldfarb. Since then, my research in the Eddy lab at Janelia Farm Research Campus in Ashburn, Virginia has been financially supported by HHMI.

This dissertation is based on and benefits from important research from many other scientists. Zasha Weinberg and Larry Ruzzo's excellent work on RNA homology search, Michael Brown and David Haussler's clever methods for accelerating structural RNA alignment, and the wealth of freely available information at Robin Gutell's Comparative RNA Website (CRW) have been particularly crucial.

In addition, I have been extremely fortunate to work in the lab of Sean Eddy. Sean's tireless work ethic and depth of knowledge in the biological and computational sciences are staggering. The past and present members of his lab bring dedication, skill and enthusiasm to their work that is unmatched. It has been an honor to learn from and work with each of them.

Finally, I want to thank my wife, Michael Hopkins, for her unwavering support and confidence, and for adding so much fun to the last six years of my life - none of this would have been possible without her.

This work is dedicated to my parents, Kathy and Ed Nawrocki.

Thank you for all the love, support, and opportunities that you have provided for me.

Contents

Acknowledgements **iv**

Part 1: RNA homology search **1**

1 Introduction to RNA homology search **2**

1.1 Functional RNA elements 3

1.2 Comparative sequence analysis 9

1.3 Computational methods for sequence homology search 15

1.4 Exploiting conserved structure in RNA homology searches 34

1.5 Stochastic context-free grammars for RNA sequence and structure modeling 41

1.6 Covariance models: profile stochastic context-free grammars 46

1.7 Addressing the limitations of covariance models 57

1.8 Outline of Part 1 of this work 63

2 QDB for faster RNA similarity searches **65**

2.1 Abstract 65

2.2 Introduction 66

2.3 Results 69

2.4 Discussion 91

2.5 Materials and Methods 95

3 Infernal 1.0: inference of RNA alignments **96**

3.1 Abstract 96
3.2 Introduction 96
3.3 Usage 97
3.4 Performance 98

4 A filter pipeline for accelerating covariance model searches 102

4.1 Previous work on accelerating CM searches using filters 103
4.2 Optimized implementations of the CYK and Inside algorithms 104
4.3 Infernal's two-stage filter acceleration pipeline 106
4.4 Evaluation 118
4.5 Implementation 127
4.6 Conclusion and future directions 130

5 Computational identification of functional RNA homologs in metagenomic data 132

5.1 Exploiting conserved structure in RNA similarity searches 134
5.2 Infernal: software for RNA homology search and alignment 138
5.3 Rfam: high-throughput RNA homology search and annotation 141
5.4 Limitations of CMs 145
5.5 Conclusion 148

6 Conclusion to RNA homology search 149

Part 2: RNA alignment 151

7 Introduction to small subunit ribosomal RNA alignment 152

7.1 SSU and the tree of life 153
7.2 Rapid culture-independent SSU sequence determination 154
7.3 Environmental sequencing surveys 155
7.4 Comparative SSU analysis for environmental surveys 161

7.5 SSU alignment methods 169
7.6 Dedicated SSU databases and alignment tools 175
7.7 Developing a new SSU alignment tool 181

8 HMM banding for faster structural RNA alignment 184

8.1 Faster alignment using banded dynamic programming 184
8.2 HMM banded alignment in Infernal 186
8.3 Comparison of HMM and query-dependent banding 194
8.4 Benchmarking 195
8.5 Constructing an HMM that is maximally similar to a CM 201

9 SSU-align: a tool for structural alignment of SSU rRNA sequences 206

9.1 Aligning SSU sequences with SSU-align 207
9.2 Automated probabilistic alignment masking 208
9.3 Implementation 214
9.4 SSU-align's SSU rRNA sequence and structure models 217

Bibliography 236

Vita 265

List of Figures

1.1 RNA secondary structure elements. 8

1.2 Predicted secondary structure of SSU rRNA from *E. coli* (a bacterium) and *M. vannielii* (an archaeon). 10

1.3 Predicted secondary structure of RNase P RNA from *E. coli* (a bacterium) and *M. vannielii* (an archaeon). 11

1.4 Toy example of a multiple sequence alignment. 12

1.5 Dynamic programming matrix filled during a Needleman-Wunsch-Sellers alignment of two sequences. 19

1.6 A banded pairwise alignment dynamic programming matrix. 21

1.7 A toy hidden Markov model. 27

1.8 A simple profile HMM. 29

1.9 Additional information gained from modeling structure 39

1.10 Information in a sequence-only versus a sequence and structure profile. . . . 42

1.11 A parse tree for a dual-stem loop grammar and the corresponding RNA secondary structure 45

1.12 Input alignment, CM guide tree, CM state graph and parse trees. 48

1.13 Examples of CM local begin and end transitions. 53

1.14 Histogram of number of sequences in RFAM 7.0 seed alignments 56

1.15 A pattern defining the E-loop RNA structural motif used by the RNAMOTIF program. 62

2.1 An example RNA family and corresponding CM 71
2.2 Effect of transition priors on band calculation 78
2.3 Effect of varying the β parameter on sensitivity, specificity, and speedup . . 88
2.4 CPU time required by CM searches with and without QDB 89
2.5 ROC curves for the benchmark (QDB chapter) 90

3.1 ROC curves for the benchmark (INFERNAL 1.0 chapter) 99

4.1 Average QDB CYK and HMM Forward scores versus Inside scores for various RNA families. 111
4.2 CM score histograms of random, known, and sampled sequences for three RNA families. 115
4.3 MER versus time for the benchmark. 123
4.4 ROC curves for the benchmark (filter pipeline chapter). 126
4.5 CM Inside scores versus HMM Forward scores during FST calibration. . . . 129

5.1 Secondary structure of three cobalamin riboswitches. 140

6.1 Rfam benchmark ROC curves for INFERNAL v0.55 and v1.01 149

7.1 Flow-chart of the key steps of environmental sequencing surveys. 157
7.2 Number of SSU rRNA sequences deposited in GENBANK per year since 1996. 159
7.3 Schematic of SSU sequence analysis for environmental surveys. 163
7.4 Toy example of phylogenetic inference from an alignment. 165
7.5 David Lane's SSU alignment mask overlaid on the SSU secondary structure of *Escherichia coli*. 167
7.6 Phil Hugenholtz's SSU alignment mask overlaid on the SSU secondary structure of *Escherichia coli*. 168
7.7 Schematic of nearest-neighbor and profile based alignment strategies. 171

8.1 Empirical run times of different alignment algorithms. 199

8.2 CM Plan 9 HMM and HMMER2 Plan 7 HMM architectures. 202

9.1 Schematic of the SSU-ALIGN alignment pipeline 209
9.2 Example of alignment ambiguity in a hairpin loop. 211
9.3 Similarity of an SSU-ALIGN automatically constructed mask and David Lane's SSU alignment mask. 215
9.4 Secondary structure diagram displaying frequency of deletions of each consensus position in a masked bacterial alignment of the SSU seed sequences . 216
9.5 The procedure for converting SSU alignments and individual structures from CRW to seed alignments for SSU-ALIGN. 220
9.6 Example of conflicting base-pairs between two aligned individual SSU structure predictions from CRW. 221
9.7 Secondary structure diagram displaying primary sequence information content per consensus position of the archaeal SSU seed alignment 224
9.8 Secondary structure diagram displaying extra information from conserved structure per consensus position of the archaeal SSU seed alignment 225
9.9 Secondary structure diagram displaying frequency of deletions per consensus position of the archaeal SSU seed alignment 226
9.10 Secondary structure diagram displaying frequency of insertions after each consensus position in the archaeal SSU seed alignment 227
9.11 Secondary structure diagram displaying primary sequence information content per consensus position of the bacterial SSU seed alignment 228
9.12 Secondary structure diagram displaying extra information from conserved structure per consensus position of the bacterial SSU seed alignment 229
9.13 Secondary structure diagram displaying frequency of deletions per consensus position of the bacterial SSU seed alignment 230
9.14 Secondary structure diagram displaying frequency of insertions after each consensus position in the bacterial SSU seed alignment 231

9.15 Secondary structure diagram displaying primary sequence information content per consensus position of the eukaryotic SSU seed alignment 232

9.16 Secondary structure diagram displaying extra information from conserved structure per consensus position of the eukaryotic SSU seed alignment . . . 233

9.17 Secondary structure diagram displaying frequency of deletions per consensus position of the eukaryotic SSU seed alignment 234

9.18 Secondary structure diagram displaying frequency of insertions after each consensus position in the eukaryotic SSU seed alignment 235

List of Tables

1.1 Examples of the power of amino acid versus nucleic acid sequence comparison 36
1.2 Accessions for examples in Table 1.1 37
1.3 Growth of the RFAM database. 54

2.1 Dirichlet priors for transitions (table 1 of 2) 80
2.2 Dirichlet priors for transitions (table 2 of 2) 81
2.3 Summary statistics for the dataset used for emission prior estimation 83
2.4 Parameters of the 9 component Dirichlet mixture emission prior for base pairs 83
2.5 Parameters of the 8 component Dirichlet mixture emission prior for singlets 84
2.6 Rfam benchmark families with timing and MER statistics 86
2.7 Rfam benchmark MER summary statistics 91

3.1 Calibration, search, and alignment running times for seven known structural RNAs of various sizes. 100

4.1 Running time of non-banded CYK and Inside algorithms in INFERNAL 1.0 for four CMs. 103
4.2 Running time of different non-banded CYK and Inside implementations in INFERNAL v0.81 and v1.0 for four CMs. 105
4.3 Benchmark MER and timing statistics for different search strategies. 121
4.4 Comparison of filter sensitivity and benchmark acceleration for queries with different FST predicted filter survival fractions. 124

4.5 Comparison of filter sensitivity and benchmark acceleration for different final algorithm reporting E-value thresholds. 125

5.1 Riboswitch search results. 144

7.1 Sampling of SSU rRNA environmental surveys. 158

7.2 Summary of SSU rRNA sequence databases and alignment strategies. . . . 176

8.1 Transfer of HMM sequence bands to CM i and j bands. 190

8.2 Banded CYK performance on the modified Kolbe09 benchmark. 198

8.3 Average alignment accuracy on the Emit and Random datasets for QDB CYK and HMM banded CYK 200

8.4 Timings for steps of HMM banded alignment on the modified Kolbe09 benchmark. 201

9.1 Effect of different masking strategies on CM alignment accuracy and coverage on the Kolbe09 SSU alignment benchmark 213

9.2 Statistics on the conversion of CRW data to seed alignments for SSU-ALIGN. 222

9.3 Statistics of the three seed alignments used by SSU-align. 223

Part 1:

RNA homology search

Chapter 1

Introduction to RNA homology search

The first complete genome sequence of a cellular organism, the bacterium *Haemophilus influenza*, was reported in 1995 [77]. Since then, roughly one thousand other genomes have been sequenced [159]. While the determination of these sequences has provided scientists with a vast amount of new data, it is only by understanding how cells use the information encoded within their genomes that the benefits of sequencing will ultimately be realized.

A key step towards understanding the information in genomes is identifying and determining the roles of functional sequence elements within them, such as genes and regulatory sequences involved in controlling gene expression. Computer programs that can identify patterns of *similarity* indicative of shared evolutionary ancestry, or *homology*, between members of families of sequence elements have proved useful toward this goal. In my thesis work, I have developed homology search software for a particular kind of sequence element - functional RNA elements. In this introduction, I will first explain why functional RNAs are important, and then why homology search is useful. Finally, I will discuss the development of homology search tools and the advantages and disadvantages of various methods for RNA homology search.

1.1 Functional RNA elements

Unlike messenger RNAs, functional RNA elements are not translated to proteins but rather carry out their biological function directly as RNA. They include RNA genes as well as structural elements within untranslated regions of messenger RNAs. These RNAs play important roles in protein synthesis, gene regulation, protein transport, intron splicing and other fundamental cellular processes. For convenience, I will refer to functional RNAs as simply RNAs throughout this work.

RNAs play many essential roles

The first functional RNAs to be discovered were transfer RNAs (tRNA) and ribosomal RNAs (rRNA). Their roles in protein synthesis were teased apart concurrently with the discovery of messenger RNA in the early 1960s. At that time, based partly on the presence of RNAs in ribosomes (rRNA), the popular "one gene - one ribosome - one protein" hypothesis stated that for each gene there was a unique ribosome responsible for synthesizing that gene's protein product. This hypothesis was disproved as it became clear that the ribosome was a general protein making machine capable of synthesizing any gene's protein product [17, 42]. The specific information of a gene is not in rRNA, but rather is carried from the DNA to the ribosome via a different type of RNA: messenger RNA (mRNA) which is a template of the gene. Yet a third type of RNA, tRNA, serves as an "adaptor" [42], matching the triplet codons in the mRNA to their corresponding amino acids at the active site of the ribosome as the protein is synthesized.

In the late 1960s, biochemical studies began to reveal several types of non-messenger RNAs of sizes markedly different from tRNAs and rRNAs [185], but the function of these non-messenger RNAs was unclear. For about 20 years, knowledge of RNAs was primarily limited to the three types involved in protein synthesis. Since the early 1980s, however, many other types of RNAs have been discovered, leading to a new understanding of the importance of RNAs [41, 55, 240, 242, 248].

RNA and protein complexes

Many RNAs collaborate with proteins in ribonucleoprotein complexes (RNPs) to carry out various ancient and essential functions in the cell. One such RNA was discovered to be a vital component of the inappropriately named signal recognition protein (SRP). SRP is involved in transporting specific proteins to the cellular membrane in all three domains of life. The discovery of an RNA component prompted its renaming to the signal recognition particle [153]. Other examples of RNPs include the eukaryotic major and minor spliceosomes, which include multiple RNAs and are responsible for mRNA processing [28]. RNase P is a universally conserved ribonuclease RNP that cleaves the leader sequence off of precursor tRNA molecules, converting them to active tRNAs [84]. Telomerase is an RNP that uses the telomerase RNA as the template for the addition of specific DNA repeats to the 3' end of eukaryotic chromosomes [14]. Small nucleolar RNAs (snoRNAs) organize with protein components in RNPs that guide chemical modifications necessary for the maturation of rRNAs and other RNA genes [68].

Catalytic RNAs

Not all RNAs require proteins to carry out their biological function. In 1968, the idea of protein-independent RNA led Carl Woese to propose that an RNA world may have predated the current DNA and protein-based world [274]. Central to the RNA world hypothesis (as it later came to be known [92]), was the capacity of RNA to both store information like DNA, and catalyze chemical reactions like proteins. The former role was well established, but the latter was not demonstrated until 1982 when Tom Cech determined that the catalysis of a self-splicing intron in the ciliate *Tetrahymena* was performed solely by RNA [143]. A year later, Sidney Altman revealed the ribonuclease activity of bacterial RNase P as the second known example of RNA-based catalysis [104]. Cech and Altman received the 1989 Nobel prize in chemistry for their work on catalytic RNAs, which are called *ribozymes.* In 2000, analysis of the atomic structure of the large ribosomal subunit from *Haloarcula marismortui* [9] revealed that the ribosome is also a ribozyme with an all-RNA active site [195], lending further support to the RNA world hypothesis.

Regulatory RNAs

RNAs are also extensively involved in the regulation of gene expression, by either modulating mRNA transcription, stability or translation (reviewed in [69, 243]). Many regulatory RNAs are *trans*-acting elements encoded at different genomic loci than their target mRNAs and function through imperfect base-pairing to their targets. Some of these allow the cell to respond appropriately to its environment. For example, the bacteria *Pseudomonas aeruginosa*'s *PrrF1* and *PrrF2* RNAs are expressed when iron levels are low and promote the degradation of transcripts encoding iron-containing enzymes [269].

In bacteria, some *trans*-encoded base-pairing regulatory RNAs carry out their functions within RNPs. A well-characterized example is *Escherichia coli*'s *Hfq* protein which interacts with more than a dozen regulatory RNAs and facilitates binding to their targets that leads to mRNA destabilization, translational repression or activation. [95]. The *Hfq*-binding RNAs regulate their targets by different mechanisms. For example, the *MicF* RNA base-pairs near the ribosome binding site of the *OmpF* mRNA, blocking translation [45]. Alternatively, the *RprA* and *DsrA* RNAs promote translation of the *rpoS* mRNA by preventing the formation of a inhibitory secondary structure that normally occurs in the *rpoS* mRNA [174].

The 21-25 nucleotide (nt) microRNAs (miRNAs) are an example of *trans*-acting base-pairing RNAs in eukaryotes, which mainly act to downregulate gene expression (reviewed in [10, 30]). The primary transcript of a miRNA (pri-miRNA) is transcribed and processed into a short stem-loop structure called a pre-miRNA and finally into a functional miRNA by the RNA-induced silencing complex (RISC). A miRNA is integrated into the RISC complex and controls the expression of target mRNAs by base-pairing. Most known miRNAs, including *lin-4* and *let-7*, the first two miRNAs to be discovered [149], repress translation of their target mRNAs, many of which function in developmental pathways. Alternatively, the interaction of some miRNAs with their targets leads to target degradation. Notably, the first miRNAS were discovered recently, in the early 1990s.

Another class of regulatory RNAs are the 21-25 nt small-interfering RNAs (siRNAs) (reviewed in [179]). siRNAs are usually derived from exogenous RNAs, and are believed to be part of a defense system against foreign RNA. When foreign RNA enters the cell it is randomly cleaved into double stranded fragments by the RNA endonuclease Dicer. These fragments are recognized by the protein complex RISC (RNA-induced-silencing-complex) which separates the two strands and enables base-pairing of one strand to target RNA (other copies of the same original foreign RNA in the cell), which is subsequently cleaved. However, not all siRNAs target exogenous RNAs. Some act to silence the expression of the endogenous RNAs from which they are derived, by either promoting degradation or modifying chromosome structure [67, 160].

A separate class of regulatory RNAs bind directly to proteins instead of base-pairing target mRNAs. For example, 6S RNA in bacteria binds to σ^{70}-RNA polymerase and represses its transcriptional activity during stationary phase when nutrient levels become low [261]. Another example is the mammalian 7SK RNA which prevents transcription by binding to the *HEXIM1/MAQ1* protein and inactivating transcription elongation factor *P-TEFb* [182].

Finally, some mRNAs are regulated by *cis*-encoded RNA structures within their own 5' or 3' untranslated regions (UTRs). For example, the iron response element structure (IRES), a short, roughly 30 nucleotide stem-loop structure occurs in 5' and 3' UTRs of eukaryotic mRNAs encoding genes related to iron metabolism and binds to iron response proteins to regulate their expression [113]. The first of many known IRES was identified in the 5' UTR of the mRNA of *ferritin*, an iron storage protein; when bound it results in translational repression.

Another example of *cis*-regulatory RNA are *riboswitches* - structured RNA elements which bind to small metabolites causing a structural change in the UTR that has a regulatory effect on the expression of the mRNA. Riboswitches often control genes that encode proteins involved in the transport or biosynthesis of the metabolite sensed by the riboswitch (reviewed in [112, 176]). For example, the lysine riboswitch, found upstream of several genes involved in lysine metabolism (such as *lysC* in *E. coli*), binds the amino acid lysine and causes termination of mRNA transcription [223, 246]. Riboswitches are widespread in bacteria, in which more than a dozen separate candidate classes have been identified [112]. One riboswitch, which binds thiamine pyrophospate (TPP), has also been discovered in the 3' UTRs of plant and fungi genes [245]. Riboswitches, like miRNAs, were only recently discovered. The first known riboswitch, the coenzyme-B_{12} cobalamin riboswitch, was described in 2002 [187]. Due to the presence of some riboswitches across wide phylogenetic ranges, notably the TPP and coenzyme-B_{12} cobalamin riboswitches, it has been posited that riboswitches could have ancient origins, and may have provided important gene regulatory mechanisms to organisms of the RNA world [16].

This is an incomplete survey of the types and roles of RNA. There are others that have not been described, such as hammerhead RNAs, piwi-interacting RNAs (piRNA), and more. Given the recent discovery of new types, the list of known RNAs as well as our understanding of their importance is likely to continue to grow.

Conserved RNA structure

Many RNAs conserve a particular three dimensional structure that is energetically favorable, integral to their function, and often conserved across evolutionary timescales. For example, the specific structures of tRNAs and rRNAs provide the precise structural environment necessary for protein synthesis [9]. The particular structure of an unbound riboswitch element in an mRNA is essential for binding the target ligand, which causes a structural change regulating the mRNA's expression [176]. In bacterial RNase P RNA, coaxial stacking of helical regions results in a flat structure that enables binding and cleavage of precursor tRNA substrates [71].

An RNA's structure is determined by intramolecular interactions between different residues in the polynucleotide chain, as well as by intermolecular interactions with other nearby RNAs or proteins. Many of these interactions are hydrogen bonds formed by the base-pairing of two RNA residues. In 1959, Doty and colleagues suggested that about half the residues in RNA molecules form base-pairs [49], an estimate that is roughly accurate for

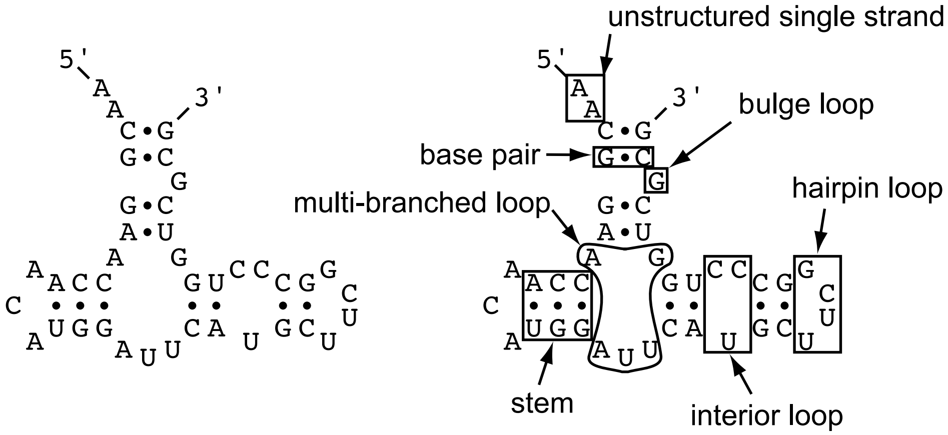


Figure 1.1: RNA secondary structure elements.

most RNAs [178]. The most common and thermodynamically stable pairs are the canonical Watson-Crick `A-U` and `C-G` base-pairs. The next most common are the *wobble* `G-U` pairs, which are typically slightly less stable than Watson-Crick pairs. Base-pairs commonly occur in groups, or *stems*, that form helices because they allow thermodynamically favorable *stacking* of the pi (π) bonds of the bases' aromatic rings. The set of base-pairs in an RNA defines its secondary structure. The secondary structure of a toy example RNA is shown in Figure 1.1.

The level of structural conservation varies between different RNAs as well as between different structural elements of an individual RNA family. In general, the levels of conservation of an RNA's structural elements correlate with their functional importance. The RNAs involved in protein synthesis, the tRNAs and rRNAs, conserve nearly their entire structure very strongly due to the precise structural requirements of a functioning ribosome. Figure 1.2 shows the global similarity between the predicted SSU rRNA secondary structure for the archaeon *Methanococcus vannielii* and the bacterium *Escherichia coli*. In constrast, RNase P RNA conserves structure in a more local fashion, as shown by comparing the predicted secondary structures from the same two organisms (Figure 1.3). In RNase P, only the structural core of the molecule that is necessary and responsible for catalytic activity is highly conserved [34]. Eukaryotic telomerase RNA also conserves only some of its structural features. The telomerase RNAs of ciliates, yeast, and vertebrates are about 200, 400, and

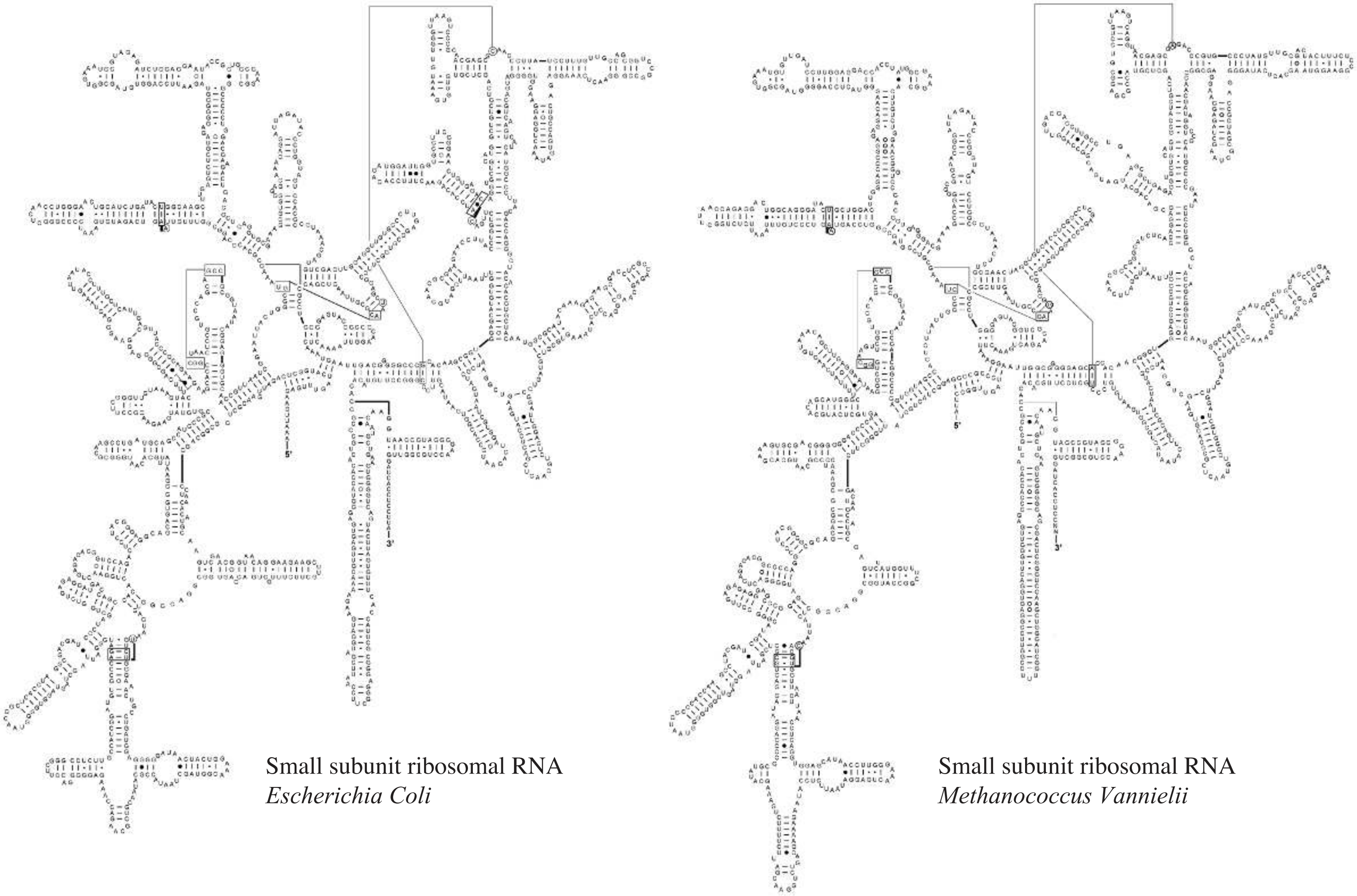
1200 nt respectively. All three are highly structured, with three or more stems each, but only the two stems that interact with the telomerase protein TERT are clearly homologous [33]. Finally, some RNAs, such as the regulatory siRNAs and miRNAs, are unstructured. These molecules function as single-stranded RNAs by base-pairing to their targets without the need for structural elements. These RNAs can be viewed as utilizing RNA's information storing capacity more so than its ability to adopt complex structures for binding to proteins or catalysis.

Biochemists have used x-ray crystallography to determine the atomic structure of several RNAs, including ribosomal RNAs [9, 286]. Nuclear magnetic resonance (NMR) has been used to solve the structure of short RNA motifs, but the technique currently cannot be used for large RNAs (the limit is about 100 nucleotides) [114, 255]. Both of these approaches are expensive and time-consuming. An attractive alternative method for inferring RNA structure is based solely on sequence analysis, by comparing examples of evolutionarily related RNAs from different organisms. This technique is described in more detail below.

1.2 Comparative sequence analysis

A powerful method for learning about any functional sequence element, including RNAs, is *comparative sequence analysis* - by collecting examples of the element and comparing them. The cornerstone of comparative sequence analysis is the fact that the genomes of all modern organisms are evolutionarily related, having ultimately descended from the last common ancestor of all life on the planet. As a result, many genes within organisms' genomes can usefully be organized into gene *families*, composed of evolutionarily related genes, or homologs.

Families of homologous genes present in modern genomes can be related by a phylogenetic tree rooted at the oldest ancestral sequence of the family with nodes in the tree representing gene duplication and speciation events. Along each branch the sequences have evolved independently, accumulating sequence mutations, but, importantly, the function of the molecule has acted as an evolutionary constraint. Sequences with mutations that

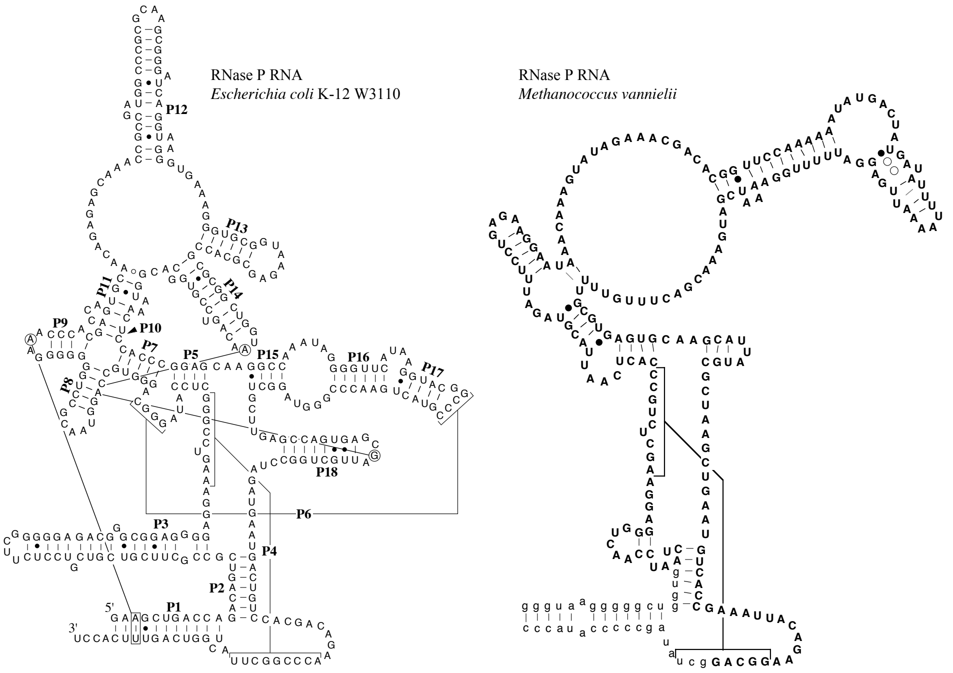


Small subunit ribosomal RNA
Escherichia Coli

Small subunit ribosomal RNA
Methanococcus Vannielii

source: Comparative RNA Website, Cannone et al. 2002 BMC Bioinformatics 2:3

Figure 1.2: Predicted secondary structure of SSU rRNA from *E. coli* (a bacterium, left) and *M. vannielii* (an archaeon, right). Diagrams are from of the COMPARATIVE RNA WEBSITE [32].



source: The Ribonuclease P database. Brown, 1999 NAR 27:3414

Figure 1.3: Predicted secondary structure of RNase P RNA from *E. coli* (a bacterium, left) and *M. vannielii* (an archaeon,right). Diagrams are from Jim Brown's RIBONUCLEASE P DATABASE [22].

```
human AAGACUUCGGAUCUGGC-GACAC-CC
  rat AAG-CUUCGCAUACGGC-GCCAC-CG
  orc AGGUCUUCGCA-CGGGCAGCCACUUC
      * * ***** *   *** * ***
```

Figure 1.4: **Toy example of a multiple sequence alignment.** The "*"s in the final line indicates columns with identical residues in all three aligned sequences.

negatively affect function are less likely to survive to the next generation, and the larger the functional deficit, the less likely survival is. Thus, conservation levels of particular residues or other features in homologous sequences are often related to their functional importance. An example of a functional insight based on recognition of sequence conservation is helix 18 of SSU rRNA that occurs between sequence positions 500 and 545 of the commonly used reference *E.coli* SSU rRNA (GENBANK accession J01695) [280]. The highly conserved length and sequence of this helix compelled biologists to test its functional significance via mutagenesis studies, eventually suggesting its crucial role in the binding of tRNA to the ribosomal *A* site during translation [212, 277].

A necessary first step towards identifying conservation is alignment of homologous sequences. The goal of alignment is to juxtapose sequences so that homologous residues (residues that descended from a common ancestral residue) in each sequence occur in the same column of the alignment. Gaps are introduced to represent insertions or deletions (collectively referred to as *indels*) of residues in some of the sequences. Figure 1.4 shows a toy alignment of three RNA sequences.

One common multiple alignment technique is *progressive alignment*. Progressive alignment begins by aligning the most similar sequences first, in a pairwise fashion, and then combining the alignments together until a single alignment of all sequences is determined [73, 249]. Alternatively, optimal multiple sequence alignment considers the alignment of all the sequences simultaneously and returns the one with the maximum score [106]. The necessary algorithms for computing optimal multiple alignments, however, are too computationally expensive to be practical for most problems [53].

A multiple alignment of homologous sequences is a useful starting point for many applications of comparative sequence analysis, including estimation of the evolutionary tree that explains the descent of those sequences since their common ancestor [72]. Phylogenetic trees inferred from SSU rRNA alignments are commonly used by biologists to identify which organisms live in various environments [200]. Over the past 20 years, SSU environmental surveys have greatly expanded the recognized biodiversity on the planet [216]. In fact, the discovery that archaea are a distinct, third domain of life resulted from a phylogenetic inference based on SSU rRNA sequences [276]. (SSU alignment and phylogenetic inference is discussed in more detail in Part 2 of this work.)

Inferring RNA structure

An alignment also serves as a starting point for comparative analysis to infer the conserved structure of an RNA family. Structural inference exploits the fact that residues that form base-pairs in RNA structure tend to covary (change in concert) throughout evolution. For example, if the `C` in a `C:G` base-pair mutates to a `U`, the `G` will commonly eventually mutate to an `A`, thus maintaining a Watson-Crick base-pair at these positions. These compensatory changes create striking patterns in multiple sequence alignments that are sometimes even recognizable by eye. There are other sets of base-pairs, besides Watson-Cricks, that adopt similar three dimensional structures, and mutations between members of these sets can be observed as well. These less common base-pairs are often more dependent on other local structural features of the RNA [150].

By grouping together consistent base-pairs suggested by these covariation patterns, conserved secondary structures can be predicted. The secondary structure of many RNA families have been correctly inferred in this manner beginning with tRNA from just four sequences [115]. Others include 5S rRNA [82], group I introns [181], RNase P RNA [125], and even the two largest structural RNAs, SSU rRNA [278] and LSU rRNA [196]. Structure prediction using comparative analysis has proven to be very reliable. For example, 97% of the base-pairs in the predicted SSU and LSU rRNA structures were verified when the crystal structures of those RNAs were determined [107].

Importantly, because sequence differences are necessary for observing covariation, this structural inference approach is dependent on the sequence diversity in the alignment. The more diverse the sequences, while still adopting the same structure, the more obvious the conserved structure becomes. However, the method also relies on a correct alignment with only homologous residues aligned with each other, and alignment accuracy typically drops with sequence similarity. Ideally, because each informs the other, the alignment and structure would be inferred simultaneously. The development of techniques that can do this accurately and efficiently is an important and active area of research [51, 117, 177, 230].

Much like conserved sequence elements, conserved structural elements suggest functional importance and can inform experimentation. For example, when the subsequences of many known examples of hammerhead self-cleaving RNA were predicted to form a common structure, it was hypothesized that the common structure was responsible for catalysis. Deletion analysis of the sequences revealed that the catalytic structure could indeed be reduced to this common structural element [79, 80].

Finally, structural models inferred from comparative analysis can also help interpret experimental results. For example, in an experiment designed to determine the active site of RNase P, tRNA was cross-linked to three separate RNase P RNAs from three different bacteria. Each experiment created cross-links in several nucleotides of the RNase P, but only a subset of them were common between all three species' RNase Ps. The authors hypothesized this subset of nucleotides was involved in the active site, which was later confirmed [29].

Homology-based annotation of functional sequence elements

Comparative analysis can also be used as the basis for detecting homologous sequences in genomes or databases. Genome sequencing projects often include annotation pipelines that use comparative analysis via *homology search* programs to identify genes belonging to characterized families. Knowledge of the genes in an organism's genome is informative about the types of metabolism and other cellular processes that take place in the organism.

Homology search programs compare known examples of a family, or *queries*, to *target* sequences. In general, targets with high levels of similarity to queries are likely homologous and can be classified as family members themselves. Because the power of comparative analysis increases as the diversity of the sequences being compared increases, finding new homologs is generally useful for other comparative analysis applications. For example, the new homologs may contain compensatory mutations relative to known examples that suggest base-pairs in the conserved structure of the family.

Because the homology search task itself is aided when new homologs are found, these programs are often used iteratively to progressively find more distant homologs in multiple rounds of searches. Any new sequences found in each round of an iterative search offer new knowledge of the family which can be exploited in the following round. Iterations can profitably continue until no new sequences are found.

Homology search is the hub of comparative sequence analysis applications, as well as an application itself. This central importance has made it a popular research topic since the dawn of sequencing over 40 years ago. In the next section, I will discuss some of the fundamental strategies and important innovations in homology search methods.

1.3 Computational methods for sequence homology search

Homology search methods compute a score based on a comparison of a query to target sequences in a database. The utility of a method lies in its ability to assign better scores to homologous targets than non-homologous ones. There are two general classes of methods: those that use a single sequence as a query (pairwise methods), and those in which the query is based on multiple sequences. Each of these is discussed in detail below. Because the majority of the first biological sequences to be determined were protein sequences, early homology search methods concentrated on proteins instead of DNA or RNA. However, virtually all of these methods are generally applicable to any alphabet, so they are useful for DNA and RNA as well.

Some general terms regarding homology search performance that will be used in this section are worth introducing here. Roughly speaking, a method's *sensitivity* is a measure of how many real homologs are assigned high scores, and its *specificity* is a measure of how many non-homologs are assigned low scores. To enable a more precise definition, consider a database in which all the homologs for a given query are known and called *trues* and all other sequences are *falses*. After searching the database with a query, all target sequences that score above a reporting score threshold are called *hits* and are considered positives. Any true hit is a true positive (TP) and any false hit is a false positive (FP). Any true which is not a hit is a false negative (FN) and any false which is not a hit is a true negative (TN). A method's sensitivity measures how many trues are hits and is often defined as the ratio of true positives to trues ($\frac{TP}{TP+FN}$). A method's specificity measures how many hits are trues, and is often defined as ratio of true negatives to falses ($\frac{TN}{TN+FP}$).

Pairwise sequence alignment based methods

A common way to compare two sequences is to align them. As discussed above in the context of comparative sequence analysis, an alignment of two sequences is a mapping of the homologous residues of one sequence to another, with the introduction of gaps in either sequence as necessary. An example three sequence alignment is shown in Figure 1.4.

The first sequence alignments were performed by hand. In 1961, Kendrew and Watson compared whale myoglobin to human hemoglobin by manually aligning the amino acid residues based on their expert knowledge of the structure and function of globins [262]. Manual alignment is subjective and time-consuming and as the number of available sequences and potential alignments between them increased, an objective, automated method was desired [48].

In 1970, Needleman and Wunsch described an algorithm for computing the optimal scoring alignment of two sequences A and B [193]. A very slightly modified version of their algorithm, that was developed by Sellers in 1974 [232], is well known in computational biology as the Needleman-Wunsch algorithm. Here, I will refer to it as Needleman-Wunsch-

Sellers (NWS). The NWS algorithm proceeds by constructing a matrix S of size $|A| \times |B|$ for the sequences of length $|A|$ and $|B|$, and filling in each of the cells with a score. Cell $S_{i,j}$ includes the alignment score for subsequences $A_1..A_i$ aligned to $B_1..B_j$. The score is defined as the maximum of three alternatives of scores of previously calculated adjacent cells plus either a gap penalty or a score for aligning residues A_i and B_j. More formally $S_{i,j}$ is computed as:

$$\text{if } i = 0 \text{ or } j = 0 \quad S_{i,j} = (i + j) * g$$
$$\text{else}$$

$$S_{i,j} = max \begin{cases} S_{i-1,j-1} + M_{A_i,B_j} & \text{(rule 1)} \\ S_{i,j-1} + g & \text{(rule 2)} \\ S_{i-1,j} + g & \text{(rule 3)} \end{cases}$$

M is a substitution matrix which defines the score for aligning any two residues to each other and g is the gap penalty incurred for introducing a gap in either sequence. The simplest scoring metric that Needleman and Wunsch suggested was assigning a 1 for a match ($M_{a,b} = 1$ if $a = b$), a 0 for a mismatch ($M_{a,b} = 0$ if $a \neq b$) between amino acid residues, and a score of -1 for a gap in either sequence ($g = -1$). The complexity of this algorithm in both time and memory is $O(|A||B|) \sim O(N^2)$.

NWS is a *dynamic programming* (DP) algorithm that is guaranteed to find the optimal (maximum) scoring alignment of the two sequences given the scoring system. By following the simple recursion above, the score in any cell $S_{i,j}$ is the optimal score for the subsequences $A_1..A_i$ to $B_1..B_j$. Thus, after filling in the full matrix, $S_{|A|,|B|}$ will contain the score of the optimal alignment of the full sequences. Retrieval of the alignment requires a *traceback* through the matrix based on which of the three rules for calculating $S_{i,j}$ were used in each cell. At cell i, j: if rule 1 was used then residue i is aligned to j; if rule 2 was used then residue j in sequence 2 is aligned to a gap in sequence 1; if rule 3 was used then residue i in sequence 1 is aligned to a gap in sequence 2. An example of aligning two sequences with

NWS is shown in Figure 1.5.

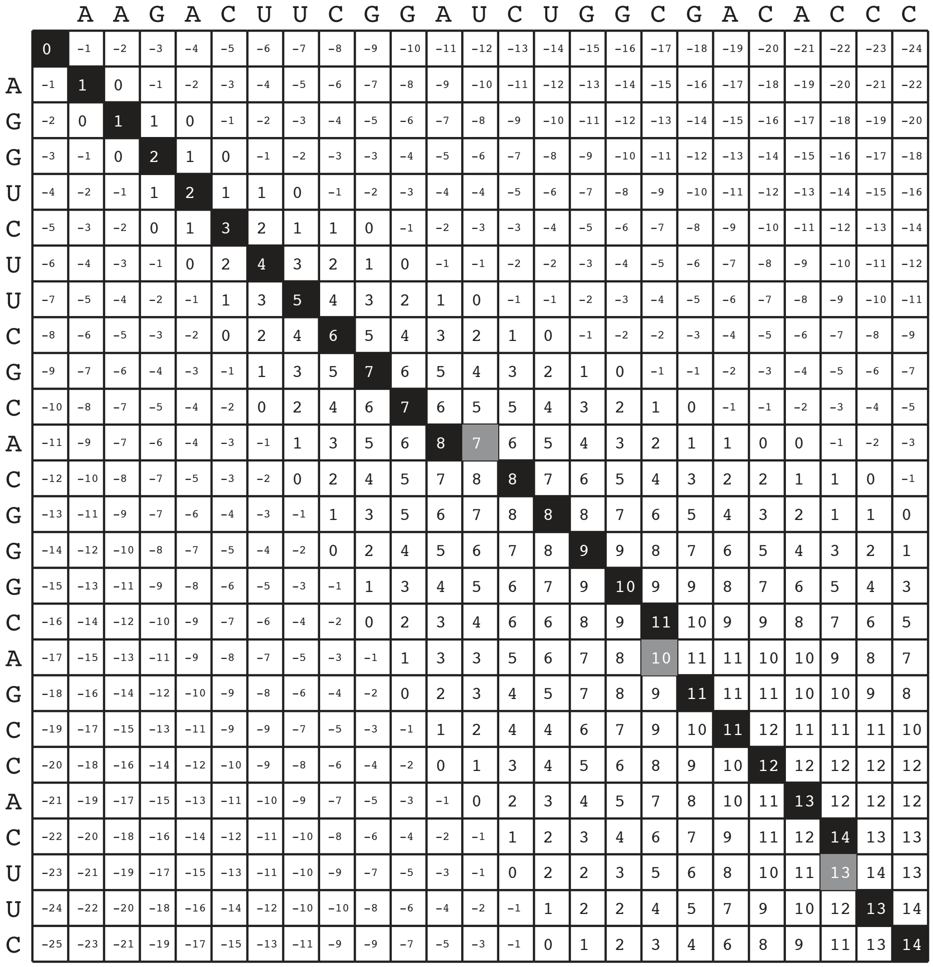
Note that the score for either aligning two residues in a column or matching a residue in one sequence to a gap in the other does not depend on the score in any other cell or any other residues. NWS assumes that each column is *independent*, which is mathematically convenient and makes the calculation of the alignment relatively efficient. This assumption can be relaxed to allow dependencies between columns like those introduced by base-pairs in conserved RNA structure at a cost to computational efficiency as discussed later in this chapter.

Affine gap penalties

As described here, NWS uses a *linear* gap penalty. The score for a gap of size l is simply $\omega(l) = lg$. A more biologically realistic scheme reflecting the empirical observation that gaps tend to be greater than length 1 is the *affine* gap scheme, which defines a separate cost for opening (c) and extending a gap (d). An affine gap score is calculated as $\omega(l) = c + (l-1)d$. An $O(N^2)$ version of NWS with linear gap penalties was introduced by Gotoh [94].

The Smith-Waterman algorithm performs local alignment

NWS is called a *global* alignment algorithm because it finds the highest scoring alignment of the full sequences it is aligning. Sometimes a *local* alignment between subsequences is more biologically relevant. For example, many protein sequences have domains that are critical to function and consequently evolutionarily conserved while surrounding sequence is less important and less conserved. A local alignment is also more meaningful for two homologous RNA sequences embedded within much longer chromosomes. The Smith-Waterman (SW) algorithm is very similar to NWS but returns the highest scoring alignment of any two subsequences within two sequences [236]. Notably, only a subtle modification to NWS's recursion is necessary to create the SW algorithm: allowing a new alignment with an initial score of 0 to begin at any cell in the scoring matrix (i.e. adding a fourth possible rule for determining $S_{i,j}$ as setting it equal to 0). After filling in the matrix, the optimal local



		A	A	G	A	C	U	U	C	G	G	A	U	C	U	G	G	C	G	A	C	A	C	C	C
	0	−1	−2	−3	−4	−5	−6	−7	−8	−9	−10	−11	−12	−13	−14	−15	−16	−17	−18	−19	−20	−21	−22	−23	−24
A	−1	1	0	−1	−2	−3	−4	−5	−6	−7	−8	−9	−10	−11	−12	−13	−14	−15	−16	−17	−18	−19	−20	−21	−22
G	−2	0	1	1	0	−1	−2	−3	−4	−5	−6	−7	−8	−9	−10	−11	−12	−13	−14	−15	−16	−17	−18	−19	−20
G	−3	−1	0	2	1	0	−1	−2	−3	−3	−4	−5	−6	−7	−8	−9	−10	−11	−12	−13	−14	−15	−16	−17	−18
U	−4	−2	−1	1	2	1	1	0	−1	−2	−3	−4	−4	−5	−6	−7	−8	−9	−10	−11	−12	−13	−14	−15	−16
C	−5	−3	−2	0	1	3	2	1	1	0	−1	−2	−3	−3	−4	−5	−6	−7	−8	−9	−10	−11	−12	−13	−14
U	−6	−4	−3	−1	0	2	4	3	2	1	0	−1	−1	−2	−2	−3	−4	−5	−6	−7	−8	−9	−10	−11	−12
U	−7	−5	−4	−2	−1	1	3	5	4	3	2	1	0	−1	−1	−2	−3	−4	−5	−6	−7	−8	−9	−10	−11
C	−8	−6	−5	−3	−2	0	2	4	6	5	4	3	2	1	0	−1	−2	−2	−3	−4	−5	−6	−7	−8	−9
G	−9	−7	−6	−4	−3	−1	1	3	5	7	6	5	4	3	2	1	0	−1	−1	−2	−3	−4	−5	−6	−7
C	−10	−8	−7	−5	−4	−2	0	2	4	6	7	6	5	5	4	3	2	1	0	−1	−1	−2	−3	−4	−5
A	−11	−9	−7	−6	−4	−3	−1	1	3	5	6	8	7	6	5	4	3	2	1	1	0	0	−1	−2	−3
C	−12	−10	−8	−7	−5	−3	−2	0	2	4	5	7	8	8	7	6	5	4	3	2	2	1	1	0	−1
G	−13	−11	−9	−7	−6	−4	−3	−1	1	3	5	6	7	8	8	8	7	6	5	4	3	2	1	1	0
G	−14	−12	−10	−8	−7	−5	−4	−2	0	2	4	5	6	7	8	9	9	8	7	6	5	4	3	2	1
G	−15	−13	−11	−9	−8	−6	−5	−3	−1	1	3	4	5	6	7	9	10	9	9	8	7	6	5	4	3
C	−16	−14	−12	−10	−9	−7	−6	−4	−2	0	2	3	4	6	6	8	9	11	10	9	9	8	7	6	5
A	−17	−15	−13	−11	−9	−8	−7	−5	−3	−1	1	3	3	5	6	7	8	10	11	11	10	10	9	8	7
G	−18	−16	−14	−12	−10	−9	−8	−6	−4	−2	0	2	3	4	5	7	8	9	11	11	11	10	10	9	8
C	−19	−17	−15	−13	−11	−9	−9	−7	−5	−3	−1	1	2	4	4	6	7	9	10	11	12	11	11	11	10
C	−20	−18	−16	−14	−12	−10	−9	−8	−6	−4	−2	0	1	3	4	5	6	8	9	10	12	12	12	12	12
A	−21	−19	−17	−15	−13	−11	−10	−9	−7	−5	−3	−1	0	2	3	4	5	7	8	10	11	13	12	12	12
C	−22	−20	−18	−16	−14	−12	−11	−10	−8	−6	−4	−2	−1	1	2	3	4	6	7	9	11	12	14	13	13
U	−23	−21	−19	−17	−15	−13	−11	−10	−9	−7	−5	−3	−1	0	2	2	3	5	6	8	10	11	13	14	13
U	−24	−22	−20	−18	−16	−14	−12	−10	−10	−8	−6	−4	−2	−1	1	2	2	4	5	7	9	10	12	13	14
C	−25	−23	−21	−19	−17	−15	−13	−11	−9	−9	−7	−5	−3	−1	0	1	2	3	4	6	8	9	11	13	14

```
final alignment:
AAGACUUCGGAUCUGGC-GACAC-CC
AGGUCUUCGCA-CGGGCAGCCACUUC
* * ***** * * *** * ***  *
```

Figure 1.5: **Dynamic programming matrix filled during a Needleman-Wunsch-Sellers alignment of two sequences.** Black and gray cells are part of an optimal alignment path. Black cells were reached using rule 1 of the algorithm and gray cells were reached using rules 2 or 3 of the algorithm and correspond to a gap in one of the aligned sequences (see text for rules). The black upper left cell is an exception, it was not reached using rule 1 but instead as part of the initialization condition. Negative scores are in a smaller font. In the final alignment, `*`'s indicate identities. The final alignment score of 14 derives from 17 identities, each contributing +1 to the alignment score, and 3 gaps, each contributing −1 to the score. See text for details on the algorithm.

alignment score is the maximum score in the matrix, and the alignment is retrieved by starting at that maximum scoring cell and tracing back as in NWS, but stopping when the $S_{i,j} = 0$ rule was used, which indicates the starting cell of the alignment.

Banded dynamic programming

As databases grew, the speed of the homology search algorithms became important. The current NCBI non-redundant protein database contains about 8 million protein sequences totaling about 3 billion residues. To search a query sequence of length 300 against the full database would require on the order of 10^{12} matrix calculations. Running on a computer that can perform roughly 100 million matrix cells per second (which is reasonable for a desktop computer at the time of writing) would require 10^4 seconds, or about three hours. This means to search all of the proteins in a prokaryotic genome (roughly 4000) against the full database would take about a year and half. Faster, heuristic alignment algorithms were developed that performed the same core DP recursion as NWS and SW, but only for a limited set of cells in the matrix - those that were predicted, using a very fast method, to be involved in the highest scoring alignment. The first such widely used programs were the FASTA package [208] and the BLAST package [2]. (Different implementations of BLAST exist for DNA and RNA (BLASTN) and for proteins (BLASTP). They both use the same basic strategy discussed here, and I will use BLAST to refer to both.)

FASTA and BLAST are fast because they exploit the fact that high scoring sequence alignments usually include at least one identical or nearly identical stretch of residues between the two sequences. Both programs first find these short high-scoring matches, then combine consistent matches together into an initial alignment and use *banded* DP to fill in any regions not involved in the exact matches. [1] Banded DP performs the SW DP recursion only for a limited number of cells within bands of the DP matrix. In this case the band involves only the regions of the matrix surrounding the high scoring matches. The effect of using bands in a DP matrix is demonstrated in Figure 1.6.

---

[1] The original BLAST [2] does not include gaps and has no need for banding, but gapped BLAST [3] does use banding.

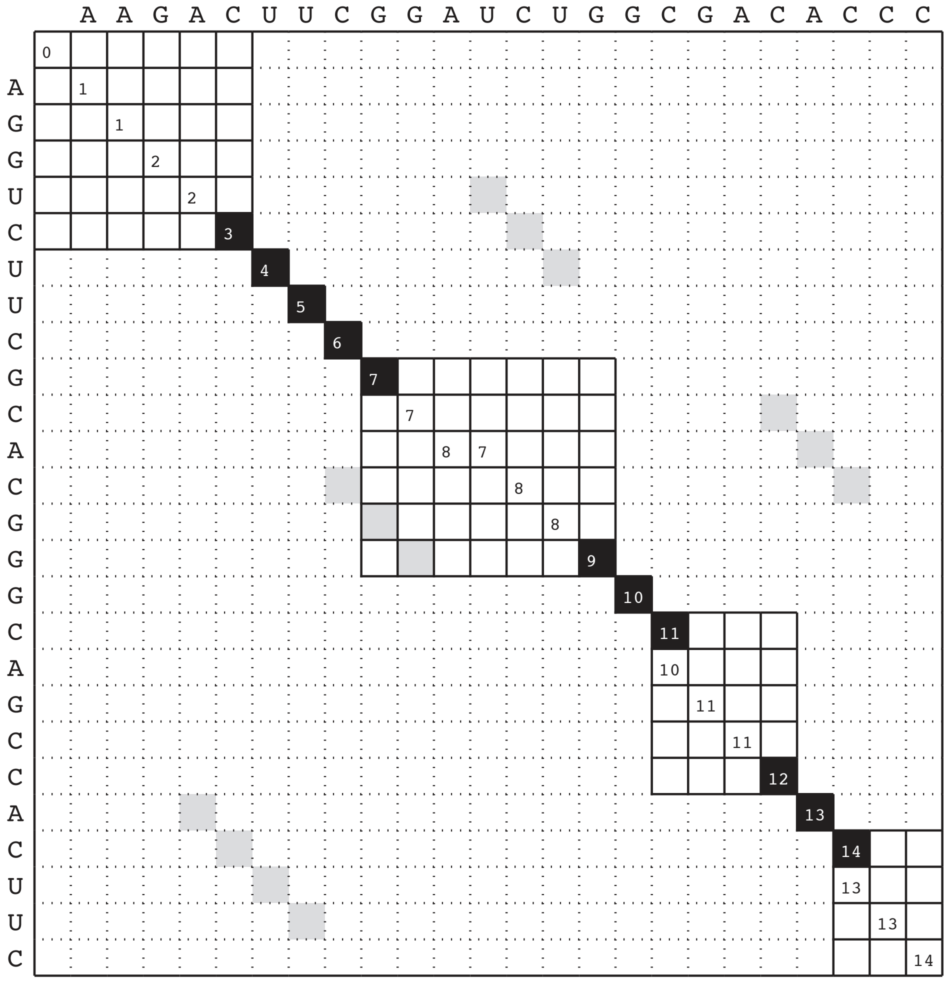

A A G A C U U C G G A U C U G G C G A C A C C C
A G G U C U U C G C A C G G G C A G C C A C U U C
0
1
1
2
2
3
4
5
6
7
7
8
7
8
8
9
10
11
10
11
11
12
13
14
13
13
14

Figure 1.6: **A banded pairwise alignment dynamic programming matrix.** The bands shown here were derived from a simple BLAST-like banding procedure. First, all stretches of at least three consecutive identities were found (black and gray cells). Then the largest consistent set of these stretches that could exist in a single path was defined (black cells). The Needleman-Wunsch-Sellers (NWS) algorithm was executed using banded DP - only alignments that include all of the black cells were considered. Cells within the bands are outlined with solid lines. Those outside the bands are outlined with dotted lines. NWS scores are shown in the cells in the optimal alignment from Figure 1.5. Note that the optimal alignment is within the bands.

An important difference between the two methods is that while FASTA requires identical matches in the first step, BLAST finds any ungapped match that scores above a pre-specified score threshold, which may include some substitutions. FASTA and BLAST are roughly 20 and 100 times faster, respectively, on typical protein queries than full SW DP implementations. Searches that took a few hours could now be performed in a few minutes.

FASTA, BLAST, and all banded DP algorithms are heuristics that sacrifice the *guarantee* that the optimal alignment will be found for increased speed. If the optimal alignment is missed, then the performance of the program may be negatively affected. This becomes more likely as the evolutionary distance, and number of substitutions and indels, between sequences increases. Some alignment benchmarks designed to test for remote homology detection reflect this and have shown that the heuristic BLAST and FASTA programs are less sensitive than full SW implementations [18].

Substitution matrices introduce probabilistic scoring

An important feature of the NWS and SW algorithms, as well as any accelerated heuristic versions of them, is that they can be used with any scoring system, i.e. any way of defining a score for substitutions and gaps. The initial scoring schemes presented with the NWS algorithm were very simple and included identical scores for many different substitutions [193]. A large amount of subsequent work has focused on improving these scoring systems so they produce more biologically meaningful alignments.

In reality, different amino acid substitutions occur with very different frequencies, and an alignment scoring system should reflect this. This is largely because some amino acids are more biochemically similar than others, so a mutation to a similar amino acid will have a smaller impact on the structure and function of a protein than a mutation to a drastically different one. Another factor is that, due to the triplet nature of the genetic code, different numbers of DNA mutations are required to mutate one amino acid to another.

To address these issues, Margaret Dayhoff and colleagues introduced the first *mutational matrix* (what is referred to today as a *substitution matrix*) with scores derived from empirical substitutions observed in real protein sequences in 1978 [43]. Dayhoff estimated the first matrices by counting substitutions in trusted pairwise alignments of very similar sequences in the *Atlas of Protein Sequence and Structure* database, which at the time contained about 1000 sequences. The observed changes were normalized to determine the frequency of each possible substitution when the expected total number of substitutions was 1%. This 1% matrix (called PAM1 for percent accepted mutation at 1%) was exponentiated to create new matrices for longer evolutionary timescales based on the idea that repeated mutations would follow the same patterns captured in PAM1, and multiple mutations can occur at the same site. Not surprisingly, PAM matrices proved superior to the simple scoring systems mentioned above and became widely used used in the community. The examples in the original FASTA [208] and BLAST [2] papers used PAM matrices.

By 1992 sequence databases had grown considerably and the problem of substitution matrix estimation was revisited by Steve and Jorja Henikoff. They derived new matrices from the BLOCKS database of ungapped alignments, which differed from the ATLAS database that Dayhoff used in several important ways: it was larger, contained multiple sequence alignments with more divergent sequences, and only contained sequences from conserved protein cores instead of full length sequences [110, 111]. The Henikoffs' matrices are called the BLOSUM matrices, and are used by current versions of BLAST and FASTA.

The BLOSUM matrices have a probabilistic interpretation [1, 54]. The scores are logarithms of a ratio of probabilities, or *log-odds* scores:

$$s(a, b) = log(\frac{p_{ab}}{q_a q_b}) \tag{1.1}$$

The value p_{ab} is the probability of observed residue a aligned to residue b in alignments of homologous sequences (derived from observations in BLOCKS), and q_a and q_b are the background probabilities of observing residue a and b, calculated as $q_a = \sum_{b'} p_{ab'}$ (i.e. the frequency of each residue in the database).

A log-odds score is the most efficient possible test for distinguishing between two alternative hypotheses [126, 194]. In this case, the two hypotheses being tested are that the residues are homologous or not. The assumption, mentioned above, that alignment columns are independent means that the summed score for all columns of an ungapped alignment is the log-odds score for the entire aligned sequences being homologous or not.

Estimating statistical significance of alignment scores

A log-odds score is not the only useful statistic for homology searches. For database hits reported from a given search, the *E-value* of a hit with score x is the expected number of hits with a score $>= x$ that would be found if the same search were repeated with the same query on a random database devoid of real homologs. E-values can be empirically estimated via simulation: by generating a random database of sequences and searching it. The logic here is that any hit from a real search with a higher score than any from the random database search is unlikely to have arisen by chance, and so is likely to indicate homology. However, simulations like these are expensive and an analytical technique for determining statistical significance without simulation was desired.

In 1990, such a technique was described by Sam Karlin and Stephen Altschul, who realized that ungapped BLAST scores of random sequences followed a Gumbel (extreme value type 1) distribution [129]. A key result was that the expected number of random matches between a query sequence of length M and a target of length N with score exceeding x is:

$$E = KMNe^{\lambda x}$$

Karlin and Altschul determined an analytical solution and could derive the relevant parameters (K and λ) for a given ungapped BLAST scoring system (i.e. BLOSUM matrix), but that did not apply when gaps were allowed, as in current versions of BLAST. In this case, to obtain the appropriate K and λ parameters for a given scoring system it is necessary to empirically fit a Gumbel to the distribution of high scores obtained by searching a large number of random sequences. For BLAST, this is not a serious problem because most searches typically employ one of a small number of scoring systems (combination of substitution matrix and gap penalties). Thus, the appropriate K and λ parameters can be precalculated once for each scoring system and used for all subsequent searches using that system. The ability of BLAST to calculate accurate E-values, along with its speed, are two of the major reasons it is so widely used.

Profiles: more powerful searches using queries based on multiple sequences

The pairwise scoring systems discussed thus far are all *position-independent*, i.e. substitutions and gaps receive the same score regardless of where they occur in the protein or nucleic acid. These methods implicitly assume that changes are equally likely at all positions. It is difficult to override this assumption given only one sequence, but when multiple homologs of a sequence family are known it becomes clear that some positions, such as those involved in the active site of a protein, are very highly conserved and rarely tolerate mutations or indels, while others, such as those in less important linker regions of proteins, are more tolerant to mutations and indels. These conservation levels can be reflected in a *position-specific* scoring system for more powerful homology searches for the particular sequence family at hand.

The introduction of *profiles* by Gribskov et al. [97] for protein families was an early use of position-specific scoring systems. Gribskov profiles are constructed for a particular sequence family from a multiple alignment of homologs and include position-specific scores for both indels and substitutions. The substitution scores are calculated using both observed substitutions in the alignment as well as PAM matrix values.

Given a position-specific scoring system, a target sequence can be aligned to the profile using a slightly modified version of the NWS or SW algorithms and a different scoring system. Instead of using a single substitution matrix of size 20×20 (for proteins), a matrix of size $20 \times N$ is used, with one column for each of the N positions of the query sequence. Similarly, there are now $2 \times N$ gap scores - two for each position (for affine gap scoring, or just one for linear gap scoring). Thus alignment to a profile takes roughly the same amount of time as a single pairwise alignment using full dynamic programming. Interestingly, although heuristic acceleration approaches like those in BLAST and FASTA were introduced about ten years ago for pairwise methods, similar heuristics have only been applied to profiles within the past year in the HMMER 3 package [62].

A major advantage of profiles is that they reduce the problem of "all versus all" pairwise comparisons to "all versus many", where "many" refers to a collection of profiles, because one profile can be used to describe a family of many sequences. As of this writing, the PFAM database contains about $10,000$ protein domain family profiles which cover about 75% of the $5 * 10^6$ sequences in the non-redundant UniProtKB database [76]. Searching all of these profiles against all the sequences requires roughly $5 * 10^{10}$ comparisons, while an all versus all pairwise approach would require $2.5 * 10^{13}$, about 500 times as many.

Another advantage of profiles is that they have proven more powerful for remote homology detection than pairwise methods. A single profile built from N family members can often detect remote homologies that none of the N pairwise searches detect. The higher performance of profiles versus pairwise methods was reported initially by Gribskov et al. [97], but has been repeatedly shown in empirical benchmarks since then [18, 158, 203].

A potential disadvantage of profiles relative to pairwise methods is the requirement of calculating many more scoring parameters. For pairwise methods, the scores from substitution matrices are well-defined from a large amount of training data [110, 111]. But how can the position-specific scores of a profile be appropriately set? A well-principled mathematical framework using probabilistic modeling techniques was introduced with profile hidden Markov models, as described below.

Probabilistic profiles

Probabilistic modeling further improves upon the homology search methodologies mentioned above. Probabilistic models offer a well-founded theoretical basis for estimating parameters and for calculating other useful quantities of interest for sequence analysis applications. (The methods described in Chapters 2, 4, and 8 are all examples of the latter quality.)

One of the simplest types of probabilistic models used for sequence analysis are hidden

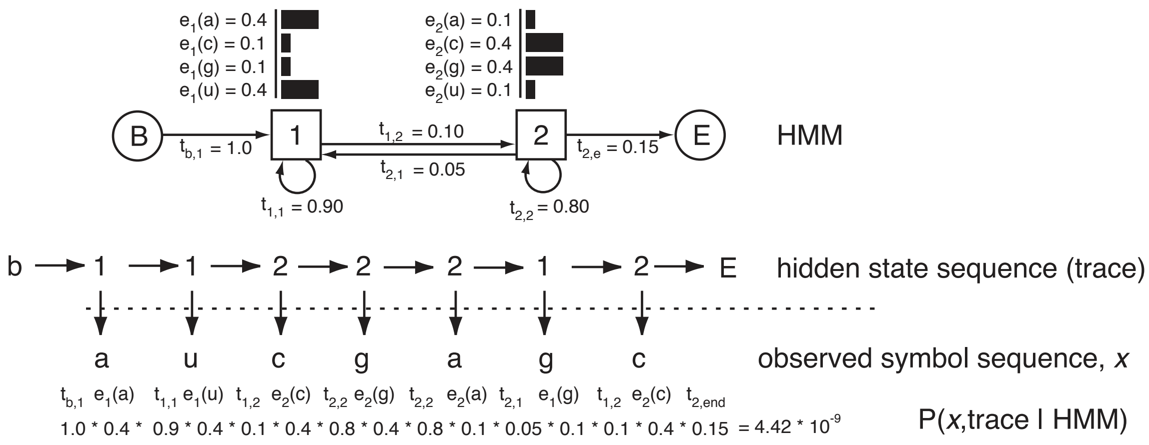


Figure 1.7: **A toy hidden Markov model.** States are labeled, `B`, `1`, `2`, and `E`. Transitions connecting states are denoted by arrows and labeled with transition probabilities t. Emission probabilities, e, are shown above the two emitting states `1` and `2`. The "hidden state sequence (trace)" is an example path through the model and the "observed symbol sequence" x is a corresponding set of emissions from that path. The particular trace shown is one possible path through the model that could have generated x. The probability of this particular trace and observed sequence x is calculated as $4.42 * 10^{-9}$, the product of emission and transition probabilities that generated it. This figure is based on a similar one in [60].

Markov models (HMMs) [215]. HMMs are organized as sets of states that generate, or *emit*, residues and then transition to other states. The frequencies of emissions and transitions from a state are defined as probabilities that sum to 1. A very simple four state HMM is shown in Figure 1.7. The `B` and `E` states are the requisite begin and end states necessary for beginning and ending sequences. State `1` emits `AU` rich sequence as defined by its emission probability e_1 shown above the state and state `2` emits `GC` rich sequence as defined by e_2. Sequences are generated from an HMM by choosing a residue to emit and a next state to transition to based on the emission and transition probabilities of the current state. An example state sequence through the model, or *trace*, and the observed symbol sequence that is emitted is shown in Figure 1.7.

For many applications, the utility of HMMs, and probabilistic models in general, is not their ability to generate sequences but rather their ability to score target sequences given the model (the query), or to align a target to a model. For scoring, the desired quantity is the probability that an HMM generated a given sequence. For alignment, one wants the most likely, or optimal, trace of the sequence to the model and its probability. In these cases, it is assumed that the model generated the sequence and that the state sequence responsible is hidden (hence *hidden* Markov models). The Forward and Viterbi dynamic programming algorithms compute these quantities [53]. Specifically, Forward computes $P(x|\text{ HMM })$, and Viterbi computes $P(x, \hat{\pi}|\text{HMM})$, where $\hat{\pi}$ represents the optimal trace of x to the model. These algorithms are generally applicable for any type of HMM and scale in time and memory with the product of M, the number of states in the HMM, and L, the length of the target sequence.

HMMs have been successfully applied in many different sequence analysis applications, including gene-finding [19, 27], pairwise alignment [26], multiple alignment [119], and structure prediction [132]. Additionally, a special type of HMMs, called *profile HMMs*, have been particularly useful for the problem of homology search.

Profile HMMs merge the ideas of probabilistic modeling with Gribskov's profiles. They were introduced for protein homology search by Anders Krogh, David Haussler and coworkers [142]. Given an alignment of sequence family members, a profile HMM can be built that models that family. An example profile HMM for a small protein family is shown in Figure 1.8. Position-specificity is achieved by organizing the states of a profile HMM into *nodes*, with each node modeling a separate *consensus* position of the input alignment. (Consensus positions are commonly defined as any input alignment column in which fewer than half of the sequences are gaps.) Each node contains three states: a *match* state, an *insert* states, and a *delete* state. Match states model an emitted residue, one of the 20 possible amino acids in the corresponding consensus positions. Insert states emit residues in between consensus positions. Delete states allow consensus positions to be skipped in target sequences; these states are special in that they do not emit residues. Insert states can *self-transition* to themselves in a profile HMM (Figure 1.8). The distinction between the transition for entering an insert states from a match state, and re-entering it via this self-transition equate to an affine gap scoring system.

1	2	3
c	a	f
c	g	w
c	d	y
c	v	f
c	k	y

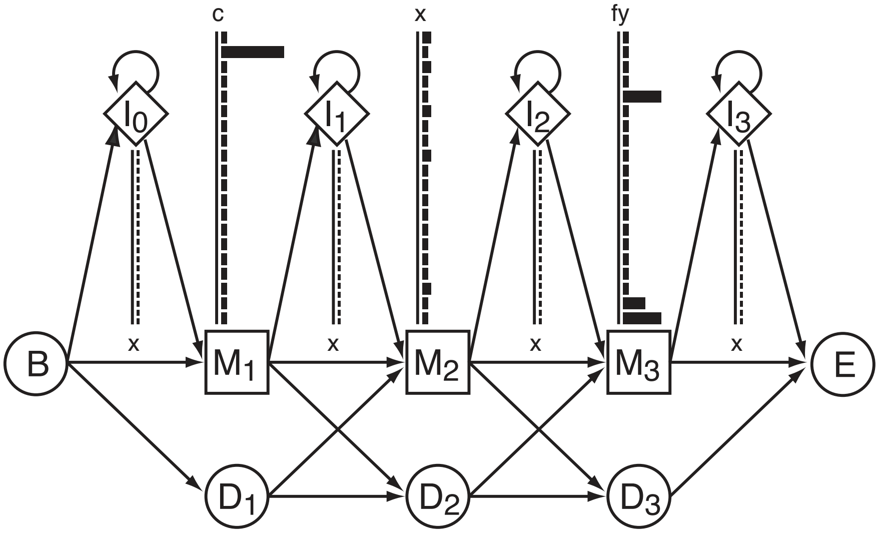


Figure 1.8: **A simple profile HMM.** The profile HMM (right) was built from the five-sequence alignment (left). The model contains three nodes representing the three consensus columns of the alignment, each containing a match (`M`), insert (`I`), and delete (`D`) state as described in the text. The emission probability distributions are depicted above each emitting match or insert state. `B` and `E` states begin and end traces/alignments. Possible transitions are depicted as arrows connecting states. This figure is reprinted with permission from [60].

Profile HMM parameterization

Similarly to Gribskov's profiles, profile HMMs use an alignment of homologous sequences of a protein family to define position-specific scoring parameters. I will refer to this alignment as the *seed* alignment. In profile HMMs, these parameters are the model's emission and transition probabilities. They are calculated as mean posterior probability estimates based on the observations in the input seed alignment and a prior probability as follows:

$$p_x = \frac{c_x + \alpha_x}{\sum_y (c_y + \alpha_y)} \tag{1.2}$$

Where x corresponds to a particular transition or emitted symbol for a particular state and $\sum_y$ is summing over all possible transitions or emissions from that particular state [53]. This equation is used to calculate the transition and emission probabilities for all states in the model. The c_x values are observed counts of the emissions and transitions in the implicit traces of the aligned sequences in the seed alignment. For example, the sequence `CAF` in the alignment from Figure 1.8 implies a path through the three match states of the model `M1`, `M2`, and `M3`, emitting `C`, `A`, and `F` respectively from each. The `C` emission adds 1 to the c value for emitting `C` from state `M1`.

Uninformative and informative priors

In equation 1.2, the α terms denote the *prior*; these augment the observed counts in the seed alignment and prevent emission or transitions not observed in the seed from having 0.0 probability [53]. If all α_y values are all equal, the prior is unbiased and called flat or *uninformative.* The use of an *informative* prior, in which α values differ, allows the injection of prior, biologically meaningful information into the model. Take again for example the alignment in Figure 1.8. If an uninformative prior of $\alpha = 1$ is used (corresponding to a Laplace "plus-one" prior [53]), the emission distribution for match state `M1` will be defined as:

$$\begin{array}{lll} \text{if } x = \texttt{C} & p(e_x) = \frac{3+1}{23} = & 0.1739 \\ \text{else} & p(e_x) = \frac{0+1}{23} = & 0.0435 \end{array}$$

In this case, our model will assign the highest probability to a `C` emission from this state, and about a quarter of that probability to each of the other 19 possible emissions. This is similar to the initial scoring scheme used with the NWS algorithm, of `+1` for a match and `0` for a mismatch. The equal score for all non-`C` residues implies they are all equally likely to appear at the position in the alignment. Because it is observed that the first position preferentially includes cysteine (`C`), one may want to assign probabilities to the other amino acids based on their biochemical similarity with cysteine. This could be achieved using different α values for each residue. Such a scheme may be appropriate for cysteine rich columns, but a single set of α values would not generalize very well to different contexts.

In the 1990s, Kimmen Sjölander, David Haussler and colleagues pioneered the use of *mixture Dirichlet priors* as informative priors in profile HMM parameterization [23, 131, 234]. A Dirichlet mixture is estimated from a large set of multiple alignment columns with the goal of capturing the typical amino acid frequencies present in those columns. The mixture is composed of components, each with its own own α vector. Each component can be thought of as modeling a separate environment with different amino acid preferences, as encoded in their α values. For example, small amino acids would have high corresponding α values in a component for buried residues in the core of a protein.[2]

Using Dirichlet mixture priors has been shown to significantly improve the sensitivity of profile HMMs, especially for models built from few (less than 20) sequences where the relative weight of the α values in equation 1.2 is high (because $\sum_y c_y$ is low) [131, 234]. In 1996, Sjölander et al. [234] estimated a nine component mixture from the BLOCKS database [110] that is currently employed by the two widely used profile HMM packages SAM [133, 142] and HMMER [62].

---

[2] In fact, the Dirichlet mixture estimation is unsupervised and there is no bias towards deriving components that are particular for a certain class of amino acids. However, in many cases a biological explanation of the resulting components is possible. The technical details of Dirichlet mixture training are outside the scope of this work. For details see [23, 234].

Entropy weighting

In 1998, Kevin Karplus introduced *entropy weighting* for profile HMM parameterization [133]. This technique defines a weighting factor w, where $w < 1$, by which the c values are multiplied in the parameterization equation given in equation 1.2:

$$p_x = \frac{(w * c_x) + \alpha_x}{\sum_y((w * c_y) + \alpha_y)} \tag{1.3}$$

This has the effect of reducing the contribution of the observed emissions and transitions from the input alignment, and increasing the contribution of the prior to the final parameters. The w value that is used varies from model to model. It is defined as the value that gives a pre-specified, target average match state entropy. The entropy r of a match state is defined as $r = -\sum_x e(x) \log_2 e(x)$.

The key idea of entropy weighting is that only a certain amount of sequence conservation need be modeled by a profile HMM for it to be able to effectively discriminate between true homologs and unrelated sequences in database searches. Models built from alignments with very highly similar sequences may exceed this conservation limit and be overly specific towards finding sequences like those in the seed alignment, at the expense of missing more remote homologs. Entropy weighting counteracts this scenario by reducing the contribution of observations from the seed (the c values) as appropriate. Entropy weighting is a heuristic that is not well-founded in probability theory. However, the technique substantially improves a profile HMM's ability to find remote homologs [127, 133] and is used by both SAM and HMMER.

Log-odds scoring

Before a profile HMM is used for homology search, the transition and emission probabilities are converted into log-odds scores using a *null* model of random sequence. A simple null model that is commonly used is a single state HMM with an emission probability distribution equal to the background distribution of a large database (similar to the q values for BLOSUM scores) and a single possible transition, a self-transition with probability 1.0. Log-odds scores s_x for each possible HMM emission and transition are defined by simply dividing the relevant model probability p_x by the null model probability r_x (either the background probability of a residue for emissions, or 1.0 for transitions) and taking the base 2 logarithm: $s_x = \log_2 \frac{p_x}{r_x}$.

In practice, the profile HMM versions of the Viterbi and Forward DP algorithms typically use s_x values instead of p_x to calculate log-odds scores instead of probabilities as described above. Viterbi calculates the log-odds score that the sequence was generated from the HMM using the most probable path through the model versus from the null model. Forward calculates the log-odds score that the sequence was generated from the HMM (summed over all possible paths) versus from the null model.

Estimating statistical significance of profile HMM database hits

Several profile HMM packages, including SAM and versions 1 and 2 of HMMER, estimate statistical significance of database hits in a manner similar to gapped BLAST - via simulation. Empirically, high-scoring profile HMM hits to random sequences follow a Gumbel distribution with parameters λ and K. These are fit via simulation for each scoring system [62, 134]. While this is not a problem for BLAST because only a handful of scoring systems are typically used, it has more serious implications with profile HMMs because each model defines its own scoring system and requires its own expensive simulation to define the Gumbel parameters. In 2008, Sean Eddy determined that by redesigning the HMMER profile HMM architecture to make a truly local probabilistic model, values of λ and K could be analytically derived for any given model that resulted in highly accurate empirical E-value estimation, thus obviating the need for an expensive simulation. Eddy stated that the conjectures used to analytically derive the profile HMM Gumbel parameters were expected to apply to any appropriately defined local probability model, including local probability models of SW and more complex sequence and structure profile models described below.

1.4 Exploiting conserved structure in RNA homology searches[3]

Protein homology search by amino acid primary sequence comparison is powerful. At the amino acid level, BLASTP has no trouble detecting significant similarity down to about 25-30% amino acid sequence identity. Many protein coding regions conserve this level of similarity even across the deepest divergences in the tree of life amongst archaea, bacteria, and eukaryotes.

In contrast, RNA homology search by nucleotide primary sequence comparison is much less able to detect distant RNA homologies. BLASTN typically requires about 60-65% sequence identity to detect a statistically significant similarity for RNAs of typical length. Although some RNAs are very highly conserved over evolution (notably large and small subunit ribosomal RNAs, which are readily detected by sequence comparison in all species; the so-called human "ultraconserved" regions included regions of rRNA [11]), this is not the rule. Many functional RNA homologies are undetectable at the primary sequence level in cross-phylum comparisons (such as nematode/human or fly/human), because weakly or moderately conserved nucleic acid sequences can diverge to the 65% identity level in just a few tens of millions of years.

A striking example of this difference comes when searching for homologs of the components of some ribonucleoprotein (RNP) complexes. It is not uncommon to detect homologs of the protein components but not the RNA components of complexes such as SRP, RNase P, small nucleolar RNPs, and telomerase. The interpretation upon finding only the protein component is usually (and almost certainly correctly) that the RNP complex is present in the organism, but the RNA component(s) are too difficult to detect. For example, the probable presence of small nucleolar RNAs in archaea could be inferred from the presence of homologs of snoRNP protein components like fibrillarin well before snoRNA homologs were discovered [5, 199]. A similar situation can occur when identifying homologous cis-

---

[3] Part of this section is identical to the "Exploiting conserved structure in RNA similarity searches" section from Chapter 5 which I co-authored with Sean Eddy and submitted to be published as a book chapter.

regulatory RNA elements (such as riboswitches) for clearly homologous coding genes.

Table 1.1 shows some specific anecdotal examples. These data are fairly typical of searching databases with protein versus RNA queries. They demonstrate two key points about the relative difficulty in detecting homologs of functional RNAs. First, notice that for the protein coding genes, the statistical significance of the similarity (the E-value) is always much better (lower and more significant) when comparing their amino acid sequences rather than when comparing their DNA sequences, highlighting the additional statistical power inherent in searches at the amino acid level. This is the reason for the recommended practice of always comparing protein sequences at the amino acid level [207]. Second, notice that RNA components are usually much shorter than the coding sequence of the protein components, further compromising statistical signal and the ability of primary sequence analysis (BLASTN) to resolve homologous relationships from background. (To enable reproduction of these results, the accessions for the sequence data used in these searches is provided in Table 1.2.)

Primary sequence-based methods for detecting functional RNAs can be bolstered by exploiting the statistical signal present in the conserved secondary structure of many RNAs. Both RNAs and proteins tend to conserve a characteristic structure that is integral to their function, but structure-based homology search methods are used much more commonly for RNA than they are for proteins. What makes RNA secondary structure constraints of particular utility for computational sequence analysis is their simplicity and relative contribution of statistical signal. As mentioned earlier, RNA base-pairs induce strong pairwise correlations within RNA sequences that can be detected as covariations in multiple sequence alignment columns using comparative sequence analysis. The consensus structures of many RNAs have been accurately inferred in this way [107, 115, 181, 202].

How much extra information does RNA secondary structure conservation contribute in addition to primary sequence conservation? This question can be addressed using probabilistic profile log-odds scores that estimate a profile's ability to distinguish between a homologous and a non-homologous RNA. Figure 1.9 shows the average score of profiles for

			amino acid sequence			coding sequence				RNA				
organism 1	organism 2	protein		BLASTP			BLASTN		RNA		BLASTN		Infernal	
query	target	name	len	%id	E-value	len	%id	E-value	name	len	%id	E-value	%id	E-value
Methanocaldococcus jannaschii	Pyrococcus horikoshii	Rpp29	95	50%	2.3e-19	288	62%	4.2e-09	RNase P RNA	258	72%	9.2e-06	51%	6.7e-09
Bacillus cereus	Bacillus subtilus	lysC	409	67%	1.0e-135	1230	66%	5.4e-94	Lysine riboswitch	187	63%	2.0e-05	59%	2.9e-18
Sulfolobus solfataricus	Thermococcus kodakarensis	rpl30p	158	42%	4.9e-29	477	61%	0.38	5S rRNA	115	73%	4.3e-05	62%	7.6e-10
Bacillus subtilus	Clostridium acetobutylicum	glmS	600	46%	3.2e-138	1803	57%	2.5e-66	glmS riboswitch	168	63%	3.5e-04	56%	1.6e-14
Escherichia coli	Bacillus subtilus	ffh	453	54%	9.8e-124	1362	63%	2.3e-85	SRP RNA	100	66%	2.7e-03	62%	2.2e-13
	Candidatus P. amoebophila			44%	2.5e-102		57%	8.6e-36			68%	0.78	47%	6.7e-06
Escherichia coli	Klebsiella pneumoniae	btuB	614	57%	2.6e-192	1845	65%	7.0e-113	Cobalamin riboswitch	191	78%	5.7e-18	77%	2.5e-33
	Yersinia enterocolitico			52%	1.5e-173		60%	6.4e-83			74%	3.2e-09	67%	9.3e-21
	Vibrio cholerae			38%	8.1e-107		57%	6.4e-34			72%	0.043	57%	4.5e-05
	Acinetobacter baumannii			26%	5.0e-46		61%	2.4e-06			65%	2.6	49%	3.7e-05

Table 1.1: **Examples of identifying coding region homologies by amino acid sequence versus nucleic acid sequence comparison (blastp vs. blastn), compared to identifying RNA homologies by primary sequence versus structure/sequence comparison (blastn vs. infernal)**. For each query/target pair, the query sequence was searched against the target genome (for coding sequence and RNA searches) or predicted proteome (for amino acid sequence searches) using the indicated search programs. "Len" indicates unaligned length of the query. "% id" indicates percent identity of local alignments of hits returned by each method. (BLASTN RNA alignments are always higher percentage id than INFERNAL alignments, but are also usually significantly shorter.) Query RNAs were selected from candidates found by INFERNAL in each listed query's genome sequence using the the RFAM 9.1 CM for the appropriate family (listed below). Each query RNA was used to build a CM using the INFERNAL imposed RFAM structure, and each CM was calibrated and used to search the target genomes. RFAM family IDs for each family listed in "RNA name", in row order, are: RF00373, RF00168, RF00001, RF00234, RF00169, RF00174. For riboswitches, the protein components are always immediately downstream of the RNA components. Versions used: `WU-BLASTN-2.0MP`, `WU-BLASTP-2.0MP` and `cmsearch` from INFERNAL version 1.0. For BLASTN, the `-w=5` option was always used, and the `-kap` option was used only if it resulted in a more significant (lower) E-value for the target sequence.

organism name	genome accession	protein name	protein accession	RNA name	RNA genomic coordinates
Methanocaldococcus jannaschii	NC_000909.1	Rpp29	NP_247439.1	RNase P RNA	643504-643761
Pyrococcus horikoshii	NC_000961.1	Rpp29	NP_143607.1	RNase P RNA	168208-168414
Bacillus cereus	NC_003909.8	lysC	NP_0978199.1	Lysine riboswitch	1818638-1818452
Bacillus subtilis	NC_000964.2	lysC	NP_390725.1	Lysine riboswitch	2910116-2909946
Sulfolobus solfatarius	NC_002754.1	rpl30p	NP_342208.1	5S rRNA	78064-77946
Thermococcus kodakarensis	NC_006624.1	rpl30p	YP_183933.1	5S rRNA	1769482-1769599
Bacillus subtilis	NC_000964.2	glmS	NP_388059.1	glmS riboswitch	200006-200173
Clostridum acetobutylicum	AE001437.1	glmS	AAK78142.1	glmS riboswitch	179915-180074
Escherichia coli	NC_000913.2	ffh	NP_417101.1	SRP RNA	475679-475778
Bacillus subtilis	NC_000964.2	ffh	NP_389480.1	SRP RNA	26531-26633
Candidatus P. amoebophila	NC_005861.1	ffh	YP_007653.1	SRP RNA	871975-872074
Escherichia coli	NC_000913.2	btuB	NP_418401.1	Cobalamin riboswitch	4161407-4161597
Klebsiella pneumoniae	CP000647.1	btuB	ABR78634.1	Cobalamin riboswitch	4660061-4660248
Yersinia enterocolitica	NC_08800.1	btuB	YP_001004531.1	Cobalamin riboswitch	157101-157301
Vibrio cholerae	NC_009457.1	btuB	YP_001218242.1	Cobalamin riboswitch	2498535-2498369
Acinetobacter baumannii	NC_011586.1	btuB	YP_002320687.1	Cobalamin riboswitch	3485342-3485537

Table 1.2: **GenBank genome and protein accessions and RNA genomic coordinates for examples from Table 1.1**.

about 100 RNA sequence families, comparing models of sequence conservation alone (profile HMMs) to models of sequence plus RNA secondary structure conservation (*covariance models*, CMs). CMs are discussed in more detail below; for the present point, it is sufficient to know that they are profile probabilistic models that derive their parameters from a seed alignment just like profile HMMs do, but that they additionally model the conserved secondary structure of an RNA family.

Because profile HMM and CM log-odds scores are derived using base 2 logarithms, the unit of score is a *bit*, which is a measure of information content [166, 233]. A toy example seed alignment and the corresponding information for a sequence profile and a sequence and structure profile is depicted in Figure 1.10. Consider a match state that models a seed alignment column with a perfectly conserved RNA residue (for example, column 5 in Figure 1.10), with an emission probability of 1.0 for that residue.[4] The probability of 1.0 for that residue compared with a null model probability of 0.25 corresponding to the null model of uniform expected background means that the state is contributing $\log_2 \frac{1.0}{0.25} = 2$ bits of information - two yes/no questions must be asked to narrow four possibilities down to one, thus two "bits" (binary units) of information. A match state modeling a column where each residue occurs with equal probability (same as expected background, column 9 in Figure 1.10) has zero bits of information. Imagine a single match state modeling two positions that contain a covarying Watson-Crick base-pair in which each of the four possible base-pairs occur with equal probability $\frac{1}{4}$ (columns 3 and 8 in Figure 1.10). In a sequence only model the two positions contribute zero bits of information, but in a structure/sequence model this pair contributes two bits of information from the pairwise correlation (the expected background in these columns is 0.0625 for each of the 16 possible base-pairs, but only 4 are observed with probability 0.25 each). In contrast, two columns that form a Watson-Crick base-pair that is perfectly conserved (a `GC` with probability 1.0 for example, columns 1 and 11 in Figure 1.10) always contribute four bits of information, regardless of whether they are modeled together as a pair ($\log_2 \frac{1.0}{0.0625} = 4$), or independently ($\log_2 \frac{1.0}{0.25} + \log_2 \frac{1.0}{0.25} = 4$).

---

[4] For this explanation, the contribution of the prior in emission parameter estimation will be ignored (the α in equation 1.2). This equates to *maximum likelihood* parameter estimation.

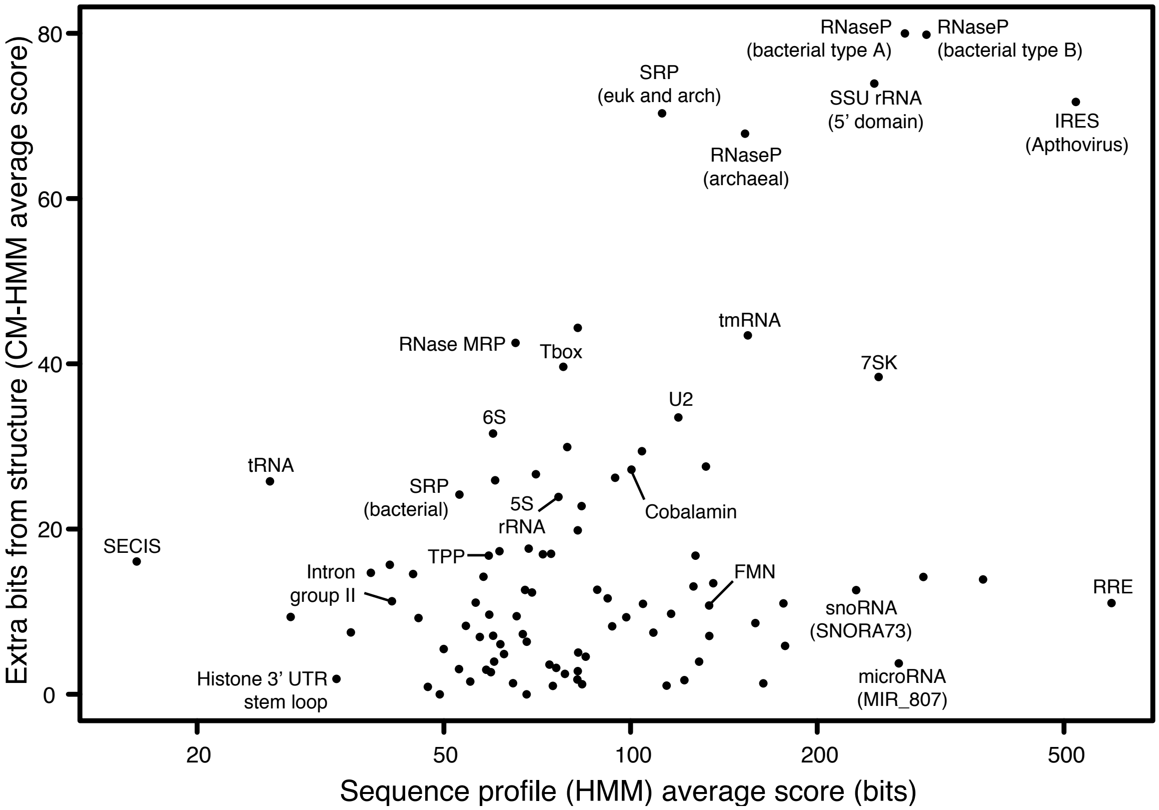


Figure 1.9: **Additional information (in bits) gained by structure/sequence profiles versus sequence-only profiles for various RNA families.** Structure/sequence profiles are most advantageous for families with less primary sequence information (towards left) and more secondary structure information (towards top), so RFAM families that gain the most from including secondary structure terms in a homology search are those toward the upper left quadrant. Data shown for the 95 RFAM release 9.1 [89] families with 50 or more sequences in the "seed" alignment. For each family, the seed alignment was used to build two profile models, one with structure (sequence/structure profile CM model) and one without (sequence profile HMM model). From each model, 10,000 sequences were generated and scored, and the average score per sampled sequence was calculated. Several of the outlying points are labeled by the name of RNA family as given by RFAM. Note that the x-axis is drawn on a log scale. Models were built and sequences were generated and scored using INFERNAL version 1.0 programs `cmbuild`, `cmemit` and `cmalign`.

Thus, the best case for extracting useful sequence information from RNA secondary structure are covarying base-pairs that are individually not conserved in primary sequence at all. The more highly conserved the aligned RNA sequences are, the more primary sequence information content and less covariation will be seen.

Importantly, for local sequence alignment searches using probabilistic models, there is a direct, intuitive connection between the score in bits and the statistical significance (E-value) of a detected match [59]. Roughly speaking, every 3 or so bits of score improves the E-value by a factor of ten-fold (for high scores, the E-value is an exponential function of the bit score x; E is proportional to 2^{-x}). So, as a rule of thumb, extracting ten more bits of information for a homology search means shifting E-values by three orders of magnitude. This increase in resolution doesn't matter much if a sequence is already readily detected by primary sequence comparison (improving an already significant E-value of 10^{-30} to 10^{-33}, for example), but it becomes important when lifting a marginally insignificant E-value to significance (0.1 to 10^{-4}, for example).

Figure 1.9 shows the extra bits of information contributed by including RNA secondary structure in "typical" RNA search models. These models are all position-specific profiles built from alignments in the RFAM RNA families database, described below. There is substantial variation from family to family, but the extra information contributed by secondary structure is usually on the order of 10-20 bits or more, depending on the length and conservation of the alignment, which would be expected to improve E-values of homologs by about 3-6 orders of magnitude. This improvement can be seen in the results of the anecdotal searches of Table 1.1 comparing the E-values obtained by primary sequence BLASTN searches to INFERNAL, a sequence and secondary structure RNA homology search, as discussed in more detail below. The conclusion here is that while primary sequence is still the dominant source of information (at least for these particular "typical" searches; it is, of course, possible to imagine searching for RNAs with zero sequence information and only secondary structure information), adding secondary structure contributes enough information content that we can expect a structure and sequence method to resolve some homologs that were not quite resolvable by sequence analysis alone.

I now turn to the question of how to efficiently computationally model conserved RNA secondary structure. Several different techniques have developed, but my discussion will mainly focus on the covariance model approach mentioned above, on which the remainder of work in this dissertation is based. CMs are profiles belonging to a class of probabilistic models called *stochastic context-free grammars* (SCFGs). I will first describe why SCFGs are well-suited for modeling RNA structure, and then detail the construction, parameterization and application of CMs for RNA homology search.

1.5 Stochastic context-free grammars for RNA sequence and structure modeling

Stochastic context-free grammars (SCFGs) are extensions of HMMs that can model both the sequence and structure of RNAs. SCFGs are most easily understood in the context of formal grammar theory developed in computational linguistics to help understand the structure of natural languages. Noam Chomsky's hierarchy of formal grammars [35, 36] provides a general theory for modeling strings of symbols that is applicable to biological sequence analysis [53]. Formal grammars are generative models capable of generating sequences, but in many cases their utility lies in the ability to *parse* sequences of symbols to determine if they could have been generated from the grammar. A grammar is defined by a set of *terminal* and *nonterminal* symbols and a set off *production rules* P defining how symbols can be generated. For example, the following set of production rules defines a grammar that can generate any sequence of `a`s and `b`s and `n`s:

$$S \rightarrow aS \mid bS \mid nS \mid \epsilon$$

In this grammar, the a, b, n and ϵ (a special case, the null string) are terminal symbols, and S is the lone nonterminal symbol. In this notation, for brevity, each possible production from S is separated by a $|$. The string *banana* can be *derived* from this grammar using the

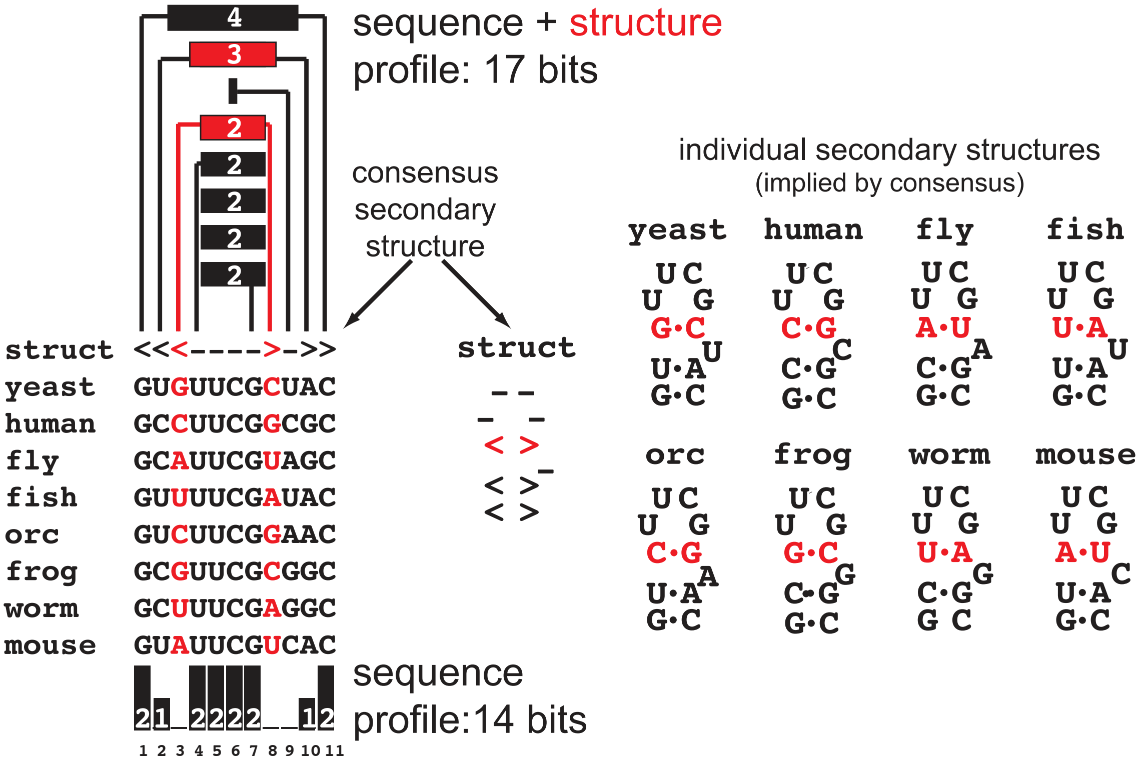


Figure 1.10: **Information in a sequence-only versus a sequence and structure profile.** The eight sequence seed alignment for a fabricated RNA family used to build both of the profiles is on the left. The `struct` line denotes the consensus secondary structure of the family, which is ignored by the sequence profile but used in the sequence and structure profile to define dependencies between base-paired columns indicated by matching nested `<` and `>` characters and connected by lines at top of figure. The `<` and `>` characters are matched like nested parentheses in a mathematical formula. The eight individual secondary structures, defined by imposing the consensus structure on each sequence are shown on the right.

following rules:

$$S \Rightarrow bS \Rightarrow aS \Rightarrow nS \Rightarrow aS \Rightarrow nS \Rightarrow aS \Rightarrow \epsilon$$

HMMs are stochastic regular grammars

Chomsky's hierarchy divides formal grammars based on the complexity of the types of languages they can recognize as well as the computational complexity required to parse sequences of symbols using the grammars. *Regular grammars* are the lowest level of the hierarchy. The example above is an regular grammar. Regular grammar production rules are heavily restricted to only the forms: $S \rightarrow aS$ or $S \rightarrow a$, where S and a represent possible nonterminal and terminal symbols respectively.

HMMs are examples of *stochastic* regular grammars, which extend a probabilistic component to grammars and their associated production rules. In a stochastic grammar, "yes" or "no" binary pattern-matching for sequences is replaced with the probability that the sequence was generated from the grammar. This is achieved by associating each production rule with a probability with the constraint that all rules from each possible nonterminal sum to 1. Roughly speaking, HMM states correspond to nonterminals, emitted residues correspond to terminals, and state transitions correspond to production rules.

For example, the simple HMM depicted in Figure 1.7 can be defined by the following set of production rules:

$$\begin{array}{rcl} B & \rightarrow & \epsilon 1 \\ 1 & \rightarrow & a1 \mid c1 \mid g1 \mid u1 \mid a2 \mid c2 \mid g2 \mid u2 \\ 2 & \rightarrow & a2 \mid c2 \mid g2 \mid u2 \mid a1 \mid c1 \mid g1 \mid u1 \mid E \\ E & \rightarrow & \epsilon \end{array}$$

A profile HMM architecture defines a characteristic way of organizing regular grammar production rules that is useful for setting position-specific probabilities of a sequence family. For example, the rules in the profile HMM architecture in Figure 1.8 can be summarized as:

$$
\begin{aligned}
B &\rightarrow \epsilon I_0 \mid \epsilon M_1 \mid \epsilon D_1 \\
\\
M_y &\rightarrow x M_{y+1} \mid x D_{y+1} \mid x I_y \\
I_y &\rightarrow x M_{y+1} \mid x I_y \\
D_y &\rightarrow \epsilon M_{y+1} \mid \epsilon D_{y+1} \\
\\
M_n &\rightarrow x E \mid x I_n \\
I_n &\rightarrow x E \mid x I_n \\
D_n &\rightarrow \epsilon E
\end{aligned}
$$

Here, n is the number of nodes, y is any number from 1 to $n-1$, and x represents any possible amino acid. This is an abbreviated list of the complete set of rules but it demonstrates the modular, repetitive nature of the node-based architecture of profile HMMs.

Context-free grammars

Context-free grammars (CFGs) are one level of complexity above regular grammars in the Chomsky hierarchy [35, 36]. The form of CFG production rules is slightly less restricted than that of regular grammars. The restrictions on the left hand side of CFG and regular grammar production rules are equivalent - both must contain a single nonterminal. However, the right hand side of CFG production rules can contain any combination of terminals and nonterminals, allowing CFGs to generate two symbols with a single production rule. This is appropriate when the two symbols are dependent on each other, such as between covarying base-paired residues in RNA sequences. Regular grammars, which must generate a single terminal at a time, cannot model these dependencies in RNA sequences. This is the key reason that CFGs have been applied to RNA sequence analysis problems.

An example of a simple CFG that generates a simple dual stem-loop RNA sequence and structure is given below[5]:

---

[5] This grammar is nearly identical to one in [53]

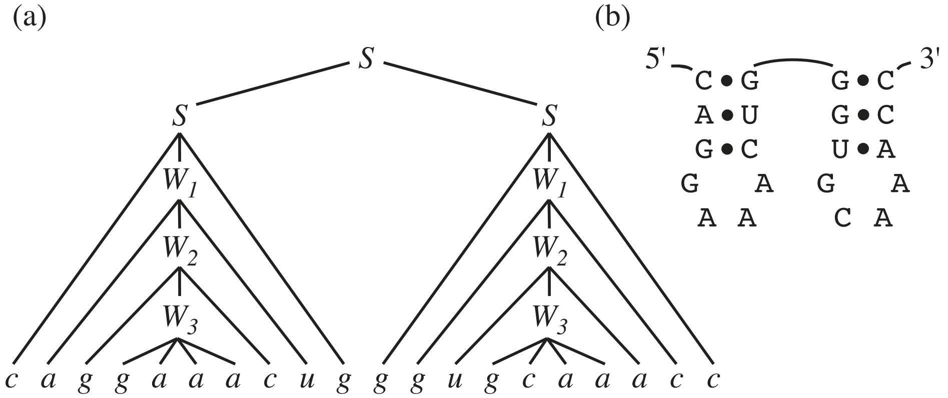


Figure 1.11: (a) A parse tree for `CAGGAAACUGGGUGCAAACC` and the dual stem-loop grammar. (b) The RNA secondary structure for the same sequence, which corresponds closely to the parse tree representation. This figure is reprinted with permission from [53].

$$
\begin{aligned}
S &\rightarrow SS, \\
S &\rightarrow aW_1u \mid cW_1g \mid gW_1c \mid uW_1a, \\
W_1 &\rightarrow aW_2u \mid cW_2g \mid gW_2c \mid uW_2a, \\
W_2 &\rightarrow aW_3u \mid cW_3g \mid gW_3c \mid uW_3a, \\
W_3 &\rightarrow gaaa \mid gcaa.
\end{aligned}
$$

Figure 1.11 shows a parse of an RNA sequence using this CFG. A CFG parse is commonly called a *parse tree* which, in the case of RNA sequences, has a tight correspondence with the secondary predicted structure of the sequence. In biological sequence analysis, a parse tree implies a unique *alignment* of symbols to nonterminals of a CFG, so sometimes the terms are used interchangeably in sequence analysis even though the converse is not always true: an alignment does not necessarily imply a unique parse tree [50, 91].

Stochastic context-free grammars (SCFGs) are the probabilistic analogs of CFGs, mirroring the relationship of HMMs to regular grammars. SCFGs have states that correspond to CFG nonterminals, with emissions and transitions and associated probabilities that correspond to production rules. SCFGs provide a general toolkit for RNA sequence and structure analysis. They have been applied to RNA gene-finding [209, 220], structural inference of

RNAs [139, 140], and simultaneous structural inference and alignment of RNAs [15, 51, 117]. For this work, the relevant applications are sequence- and structure-based RNA homology search (for part 1 of this work) and alignment (for part 2 of this work) using profile SCFGs.

1.6 Covariance models: profile stochastic context-free grammars

The concept of profiles was extended to SCFGs soon after it was for HMMs. Profile SCFGs were introduced independently in 1994 by Yasu Sakakibara and David Haussler's group [224, 226] and by Sean Eddy and Richard Durbin [65]. Similarly to profile HMMs for protein sequence analysis, the primary application of profile SCFGs has been RNA sequence- and structure-based homology search

My work has focused on Eddy and Durbin's *covariance model* (CM) formulation of profile SCFGs as implemented in the INFERNAL software package [191]. This discussion pertains to version 0.55 of INFERNAL, the current version at the onset of this work in 2004 [57]. In this section, I will discuss how CMs are built and parameterized, list some of their important limitations, and describe some existing methods for addressing those limitations. For more details on CMs see [53, 57, 65, 141, 189, 190, 192].

CM construction

A CM is built from a multiple sequence alignment of an RNA family that is annotated with a consensus, well-nested secondary structure. CMs are composed of a series of nodes organized as a *guide tree* that corresponds closely with the consensus structure of the input alignment. There are eight node types, each corresponding to a different type of structural element:

node type	corresponding structural element
`MATP`	base-pair
`MATL`	single stranded residue
`MATR`	single stranded residue
`ROOT`	beginning of complete structure
`BIF`	bifurcation
`BEGL`	beginning of substructure
`BEGR`	beginning of substructure
`END`	end of substructure

INFERNAL defines a set of rules for guide tree construction that ensure that there is exactly one guide tree for each possible consensus structure [192]. A toy alignment and its resulting guide tree is shown in Figure 1.12. As with profile HMMs, only *consensus* columns (those columns in which fewer than half the residues have gaps) are modeled by nodes. Note that each consensus column corresponds to either a `MATP`, `MATL`, or `MATR` node. These nodes are responsible for the *match* emissions of the consensus positions of the model. `MATL` and `MATR` nodes model single stranded consensus positions and are very similar to profile HMM nodes. `MATP` nodes emit two base-paired residues at once. The other node types (`ROOT, BIF, BEGL, BEGR and END`) are necessary only to model the possible branching patterns of RNA structures. Each model contains a single `ROOT` node at the top, the root of the guide tree. The number of `BIF` nodes varies with the complexity of the consensus structure, one is necessary for each *multi-branched loop* (Figure 1.1), which are also called *bifurcations.* There is exactly one `BEGL` and one `BEGR` node for each `BIF` node, that start, or root, each of the two subtrees created by the bifurcation. In the special case when the consensus structure contains zero base-pairs, the resulting guide tree will contain only `MATL` nodes and a single `ROOT` and `END` node. Such a model would correspond nearly exactly to a profile HMM (Figure 1.8).

The guide tree represents the fixed length consensus sequence and secondary structure of the family being modeled. To allow for insertions and deletions at any position relative

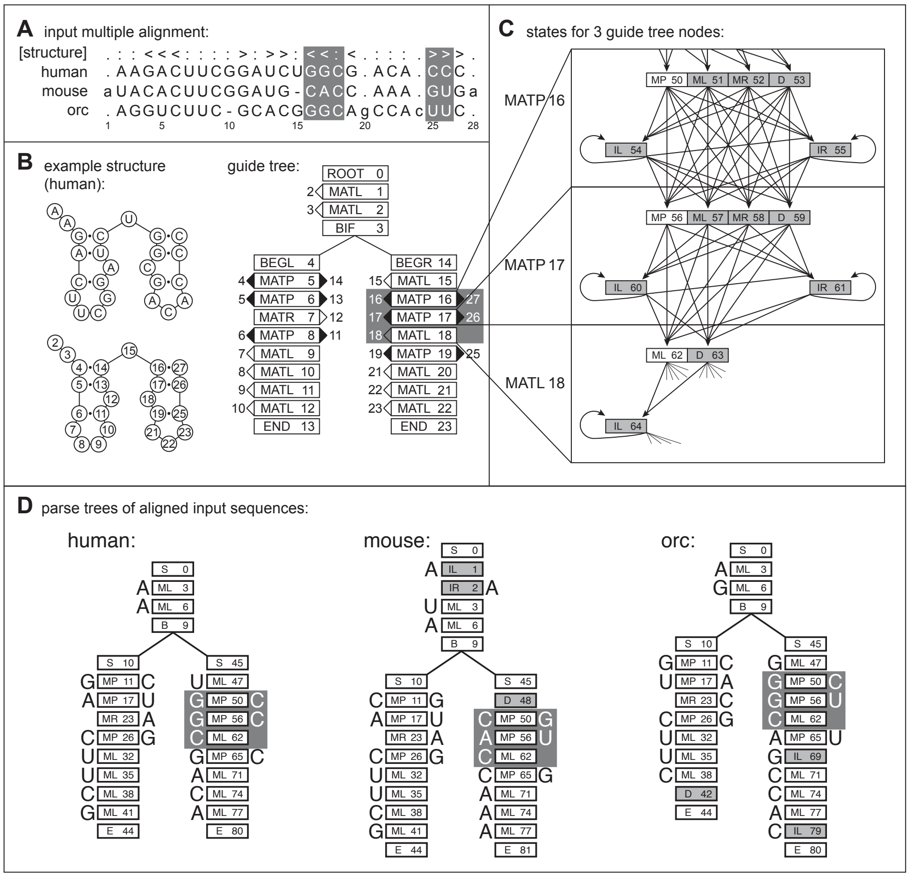

A input multiple alignment:
[structure] . : : < < < : : : : > : > > : < < : < . : : : : . > > > .
human . A A G A C U U C G G A U C U G G C G . A C A . C C C .
mouse a U A C A C U U C G G A U G - C A C C . A A A . G U G a
orc . A G G U C U U C - G C A C G G G C A g C C A c U U C .
1 5 10 15 20 25 28
B example structure (human):
guide tree:
ROOT 0
MATL 1
MATL 2
BIF 3
BEGL 4
MATP 5
MATP 6
MATR 7
MATP 8
MATL 9
MATL 10
MATL 11
MATL 12
END 13
BEGR 14
MATL 15
MATP 16
MATP 17
MATL 18
MATP 19
MATL 20
MATL 21
MATL 22
END 23
C states for 3 guide tree nodes:
MATP 16
MATP 17
MATL 18
MP 50
ML 51
MR 52
D 53
IL 54
IR 55
MP 56
ML 57
MR 58
D 59
IL 60
IR 61
ML 62
D 63
IL 64
D parse trees of aligned input sequences:
human:
mouse:
orc:

Figure 1.12: **Input alignment, CM guide tree, state graph and parse trees.** A toy multiple alignment of three RNA sequences, with 28 total columns, 24 of which will be modeled as consensus positions. In the consensus secondary structure (`[structure]`) `<` and `>` symbols mark base-pairs, `:`s mark consensus single stranded positions, and `.`s mark non-consensus insert columns in which more than half the sequences contain gaps. Columns with white text on gray background correspond to the three nodes in (C). **(B):** The structure of one sequence from (A), the same structure with positions numbered according to alignment columns, and the guide tree of nodes corresponding to that structure, with alignment column indices assigned to nodes (for example `MATP` node 5 will model the base-pair between columns 4 and 14). Nodes shown on gray background are expanded to states in (C). **(C):** The state topology of three selected nodes of the CM. The consensus pair and singlet states (two `MP`s and one `ML`) are white, and the insertion/deletion states are gray. State transitions are indicated by arrows. **(D):** Parse trees of the three aligned sequences from (A) given the CM. The full sequences can be read by starting at the top left of each parse tree and following counterclockwise along the yield of the tree. States corresponding to the consensus path through the model are white, and insertions/deletions to the consensus are gray. The states and associated residues from the three nodes in (C) are shown on gray background. This figure is similar to one I published with Sean Eddy in [189] and originally derives from [192] which includes a figure of the full CM state graph for this example family.

to this fixed model, insert and delete states must exist for each node. An analogous, but implicit, step occurs in profile HMM construction when a node for each consensus column of the input alignment is populated with separate match, insert, and delete states.

There are nine CM state types, many of which occur in more than one node type and each of which corresponds to a particular production rule:

			node types containing each state type							
state type	description	production rule	`MATP`	`MATL`	`MATR`	`ROOT`	`BIF`	`BEGL`	`BEGR`	`END`
`MP`	match pair	$S \rightarrow aSb$	x							
`ML`	match left	$S \rightarrow aS$	x	x						
`MR`	match right	$S \rightarrow Sb$	x		x					
`D`	delete	$S \rightarrow S$	x	x	x					
`B`	bifurcation	$S \rightarrow SS$					x			
`S`	start	$S \rightarrow S$				x		x	x	
`E`	end	$S \rightarrow \epsilon$								x
`IL`	insert left	$S \rightarrow aS$	x	x		x			x	
`IR`	insert right	$S \rightarrow Sb$	x		x	x				

The CM grammar formulation implemented in INFERNAL defines the connectivity of states via transitions [192]. Each state may only transition to each insert state in its own node and to all non-insert states in the following node, with two exceptions. `B` states model bifurcations by taking requisite transitions to *two* states, and `E` states do not transition to any other state. Part C of Figure 1.12 shows the states and corresponding transitions for three nodes of the complete CM. (For a more complete example, see [192]). This definition of connectivity ensures that at least one non-insert state in each node is visited by any CM parse tree (except when the model is locally configured, as discussed below).

A parse tree and its corresponding sequence is generated from a CM just like in the CFG examples earlier in the chapter, with the main difference that each production rule (emission and transition) has an associated probability. Sequences are generated from the *outside-in* as opposed to from *left to right* as in regular grammars like HMMs, by starting at the `S` state of the `ROOT` node (referred to as a `ROOT_S` state by convention) and transitioning from state to state in the model following each state's production rules.

CM parameterization

A CM is parameterized using the implicit parse trees of each aligned sequence in the input alignment; that is, the parse tree that would generate the sequence as it appears in the alignment. These are determined using the mapping of the guide tree nodes to consensus positions. For example, in Figure 1.12, the `CG` consensus base-pair between alignment columns 6 and 11 is modeled by guide tree node 8, a `MATP` node. All three of the input sequences have a `CG` base-pair at these positions, which means their implicit parse trees include the use of the $S \rightarrow cSg$ production rule of node 8's `MP` state to emit a `C` and a `G` to consensus positions 6 and 11 respectively.

The emission and transition probabilities of a CM are then set in the same manner as in profile HMMs (Equation 1.2), as mean posterior probability estimates based on observed counts c_x from the implicit parse trees in the input alignment and a prior defined by α pseudocounts:

$$p_x = \frac{c_x + \alpha_x}{\sum_y (c_y + \alpha_y)} \tag{1.4}$$

Where x corresponds to a particular transition or emission for a particular state and $\sum_y$ is summing over all possible transitions or emissions from that particular state. This equation is used to calculate the transition and emission probabilities for all states in the model. In INFERNAL 0.55, all α values are set as 1.0, corresponding to an uninformative plus-one prior (as discussed for profile HMMs).

Log-odds scores (s_x) for each emission and transition x are derived from the corresponding CM probability and a null model probability in the same way they are for profile HMMs: $s_x = log_2 \frac{p_x}{r_x}$. The null model corresponds to a single state HMM with equiprobable emission probabilities for each `A`, `C`, `G`, and `U` ($r_a = r_c = r_g = r_u = 0.25$) and self transition probability of 1.0.

Scoring and aligning sequences using CMs

Given a parameterized CM, target sequences are scored using CM versions of the Cocke-Younger-Kasami (CYK) [120, 135, 285] and Inside algorithms [53]. CYK and Inside are the SCFG analogs of the HMM Viterbi and Forward algorithms, respectively. Given target sequence x, CM M, and null model R, CYK calculates:

$$S_c = \log_2 \frac{P(x, \hat{\pi}|M)}{P(x|R)},$$

the log-odds score that x was generated by the model using the most likely parse tree $\hat{\pi}$ that could have generated it. Inside calculates:

$$S_i = \log_2 \frac{P(x|M)}{P(x|R)} = \sum\nolimits_{\pi} \frac{P(x, \pi|M)}{P(x|R)},$$

the log-odds score that the target sequence was generated by the model (by summing over all possible parse trees π that could have generated it) versus the null model.

When performing homology searches, actual RNA homologs are usually embedded within much longer target sequences, such as chromosomes if a complete genome sequence is being searched. As a result, the implementations of the CM CYK and Inside algorithms used for homology search report high-scoring target *sub*sequences, and can be classified as *local with respect to the target.*

CMs can also be used to generate multiple alignments of homologous RNAs. In INFERNAL this is done using an implementation of CYK that is *global* with respect to the target to determine the most likely parse trees of full target sequences to a CM. Collections of parse trees for multiple sequences imply a multiple alignment, by placing the residues from different sequences that are associated with the same CM state in the same alignment column. For example, in Figure 1.12, the parse trees in part D correspond to the alignment in part A.

Local and global CMs

In addition to CM DP algorithms, the CM itself can be configured in both global and local modes [192]. The description up to this point has focused on global mode, for which all parse trees start at the lone `ROOT_S` state and visit at least one non-insert state in each node. This is enforced by the CM formalism's definition of transitions as explained above. In local mode, two additional types of transitions are allowed: *local begin* transitions are allowed from the `ROOT_S` state to any internal node (i.e. non-`END` and non-`ROOT` node) of the model, and *local end* transitions from any internal node to a special `EL` state. The `EL` state is similar to an insert state and includes a self-transition. Once a parse (sub)tree has entered the `EL` state it cannot transition back to another state of the model. These additional transitions allow a locally configured model to tolerate large insertions and deletions of sequences and substructures that are not part of the consensus model, but may exist in RNAs that are remotely homologous to the seed alignment sequences. Figure 1.13 provides an example of the use of local begin and end transitions.

Rfam: high-throughput homology search and annotation using CMs

The RFAM database is a curated collection of structural RNA families maintained at the Wellcome Trust Sanger Institute in Cambridge, UK that uses CMs for high-throughput annotation of putative RNA homologs in the large RFAMSEQ sequence database (currently about 120 Gb) [89]. Each family is represented by three pieces of data: a consensus structure annotated seed alignment, a CM built from the seed using INFERNAL's `cmbuild` program, and a *full* alignment of putative homologs. The database has grown rapidly since its inception in 2002 (Table 1.6). During the course of my thesis work the number of RNA families in the database has risen from less than 200 (release 5.0) to more than 1300 (release 9.1).

For each release of the database, BLAST is used as a *filter* for each family by searching RFAMSEQ for high-scoring subsequences using each seed sequence as a separate BLAST query. Hits with E-values less than a threshold ($E = 100$ by default) are saved and are reevaluated using the CM search algorithms implemented in INFERNAL's `cmsearch` program.

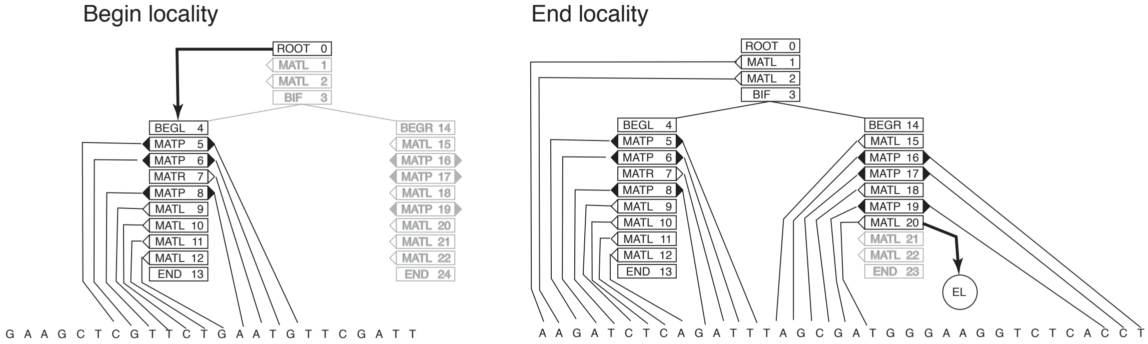


Figure 1.13: **Examples of CM local begin and end transitions.** An example of begin locality (left) and end locality (right) are shown using a guide tree representation of the CM from Figure 1.12. For begin locality (left), a *local begin* transition is used from `ROOT` node 0 to an internal node, in this case a `BEGL` node, bypassing nodes 1, 2, and 3 and the states within them. The parse tree continues only until `END` node 13 is reached, emitting (aligning to) the subsequence `CGTTCTGAATGT` in the target sequence. The remainder of the sequence is considered non-homologous to the model. For end locality (right), a *local end* transition from a state in `MATL` node 21 to the special `EL` state bypasses nodes 22, 23, and 24. The `EL` state implicitly emits (aligns to) the intervening target subsequence `GGGAAGGTCTCA` that is not aligned by the rest of the parse tree. This figure is reprinted with permission from [138].

Any sequence that receives a CM score above a manually curated family-specific bit score threshold (determined as the score just above the highest-scoring clearly false positive hit) are saved as putative homologs. The set of putative homologs is then aligned to the CM using INFERNAL's `cmalign` program to create the family's *full* alignment.

Each family's CM, seed alignment and full alignment are made freely available with each release. The alignments are useful for several purposes including phylogenetic inference and examining the phylogenetic range spanned by an RNA family. The CMs can be downloaded and used with the freely available INFERNAL to search for and annotate RNAs in other datasets, such as genomes, and to create structurally annotated multiple sequence alignments. As of this writing, according to Google Scholar, the three RFAM publications [89, 99, 100] have collectively been cited 1337 times. RFAM is an important resource. Its usage of INFERNAL serves as important motivation for the continued development and improvement of the software.

year	release	# of families	INFERNAL version	RFAM head	INFERNAL team
2002	0.1	4	0.3	SGJ	SRE
2002	0.2	12	0.3	SGJ	SRE
2002	0.3	21	0.3	SGJ	SRE
2002	1.0	25	0.4	SGJ	SRE
2002	2.0	30	0.55	SGJ	SRE
2003	3.0	36	0.55	SGJ	SRE
2003	4.0	114	0.55	SGJ	SRE
2003	4.1	165	0.55	SGJ	SRE
2003	5.0	176	0.55	SGJ	SRE
2004	6.0	350	0.55	SGJ	SRE
2004	6.1	379	0.55	SGJ	SRE
2005	7.0	503	0.55	SGJ	SRE
2007	8.0	574	0.7	PPG	EPN, DLK, SRE
2007	8.1	607	0.81	PPG	EPN, DLK, SRE
2008	9.0	603	0.81	PPG	EPN, DLK, SRE
2008	9.1	1372	0.81	PPG	EPN, DLK, SRE

Table 1.3: **Growth of the Rfam database.** All three letter abbreviations are initials: SGJ: Sam Griffiths-Jones; SRE: Sean R. Eddy; PPG: Paul P. Gardner; EPN: Eric P. Nawrocki; DLK: Diana L. Kolbe. "RFAM head" is the RFAM project leader at the time of release. "INFERNAL team" is the software development team that contributed to the INFERNAL version used for each RFAM release.

The use of BLAST filters in the RFAM pipeline is undesirable but necessary. It is undesirable because BLAST scores only sequence similarity. Some remote homologs with relatively divergent sequence but conserved structure that would receive high CM scores may not pass the filter score threshold. This is especially unfortunate because the discovery of remote homologs disproportionately expands knowledge of the family. The BLAST filter is necessary though because CM searches of large databases are prohibitively slow. Even for the RFAM curators, who have a access to a large compute cluster, it is only practical to use CMs to search the small fraction of RFAMSEQ that survives the filters. The slow speed and other limitations of CM methods are discussed in detail in the next section.

Limitations of CMs

The most serious limitation of CMs, and all types of SCFGs, is their high computational complexity. The CM CYK and Inside search algorithms scale $O(LN^2 \log N)$ for a target database of length L and query model length (number of consensus columns) of N. Here are some example running times of using the CYK algorithm implemented in INFERNAL 0.55 to search for RNAs of various sizes (on a single Intel Xeon 3.0 GHz processor):

family	length	search time (hours/Mb)
mir-10	74	0.68
U5	120	2.37
5.8S rRNA	154	4.26
Lysine riboswitch	175	7.13
SRP RNA	304	14.22
RNase P RNA	370	44.68

To search a typically sized bacterial genome (about 10 Mb, both strands of a 5 Mb genome) with a Lysine riboswitch model would take roughly 3 days. The same search with a RNase P model would take roughly two and a half weeks. Using the same two models to search the roughly 3 Gb chimpanzee genome would take roughly two years and fifteen years, respectively. These times are clearly impractical.

Another limitation of CMs is that their simple parameterization scheme requires deep multiple alignments ($>$ 20 sequences) to build sensitive and specific models. INFERNAL version 0.55 uses an uninformative plus-one prior for parameterization of emission and transition probabilities. Uninformative priors disproportionately affect models built from small numbers of sequences because the fewer sequences present in the seed alignment, the higher the weight of the prior (the α values in equation 1.4). More than half of the 503 families in RFAM 7.0 contain less than 5 sequences in the training alignment (Figure 1.14). As discussed earlier, the replacement of plus-one priors with informative mixture Dirichlet priors for profile HMM parameterization lead to a significant increase in their sensitivity

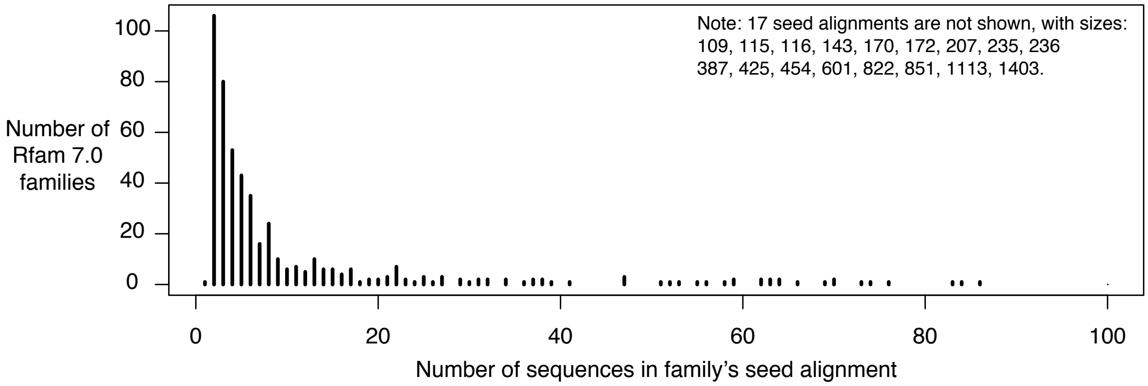


Figure 1.14: **Histogram of number of sequences in Rfam 7.0 seed alignments.** The histogram shows data for 486 of the 503 RFAM 7.0 seed alignments. The other 17 have more than 100 sequences in the seed, as listed in top right of plot.

for protein homology search [234]. The small size of the RFAM seed alignments suggest that using mixture Dirichlet priors could have a profound impact for RFAM.

An inherent limitation of CMs (and all SCFGs) is that they can only model *well-nested* base-pairing interactions. This is because of how sequences are generated from the *outside-in* by SCFGs (Figure 1.11). In an SCFG parse tree structure, like the one in Figure 1.11, there will never be overlapping lines connecting states to residues, nor will any residue be connected to more than one state. However, many RNA structures include sets of base-pairs that correspond to these scenarios. Formally, in a well-nested set of base-pairs there are no two base-pairs between positions $i : j$ and $k : l$ such that $i < k < j < l$. A *pseudoknot* is introduced when the addition of one or more new base-pairs to a well-nested set violates this criteria. *Base triples*, which occur when one position is base-paired to more than one other position, also violate the well-nested criteria and so cannot be modeled by SCFGs. Many structural RNAs have pseudoknots and base triples, such as RNase P RNA, small and large subunit ribosomal RNA and telomerase RNA. Fortunately, in many RNAs, including these four, the large majority of base-pairs can be defined in a well-nested set. The secondary structure diagrams of RNase P RNA in Figure 1.3 depict some pseudoknots, including the interaction labeled "P4". Notably, some RNA sequence- and structure-based homology search methods can model pseudoknots, such as RNATOPS [121] and ERPIN [90].

CMs only model a single consensus structure. This is a limitation because homologous RNAs vary in structure as well as sequence. While some structural features are crucial to an RNA's function and are highly conserved, others can tolerate small or even large changes. CMs can only handle this in a crude fashion via local alignment as explained above, by allowing large insertions of unstructured residues and deletions of consensus structural elements with a small score penalty. Additionally, some RNAs, such as riboswitch elements, are known to adopt at least two stable structures, both of which have functional importance. A CM must choose one of these two structures to model. (For these cases, the structure that contributes the most additional information relative to a sequence-only profile should be chosen.)

Some of these limitations are more easily addressed than others. Adding the ability to model pseudoknots and base triples or relaxing the requirement that a CM model a single consensus structure would likely require either significant changes to the CM grammar formulation and construction procedure, or augmentation of CM methods with other types of models. Alternatively, the speed and parameterization issues can, and have been, approached within the existing CM framework.

1.7 Addressing the limitations of covariance models

RSEARCH: CMs from single sequence queries

The RSEARCH program uses a CM built from a single RNA sequence and structure query to search for homologs [138]. Instead of using position-specific scores derived from a seed alignment, RSEARCH uses position-independent scores from the RIBOSUM RNA substitution matrices in the same way BLAST uses the BLOSUM matrices. This approach partially addresses the limitation of CM methods to determine appropriate parameters when very few homologs of the query family are known. The RIBOSUM matrices include a 16×16 substitution matrix defining possible base-pair substitutions used to score base-pairs and a 4×4 matrix for scoring single-stranded positions. The scores were derived similarly to the

BLOSUM scores, but using structurally annotated RNA alignments (large and small subunit ribosomal RNA alignments) instead of protein alignments. The single sequence CM transition scores in RSEARCH, which include gap penalties, are set as arbitrary penalties that were chosen based on good empirical performance [138]. While RSEARCH uses a different scoring system than standard CMs, it still uses the same CYK scoring algorithm. However, partly because RSEARCH does not use probabilistic transition scores, the Inside algorithm cannot be used to sum over all possible alignment scores for a given target sequence.

RSEARCH is able to detect similarities between RNA sequences that primary sequence based methods like BLAST cannot. One such example out of several detailed in [138] is the detection, with a statistically significant E-value less than 10^{-5}, of SRP RNA in the genome of the plant *Arabadopsis thaliana* using a mammalian (*Homo sapiens*) SRP RNA as a query. For this search, neither BLAST nor SSEARCH (a Smith-Waterman implementation) [205] find any significant hits with an E-value less than 10.

tRNAscan-SE: using CMs to annotate tRNAs in genomes

tRNAs are particularly well suited for modeling with CMs because they are short (about 70 nt) and evolve rapidly at the sequence level while strongly conserving a canonical cloverleaf structure. A sequence and structure tRNA profile derives about as much information from structure as it does from sequence (about 25 bits from each, Figure 1.9), making it an outlier relative to other RNA families. A related reason that CMs should work well for tRNAs is that thousands of examples are known, so deep seed alignments can be used to construct CMs that have well-defined parameters resulting from mean posterior estimation using a large number of observations (equation 1.4). Further, because there is very little structural variation amongst tRNAs (with the exception of mitochondrial tRNAs), modeling them using a single consensus structure is almost always appropriate.

Several tRNA specific homology search programs have been developed, but the most widely used program, tRNAscan-SE, uses CMs [162]. To circumvent the high computational complexity of CM search algorithms, the program uses optimized versions of two fast tRNA-specific alignment methods to prefilter a target database ([74, 204]. Any candidate tRNAs that survive either filter is reevaluated using two different CMs. The first, a canonical tRNA model, was built from a structural alignment of 1415 tRNAs, including 38 that contain introns (so the CM is aware that some tRNAs will contain introns). The second CM models selenocysteine tRNAs, which have a slightly different structure than canonical tRNAs. Combining CMs with the fast prefilters in this way improved the sensitivity of tRNA detection from 95.1% to 99.5% and reduced the false positive rate from 0.37 per Mb to less than 0.00007 per Mb relative to using one of the filtering methods alone [74]. The tRNAscan-SE program is commonly included in genome annotation pipelines to annotate tRNAs in genomes.

Accelerating CMs in the general case

tRNAscan-SE demonstrates the power of CM methods and removes the speed barrier using heuristic prefilters. The filters are tRNA-specific though and are useless for the hundreds of other RFAM families. RFAM uses BLAST as a filter for the general case. Several other methods have been developed for general CM acceleration, which can be divided into two classes: those that use sequence-based scoring schemes, and those that use sequence- and structure-based filters. I will list some of these techniques below.

As mentioned above, profile HMMs outperform BLAST in protein homology search, so they may be more effective than BLAST at filtering for CMs. As a graduate student with Larry Ruzzo at the University of Washington, Zasha Weinberg introduced several types of profile HMM based filters for CM searches [263–266]. Weinberg/Ruzzo *rigorous filter* profile HMMs are parameterized so as to guarantee that any sequence that will score above a score threshold to a CM will survive the filter [264]. The acceleration achieved by rigorous filters varies from family to family, and they are not practical (do not save time) for a small fraction of CMs. Weinberg/Ruzzo *maximum-likelihood ML heuristic* HMM filters provide a more consistent acceleration across families while sacrificing the guarantee that all high-scoring CM hits will survive [265]. Their strategy sets the filter score threshold to allow a predicted

0.01 fraction of the database to survive. A ML-heuristic HMM is carefully constructed to be as similar as possible to a CM. Zhang et al. [288] introduced a sequence-based filtering method for sequence- and structure models closely resembling INFERNAL CMs that requires a target subsequence contain one or more of a pre-generated list of model-specific high-scoring keywords to survive the filter. This approach is similar to how BLAST and FASTA require high-scoring keyword matches prior to using banded DP to determine final scores.

Sequence- and structure-based methods have also been developed for CM filtering. Weinberg and Ruzzo have supplemented their HMM filtering strategies with one that uses *sub-CMs*, small models constructed to model only a substructure (parse subtree) of a full CM [263]. Zhang et al. [287] developed a filtering method for single sequence CM-like searches that quickly identifies $(k, w) - stacks$, disjoint subsequences that could form high-scoring base-paired helices. Only subsequences containing one or more of these regions survive the filter. Finally, Sun and Buhler have described a method based on secondary structure profiles (SSPs) that score both sequence and structural similarity to substructures modeled by a CM [247]. A set of SSPs, each of which typically models a substructure with low probabilities of insertions and deletions, are collectively used to filter.

Other uses of CMs and profile SCFGs

INFERNAL is not the only implementation of CMs. The RAVENNA freely available package of Weinberg and Ruzzo includes INFERNAL, but also implements the filtering methods described above [266]. CMFINDER, a freely available package developed by Yao and Ruzzo, is a structural motif finder that looks for common RNA structural elements in multiple sequences, such as genomes. CMFINDER includes source code from Sean Eddy's COVE software package [61, 65], the predecessor of INFERNAL, as well as some important parameters from INFERNAL [189]. Zhang et al.'s filters and CM-like models are implemented in the FASTR program [287, 288]. The RNATOPS program implements a structure graph model that uses both profile HMMs and simplified CMs for homology search in a manner that allows pseudoknots to be modeled [121]. Finally, RNACAD includes an alternative implementation of profile SCFGs to CMs [24]. This program was developed by Michael Brown in the lab of David Haussler, in which Yasu Sakakibara and colleagues independently introduced profile SCFGs for RNA sequence analysis [224, 226] the same year as Eddy and Durbin [65]. Although RNACAD is the latest implementation from the Haussler group, it seems to be no longer actively developed as far as I can tell. RNACAD was used by the Ribosomal Database Project [39] for alignment of SSU rRNA until 2008, when it was replaced with INFERNAL [40] (this is discussed in detail in Part 2 of this work).

Alternatives to profile SCFGs

Of course, not all methods for scoring sequence and structure during RNA homology search are based on profile SCFGs. A separate, broad class implements what I will call *pattern-matching* approaches. Some of the more popular pattern-matching tools are PATSCAN [52], RNAMOTIF [167], ERPIN [90], PATSEARCH [101], RNABOB [64], LOCOMOTIF [217] and RNAPATTERN [136]. These programs specify a query pattern of an RNA family or *motif* (a common structural element present in many families) that defines constraints on sequence and structural characteristics of the family derived from known examples. Constraints include limits on possible stem lengths, positions and sizes of loops, and in many cases acceptable single-stranded and base-paired residues. Figure 1.15 depicts a pattern used to define an E-loop RNA structural motif by the RNAMOTIF program [167]. Homology search is performed by scanning target databases for matching subsequences that satisfy the pattern constraints.

A useful pattern must be sensitive enough to match to all known homologs as well as yet undiscovered ones, and also specific enough to not match a large number of spurious, non-homologous sequences. Patterns are often manually created, or automatically constructed and manually refined, to delicately balance this sensitivity/specificity trade-off.

A general drawback the pattern-matching approach as described above is the binary output of the results: a target sequence either matches the pattern or it does not. In this way all positive matches are considered equal, which makes the statistical significance

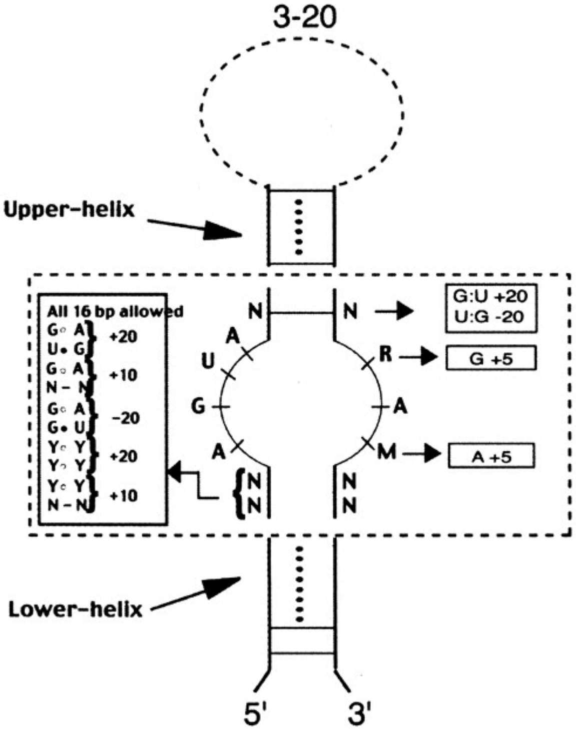


Figure 1.15: **A pattern defining the E-loop RNA structural motif used by the RNAMOTIF program.** Constraints on the possible sizes and residues of various parts of the motif are annotated. Matching sequences must satisfy all constraints. For example, a matching sequence must include the sequence `AGUA` matched to the left (5') side of the internal loop within the dotted box, and include a stretch of between 3 and 20 residues that matches to the top loop region (marked "3-20"). This figure is reprinted with permission from [167].

of a match difficult to estimate. Several recent pattern-matching tools have addressed this drawback by incorporating scores, which are sometimes position-specific, for single stranded residues as well as base-pairs. A notable example is the ERPIN program [90] which introduced the secondary structure profiles that were extended and applied as filters for CM searches by Sun and Buhler [247].

An important advantage of many pattern-matching tools is their fast speed relative to profile SCFGs. In general, the number of calculations required to determine if a sequence matches the constraints of a pattern is far less than those required to determine the optimal alignment of a sequence to a CM using CYK. An additional advantage is that some pattern-matching based programs, including RNAMOTIF and ERPIN can model pseudoknotted base-pairing interactions which SCFGs cannot.

A recent, independent, benchmark tested the performance of various homology search tools on RNA homology search. Primary sequence-based methods such as BLAST and HMMER were tested as well as sequence- and structure-based methods including CMs and a single pattern-matching tool - ERPIN. Three of the top performing methods in terms of sensitivity and specificity were CM-based (INFERNAL[6], RSEARCH and RaveNnA), each of which outperformed ERPIN, but these methods were also the slowest.

1.8 Outline of Part 1 of this work

The remainder of Part 1 of this dissertation describes my work on alleviating two serious limitations of CM homology search methods: their slow speed and simplistic parameterization. Chapter 2 details the application of informative Dirichlet priors and entropy weighting for improved CM parameterization, as well as a banded DP technique for accelerating CM searches.[7] Chapter 3 describes INFERNAL version 1.0, which implements the improved CM methods I have worked on.[8] In chapter 4, I discuss the combination of the banded DP

---

[6] Version 0.7 of INFERNAL was used in this benchmark, which includes the Dirichlet mixture priors and entropy weighting strategies discussed in Chapter 2.

[7] Chapter 2 is co-authored by myself and Sean Eddy and independently published as [189].

[8] Chapter 3 is independently published as [190] and is co-authored by myself, Diana Kolbe and Sean Eddy.

technique from 2 and profile HMMs as a two-stage filter pipeline for accelerating INFERNAL searches. Chapter 5 details how and why INFERNAL should be used for RNA homology search, with an emphasis on environmental metagenomics datasets.[9] Finally, Chapter 6 concludes part 1 of this work, with a brief discussion of its impact on CM homology search performance. Part 2 of this work, which includes its own introduction, describes improving CM methods for large-scale SSU rRNA alignment.

---

[9] Chapter 5 was co-authored with Sean Eddy and submitted as a chapter for a book on metagenomics.

Chapter 2

Query-Dependent Banding (QDB) for Faster RNA Similarity Searches[1]

2.1 Abstract

When searching sequence databases for RNAs, it is desirable to score both primary sequence and RNA secondary structure similarity. Covariance models (CMs) are probabilistic models well suited for RNA similarity search applications. However, the computational complexity of CM dynamic programming alignment algorithms has limited their practical application. Here we describe an acceleration method called *query-dependent banding* (QDB), which uses the probabilistic query CM to precalculate regions of the dynamic programming lattice that have negligible probability, independently of the target database. We have implemented QDB in the freely available INFERNAL software package. QDB reduces the average-case time complexity of CM alignment from $LN^{2.4}$ to $LN^{1.3}$ for a query RNA of N residues and a target database of L residues, resulting in a four-fold speedup for typical RNA queries.

---

[1] This chapter was published independently as Nawrocki EP, Eddy SR. Query-dependent Banding (QDB) for faster RNA similarity searches. PLoS Comput Biol. 3(3):e56. It is reprinted here without modification except where necessary to adhere to dissertation formatting guidelines.

Combined with other improvements to INFERNAL, including informative mixture Dirichlet priors on model parameters, benchmarks also show increased sensitivity and specificity resulting from improved parameterization.

2.2 Introduction

Many functional RNAs conserve a base-paired secondary structure. Conserved RNA secondary structure induces long-distance pairwise correlations in homologous RNA sequences. When performing database searches to identify homologous structural RNAs, it is desirable for RNA similarity search programs to score a combination of secondary structure and primary sequence conservation.

A variety of approaches for RNA similarity searching have been described. There are specialized programs for identifying one particular RNA family or motif, such as programs that identify tRNAs [147, 162], snoRNAs [163, 231], microRNAs [144, 157], SRP RNAs [218], and rho-independent transcription terminators [70]. There are also pattern-matching algorithms that rely on expertly designed query patterns [167]. However, the most generally useful approaches are those that take any RNA (or any multiple RNA alignment) as a query, and use an appropriate scoring system to search a sequence database and rank high-scoring similarities [90, 287], just as programs like BLAST do for linear sequence comparison [3].

In a general search program, one wants to score a combination of RNA sequence and structural conservation in a principled rather than *ad hoc* manner. A satisfactory solution to this problem is known, using probabilistic models called *stochastic context-free grammars* (SCFGs). SCFGs readily capture both primary sequence and (non-pseudoknotted) RNA secondary structure conservation [53, 225]. Just as hidden Markov models (HMMs) are useful for many different linear sequence modeling applications, including genefinding, multiple alignment, motif finding, and similarity search [53], SCFGs are a generally useful paradigm for probabilistic RNA sequence/structure analysis, with applications including secondary structure prediction and genefinding. A particular SCFG architecture called *covariance models* (CMs) was developed specifically for the RNA similarity search problem

[65]. CMs are profile SCFGs, analogous to the use of profile HMMs in sequence analysis [57, 65]. The Rfam database of RNA families [100] is based on CM software (INFERNAL) in much the same way that the Pfam database of protein families is based on profile HMM software (HMMER) [75, 237].

The most serious problem with using CMs has been their computational complexity. Applying standard SCFG dynamic programming alignment algorithms to the particular case of CMs results in algorithms that require $O(N^3)$ memory and $O(LN^3)$ time for a query of length N residues (or consensus alignment columns) and a target database sequence of length L. The memory complexity problem has essentially been solved, by extending divide-and-conquer dynamic programming methods (the Hirshberg or Myers/Miller algorithm) to the case of CMs [57], but the time complexity problem still stands.

Weinberg and Ruzzo have described several filtering methods for accelerating CM searches [263–265]. The original idea ("rigorous filters") was to score a target sequence first by a linear sequence comparison method, using a profile HMM specially constructed from the query CM such that the profile score was provably an upper bound on the CM score; the subset of hits above threshold would then be passed for rescoring with the more expensive CM alignment algorithm [264]. Subsequently a "maximum likelihood heuristic" filter profile was developed that gives up the guarantee of recovering the same hits as the unfiltered search, but offers greater speedups [265]. For most current Rfam models, Weinberg/Ruzzo filters give about a hundred-fold speed up relative to a full CM based search at little or no cost to sensitivity and specificity. However, because these filters depend on primary sequence conservation alone, they can be relatively ineffective for RNA families that exhibit poor sequence conservation – unfortunately, precisely the RNAs that benefit the most from SCFG-based search methods. Indeed, in this respect, we are concerned that the overall performance of rigorous filters on the current Rfam database may be somewhat misleading. Rfam currently uses a crude BLAST-based filtering method to accelerate the CM searches used in curating the database. This step introduces a bias towards high primary sequence similarity in current Rfam alignments. As Rfam improves and incorporates more diverse structural homologs, the effectiveness of sequence-based filters will decrease. To address this worry, Weinberg and Ruzzo have also described additional heuristics ("sub-CMs" and the "store-pair" technique) that should capture more secondary structure information in the filtering process [263]. Bafna and coworkers have described further improvements to sequence filtering methods [288]. Currently, the INFERNAL codebase includes Weinberg's C++ implementation of rigorous filters, but not, as yet, the ML heuristic, sub-CM, or store-pair methods. All these methods are important, but it also remains important to us to identify yet more methods for accelerating CMs.

Here, we describe a method for accelerating CM searches using a banded dynamic programming (DP) strategy. In banded DP, one uses some fast method to identify a band through the DP matrix where the optimal alignment is likely to lie, and then calculates computationally expensive DP recursions only within that band. In most cases, including our approach, banded DP is a heuristic that sacrifices guaranteed alignment optimality. Banding is a standard approach in many areas of sequence analysis. Gapped BLAST uses banded DP to convert ungapped high-scoring pairs (HSPs) to full gapped alignments [3]. LAGAN and MULTI-LAGAN use banded dynamic programming (referred to as limited-area dynamic programming) to stitch together alignments between anchored sequences when aligning long genomic sequences [25]. Banding has also been applied to profile SCFGs by Michael Brown in his RNACAD program by using information from a profile HMM alignment to define bands for the expensive SCFG alignment [24]. The key to developing a banded DP strategy is in deciding how the bands are identified. Usually, including all the examples just mentioned, banded DP involves performing some sort of rapid approximate sequence alignment between the query and the target.

In contrast, the method we describe here, called *query-dependent banding* (QDB), takes advantage of specific properties of CMs in order to predefine bands that are independent of any target sequence. QDB depends on the consensus secondary structure of the query, so it is complementary to acceleration methods like the Weinberg/Ruzzo filters that rely on sequence but not structure.

2.3 Results

Briefly, the key idea is the following. Each base pair and each single-stranded residue in the query RNA is represented in a CM by a *state*. States are arranged in a tree-like structure that mirrors the secondary structure of the RNA, along with additional states to model insertions and deletions. The standard CM dynamic programming alignment algorithm works by calculating the probability that a substructure of the query rooted at state v aligns to a subsequence $i..j$ in the the target sequence. The calculation is recursive, starting at the leaves of the CM (ends of hairpin loops) and subsequences of length 0, and working upwards in larger substructures of the CM, and outwards in longer and longer subsequences.

To guarantee optimality, at each v, the DP algorithm must score all possible $i..j$ subsequences in the target sequence. However, most of these subsequences are obviously too long or short, when one considers the size of the query substructure under state v. For example, when state v models the closing base pair of a consensus four-base loop, only $i..j$ subsequences of length six are likely to occur in any optimal alignment to state v; that is, $(j-5, j)$ being the base pair, and $(j-4..j-1)$ being the four bases of the hairpin loop. Likewise, the optimal subsequence aligned to the next consensus base pair in that stem is almost certainly of length eight.

Because insertions and deletions may occur in the target sequence, no subsequence length is known with certainty, but because the CM is a probabilistic model, a probability distribution for subsequence lengths under each state (including the probability of insertions and deletions) can be analytically derived from the query CM. These distributions can be used to determine a band of subsequence lengths that captures all but a negligible amount of the probability mass. A CM dynamic programming algorithm can then look not at all subsequences i, j for each state v, but only those i within a band of minimum and maximum distance relative to each j.

To formalize this idea, we start with a description of CMs, followed by the QDB algorithms for calculating the subsequence length distributions, using these length distributions to determine bands, and using the bands in a banded CM dynamic programming alignment algorithm. Calculation of the bands is sensitive to transition parameter estimation, so we describe INFERNAL's new implementation of informative Dirichlet priors for CM parameter estimation. Finally, we present results from a benchmark that suggest the sensitivity and specificity of a QDB-accelerated search is negligibly different from a non-banded search.

Covariance models

CMs are a convention for mapping an RNA secondary structure into a tree-like, directed graph of SCFG states and state transitions (or equivalently, SCFG nonterminals and production rules). The CM is organized by a binary tree of *nodes* representing base pairs and single-stranded residues in the query's structure. Each node contains a number of *states*, where one state represents the consensus alignment to the query, and the others represent insertions and deletions relative to the query. Figure 2.1 shows an example of converting a consensus structure to the guide tree of nodes, and part of the expansion of those guide tree nodes into the CM's state graph. Here we will only concentrate on the aspects of CMs necessary to understand QDB, and a subset of our usual notation. For full details on CM construction, see [57, 192].

A guide tree consists of eight types of nodes. MATP nodes represent consensus base pairs. MATL and MATR nodes represent consensus single-stranded residues (emitted to the left or right with respect to a stem). BIF nodes represent bifurcations in the secondary structure of the family, to deal with multiple stem-loops. A ROOT node represents the start of the model. BEGL and BEGR nodes represent the beginnings of a branch on the left and right side of a bifurcation. END nodes end each branch.

The CM is composed of seven different types of states, each with a corresponding form of production rule, with notation defined as follows:

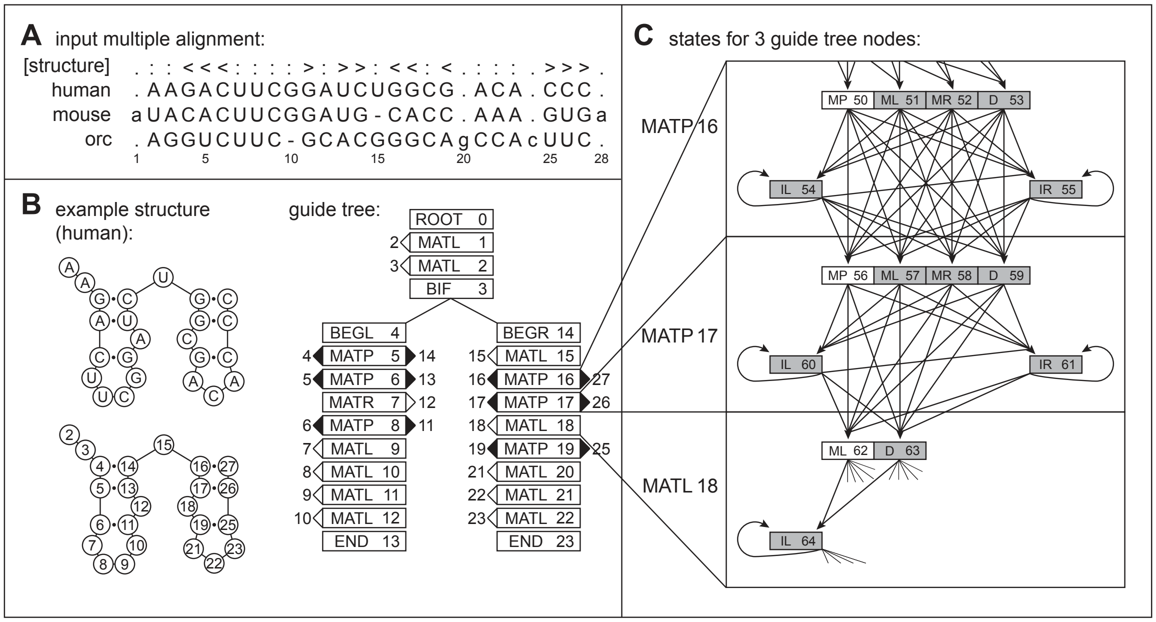

A input multiple alignment:
[structure] . : : < < < : : : : > : > > : < < : < . : : : : . > > > .
human . A A G A C U U C G G A U C U G G C G . A C A . C C C .
mouse a U A C A C U U C G G A U G - C A C C . A A A . G U G a
orc . A G G U C U U C - G C A C G G G C A g C C A c U U C .
1 5 10 15 20 25 28
B example structure (human):
guide tree:
ROOT 0
MATL 1
MATL 2
BIF 3
BEGL 4
MATP 5
MATP 6
MATR 7
MATP 8
MATL 9
MATL 10
MATL 11
MATL 12
END 13
BEGR 14
MATL 15
MATP 16
MATP 17
MATL 18
MATP 19
MATL 20
MATL 21
MATL 22
END 23
C states for 3 guide tree nodes:
MATP 16
MP 50
ML 51
MR 52
D 53
IL 54
IR 55
MATP 17
MP 56
ML 57
MR 58
D 59
IL 60
IR 61
MATL 18
ML 62
D 63
IL 64

Figure 2.1: **An example RNA family and corresponding CM. (A):** A toy multiple alignment of three RNA sequences, with 28 total columns, 24 of which will be modeled as consensus positions. The [structure] line annotates the consensus secondary structure: < and > symbols mark base pairs, :'s mark consensus single stranded positions, and .'s mark "insert" columns that will not be considered part of the consensus model because more than half the sequences in these columns contain gaps. **(B):** The structure of one sequence from (A), the same structure with positions numbered according to alignment columns, and the guide tree of nodes corresponding to that structure, with alignment column indices assigned to nodes (for example, node 5, a MATP "match-pair" node, will model the consensus base pair between columns 4 and 14). **(C):** The state topology of three selected nodes of the CM, for two MATP nodes and one consensus "leftwise" single residue bulge node (MATL, "match-left"). The consensus pair and singlet states (two MPs and one ML) are white, and the insertion/deletion states are gray. State transitions are indicated by arrows.

State type	Description	Production	Δ_v^L	Δ_v^R	Emission	Transition
P	(pair emitting)	$P \rightarrow aYb$	1	1	$e_v(a,b)$	$t_v(Y)$
L	(left emitting)	$L \rightarrow aY$	1	0	$e_v(a)$	$t_v(Y)$
R	(right emitting)	$R \rightarrow Ya$	0	1	$e_v(a)$	$t_v(Y)$
B	(bifurcation)	$B \rightarrow SS$	0	0	1	1
D	(delete)	$D \rightarrow Y$	0	0	1	$t_v(Y)$
S	(start)	$S \rightarrow Y$	0	0	1	$t_v(Y)$
E	(end)	$E \rightarrow \epsilon$	0	0	1	1

That is, for instance, if state v is a pair state, it produces (aligns to and scores) two correlated residues a and b and moves to some new state Y. The probability that it produces a residue pair a, b is given by an *emission probability* $e_v(a,b)$. The probability that it moves to a particular state Y is given by a *transition probability* $t_v(Y)$. The set of possible states Y that v may transit to is limited to the states in the next (lower) node in the guide tree (and insert states in the current node); the set of possible children states Y is called C_v, for "children of v". The indicators Δ_v^L and Δ_v^R are used to simplify notation in CM dynamic programming algorithms. They are the residues emitted to the left and right of state v, respectively. Bifurcation rules are special, in that they always transition to two particular start (S) states, at the root of subtrees in the guide tree, with probability 1.0.

These state types essentially define a "normal form" for SCFG models of RNA, akin to SCFGs in Chomsky normal form where all productions are in one of two forms, $Y \rightarrow a$ or $Y \rightarrow YY$. We describe CM algorithms (including QDB) in terms of this normal form. CMs define a specific way that nodes in the guide tree are expanded into states, and how those states are connected within each node and to states in the next node in the guide tree. For example, a MATP node that deals with a consensus base pair contains six states called MATP_MP (a P state for matching the base pair), MATP_ML and MATP_MR (a L and R state for matching only the leftmost or rightmost base and deleting the right or left one, respectively), MATP_D (a D state for deleting the base pair), and MATP_IL and MATP_IR (L and R states with self-transitions, for inserting one or more residues to the left and/or right before going to the next node).

Thus a CM is a generative probabilistic model of homologous RNAs. A sequence is emitted starting at the root, moving downwards from state to state according to state transition probabilities, emitting residues and residue pairs according to emission probabilities, and bifurcating into substructures at bifurcation states. An important property of a CM is the states can be numbered from $0..M-1$ (from root to leaves) such that for any state v, the states y that it can transit to must have indices $y \geq v$. There are no cycles in a CM, other than self-transitions on insert states. This is the property that enables the recursive calculations that both CM DP alignment algorithms and QDB rely on.

Without any change in the above description, CMs apply to either global or local alignment, and to either pairwise alignment to single RNA queries or profile alignment to a consensus query structure of a multiple RNA sequence alignment. CMs for single RNA queries are derived identically to profiles of a consensus structure, differing only in the parameterization method [138]. Local structural alignment to substructures and truncated structures (as opposed to requiring a global alignment to the whole RNA structural model) is achieved by adding state transitions from the ROOT that permit entering the model at any internal consensus state with some probability, and state transitions from any internal consensus state to an END with some probability [138, 192].

QDB algorithm

Observe that for any state v, we could enumerate all possible paths down the model from v to the END(s). Each path has a certain probability (the product of the transition probabilities used by the path), and it will emit a certain number n of residues (2 per P state, 1 per L or R state in the path). The sum of these path probabilities for each n defines a probability distribution $\gamma_v(n)$, the probability that the CM subgraph rooted at v will generate a subsequence of length n. Given a finite limit Z on maximum subsequence length (defined later), we can calculate $\gamma_v(n)$ by an efficient recursive algorithm, working from the leaves of the CM towards the root, and from smallest subsequences to largest:

for $v = M - 1$ down to 0:

$$\begin{array}{l|l|l}
v = \text{end state } (E): & \gamma_v(0) = 1 & \\
 & \gamma_v(d) = 0 & \text{for } d = 1 \text{ to } Z \\
 & & \\
v = \text{bifurcation } (B): & \gamma_v(d) = \sum_{n=0}^{d} \gamma_y(n) * \gamma_z(d-n) & \text{for } d = 0 \text{ to } Z \\
 & & \\
\text{else } (v = S, P, L, R): & \gamma_v(d) = 0 & \text{for } d = 0 \text{ to } (\Delta_v^L + \Delta_v^R - 1) \\
 & \gamma_v(d) = \sum_{y \in C_v} \gamma_y(d - (\Delta_v^L + \Delta_v^R)) * t_v(y) & \text{for } d = (\Delta_v^L + \Delta_v^R) \text{ to } Z
\end{array}$$

For example, if we are calculating $\gamma_v(d)$ where v is is a pair state, we know that v must emit a pair of residues then transit to a new state y (one of its possible transitions C_v), and then a subgraph rooted at y will have to account for the rest of the subsequence of length $d - 2$. Therefore, $\gamma_v(d)$ must be the sum, over all possible states y in C_v, of the transition probability $t_v(y)$ times the probability that the subtree rooted at y generates a subsequence of length $d - 2$ – which is $\gamma_y(d - 2)$, guaranteed to have already been calculated by the recursion. For the B state (bifurcation) calculation, indices y and z indicate the left and right S (start) state that bifurcation state v must connect to.

A band dmin(v)..dmax(v) of subsequence lengths that will be allowed for each state v is then defined as follows. A parameter β defines the threshold for the negligible probability mass that we are willing to allow outside the band. (The default value of β is set to 10^{-7}, as described later.) We define dmin(v) and dmax(v) such that the cumulative left and right tails of $\gamma_v(n)$ contain less than a probability $\frac{\beta}{2}$:

$$\sum_{d=0}^{\mathrm{dmin}(v)-1} \gamma_v(d) < \frac{\beta}{2},$$

$$\sum_{d=\mathrm{dmax}(v)+1}^{Z} \gamma_v(d) < \frac{\beta}{2}.$$

Larger values of β produce tighter bands and faster alignments, but at a cost of increased risk of missing the optimal alignment. β is the only free parameter that must be specified to QDB.

Because CMs have emitting self-loops (i.e. insert states), there is no finite limit on subsequence lengths. However, we must impose a finite limit Z to obtain a finite calculation. Z can be chosen to be sufficiently large that it does not affect $\mathrm{dmax}(v)$ for any state v. On a digital computer with floating point precision ϵ (the largest value for which $1+\epsilon = 1$), it suffices to guarantee that, for all v:

$$\frac{\sum_{d=Z+1}^{\infty} \gamma_v(d)}{\sum_{d'=\mathrm{dmax}(v)+1}^{Z} \gamma_v(d')} \leq \epsilon$$

Empirically, we observe that the tails of the $\gamma_v(d)$ densities decrease approximately geometrically. We can estimate the mass remaining in the unseen tail by fitting a geometric tail to the observed density $\gamma_v(d)$. Our implementation starts with a reasonable guess at Z and verifies that the above condition is true for each v, assuming these geometrically decreasing tails; if it is not, Z is increased and bands are recalculated until it is.

A QDB calculation only needs to be performed once per query CM to set the bands. Overall, a QDB calculation requires $\Theta(MZ)$ in time and space; or equivalently, because both M and Z scale roughly linearly with the length L in residues of the query RNA, $\Theta(L^2)$. The time and space requirement is negligible compared to the requirements of a typical CM search.

Banded CYK database search algorithm for CMs

A standard algorithm for obtaining the maximum likelihood alignment (parse tree) of an SCFG to a target sequence is the Cocke-Younger-Kasami (CYK) dynamic programming algorithm [120, 135, 285]. Formally, CYK applies to SCFGs reduced to Chomsky normal form, and it aligns to the complete sequence. The CM database search algorithm is a CYK variant, specialized for the “normal form” of our seven types of RNA production rules, and for scanning long genomic sequences for high-scoring subsequences (hits) [53].

The CM search algorithm recursively calculates $\alpha_v(j,d)$, the log probability of the most likely CM parse subtree rooted at state v that generates (aligns to) the length d subsequence $x_{j-d+1}..x_j$ that ends at position j of target sequence x [53, 65]. This calculation initializes at the smallest subgraphs (E states) and shortest subsequences ($d = 0$) and iterates upwards and outwards to progressively larger subtrees and longer subsequences up to a preset window size W. The outermost loop iterates over the end position j on the target sequence, enabling an efficient scan across a long target like a chromosome sequence. Banding is achieved simply by limiting all loops over possible subsequence lengths d to the bounds $\mathrm{dmin}(v)..\mathrm{dmax}(v)$ derived in the band calculation algorithm, rather than all possible lengths $0..W$. The banded version of the algorithm is as follows:

Initialization (impose bands): for $j = 0$ to L, $v = M - 1$ down to 0:
for $d = 0$ to $\min((\mathrm{dmin}(v) - 1), j)$ $\quad \alpha_v(j, d) = -\infty$;
for $d = (\mathrm{dmax}(v) + 1)$ down to j $\quad \alpha_v(j, d) = -\infty$.

Initialization at $d = 0$: for $j = 0$ to L, $v = M - 1$ down to 0:
$v =$ end state (E): $\quad \alpha_v(j, 0) = 0$;
$v =$ bifurcation (B): $\quad \alpha_v(j, 0) = \alpha_y(j, 0) + \alpha_z(j, 0)$;
$v =$ delete or start (D, S): $\quad \alpha_v(j, 0) = \max_{y \in C_v}[\alpha_y(j, 0) + \log t_v(y)]$;
else ($v = P, L, R$): $\quad \alpha_v(j, 0) = -\infty$.

Recursion: for $j = 1$ to $L, d = \max(1, \mathrm{dmin}(v))$ to $\min(\mathrm{dmax}(v), j), v = M - 1$ down to 0
$v =$ end state (E): $\quad \alpha_v(j, d) = -\infty$;
$v =$ bifurcation (B): $\quad \mathrm{kmin} = \max(\mathrm{dmin}(z), (d - \mathrm{dmax}(y)))$,
$\quad \mathrm{kmax} = \min(\mathrm{dmax}(z), (d - \mathrm{dmin}(y)))$,
$\quad \alpha_v(j, d) = \max_{\mathrm{kmin} \leq k \leq \mathrm{kmax}}[\alpha_y(j - k, d - k) + \alpha_z(j, k)]$;
$v =$ delete or start (D, S): $\quad \alpha_v(j, d) = \max_{y \in C_v}[\alpha_y(j, d) + \log t_v(y)]$;
else ($v = P, L, R$) : $\quad \alpha_v(j, d) = \max_{y \in C_v}[\alpha_y(j - \Delta_v^R, d - (\Delta_v^L + \Delta_v^R)) + \log t_v(y)]$
$\quad + \log e_v(x_i, x_j)$.

For example, if we are calculating $\alpha_v(j, d)$ and v is a pair state (P), v will generate the basepair x_{j-d+1}, x_j and transit to a new state y (one of its possible transitions C_v) which then will have to account for the smaller subsequence $x_{j-d+2}..x_{j-1}$. The log odds score for a particular choice of next state y is the sum of three terms: an emission term $\log e_v(x_{j-d+1}, x_j)$, a transition term $\log t_v(y)$, and an already calculated solution for the smaller optimal parse tree rooted at y, $\alpha_y(j-1, d-2)$. The value assigned to $\alpha_v(j, d)$ is the maximum over all possible choices of child states y that v can transit to.

The W parameter defines the maximum size of a potential hit to a model. Previous INFERNAL implementations required an *ad hoc* guess at a reasonable W. The band calculation algorithm delivers a probabilistically derived W for database search in dmax(0), the upper bound on the length of the entire sequence (the sequence generated from the root state of the CM).

QDB does not reduce the asymptotic computational complexity of the CM search algorithm. Both the banded algorithm and the original algorithm are $O(MW + BW^2)$ memory and $O(L(MW + BW^2))$ time, for a model of M states containing B bifurcation states, window size W of residues, and target database length L. M, B, and W all scale with the query RNA length N, so roughly speaking, worst-case asymptotic time complexity is $O(LN^3)$.

Informative Dirichlet priors

The subsequence length distributions calculated by QDB depend on the CM's transition probabilities. Transition probability parameter estimation is therefore crucial for obtaining predicted subsequence length bands that reflect real subsequence lengths in homologous RNA targets. Transition parameters in INFERNAL are mean posterior estimates, combining (*ad hoc* weighted) observed counts from an input RNA alignment with a Dirichlet prior [192]. Previous to this work, INFERNAL used an uninformative uniform Dirichlet transition prior, equivalent to the use of Laplace "plus-1" pseudo-counts. However, we found that transition parameters derived under a uniform prior inaccurately predict target subsequence lengths, as shown in an example in Figure 2.2. The problem is exacerbated when there are few sequences in the query alignment, when the choice of prior has more impact on mean posterior estimation. To alleviate this problem, we estimated informative single component Dirichlet prior densities for CM transition parameters, as follows.

The training data for transition priors consisted of the 381 seed alignments in the Rfam

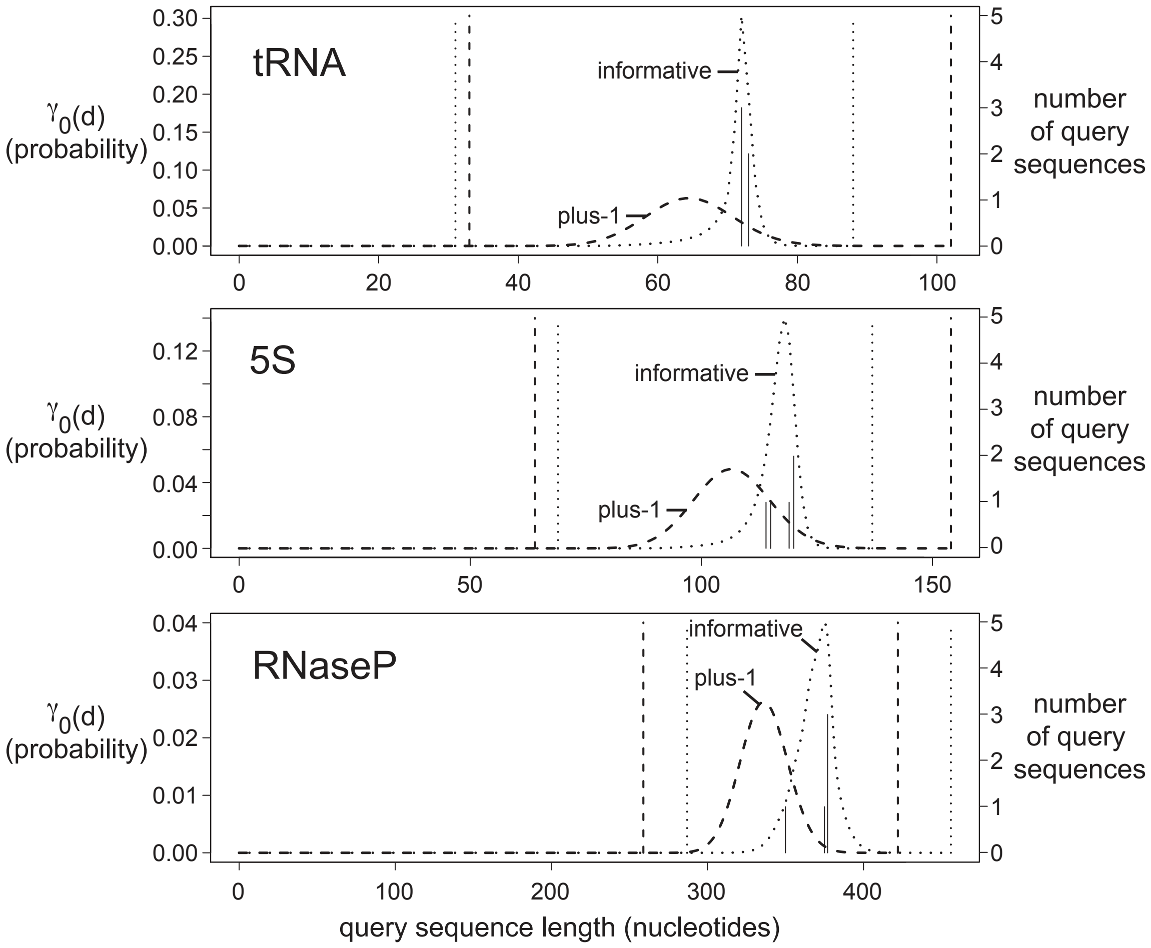


Figure 2.2: **Effect of transition priors on band calculation.** Predicted and actual target lengths are shown for three CMs built from alignments of five tRNA, 5S rRNA, and RNaseP sequences, which are about 75, 120, and 380 residues long, respectively. Solid vertical lines are histogram bars of the actual lengths of the query sequences in each alignment, corresponding with the right vertical axis labels. Dashed and dotted curves show QDB calculations for $\gamma_0(d)$ for the root state of each model, for uninformative versus informative Dirichlet priors, respectively. Dashed and dotted vertical lines show the band bounds (dmin(0) (left) and dmax(0) (right)) derived from the $\gamma_0(d)$ distributions using $\beta = 10^{-7}$. The uninformative plus-one prior results in consistent underprediction of target sequence lengths, with a broad distribution. The new informative priors produce tighter distributions that are centered on the actual subsequence lengths. We observe the same result for all other states (data not shown).

database, version 6.1 [100]. For each alignment, we built CM structures by INFERNAL's default procedure, and collected weighted counts of observed transitions in the implied parse trees of the training sequences. Considering all possible combinations of pairs of adjacent node types, there are 73 possible distinct types of transition probability distributions in CMs. To reduce this parameter space, we tied these 73 distributions into 36 groups by assuming that certain distributions were effectively equivalent. 36 Dirichlet densities were then estimated from these pooled counts by maximum likelihood as described in [234], with the exception that we optimize by conjugate gradient descent [213] rather than by expectation-maximization (EM). The results, including the Dirichlet parameters, are given in Tables 2.1 and 2.2. Using these priors for transition probability parameter estimation results in an improvement in the utility of QDB calculations, often yielding tighter, yet accurate subsequence length distributions, as illustrated by anecdotal example in Figure 2.2.

We also estimated informative mixture Dirichlet density priors for emission probabilities. Emission probabilities have no effect on QDB, but informative emission priors should improve sensitivity and specificity of CM searches, as they do for profile hidden Markov models [23, 234]. We collected filtered counts of aligned single-stranded residues and base-pairs from annotated ribosomal RNA alignments from four alignments in the 2002 version of the European Ribosomal RNA Database [283, 284]: large subunit rRNA (LSU), bacterial/archaeal/plastid small subunit rRNA (SSU-bap), eukaryotic SSU rRNA (SSU-euk), and mitochondrial SSU rRNA (SSU-mito). These alignments were filtered, removing sequences in which either less than 40% of the base paired positions are present or more than 5% of the nucleotides are ambiguous, and removing selected sequences based on single-linkage-clustering such that no two sequences in a filtered alignment were greater than 80% identical (in order to remove closely related sequences). Summary statistics for the filtered alignments and collected counts in the training dataset are given in Table 2.3. These data were used to estimate a nine-component Dirichlet mixture prior for base pairs, and an eight-component Dirichlet mixture prior for single stranded residues. The base pair prior is given in Table 2.4, and the singlet residue prior is given in Table 2.5.

dist #	tied to	dist counts	grp counts	node state	next node	Dirichlet α parameters												
						$\|\alpha\|$	t	$\frac{\alpha_t}{\|\alpha\|}$	t	$\frac{\alpha_t}{\|\alpha\|}$	t	$\frac{\alpha_t}{\|\alpha\|}$	t	$\frac{\alpha_t}{\|\alpha\|}$	t	$\frac{\alpha_t}{\|\alpha\|}$	t	$\frac{\alpha_t}{\|\alpha\|}$
1		6	11	MATP_MP	BIF	0.5509	IL	0.1229	IR	0.0001	B	0.8770						
2		7023	7119	MATP_MP	MATP	7.2986	IL	0.0023	IR	0.0024	MP	0.9816	ML	0.0056	MR	0.0046	D	0.0035
3		1600	1830	MATP_MP	MATL	1.5914	IL	0.0179	IR	0.0155	ML	0.9200	D	0.0466				
4		145	195	MATP_MP	MATR	1.9038	IL	0.0173	IR	0.0073	MR	0.8903	D	0.0852				
5	1	2	11	MATP_MP	END	0.5509	IL	0.1229	IR	0.0001	E	0.8770						
6*		1	2	MATP_ML	BIF	3.0000	IL	0.3333	IR	0.3333	B	0.3333						
7		577	577	MATP_ML	MATP	0.6941	IL	0.0131	IR	0.0103	MP	0.4032	ML	0.4983	MR	0.0115	D	0.0636
8		133	133	MATP_ML	MATL	0.9316	IL	0.0739	IR	0.0651	ML	0.7038	D	0.1571				
9		15	15	MATP_ML	MATR	0.3272	IL	0.1884	IR	0.0432	MR	0.4082	D	0.3602				
10*	6	1	2	MATP_ML	END	3.0000	IL	0.3333	IR	0.3333	E	0.3333						
11*		1	2	MATP_MR	BIF	3.0000	IL	0.3333	IR	0.3333	B	0.3333						
12		531	531	MATP_MR	MATP	0.7987	IL	0.0079	IR	0.0190	MP	0.3241	ML	0.0193	MR	0.5631	D	0.0666
13		151	151	MATP_MR	MATL	0.6933	IL	0.0357	IR	0.0699	ML	0.3066	D	0.5879				
14		15	15	MATP_MR	MATR	0.3574	IL	0.0582	IR	0.0002	MR	0.7629	D	0.1787				
15*	11	1	2	MATP_MR	END	3.0000	IL	0.3333	IR	0.3333	E	0.3333						
16*		0	2	MATP_D	BIF	3.0000	IL	0.3333	IR	0.3333	B	0.3333						
17		575	575	MATP_D	MATP	0.5450	IL	0.0019	IR	0.0047	MP	0.0857	ML	0.0534	MR	0.0528	D	0.8015
18		149	149	MATP_D	MATL	0.5831	IL	0.0421	IR	0.0526	ML	0.2080	D	0.6973				
19		14	14	MATP_D	MATR	0.1164	IL	0.0001	IR	0.0001	MR	0.2439	D	0.7559				
20*	16	2	2	MATP_D	END	3.0000	IL	0.3333	IR	0.3333	E	0.3333						
21		2	4	MATP_IL	BIF	1.4397	IL	0.6553	IR	0.0445	B	0.3002						
22		121	126	MATP_IL	MATP	0.9402	IL	0.1673	IR	0.1394	MP	0.5904	ML	0.0443	MR	0.0259	D	0.0327
23		114	119	MATP_IL	MATL	0.8046	IL	0.3108	IR	0.1936	ML	0.4610	D	0.0346				
24		14	15	MATP_IL	MATR	1.0926	IL	0.1419	IR	0.0501	MR	0.6538	D	0.1541				
25	21	2	4	MATP_IL	END	1.4397	IL	0.6553	IR	0.0445	E	0.3002						
26		1	31	MATP_IR	BIF	0.9361	IR	0.2827	B	0.7173								
27		145	227	MATP_IR	MATP	1.5494	IR	0.1884	MP	0.7090	ML	0.0165	MR	0.0588	D	0.0273		
28		129	701	MATP_IR	MATL	1.6332	IR	0.3681	ML	0.5752	D	0.0566						
29		8	160	MATP_IR	MATR	1.2428	IR	0.2633	MR	0.6809	D	0.0558						
30	26	0	31	MATP_IR	END	0.9361	IR	0.2827	E	0.7173								
31		108	1130	MATL_ML	BIF	1.2298	IL	0.0078	B	0.9922								
32		420	1319	MATL_ML	MATP	2.4162	IL	0.0132	MP	0.9520	ML	0.0150	MR	0.0129	D	0.0070		
33		19013	19371	MATL_ML	MATL	1.8632	IL	0.0082	ML	0.9711	D	0.0207						
34		859	6692	MATL_ML	MATR	72.1283	IL	0.0058	MR	0.9755	D	0.0187						
35	31	801	1130	MATL_ML	END	1.2298	IL	0.0078	E	0.9922								
36		28	172	MATL_D	BIF	6.8008	IL	0.0029	B	0.9971								
37		103	103	MATL_D	MATP	0.7288	IL	0.0321	MP	0.5730	ML	0.0536	MR	0.1654	D	0.1758		
38		3152	3152	MATL_D	MATL	0.4101	IL	0.0138	ML	0.3105	D	0.6756						
39		154	154	MATL_D	MATR	0.6736	IL	0.0203	MR	0.6014	D	0.3782						
40	36	144	172	MATL_D	END	6.8008	IL	0.0029	E	0.9971								
41	26	13	31	MATL_IL	BIF	0.9361	IL	0.2827	B	0.7173								
42	27	35	227	MATL_IL	MATP	1.5494	IL	0.1884	MP	0.7090	ML	0.0588	MR	0.0165	D	0.0273		
43	28	549	701	MATL_IL	MATL	1.6332	IL	0.3681	ML	0.5752	D	0.0566						
44	29	45	160	MATL_IL	MATR	1.2428	IL	0.2633	MR	0.6809	D	0.0558						
45	26	0	31	MATL_IL	END	0.9361	IL	0.2827	E	0.7173								

Table 2.1: **Dirichlet priors for transitions (table 1 of 2).** "dist #"= index for the 45 of the 73 different types of transition distributions in CMs. The remaining 28 appear in Table 2.2. Asterisks mark six distributions which had very few observed counts even after tying into groups, and for which we assigned a uniform plus-one Laplace prior. "tied to"= if an index is shown in this column, this distribution was estimated in a group (pooling observed counts) with the indicated distribution. "dist counts"= total counts observed for this distribution in the training data, before pooling into groups. "grp counts"= total counts for a group of one or more pooled distributions; this is the size of the training data sets for 36 different single-component Dirichlet priors. "node state"= unique CM state type the transition is from. "next node"= the node type that the transitions are going to. The codes for node types and state types in a CM are more fully explained in [57].

						Dirichlet α parameters												
dist #	tied to	dist counts	grp counts	node state	next node	$\lvert\alpha\rvert$	t	$\frac{\alpha_t}{\lvert\alpha\rvert}$	t	$\frac{\alpha_t}{\lvert\alpha\rvert}$	t	$\frac{\alpha_t}{\lvert\alpha\rvert}$	t	$\frac{\alpha_t}{\lvert\alpha\rvert}$	t	$\frac{\alpha_t}{\lvert\alpha\rvert}$	t	$\frac{\alpha_t}{\lvert\alpha\rvert}$
46	31	206	1130	MATR_MR	BIF	1.2298	IR	0.0078	B	0.9922								
47	32	848	1319	MATR_MR	MATP	2.4162	IR	0.0132	MP	0.9520	ML	0.0150	MR	0.0129	D	0.0070		
48	34	5833	6692	MATR_MR	MATR	2.1283	IR	0.0058	MR	0.9755	D	0.0187						
49		39	39	MATR_D	BIF	0.4664	IR	0.0463	B	0.9537								
50		176	176	MATR_D	MATP	0.8689	IR	0.0245	MP	0.6126	ML	0.1269	MR	0.0471	D	0.1890		
51		771	771	MATR_D	MATR	0.4869	IR	0.0119	MR	0.3373	D	0.6507						
52	26	15	31	MATR_IR	BIF	0.9361	IR	0.2827	B	0.7173								
53	27	39	227	MATR_IR	MATP	1.5494	IR	0.1884	MP	0.7090	ML	0.0165	MR	0.0588	D	0.0273		
54	29	107	160	MATR_IR	MATR	1.2428	IR	0.2633	MR	0.6809	D	0.0558						
55		338	338	BEGL_S	MATP	5.0422	MP	0.9579	ML	0.0121	MR	0.0183	D	0.0117				
56	31	15	1130	BEGR_S	BIF	1.2298	IL	0.0078	B	0.9922								
57	32	51	1319	BEGR_S	MATP	2.4162	IL	0.0132	MP	0.9520	ML	0.0150	MR	0.0129	D	0.0070		
58	33	358	19371	BEGR_S	MATL	1.8632	IL	0.0082	ML	0.9711	D	0.0207						
59	26	2	31	BEGR_IL	BIF	0.9361	IL	0.2827	B	0.7173								
60	27	3	227	BEGR_IL	MATP	1.5494	IL	0.1884	MP	0.7090	ML	0.0588	MR	0.0165	D	0.0273		
61	28	19	701	BEGR_IL	MATL	1.6332	IL	0.3681	ML	0.5752	D	0.0566						
62	1	3	11	ROOT_S	BIF	0.5509	IL	0.1229	IR	0.0001	B	0.8770						
63	2	96	7119	ROOT_S	MATP	7.2986	IL	0.0023	IR	0.0024	MP	0.9816	ML	0.0056	MR	0.0046	D	0.0035
64	3	230	1830	ROOT_S	MATL	1.5914	IL	0.0179	IR	0.0155	ML	0.9200	D	0.0466				
65	4	50	195	ROOT_S	MATR	1.9038	IL	0.0173	IR	0.0073	MR	0.8903	D	0.0852				
66	21	0	4	ROOT_IL	BIF	1.4397	IL	0.6553	IR	0.0445	B	0.3002						
67	22	5	126	ROOT_IL	MATP	0.9402	IL	0.1673	IR	0.1394	MP	0.5904	ML	0.0443	MR	0.0259	D	0.0327
68	23	5	119	ROOT_IL	MATL	0.8046	IL	0.3108	IR	0.1936	ML	0.4610	D	0.0346				
69	24	1	15	ROOT_IL	MATR	1.0926	IL	0.1419	IR	0.0501	MR	0.6538	D	0.1541				
70	26	0	31	ROOT_IR	BIF	0.9361	IR	0.2827	B	0.7173								
71	27	5	227	ROOT_IR	MATP	1.5494	IR	0.1884	MP	0.7090	ML	0.0165	MR	0.0588	D	0.0273		
72	28	4	701	ROOT_IR	MATL	1.6332	IR	0.3681	ML	0.5752	D	0.0566						
73	29	0	160	ROOT_IR	MATR	1.2428	IR	0.2633	MR	0.6809	D	0.0558						

Table 2.2: **Dirichlet priors for transitions (table 2 of 2).** “dist #”= index for the 28 of the 73 different types of transition distributions in CMs. The first 45 appear in Table 2.1. “tied to”= if an index is shown in this column, this distribution was estimated in a group (pooling observed counts) with the indicated distribution. “dist counts”= total counts observed for this distribution in the training data, before pooling into groups. “grp counts”= total counts for a group of one or more pooled distributions; this is the size of the training data sets for 36 different single-component Dirichlet priors. “node state”= unique CM state type the transition is from. “next node”= the node type that the transitions are going to. The codes for node types and state types in a CM are more fully explained in [57].

The reason to use two different datasets to estimate transition versus emission priors is the following. Rfam provides many different structural RNA alignments, but of uneven quality and varying depth (number of sequences). The European rRNA database provides a small number of different RNA alignments, but of high quality and great depth. A transition prior training set should be maximally diverse, so as not to bias any transition types toward any particular RNA structure, so we used the 381 different Rfam alignments for transitions. Emission prior estimation, in contrast, improves with alignment depth and accuracy, but does not require broad structural diversity per se, so we used rRNA data for emissions.

Inspection of the Dirichlet α parameters shows sensible trends. In the transition priors, transitions between main (consensus) states are now favored (higher α values) relative to insertions and deletions. In the base pair emission mixture prior, all components favor Watson-Crick and G-U pairs, with different components preferring different proportions of pairs in a particular covarying aligned column (for instance, component 1 likes all four Watson-Crick pairs, component 2 describes covarying conservation of CG,UA,UG pairs, and component 3 specifically likes conserved CG pairs), and the mean α parameters prefer GC/CG pairs over AU/UA pairs. In the singlet emission mixture prior, some components are capturing strongly conserved residues (component 1 favors conserved U's, for example) while other components favor more variation (components 4 and 5, for example), and the marginal α parameters show a strong A bias, reflecting the known bias for adenine in single-stranded positions of structural RNAs (especially ribosomal RNAs).

There is redundancy between some components (notably 5 and 8 in the base pair mixture and 2, 3 and 8 in the singlet mixture). This is typical for statistical mixture estimation, which (unlike, say, principal components analysis) does not guarantee independence between components. The decision to use nine pair and eight singlet components was empirical, as these priors performed better than priors with fewer components on the benchmark we describe below (data not shown).

Note that all singlet positions are modeled with one singlet mixture prior distribution, and all base pairs are modeled with one base pair mixture prior. These priors do not

alignment	# seqs	# filtered seqs	# aln columns	# consensus base pairs	# consensus SS columns	base pair counts	SS counts
LSU	1551	139	7270	601	1532	65229	180558
SSU bap	12773	254	2653	421	680	97834	153565
SSU euk	7151	207	4558	407	959	72521	174260
SSU mito	1039	107	3791	216	524	19803	56510

Table 2.3: **Summary statistics for the dataset used for emission prior estimation.** "SS" = single-stranded.

component i	1	2	3	4	5	6	7	8	9
q_i	0.0305	0.0703	0.1185	0.1810	0.1888	0.1576	0.0417	0.0959	0.1156
$\|\alpha\|$	14.3744	2.9920	26.2757	0.5342	4.2716	13.3232	33.8619	22.2258	33.1991

ab	mean α	$\frac{\alpha_{ab}}{\|\alpha\|}$	$\frac{\alpha_{ab}}{\|\alpha\|}$	$\frac{\alpha_{ab}}{\|\alpha\|}$	$\frac{\alpha_{ab}}{\|\alpha\|}$	$\frac{\alpha_{ab}}{\|\alpha\|}$	$\frac{\alpha_{ab}}{\|\alpha\|}$	$\frac{\alpha_{ab}}{\|\alpha\|}$	$\frac{\alpha_{ab}}{\|\alpha\|}$	$\frac{\alpha_{ab}}{\|\alpha\|}$
AA	0.0063	0.0398	0.0390	0.0011	0.0017	0.0005	0.0062	0.0064	0.0058	0.0002
AC	0.0092	0.0421	0.0176	0.0009	0.0152	0.0018	0.0125	0.0115	0.0051	0.0046
AG	0.0052	0.0381	0.0226	0.0046	0.0034	0.0008	0.0032	0.0040	0.0053	0.0001
AU	**0.1663**	**0.1092**	0.0864	0.0194	**0.2138**	**0.1464**	**0.2563**	**0.7360**	**0.1295**	0.0404
CA	0.0086	0.0412	0.0510	0.0054	0.0027	0.0044	0.0018	0.0030	0.0138	0.0002
CC	0.0038	0.0327	0.0115	0.0030	0.0001	0.0003	0.0036	0.0039	0.0035	0.0041
CG	**0.2412**	**0.1007**	**0.1392**	**0.8310**	**0.1359**	**0.3211**	0.0889	0.0340	**0.2870**	0.0147
CU	0.0066	0.0418	0.0172	0.0027	0.0104	0.0019	0.0045	0.0076	0.0052	0.0003
GA	0.0061	0.0362	0.0266	0.0002	0.0074	0.0002	0.0058	0.0045	0.0042	0.0021
GC	**0.2547**	**0.1299**	0.0544	0.0206	**0.1786**	**0.1613**	**0.4079**	0.0945	**0.1155**	**0.8858**
GG	0.0063	0.0327	0.0142	0.0045	0.0091	0.0005	0.0072	0.0023	0.0044	0.0030
GU	0.0567	0.0811	0.0412	0.0049	**0.1355**	0.0451	0.0668	0.0303	0.0356	0.0218
UA	**0.1571**	**0.1063**	**0.3085**	0.0672	**0.1856**	**0.2293**	0.0902	0.0363	**0.3108**	0.0151
UC	0.0063	0.0477	0.0263	0.0006	0.0048	0.0002	0.0056	0.0042	0.0060	0.0038
UG	0.0543	0.0746	**0.1054**	0.0317	0.0807	0.0814	0.0299	0.0120	0.0551	0.0032
UU	0.0114	0.0459	0.0389	0.0022	0.0151	0.0048	0.0098	0.0095	0.0133	0.0008

Table 2.4: **Parameters of the 9 component Dirichlet mixture emission prior for base pairs.** q_i = mixture coefficient for component i. Normalized α values > 0.10 are in bold faced type (0.10 was arbitrarily chosen to highlight higher values).

distinguish between singlet residues in different types of loops, for example, nor between a stem-closing base pair versus other base pairs. In the future it may prove advantageous to adopt more complex priors to capture effects of structural context on base pair and singlet residue preference.

In another step to increase sensitivity and specificity of the program, we adopted the "entropy weighting" technique described for profile HMMs [133] for estimating the total effective sequence number for an input query alignment. This is an *ad hoc* method for reducing the information content per position of a model, which helps a model that has

component i	1	2	3	4	5	6	7	8
q_i	0.0851	0.0159	0.1020	0.4160	0.0745	0.0554	0.1184	0.1327
$\|\alpha\|$	15.4467	154.4640	180.2862	5.4562	0.2199	16.4089	13.4592	19.9059

a	mean α	$\frac{\alpha_a}{\|\alpha\|}$	$\frac{\alpha_a}{\|\alpha\|}$	$\frac{\alpha_a}{\|\alpha\|}$	$\frac{\alpha_a}{\|\alpha\|}$	$\frac{\alpha_a}{\|\alpha\|}$	$\frac{\alpha_a}{\|\alpha\|}$	$\frac{\alpha_a}{\|\alpha\|}$	$\frac{\alpha_a}{\|\alpha\|}$
A	**0.3951**	0.0373	**0.9961**	**0.9787**	**0.3109**	**0.3383**	0.0375	0.0864	**0.8247**
C	**0.1635**	0.0490	0.0015	0.0052	**0.2067**	**0.1782**	**0.8916**	0.0303	0.0493
G	**0.2041**	0.0220	0.0023	0.0072	**0.1751**	**0.2905**	0.0182	**0.8313**	0.0569
U	**0.2372**	**0.8917**	0.0000	0.0090	**0.3073**	**0.1930**	0.0527	0.0519	0.0691

Table 2.5: **Parameters of the 8 component Dirichlet mixture emission prior for singlets.** q_i = mixture coefficient for component i. Normalized α values > 0.10 are in bold faced type (0.10 was arbitrarily chosen to highlight higher values).

been trained on closely related sequences to recognize distantly related homologues [1]. In entropy weighting, one reduces the total effective sequence number (which would normally be the actual number of sequences in the input alignment), thereby increasing the influence of the Dirichlet priors, flattening the transition and emission distributions, and reducing the overall information content. We approximate a model's entropy as the mean entropy per consensus residue, as follows. Let C be the set of all MATP_MP states emitting consensus base pairs (a,b), and let D be the set of all MATL_ML and MATR_MR states emitting consensus singlets (a); the entropy is then calculated as:

$$\frac{\sum_{v \in C} - \sum_{a,b} e_v(a,b) \log e_v(a,b) - \sum_{v \in D} \sum_a e_v(a) \log e_v(a)}{2|C| + |D|}$$

For each input multiple alignment, the effective sequence number is set (by bracketing and binary search) so as to obtain a specified target entropy. The target entropy for INFERNAL is a free parameter, which we optimized on the benchmark described below to identify our default value of 1.46 bits.

Benchmarking

To assess the effect of QDB, informative priors, and entropy weighting on the speed, sensitivity, and specificity of RNA similarity searches, we designed a benchmark based on the Rfam database [100]. The benchmark was designed so that we would test many RNA query/target pairs, with each query consisting of a given RNA sequence alignment, and each target consisting of a distantly related RNA homolog buried in a context of a random genome-like background sequence.

We started with seed alignments from Rfam version 7.0. In each alignment, sequences shorter than 70% the median length were removed. We clustered the sequences in each family by single-linkage-clustering by % identity (as calculated from the given Rfam alignment), then split the clusters such that the training set and test sequences satisfied three conditions: 1) no training/test sequence pair is more than 60% identical; 2) no test sequence pair is greater than 70% identical; 3) at least 5 sequences are in the training set. Fifty-one families satisfy these criteria (listed in Table 2.6), giving us 51 different query alignments (containing 5 to 1080 sequences each) and 450 total test sequences (from 1 to 66 per query). We embedded the test sequences in a one megabase "pseudo-genome" consisting of twenty 50 kilobase "chromosomes", generated as independent, identically distributed (iid) random sequences with uniform base frequencies. The 450 test sequences were embedded into this sequence by replacement, by randomly choosing a chromosome, orientation, and start position, and disallowing overlaps between test sequences. The total length of the 450 test sequences is 101,855 nt, leaving 898,145 nt of random background sequence.

The benchmark proceeds by first building a CM for each query alignment, then searching the pseudo-genome with each CM in local alignment mode. All hits above a threshold of 8.0 in raw bit score for each of the 51 queries were sorted by score into 51 ranked family specific lists, as well as one ranked master list of all 51 sets of scores. Each hit is classified into one of three categories, "positive", "ignore", or "negative". A "positive" is a hit that significantly overlaps with a true test sequence from the same family as the query. An "ignore" is a hit that significantly overlaps with a test sequence from a different family, where "significantly overlap" means that the length of overlap between two sequences (either two hits, or one hit and one test sequence embedded in the pseudo-genome) is more than 50% the length of the shorter sequence. (Although it would be desirable to measure the false positive rate on nonhomologous structural RNAs, we cannot be sure that any given pair of Rfam families is truly nonhomologous. Like most sequence family databases, Rfam

Rfam 7.0	family	#	#	avg len		non-banded	QDB ($\beta = 10^{-7}$)		QDB ($\beta = 10^{-7}$)			
ID	name	query	test	query	W	time	time	spd up	MER	FP	FN	thr
RF00177	SSU_rRNA_5	145	21	593	690	96.97	7.61	12.74	0	0	0	9.90
RF00024	Telomerase-vert	20	11	436	505	52.99	4.44	11.94	0	0	0	11.30
RF00011	RNaseP_bact_b	30	1	366	441	33.44	3.72	8.98	0	0	0	11.31
RF00018	CsrB	8	1	351	403	28.76	2.27	12.68	0	0	0	12.98
RF00040	rne5	6	1	338	368	19.69	2.60	7.57	0	0	0	11.79
RF00023	tmRNA	19	40	334	463	24.58	2.47	9.95	11	0	11	11.20
RF00010	RNaseP_bact_a	233	1	332	514	33.82	3.29	10.27	0	0	0	12.61
RF00009	RNaseP_nuc	26	21	320	530	27.89	7.11	3.93	19	0	19	11.67
RF00017	SRP_euk_arch	28	21	303	328	14.78	2.70	5.48	6	0	6	10.40
RF00028	Intron_gpI	5	24	300	381	17.55	3.59	4.88	19	2	17	10.70
RF00373	RNaseP_arch	20	13	290	337	15.06	3.12	4.82	0	0	0	12.23
RF00030	RNase_MRP	18	3	284	394	19.75	3.12	6.34	3	0	3	12.46
RF00101	SraC_RyeA	6	1	250	278	9.50	1.54	6.19	0	0	0	11.88
RF00230	T-box	10	35	244	298	8.23	1.80	4.58	1	0	1	12.34
RF00448	IRES_EBNA	7	1	213	238	7.25	1.26	5.76	1	0	1	11.99
RF00012	U3	6	5	212	240	7.17	1.61	4.46	2	0	2	13.02
RF00174	Cobalamin	87	66	203	326	9.49	2.85	3.32	0	0	0	11.28
RF00004	U2	76	1	184	215	5.95	1.12	5.29	0	0	0	10.01
RF00234	glmS	8	3	181	303	7.98	1.83	4.36	0	0	0	11.22
RF00168	Lysine	33	17	180	223	5.71	1.45	3.94	0	0	0	15.98
RF00380	ykoK	35	3	168	192	4.33	1.23	3.52	0	0	0	13.10
RF00003	U1	46	6	159	184	4.14	0.89	4.63	0	0	0	11.24
RF00025	Telomerase-cil	10	2	157	188	3.89	1.00	3.88	2	0	2	13.97
RF00002	5_8S_rRNA	62	1	151	183	3.44	0.97	3.55	0	0	0	11.28
RF00379	ydaO-yuaA	31	4	147	227	4.38	1.63	2.69	0	0	0	12.25
RF00067	U15	9	3	146	178	2.71	0.98	2.76	0	0	0	11.11
RF00029	Intron_gpII	7	11	141	276	4.75	1.32	3.59	1	0	1	11.08
RF00015	U4	25	1	141	187	3.66	1.04	3.53	1	0	1	13.46
RF00096	U8	5	1	135	177	2.98	0.93	3.20	0	0	0	11.56
RF00080	yybP-ykoY	20	33	129	173	3.05	1.13	2.69	1	0	1	10.78
RF00114	S15	10	1	117	138	1.78	0.60	2.99	0	0	0	13.12
RF00020	U5	29	3	115	139	2.06	0.74	2.80	0	0	0	13.64
RF00059	THI	228	8	109	222	3.34	1.46	2.29	0	0	0	13.66
RF00504	gcvT	109	5	102	199	2.37	1.40	1.70	0	0	0	13.40
RF00167	Purine	33	4	99	119	1.49	0.57	2.62	0	0	0	13.02
RF00169	SRP_bact	46	15	96	120	1.47	0.65	2.26	0	0	0	11.58
RF00055	snoZ37	5	1	94	117	1.14	0.53	2.16	1	0	1	13.96
RF00019	Y	15	1	94	128	1.42	0.73	1.94	1	0	1	14.25
RF00033	MicF	8	1	93	114	1.28	0.51	2.50	0	0	0	13.18
RF00213	snoR38	7	3	88	147	1.36	0.70	1.94	0	0	0	16.07
RF00054	U25	5	1	87	107	0.96	0.46	2.09	1	0	1	16.66
RF00206	U54	12	1	81	115	0.94	0.53	1.76	1	0	1	15.80
RF00104	mir-10	9	2	73	94	0.85	0.52	1.64	2	0	2	16.13
RF00005	tRNA	1080	19	73	127	1.35	0.48	2.81	5	1	4	12.62
RF00170	msr	5	3	70	112	0.86	0.45	1.92	3	0	3	13.49
RF00163	Hammerhead_1	65	1	68	233	1.60	0.90	1.77	0	0	0	15.82
RF00031	SECIS	11	24	64	87	0.69	0.42	1.63	13	2	11	14.58
RF00165	Corona_pk3	10	1	63	80	0.55	0.31	1.78	1	0	1	14.72
RF00066	U7	28	2	62	85	0.59	0.34	1.73	0	0	0	14.23
RF00008	Hammerhead_3	82	1	55	101	0.71	0.44	1.61	0	0	0	14.71
RF00037	IRE	36	1	28	45	0.17	0.12	1.39	1	0	1	17.64
MER statistics summed across all families									96	5	91	N/A
Summary MER statistics (using one threshold for all families)									114	3	111	16.38
average timing statistics						10.02	1.64	4.21				
total timing statistics						510.86	83.48	6.12				

Table 2.6: **Rfam benchmark families with timing and MER statistics.** "W" = window length, maximum size of a hit per family, calculated as dmax(0). Running times for standard (non-banded) and QDB ($\beta = 10^{-7}$) searches are given for each family, in CPU-hours per Mb. The MER threshold ("thr" column) is the bit score for a given family at which the sum of false positives ("FP") and false negatives ("FN") is minimized. "MER" = minimum error rate, FP+FN at threshold. In the row labeled "Summary MER statistics", these are derived from a single score threshold in a ranked list of all hits across all families. All statistics are for INFERNAL version 0.72 in local alignment mode.

is clustered computationally, and more sensitive methods will reveal previously unsuspected relationships that should not be benchmarked as "false positives".) A "negative" is a hit that is not a positive or an ignore. For any two negatives that significantly overlap, only the one with the better score is counted.

The minimum error rate (MER) ("equivalence score") [206] was used as a measure of benchmark performance. The MER score is defined as the minimum sum of the false positives (negative hits above the threshold) and false negatives (true test sequences which have no positive hit above the threshold), at all possible choices of score threshold. The MER score is a combined measure of sensitivity and specificity, where a lower MER score is better. We calculate two kinds of MER scores. For a *family-specific* MER score, we choose a different optimal threshold in each of the 51 ranked lists, and for a *summary* MER score, we choose a single optimal threshold in the master list of all hits. The summary MER score is the more relevant measure of our current performance, because it demands a single query-independent bit score threshold for significance. A family-specific MER score reflects the performance that could be achieved if INFERNAL provided P-values (currently it reports only raw bit scores).

For comparison, BLASTN was also benchmarked on these data using a family-pairwise-search (FPS) procedure [103]. For each query alignment, each training sequence is used as a query sequence to search the pseudo-genome, all hits with an E-value of less than 1.0 were sorted by increasing E-value, and the lowest E-value positive hit to a given test sequence is counted.

Using this benchmark, we addressed several questions about QDB's performance.

What is the best setting of the single QDB free parameter, β, which specifies how much probability mass to sacrifice? Figure 2.3 shows the average speedup per family and summary MER score as a function of varying β. There is no clear choice. The choice of β is a tradeoff of accuracy for speed. We chose a default of $\beta = 10^{-7}$ as a reasonable value that obtains a modest speedup with minimal loss of accuracy.

How well does QDB accelerate CM searches? Figure 2.4 shows the time required for

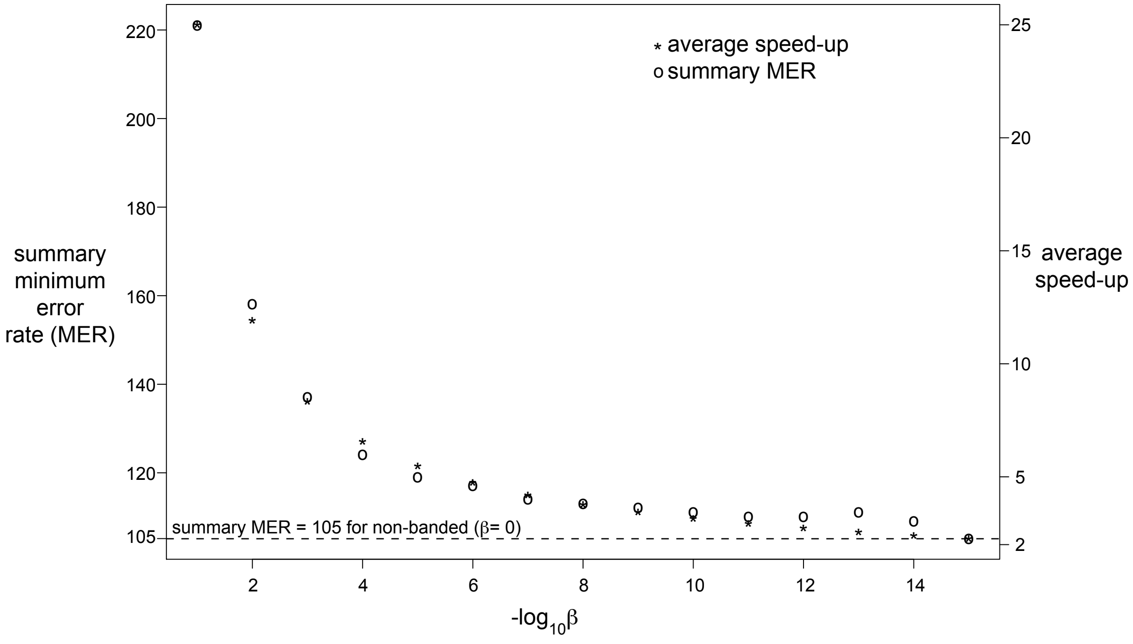


Figure 2.3: **Effect of varying the β parameter on sensitivity, specificity, and speedup.**

searching the 1 Mb benchmark target sequence with each of the 51 models, as a function of the average query RNA length. QDB reduces the average-case running time complexity of the CM search algorithm from $LN^{2.36}$ to $LN^{1.32}$. Observed accelerations relative to the standard algorithm range from 1.4-fold (for the IRE, iron response element) to 12.7-fold (for CsrB RNA), with an average speed-up per family of 4.2-fold. In total search time for the benchmark (sum of all 51 searches), the acceleration is six-fold, because large queries have disproportionate effect on the total time.

How much does QDB impact sensitivity and specificity? Optimal alignments are not guaranteed to lie within QDB's high-probability bands. This is expected to compromise sensitivity. The hope is that QDB's bands are sufficiently wide and accurate that the loss is negligible. Figure 2.5 shows ROC plots (sensitivity versus false positive rate) on the benchmark for the new version of INFERNAL (version 0.72) in standard versus QDB mode. These plots are nearly superposed, showing that the loss in accuracy is small at the default QDB setting of $\beta = 10^{-7}$.

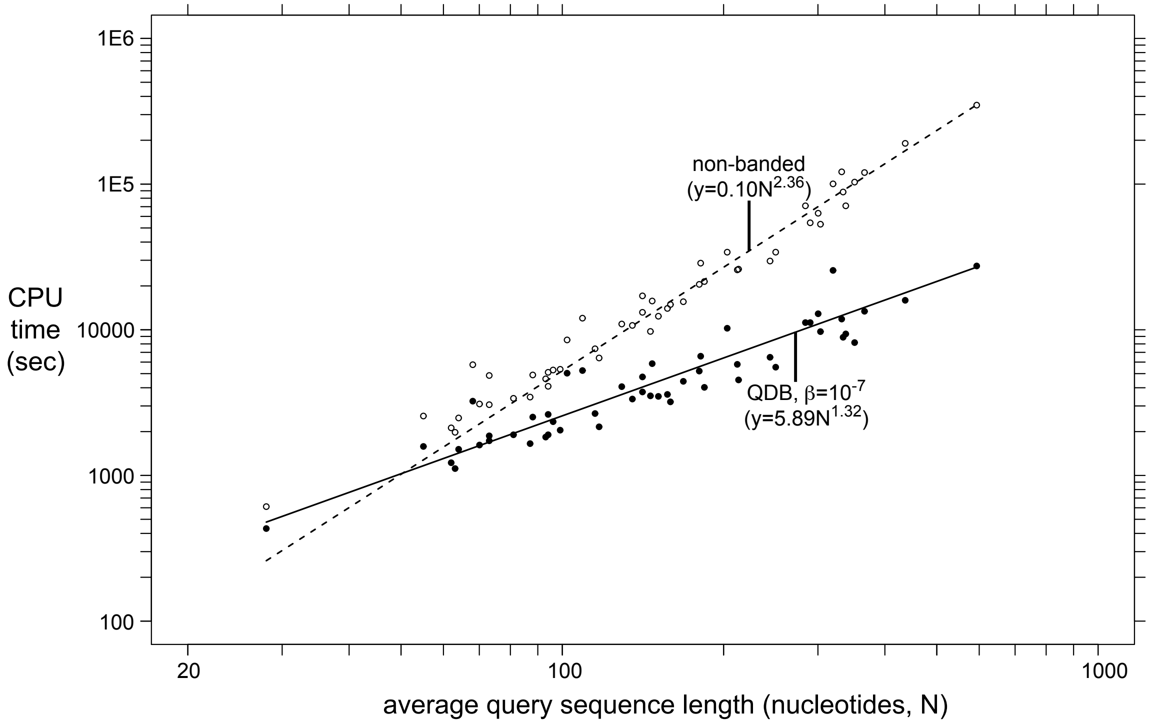


Figure 2.4: **CPU time required by CM searches with and without QDB.** The time required for searching the 1 Mb target pseudogenome with each of the 51 benchmark models is shown as a point, plotted on a log-log graph as a function of the average length of the RNA sequences in the query alignment; open circles are without QDB, and filled circles are with QDB (with the default $\beta = 10^{-7}$). Lines represent fits to a power law (aN^b), showing that for a fixed $L = 1$ Mb target database size, the standard CYK algorithm empirically scales as $N^{2.36}$ and the QDB algorithm scales as $N^{1.32}$. The apparent intersection of the linear fitted lines is deceptive. At small query lengths, run time is dominated by factors other than the CM alignment computation, such as i/o. QDB searches are always faster than non-banded searches even for synthetic tiny queries of less than 10 nt (data not shown).

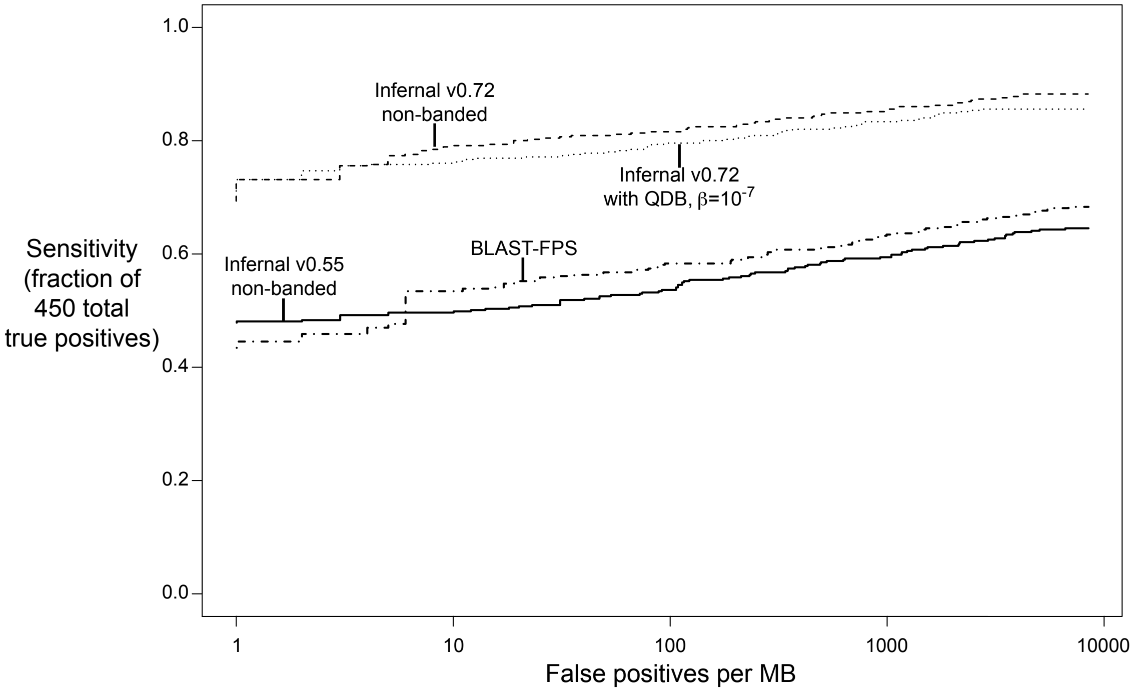


Figure 2.5: **ROC curves for the benchmark.** Plots are shown for the new INFERNAL 0.72 with and without QDB, for the old INFERNAL 0.55, and for family-pairwise-searches (FPS) with BLASTN.

How much do our changes in parameterization (the addition of informative Dirichlet priors and entropy weighting) improve sensitivity and specificity? Figure 2.5 shows that the new INFERNAL 0.72 is a large improvement over the previous INFERNAL version 0.55, independent of QDB. (On average, in this benchmark, INFERNAL 0.55 is no better than a family-pairwise-search with BLASTN.) Table 2.7 breaks this result down in more detail, showing summary and family-specific MER scores for a variety of combinations of prior, entropy weighting, and QDB. These results show that both informative priors and entropy weighting individually contributed large improvements in sensitivity and specificity.

program	prior	entropy (bits)	β	summary MER	family-specific MER
BLASTN	-	-	-	216	188
INFERNAL 0.55	plus-1	-	-	232	180
INFERNAL 0.72	plus-1	-	-	215	187
INFERNAL 0.72	plus-1	1.46	-	208	191
INFERNAL 0.72	informative	-	-	177	158
INFERNAL 0.72	informative	1.46	-	105	90
INFERNAL 0.72	informative	1.46	10^{-7}	114	96

Table 2.7: **Rfam benchmark MER summary statistics.** "prior": "plus-1" if uninformative Laplace plus-1 priors were used; "informative" if new Dirichlet priors were used. "entropy": target model entropy in bits for entropy weighting; "-" if entropy weighting was not used. "β": tail probability loss for banded calculation used; "-" if search was non-banded. "summary MER": MER across 51 benchmark families; "family-specific MER": MER for each family, summed over all 51 families. Program versions: Row 1: WU-BLASTN-2.0MP –kap -W=7. For INFERNAL version 0.55, window length values (W) were preset as calculated in version 0.72 with plus-1 priors.

2.4 Discussion

CM searches take a long time, and this is the most limiting factor in using the INFERNAL software to identify RNA similarities. Prior to this work, INFERNAL 0.55 required 508 CPU-hours to search 51 models against just 1 megabase of sequence in our benchmarks (Table 2.6). Using QDB with β banding cutoffs that do not appreciably compromise sensitivity and specificity, INFERNAL 0.72 offers a six-fold speedup, performing the benchmark in 85 hours. Our eventual goal is to enable routine genome annotation of structural RNAs: to be able to search thousands of RNA models against complete genome sequences. A search of all 503 Rfam 7.0 models against the 3 GB human genome with INFERNAL 0.72 in QDB mode would take on the order of 300 CPU-years (down from 1800 with INFERNAL 0.55). We need to be able to do it in at most a few days, so we still need to increase CM search speed by five orders of magnitude. Thus, the QDB algorithm is a partial but certainly not complete solution to the problem. However, QDB combines synergistically with other acceleration techniques. Parallelization, on large clusters (though prohibitively expensive for all but a few centers) could give us further acceleration of three orders of magnitude. Software improvement (code optimization) will also contribute, but probably only about two-fold. Hardware improvements will contribute about two-fold per year or so so long as Moore's law continues. Finally, QDB is complementary to the filtering methods recently described by Weinberg and Ruzzo [263–265]. We view QDB as part of a growing suite of approaches that we can combine to accelerate INFERNAL.

Is it really worth burning all this CPU time in the first place? Do CM searches identify structural RNA homologies that other methods miss? Obviously we think so, but one would like to see convincing results. For large, diverse RNA families like tRNA, where a CM can be trained on over a thousand well-aligned sequences with a well-conserved consensus secondary structure, CM approaches have been quite powerful. The state of the art in large scale tRNA gene identification remains the CM-based program TRNASCAN-SE [162], and CMs were also used, for example, to discover the divergent tRNA for pyrrolysine, the "22nd amino acid" [238]. But Figure 2.5 shows that on average, over 51 more or less "typical" RNA families of various sizes and alignment quality, INFERNAL 0.55 was actually no better than doing a family-pairwise-search with BLASTN. Until recently, we have spent relatively little effort on how INFERNAL parameterizes its models, and relatively more on reducing its computational requirements [57], so previous versions of INFERNAL have performed best where naive parameterization works best: on very large, high-quality alignments of hundreds of sequences, which are atypical of many interesting homology search problems.

In this work, partly because the level of acceleration achieved by QDB is sensitive to transition parameterization, we have brought INFERNAL parameterization close to the state of the art in profile HMMs, by introducing mixture Dirichlet priors [234] and entropy weighting [133]. This resulted in a large improvement in the sensitivity and specificity of searches, as judged by our benchmark (Figure 2.5). The difference between INFERNAL and family-pairwise BLASTN now appears pronounced for average-case behavior, not just best-case behavior. However, while we trust our benchmarking to tell us we have greatly improved INFERNAL relative to previous versions of itself, our benchmarking does not allow us to draw firm conclusions about our performance relative to other software. For that, we prefer to see independent benchmarks. Benchmarks by tool developers are notoriously biased, and however honest we may try to be, some biases are essentially unavoidable. For one thing, establishing an internal benchmark for ongoing code development creates an insidious form of training on the test set, because we accept code changes that improve benchmark performance. In particular, we set the entropy weighting target of 1.46 bits and the numbers of mixture prior components by optimizing against our benchmark. Further, our benchmark does not use a realistic model for the background sequence of the "pseudo-genome", because we construct the background as a homogeneous IID (independent, identically distributed) sequence, and this poorly reflects the heterogeneous and repetitive nature of genomic sequence. This benchmark should be sufficient for an internal comparison of versions 0.55 and 0.72 of INFERNAL, because we have not altered how INFERNAL deals with heterogeneous compositional bias. But we cannot safely draw conclusions from our simple benchmark about the relative performance of INFERNAL and BLAST on real searches, for example, because BLAST may (and in fact does) treat sequence heterogeneity better than INFERNAL does. In this regard, currently we are aware of only one independent benchmark BRaliBase III [85]. BRaliBase III consists of many different query alignments of 5 or 20 RNA sequences, drawn from three different RNA families (U5, 5S rRNA, and tRNA). These authors' results broadly confirm our internal observations: while INFERNAL 0.55 showed mediocre performance compared to BLASTN and several other tools, a recent version of INFERNAL stood out as a superior method for RNA similarity search.

Nonetheless, though INFERNAL 0.72 shows large improvements in speed, sensitivity, and specificity over previous versions, there are numerous areas where we need to improve further.

A significant gap in our current implementation is that INFERNAL reports only raw bit scores, and does not yet report expectation values (E-values). CM local alignment scores empirically follow a Gumbel (extreme value) distribution [138], just as local sequence alignment scores do [130], so there are no technical hurdles in implementing E-values. This will be an immediate focus for the next version of INFERNAL. E-value calculations not only have the effect of reporting statistical significance (more meaningful to a user than a raw bit score), but they also normalize each family's score distribution into a more consistent overall rank order, because different query models exhibit different null distributions (particularly in the location parameter of the Gumbel distribution). We therefore expect E-values to contribute a large increase in performance whenever a single family-independent threshold is set. Table 2.7 roughly illustrates the expected gain, by showing the large difference between summary MER scores and family-specific MER scores.

Parameterization of both CMs and profile HMMs remains problematic, because these methods continue to assume that training sequences are statistically independent, when in fact they are related (often strongly so) by phylogeny. Methods like sequence weighting and entropy weighting do help, but they are *ad hoc* hacks: unsatisfying and unlikely to be optimal. Even mixture Dirichlet priors, though they appear to be mathematically sophisticated, fundamentally assume that observed counts are drawn as independent multinomial samples, and therefore the use of Dirichlet priors is fundamentally flawed. Probabilistic phylogenetic inference methodology needs to be integrated with profile search methods. This is an area of active research [116, 118, 219] in which important challenges remain, particularly in the treatment of insertions and deletions.

Finally, QDB is not the only algorithmic acceleration method we can envision. Michael Brown described a complementary banding method to accelerate his SCFG-based RNACAD ribosomal RNA alignment software [24], in which he uses profile HMM based sequence alignment to the target to determine bands where the more rigorous SCFG-based alignment should fall (because some regions of the alignment are well-determined based solely on sequence alignment). The gapped BLAST algorithm (seed word hits, ungapped hit extension, and banded dynamic programming) can conceivably be extended from two-dimensional sequence alignment to three-dimensional CM dynamic programming lattices. Developing such algorithms – and incorporating them into a widely useful, freely available codebase – are priorities for us.

2.5 Materials and Methods

The version and options used for BLAST in our benchmark are `WU-BLASTN-2.0MP --kap -W=7`. For INFERNAL, versions 0.55 and 0.72 were used as indicated. The complete INFERNAL software package, including documentation and the Rfam-based benchmark described here, may be downloaded from `http://infernal.janelia.org`. It is developed on GNU/Linux operating systems but should be portable to any POSIX-compliant operating system, including Mac OS/X. It is freely licensed under the GNU General Public License.

The ANSI C code we used for estimating maximum likelihood mixture Dirichlet priors depends on a copyrighted and nonredistributable implementation of the conjugate gradient descent algorithm from Numerical Recipes in C [213]. Our code, less the Numerical Recipes routine, is freely available upon request.

Chapter 3

Infernal 1.0: inference of RNA alignments [1]

3.1 Abstract

Summary: INFERNAL builds consensus RNA secondary structure profiles called covariance models (CMs), and uses them to search nucleic acid sequence databases for homologous RNAs, or to create new sequence- and structure-based multiple sequence alignments.
Availability: Source code, documentation, and benchmark downloadable from http://infernal.janelia.org. INFERNAL is freely licensed under the GNU GPLv3 and should be portable to any POSIX-compliant operating system, including Linux and Mac OS/X.
Contact: {nawrockie,kolbed,eddys}@janelia.hhmi.org

3.2 Introduction

When searching for homologous structural RNAs in sequence databases, it is desirable to score both primary sequence and secondary structure conservation. The most generally useful tools that integrate sequence and structure take as input any RNA (or RNA multiple alignment), and automatically construct an appropriate statistical scoring system that allows quantitative ranking of putative homologs in a sequence database [90, 121, 287]. Stochastic context-free grammars (SCFGs) provide a natural statistical framework for com-

---

[1] This chapter was published independently as Nawrocki EP, Kolbe DL, Eddy SR. Infernal 1.0: inference of RNA alignments. Bioinformatics 2009 25(10):1335-1337. The published material is unmodified but Table 3.1 has been appended.

bining sequence and (non-pseudoknotted) secondary structure conservation information in a single consistent scoring system [24, 53, 65, 225].

Here, we announce the 1.0 release of INFERNAL, an implementation of a general SCFG-based approach for RNA database searches and multiple alignment. INFERNAL builds consensus RNA profiles called *covariance models* (CMs), a special case of SCFGs designed for modeling RNA consensus sequence and structure. It uses CMs to search nucleic acid sequence databases for homologous RNAs, or to create new sequence- and structure-based multiple sequence alignments. One use of INFERNAL is to annotate RNAs in genomes in conjunction with the RFAM database [89], which contains hundreds of RNA families. RFAM follows a seed profile strategy, in which a well-annotated "seed" alignment of each family is curated, and a CM built from that seed alignment is used to identify and align additional members of the family. INFERNAL has been in use since 2002, but 1.0 is the first version that we consider to be a reasonably complete production tool. It now includes E-value estimates for the statistical significance of database hits, and heuristic acceleration algorithms for both database searches and multiple alignment that allow INFERNAL to be deployed in a variety of real RNA analysis tasks with manageable (albeit high) computational requirements.

3.3 Usage

A CM is built from a Stockholm format multiple sequence alignment (or single RNA sequence) with consensus secondary structure annotation marking which positions of the alignment are single stranded and which are base paired [192]. CMs assign position specific scores for the four possible residues at single stranded positions, the sixteen possible base pairs at paired positions, and for insertions and deletions. These scores are log-odds scores derived from the observed counts of residues, base pairs, insertions and deletions in the input alignment, combined with prior information derived from structural ribosomal RNA alignments. CM parameterization has been described in more detail elsewhere [57, 65, 138, 189, 192].

INFERNAL is composed of several programs that are used in combination by following four basic steps:

1. Build a CM from a structural alignment with *cmbuild*.
2. Calibrate a CM for homology search with *cmcalibrate*.
3. Search databases for putative homologs with *cmsearch*.
4. Align putative homologs to a CM with *cmalign*.

The calibration step is optional and computationally expensive (4 hours on a 3.0 GHz Intel Xeon for a CM of a typical RNA family of length 100 nt), but is required to obtain E-values that estimate the statistical significance of hits in a database search. *cmcalibrate* will also determine appropriate HMM filter thresholds for accelerating searches without an appreciable loss of sensitivity. Each model only needs to be calibrated once.

3.4 Performance

A published benchmark (independent of our lab) [85] and our own internal benchmark used during development [189] both find that INFERNAL and other CM based methods are the most sensitive and specific tools for structural RNA homology search among those tested. Figure 3.1 shows updated results of our internal benchmark comparing INFERNAL 1.0 to the previous version (0.72) that was benchmarked in Freyhult et al. [85], and also to family-pairwise-search with BLASTN [3, 103]. INFERNAL's sensitivity and specificity have greatly improved, due mainly to three relevant improvements in the implementation [192]: a biased composition correction to the raw log-odds scores, the use of Inside log likelihood scores (the summed score of all possible alignments of the target sequence) in place of CYK scores (the single maximum likelihood alignment score), and the introduction of approximate E-value estimates for the scores.

The benchmark dataset used in Figure 3.1 includes query alignments and test sequences from 51 RFAM (release 7) families (details in [189]). No query sequence is more than 60% identical to a test sequence. The 450 total test sequences were embedded at random positions in a 10 Mb "pseudogenome". Previously we generated the pseudogenome sequence

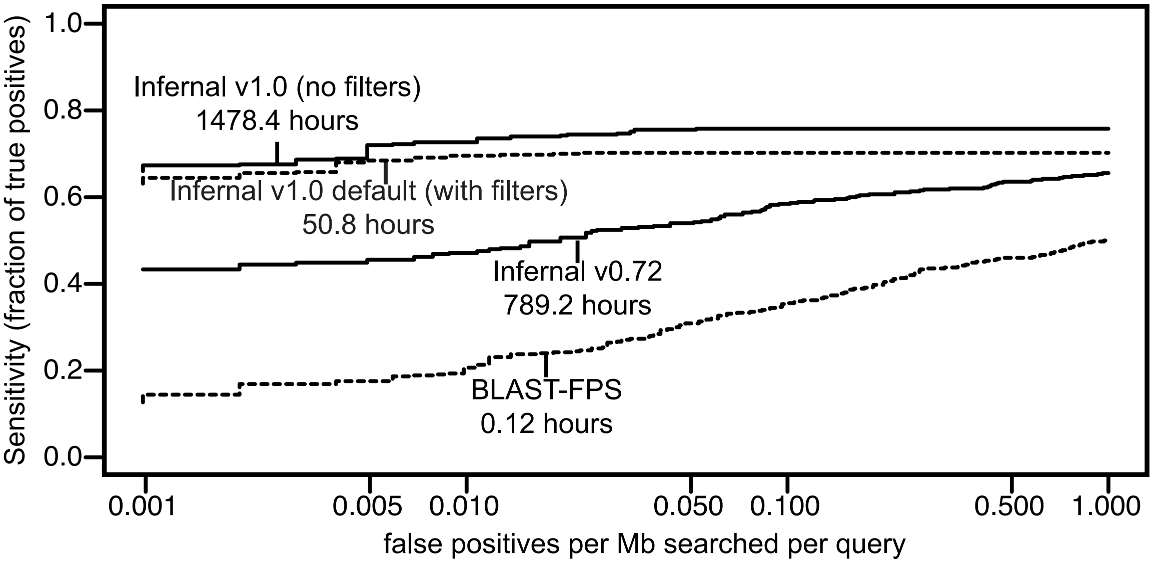


Figure 3.1: **ROC curves for the benchmark.** Plots are shown for the new INFERNAL 1.0 with and without filters, for the old INFERNAL 0.72, and for family-pairwise-searches (FPS) with BLASTN. CPU times are total times for all 51 family searches measured for single execution threads on 3.0 GHz Intel Xeon processors. The INFERNAL 1.0 times do not include time required for model calibration.

from a uniform residue frequency distribution [189]. Because base composition biases in the target sequence database cause the most serious problems in separating significant CM hits from noise, we improved the realism of the benchmark by generating the pseudogenome sequence from a 15-state fully connected hidden Markov model (HMM) trained by Baum-Welch expectation maximization [53] on genome sequence data from a wide variety of species. Each of the 51 query alignments was used to build a CM and search the pseudogenome, a single list of all hits for all families were collected and ranked, and true and false hits were defined (as described in Nawrocki and Eddy [189]), producing the ROC curves in Figure 3.1.

INFERNAL searches require a large amount of compute time (our 10 Mb benchmark search takes about 30 hours per model on average (Figure 3.1), also see Table 3.1). To alleviate this, INFERNAL 1.0 implements two rounds of filtering. When appropriate, the HMM filtering technique described by Weinberg and Ruzzo [265] is applied first with filter thresholds configured by *cmcalibrate* (occasionally a model with little primary sequence

		calibration	search (min/Mb)		alignment
family	length	(hours)	no filters	w/filters	(sec/seq)
tRNA	71	3.2h	23.5m	4.4m	0.01s
5S rRNA	119	4.4h	29.3m	1.1m	0.03s
Lysine riboswitch	183	8.9h	100.5m	1.3m	0.06s
SRP RNA	304	13.5h	166.0m	3.0m	0.18s
RNaseP	365	16.8h	205.6m	0.9m	0.19s
SSU rRNA	1466	84.5h	1265.5m	17.6m	1.10s
LSU rRNA	2909	169.7h	3907.6m	740.4m	3.34s

Table 3.1: **Calibration, search, and alignment running times for seven known structural RNAs of various sizes.** CPU times are measured on 3.0 GHz Intel Xeon processors with 8 GB RAM, running Red Hat AS4 Linux operating systems. All times were single execution threads except for SSU and LSU calibrations and searches which were run in parallel using MPI (OpenMPI) on 12 CPUs (times reported are actual times multiplied by 12). "Length" is the number of consensus positions (positions that contain gaps in fewer than 50% of the aligned sequences) in the input alignment. Randomly generated sequence of length 20 Mb (for filtered) and 2 Mb (for non-filtered) was used for the searches. Alignment files, CM files and instructions for reproduction are in the supplementary material.

conservation cannot be usefully accelerated by a primary sequence-based filter as explained in [192]). The query-dependent banded (QDB) CYK maximum likelihood search algorithm is used as a second filter with relatively tight bands ($\beta = 10^{-7}$, the β parameter is the subtree length probability mass excluded by imposing the bands as explained in [189]). Any sequence fragments that survive the filters are searched a final time with the Inside algorithm (again using QDB, but with looser bands ($\beta = 10^{-15}$)). In our benchmark, the default filters accelerate similarity search by about 30-fold overall, while sacrificing a small amount of sensitivity (Figure 3.1). This makes version 1.0 substantially faster than 0.72. BLAST is still orders of magnitude faster, but significantly less sensitive than INFERNAL. Further acceleration remains a major goal of INFERNAL development.

The computational cost of CM alignment with *cmalign* has been a limitation of previous versions of INFERNAL. Version 1.0 now uses a constrained dynamic programming approach first developed by Brown [24] that uses sequence-specific bands derived from a first-pass HMM alignment. This technique offers a dramatic speedup relative to unconstrained alignment, especially for large RNAs such as small and large subunit (SSU and LSU) ribosomal RNAs, which can now be aligned in roughly 1 and 3 seconds per sequence, respectively 3.1, as opposed to 12 minutes and 3 hours in previous versions. This acceleration has facilitated the adoption of INFERNAL by RDP, one of the main ribosomal RNA databases [40].

INFERNAL is now a faster and more sensitive tool for RNA sequence analysis. Version 1.0's heuristic acceleration techniques make some important applications possible on a single desktop computer in less than an hour, such as searching a prokaryotic genome for a particular RNA family, or aligning a few thousand SSU rRNA sequences. Nonetheless, INFERNAL remains computationally expensive, and many problems of interest require the use of a cluster. The most expensive programs (*cmcalibrate*, *cmsearch*, and *cmalign*) are implemented in coarse-grained parallel MPI versions which divide the workload into independent units, each of which is run on a separate processor.

Chapter 4

A filter acceleration pipeline for covariance model searches [1]

Covariance models (CMs) are profile stochastic context-free grammars (SCFGs), probabilistic models of the conserved sequence and well-nested secondary structure of an RNA family, that are useful for searching sequences databases for homologous RNAs. CM versions of the standard CYK and Inside SCFG algorithms exist for determining the optimal alignment (parse tree, $\hat{\pi}$) of a target sequence x to a CM M $(P(x, \hat{\pi}|M))$ and calculating the probability that x was generated from a CM $(P(x|M))$, respectively. However, these algorithms are computationally expensive and the time required to run them empirically scales with $LN^{2.4}$ for a query of length N residues (or consensus alignment columns) and a target database of length L, severely limiting the practical application of CM database searches [189]. The time required to use CYK and Inside to search both strands of a sequence of length 1 million residues (1 Mb) with CMs modeling four RNA families of various sizes is shown in Table 4.1.

Two complementary approaches have been taken to mitigate this high computational cost. The first is to accelerate the CM CYK similarity search dynamic programming algorithm. As described in Chapter 2, Sean Eddy and I introduced a banded variant of CYK that reduces the average case time complexity from $LN^{2.4}$ to $LN^{1.3}$ at a small cost to sensitivity [189]. The second approach is to reduce the search space (decrease L) by using a

---

[1] Sean Eddy and I plan to submit a version of this chapter for independent publication in the near future. This version was written solely by EPN.

		search time (min/Mb)	
family	length	CYK	Inside
5S rRNA	119	25.9	104.0
Lysine riboswitch	183	133.2	433.5
SRP RNA	304	276.4	936.2
RNase P RNA	365	733.4	1936.7

Table 4.1: **Running time of non-banded CYK and Inside algorithms in Infernal 1.0 for four CMs.** CMs are from RFAM 9.1 [89] - RF00001 (5S), RF00168 (Lysine), RF00017 (SRP), and RF00011 (RNase P). Times were measured on a single Intel Xeon 3.0 Ghz processor.

filter, typically a fast sequence-based scoring algorithm, to prune away low-scoring regions of the database that are deemed unlikely to contain high scoring hits to the CM. The CYK algorithm is then only used to search the surviving fraction of the database.

4.1 Previous work on accelerating CM searches using filters

Several filtering techniques have been developed for CMs. RFAM uses a BLAST-based filter on the RFAMSEQ target database prior to searching with CMs [89]. All sequences from the CM training alignment are used as queries, any target subsequence scored with an E-value less than a query-independent threshold to any query survives the filter and is searched with the CM. Weinberg and Ruzzo have introduced several types of filters, including two types of HMM filters: *rigorous filter* HMMs [264] and *maximum likelihood (ML)-heuristic* HMM filters [265] as well as *sub-CM* filters, which model some but not all of the CM consensus structure [263]. The rigorous filter models are parameterized to provably allow all target subsequences that score better than a preset CM score threshold to survive. ML-heuristic HMM filtering employs HMMs that are as similar as possible to the CM and aims to prune away 99% of the target database by setting the filter survival threshold such that a predicted 1% of the database will score better than it. Weinberg and Ruzzo's filtering techniques are implemented in the RaveNnA software package, which includes INFERNAL. Zhang and

Bafna introduced their own structure-based filtering method for CMs [287], as well as a key-word based method [288] for which only database subsequences that contain at least one of a pre-generated list of keywords survive the filter. Finally, Sun and Buhler have described a method that uses secondary-structure profiles (SSPs) to quickly filter for CM searches [247].

Zhang et al. [288] and Weinberg and Ruzzo [263] both experimented with combining different filters together to gain additional speed over using any of them independently. This can either be done by running each filter separately and using the CYK algorithm on any sequence that survives any of the filters, or by running the filters in succession in a pipeline in which each filter is only used on the (sub)sequences that have survived all previous filters in the pipeline. While, in principle, there is no limit to the number of different filters that can be used, in most work on CM filtering one or two rounds of filtering are used. Combining different filters in a pipeline is especially useful when the filters differ significantly in speed and sensitivity. If the fast, less sensitive filter is used first with a low (permissive) survival threshold, and the slower, more sensitive filter is run second using a higher (stricter) threshold, the search may proceed faster than using either filter independently while achieving similar sensitivity.

4.2 Optimized implementations of the CYK and Inside algorithms

The discussion above lists banded dynamic programming (DP) and filtering as two approaches towards accelerating CM searches. A third approach is to optimize the CYK and Inside DP implementations, which I attempted during development of INFERNAL version 1.0. I was able to speed up CYK about three-fold and Inside about two-fold (shown for four families in Table 4.2), a result that is consistent across all families (data not shown). These speedups are measured using GCC compiled versions of the `cmsearch` program from INFERNAL 0.81 and INFERNAL 1.0.

An important factor enabling this optimization was the availability of optimized HMM

		CYK time (min/Mb)			Inside time (min/Mb)		
family	length	v0.81	v1.0 Ref	v1.0 Fast	v0.81	v1.0 Ref	v1.0 Fast
5S rRNA	119	124.2	46.1	26.9	202.9	125.5	104.0
Lysine riboswitch	183	424.7	196.9	133.2	715.7	498.9	433.5
SRP RNA	304	966.3	421.2	276.4	1620.0	1077.8	936.2
RNaseP	365	1793.1	920.5	733.4	2944.1	2134.6	1936.7

Table 4.2: **Running time of different non-banded CYK and Inside implementations in Infernal v0.81 and v1.0 for four CMs.** CMs are from RFAM 9.1 [89] - RF00001 (5S), RF00168 (Lysine), RF00017 (SRP), and RF00011 (RNase P). Times were measured on a single Intel Xeon 3.0 Ghz processor.

Viterbi and Forward algorithm implementations from Sean Eddy's HMMER software package [62] written by Jeremy Buhler and Christopher Swope of Washington University in St. Louis. I was able to adapt many of the techniques from those implementations to CYK and Inside. The speedup between the "0.81" and "1.0 Ref" columns of Table 4.2 roughly measures the effect of these techniques: CYK runs two to three times faster and Inside runs about 50% faster.

Additionally, it proved possible to save time during the DP recursions by carefully rewriting the code to minimize the number of calculations needed to determine the score in each cell of the matrix. Because the number of calculations per cell is dependent on many factors, such as state type and number of possible transitions out of the state, this largely amounted to writing a separate block of code specific to each possible situation that only performs the minimal calculations for that situation. The speedup between the "1.0 Ref" and "1.0 Fast" columns of Table 4.2 roughly measures the effect of this careful rewrite: which accelerates CYK by roughly 25% and Inside by roughly 15%.

A side effect of optimization is longer, more complex and less easily understood code. INFERNAL version 1.0 (and 1.01) contain both optimized and reference (simpler to read and interpret) versions of CYK and Inside in the `cm_dpsearch.c` file as `FastCYKScan()`, `RefCYKScan()`, `FastIInsideScan()` and `RefIInsideScan()`. Each of these can be run in either non-banded or banded mode using query-dependent bands (chapter 2). The end of the `cm_dpsearch.c` file includes a benchmark driver that can be compiled independently, enabling running time comparisons of the various implementations.

4.3 Infernal's two-stage filter acceleration pipeline

In the remainder of this chapter I will discuss the two-stage filter pipeline implemented in INFERNAL as of version 1.0. In this section I will give a brief overview of the pipeline. Then I will discuss the critical issue of determining appropriate filter thresholds to use at each stage of the pipeline. Finally, I will present benchmark results that were used to choose the default values for various important parameters of the pipeline in INFERNAL version 1.01. The results show that, using the default set of parameters, the filter pipeline accelerates CM search roughly 200-fold compared to non-banded Inside in INFERNAL 1.0 at a small cost to accuracy, and roughly 40-fold compared to the default QDB CYK strategy of INFERNAL 0.72 (described in Chapter 2).

Choice of final scoring algorithm: CYK or Inside?

The choice of the *final* algorithm - the scoring algorithm used on the fraction of the database that survives all filters - is critical to search sensitivity. For practicality, all previous versions of INFERNAL have used the CYK algorithm to score sequences instead of the roughly two- to three-fold slower Inside algorithm (Table 4.1). However, Inside is potentially more powerful for homology search because it calculates its score by summing over all possible alignments of a target to the CM, instead of just using the single most likely alignment, as CYK does [53]. One indication that Inside may outperform CYK is that the Forward algorithm (the HMM analog of Inside) generally outperforms Viterbi (the HMM analog of CYK) for protein homology search using profile HMMs [59, 127].

Later in this chapter, I present results on the the sensitivity and specificity of Inside and CYK on a homology search benchmark and test the effect that filtering has on the running time of using either as the final algorithm. I find that best performing final algorithm is

QDB Inside, using a small β value of 10^{-15}. Additionally, I find that the two-stage filtering pipeline described below prunes away enough of the database that the difference in running time between using Inside and CYK as the final algorithm on the remaining fraction is negligible. For these reasons, the default final algorithm in INFERNAL is QDB Inside with $\beta = 10^{-15}$.

Stage 1: HMM filtering

Based on the impressive, roughly 100-fold acceleration and high sensitivity achieved by the ML-heuristic HMM filtering strategy of Weinberg and Ruzzo [265], I decided to essentially reimplement ML-heuristic HMMs in INFERNAL. These HMMs, which are slightly different from ML-heuristic HMMs, are called CM Plan 9 (CP9) HMMs in the INFERNAL codebase and are explained in more detail in Chapter 8. A CP9 version of the HMM Forward algorithm [53, 127] is used as the first filtering algorithm in INFERNAL's two-stage filtering pipeline. In [265], Weinberg and Ruzzo use a single, query-independent, thresholding scheme of setting the filter threshold such that a predicted 1% of the database will survive. I investigated an alternative query-*dependent* approach to determining thresholds, as described in detail below. Sometimes this approach predicts that using an HMM will potentially result in significant sensitivity loss, in which case the HMM filtering step is skipped.

Stage 2: QDB CYK filtering

INFERNAL uses QDB CYK as the filtering algorithm in the second-stage of its pipeline. The QDB CYK filter is only run on database subsequences surviving the first-stage HMM filter.

In chapter 2, the QDB version of the CYK algorithm was introduced and benchmarked to determine its performance for RNA similarity search as a standalone algorithm, i.e. as the final algorithm without using filters. But using QDB CYK as a filter for some other final algorithm (i.e. non-banded CYK or Inside) may make more sense. For example, when using QDB CYK as a filter it may be possible to use tighter bands for added acceleration while maintaining sensitivity relative to a standalone non-filtered implementation that uses QDB CYK as the final algorithm. This is investigated and discussed using the benchmark at the end of the chapter.

HMMs and QDB CYK are complementary filtering strategies

While QDB CYK scores both sequence and structure, HMM filters score only sequence. Ignoring structure makes the HMM faster than QDB CYK ($O(LN)$ for the HMM versus empirical scaling of $LN^{1.3}$ for QDB with $\beta = 10^{-7}$, see Chapter 2), but potentially less sensitive, especially for models in which structure significantly contributes to scoring. This qualitative difference in the scoring algorithm and its impact on filtering speed suggests that the two methods might complement each other well, specifically as a two-stage filter pipeline, with a faster HMM filter first and a slower QDB CYK filter second.

Filter thresholds control speed versus sensitivity

An important tradeoff exists between the acceleration gained and the sensitivity lost from using a filter. Acceleration is dependent on the speed of the filtering technique and the fraction of the database that survives the filter. The sensitivity loss depends on the ability of the filter to recognize (and not prune away) sequences that will score above the reporting score threshold of the final algorithm.

The previously developed filtering strategies discussed above prioritize speed versus sensitivity differently. For example, Weinberg and Ruzzo's ML-heuristic HMM filtering is designed for speed by pruning away a target fraction of the database, whereas their rigorous filter HMMs set filter survival thresholds so that 100% of the target subsequences that score better than a preset final algorithm threshold will survive, regardless of the threshold's impact on speed.

In INFERNAL, the main concern is sensitivity - but making practical implementations requires acceleration. The goal is to achieve the maximum possible acceleration that sacrifices an *acceptable* level of sensitivity relative to the final algorithm. Here, I judge speedup and sensitivity loss using a benchmark (described below) and subjectively define *acceptable* based on the results.

The speed and sensitivity tradeoff is determined by a filter's survival score threshold T.

Assuming higher filter scores are better, as T increases the number of surviving subsequences decreases, thus increasing the acceleration and potential sensitivity loss from using the filter. The choice of T is therefore critical to a filter's performance.

When speed is the main priority, it is straightforward to define a T that will eliminate the appropriate fraction of the database necessary to achieve the desired speedup. Given knowledge of the algorithm running times, this can be done either using E-value statistics of the filter and final algorithms, or by performing simulations to predict the appropriate threshold.

If the primary goal of a filter is to maintain sensitivity, as it is here, setting appropriate thresholds can be more complicated because it depends on differences between how the filter and final algorithm score target sequences. It is further complicated if those differences are dependent on the query sequence family. For example, imagine a filter that consistently scores family A homologs with an E-value no more than 10-fold higher (less significant) than the final algorithm E-value, but commonly scores family B homologs with E-values 1000-fold higher than the final algorithm E-values. Achieving the same sensitivity for families A and B will require significantly different filter thresholds. An effective thresholding strategy must be able to predict the magnitude of filter/final score discrepancies for a given family.

Query-independent versus query-dependent thresholding

Filtering strategies can be usefully divided into two groups: those that require a query-*dependent* thresholding strategy to achieve consistently high sensitivity across different queries and and those that do not, and can achieve it using a single, query-*independent* thresholding strategy. A useful way for determining which group a given strategy belongs to is to examine the correlation between the filter and final algorithm scores for true sequences from many different query families. A consistent correlation across many queries implies a query-independent strategy may be sufficient, while large differences in the final/filter score correlation between different queries suggests a query-dependent strategy would be more effective.

One might expect that query-dependent strategies should be necessary when the filter algorithm is qualitatively different than the final algorithm, as is the case when using an HMM to filter for a CM. The HMM is scoring only the conserved sequence of the family, while the CM is scoring both the conserved sequence and structure. This should affect different query families differently and so require different thresholds to achieve similar sensitivity. A family with very low sequence conservation, for example, would probably require a much looser (more permissive) HMM filter threshold to achieve high sensitivity than would a family with very high sequence conservation.

Alternatively, query-independent strategies should be sufficient when the filter algorithm is highly similar to the final algorithm, as is the case with using QDB CYK to filter for either CYK or Inside. Though, of course, the algorithms are not identical, they both score conserved sequence and structure using the same CM. Other filter strategies for which a query-independent strategy is probably sufficient include using the HMM Viterbi algorithm to filter for the HMM Forward algorithm [62] and the filter used by BLAST which finds high-scoring matches between query and target to define bands for the final algorithm, banded Smith-Waterman [3].

Figure 4.1 plots the average QDB CYK and HMM Forward scores versus Inside scores for about 100 different RNA families. The QDB CYK average scores show consistently high correlation with the Inside scores (most points are clustered around the identity line). There is significantly less correlation between the HMM scores and the Inside scores. This result supports the intuition outlined above.

Simple query-independent thresholding

Above, I defined filter strategies in the query-independent thresholding group as those for which a single thresholding strategy is sufficient for all queries. The best strategy will remove all query-dependence from the scoring system, and so an E-value threshold is often a good choice of threshold. Bit scores are usually less appropriate because for many scoring systems, including those used by HMMs and CMs, the magnitude of a significant bit score varies across different families. E-values, however, provide a measurement of the statistical significance that is meant to be consistent across different families.

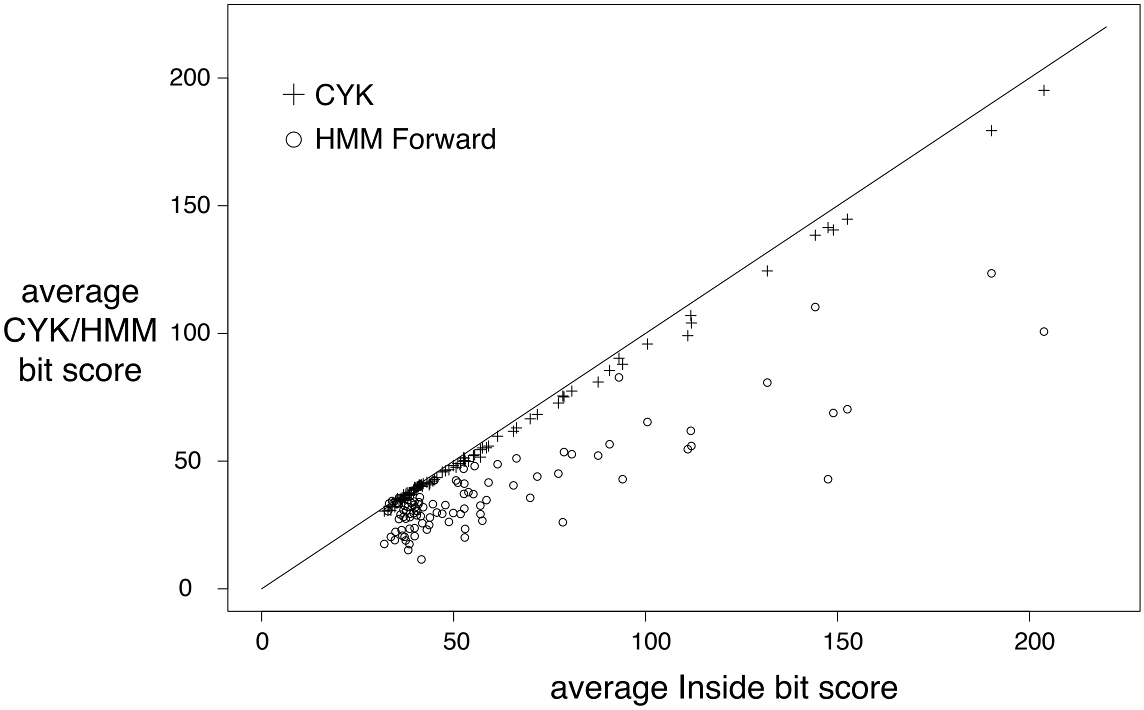


Figure 4.1: **Average QDB CYK and HMM Forward scores versus Inside scores for various RNA families.** QDB CYK scores correlate well with Inside scores. The difference between HMM Forward scores and Inside scores varies between different families. Data shown for the 95 RFAM release 9.1 [89] families with 50 or more sequences in the "seed" alignment. For each family, the seed alignment was used to build a CM using default `cmbuild` parameters. From each model, 1,000 sequences were generated and scored using the non-banded Inside, QDB CYK ($\beta = 10^{-10}$) and HMM Forward algorithms, and the average score per sampled sequence was calculated. Models were built and sequences were generated and scored using INFERNAL version 1.0 programs `cmbuild`, `cmemit` and `cmsearch`. The diagonal line is the identity line ($x = y$).

INFERNAL's filter pipeline uses an E-value of 100 times the final threshold E value for its QDB CYK filters. Later in this chapter, I show that this simple strategy performs acceptably well on a benchmark.

Filter sensitivity targeting (FST) for query-dependent thresholds

Query-dependent thresholding is more complex than query-independent thresholding because to achieve high sensitivity the filter threshold used for each query must somehow reflect the differences between the filter and final scores for that specific query. I propose a simple technique, called *filter sensitivity targeting* (FST), for determining a filter threshold that will achieve any target level of sensitivity. FST is similar to Weinberg/Ruzzo's rigorous filtering strategy in that it defines query-dependent thresholds and prioritizes sensitivity over speed, but differs in that it does not provably sacrifice zero sensitivity. A possible advantage of FST over a rigorous approach is that it may result in faster searches (by pruning away more of the database with the filter) while sacrificing an acceptably small amount of sensitivity.

It is first useful to define some terms. Here, we will assume higher scores are better, such as is the case with bit scores. Let F be the sensitivity of a filter, defined as the fraction of database hits that survive the filter (score above a filter survival score threshold T) that a non-filtered search with only the final algorithm would report as hits (score above a reporting score threshold C). The FST technique estimates the appropriate T to use to achieve F sensitivity for a search using threshold C, for a wide range of possible C values. Importantly, different values of C require different T values to achieve the same F. For example, if we assume filter scores increase with final algorithm scores, the T required to achieve $F = 0.99$ for a bit score C corresponding to an E-value of 10^{-5} will be higher (enabling a stricter filter) than if C were a lower bit score, corresponding to an E-value of 1. This is because in the latter case if the filter prunes away hits that would have final E-values between 1 and 10^{-5} it will decrease the sensitivity, whereas it will not in the former case.

FST requires as input a set of N test sequences that receive high scores with the final algorithm. For now, imagine this set of sequence is provided, I will come back to the issue of how to obtain it in the next section. To determine FST thresholds, the filter and final algorithms are used to score each sequence. From this set of scores we can approximate a T for a given C that achieves predicted sensitivity F by creating a list of the N' sequences with final scores above C, sorting them by filter score, and setting T as the k^{th} ranked filter score, where $k = \text{ceiling}(F * N')$.

If the test sequences are a representative sample of target sequences that would score above C, then in the limit of very large N and infinite database searching, using this procedure to set T will achieve sensitivity F. In other words, the larger and more representative of high-scoring final algorithm hits the set of test sequences is, the more accurate and useful this approach is.

Source of test sequences

I now return to the important issue of how to obtain the test sequences. One approach is to use known examples of homologs. Weinberg and Ruzzo essentially suggested a special case of the FST strategy to define thresholds for ML-heuristic HMM filters by using the RFAM *seed* sequences as the N sequences and requiring an F of 1.0. (They ultimately decided on using a query-independent thresholding strategy that would eliminate a predicted 99% of the target database.) The seed sequences are the sequences in the RFAM structural alignment used to build the CM. Alternatively, the RFAM *full* sequences could be used, which are all the sequences that score above an expertly curated score threshold (chosen as the score of the highest scoring obvious false positive) in a BLAST filtered CM search of the RFAMSEQ database. Because RFAMSEQ is the source of all seed sequences, the full sequences set is almost certainly a superset of the seed set.

For structural RNAs, there are two drawbacks to using known homologs as the N test sequences. First, the number of known homologs is usually small. The median number of full sequences per RNA family in RFAM release 9.1 (the largest public database of RNAs) is 50, with 100 or more sequences in 30% of the families, and 1000 or more sequences in only 6%. This is problematic because the accuracy of FST depends on N being large. For example, if F is set as 0.99 for a search with final threshold C, then at least $N' = 100$

sequences that score above C are required to derive a T using our procedure, and even then it would be defined based on a very small sample. To get a good estimate of T probably requires $N' = 1000$ or $10,000$.

Secondly, known homologs are unlikely to be a representative sample of the sequences the CM would classify as homologous with statistically significant scores. Alignments of the seed sequences are used to build and parameterize the models themselves, and as a result those sequences are a biased sample of very high scoring sequences. The full sequences have been detected using RFAM's BLAST filtering strategy and, presumably, are also a biased, high scoring sample (although it is impossible to be certain without doing a prohibitively expensive non-filtered CM search for comparison). CM parameterization has recently been significantly improved for remote homology detection, with the adaptation of informative mixture Dirichlet priors and entropy weighting from profile HMM implementations (Chapter 2). In order for a FST calibrated filter to maintain that increased sensitivity, the test sequences must include lower scoring, but still statistically significant, remotely homologous sequences.

An alternative strategy is to take advantage of the generative capacity of CMs as probabilistic models and sample the test sequences directly from the model. This approach nicely addresses the requirements of our strategy. N can be large because sampling is fast and infinitely repeatable, and sampling draws sequences from the CM's own probability distribution, which is exactly the distribution of homologs the CM is modeling. Figure 4.2 illustrates the difference in the CM score distributions of random sequences (solid lines), known (RFAM full sequences, dotted lines), and sampled sequences (dashed lines) for three RNA families: tRNA, 5S rRNA, and SRP RNA. In all three cases, the known sequences are biased towards high scores relative to the sampled sequences.

Scoring and sampling sequences

The CM CYK and Inside algorithms assign a bit score B, a log-odds score, to a target database subsequence x based on the CM model M. CYK computes:

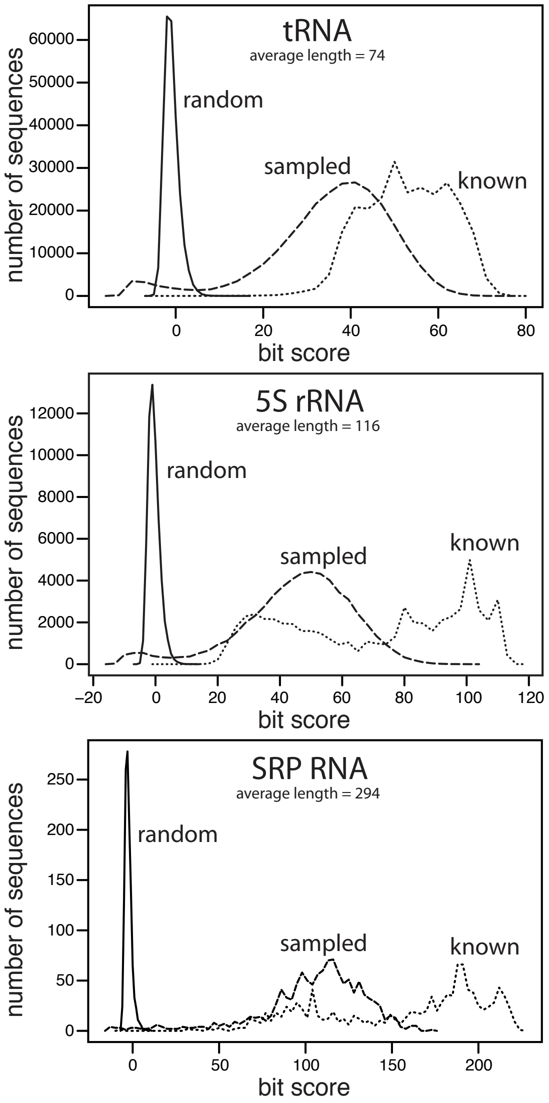


Figure 4.2: **CM score histograms of random, known, and sampled sequences for three RNA families.** CMs were built from RFAM 9.1 seed alignments using default parameters in INFERNAL 1.01 for three families: tRNA (RF00005), 5S rRNA (RF00001), and SRP RNA (RF00017). "random" sequences were generated independently for each family using a single state HMM with equiprobable emission probabilities (0.25) for the four possible RNA bases to be a specific length L, the average length of each family. The "sampled" sequences were sampled from locally configured CMs using the `cmemit` program of INFERNAL v1.01. The "known" sequences are the "seed" and "full" sequences from RFAM. All the sequences were scored using the non-banded Inside algorithm, and the scores were collated into a histogram of bit scores. The number of "random" and "sampled" sequences was set per family to be equal to the number of "known" sequences for that family: 261247 for tRNA, 57766 for 5S, and 1187 for SRP.

$$B = \log_2 \frac{P(x, \bar{\pi}|M)}{P(x|R)}$$

Where $P(x, \bar{\pi}|M)$ is the probability of the most likely parse tree (alignment) $\bar{\pi}$ of the target subsequence according to the CM.

Inside computes:

$$B = \log_2 \frac{P(x|M)}{P(x|R)}$$

Where $P(x|M)$, the probability of x given the CM, which is calculated by summing the probability of all possible paths π through the model that generate x, that is:

$$P(x|M) = \sum_{\pi} P(x, \pi|M)$$

For both CYK and Inside, $P(x|R)$ is the probability of the target sequence given a *null hypothesis* model R of the statistics of random background sequence. The null model is a simple one-state hidden Markov model (HMM) that says that random sequences are i.i.d. sequences with a specific residue composition, which by default is equiprobable across the four RNA nucleotides. The null model score is calculated as simply: $P(x|R) = 0.25^L$ for a sequence of length L. Because the null model score depends only on the length of the target sequence, and not the sequence itself, B increases monotonically with $P(x, \pi|M)$ for a constant L. This means that as the probability that a sequence and particular parse tree was generated from the CM increases, so does its score. This suggests that sampling from the distribution defined by: $P(\mathrm{seq}, \pi|\mathrm{CM})$ should yield high scoring sequences. This is confirmed for three families in Figure 4.2 for which the scores of the vast majority of sampled sequences are significantly better than random.

Sampling a sequence from a CM is a recursive procedure that begins at the root state and samples a parse tree of states (π) and sequence residues, until all branches of the tree terminate at end states. During the procedure, the choice of next state is determined by the current state's transition probability distribution. When singlet or base-pair emitting states are visited a single residue or base-pair residue, respectively, is sampled from the state's emission probability distribution. The emitted sequence associated with a parse tree is generated from outside to inside (as opposed to from left to right from an HMM, see Chapter 1) and can be read by starting at the top left of the parse tree and following the yield of the tree counterclockwise, as depicted in Figures 1.11 and 1.12.

A CM can be locally or globally configured [138, 192] (Chapter 1). In global configuration, the only way to enter and exit the model is through the root state and end states, respectively. In local configuration, begins and ends are possible from any internal node of the model. Further, when a local end takes place, a special insert state is visited that can emit additional sequence. Local ends allow CMs to tolerate insertions or deletions of entire substructures, increasing sensitivity for remote homology detection in some cases.

Practical limits on filter thresholds

Sometimes it is reasonable to define a maximally useful filter survival threshold T_{max}, and to use $min(T, T_{max})$ for all T derived from the FST procedure. Specifically, this is useful if T_{max} is chosen so that using $\min(T, T_{max})$ slows down the search only by a negligible amount, because this means that using T_{max} will possibly increase the sensitivity of the filter (because $T_{max} < T$) at a negligible cost in running time.

For example, if the FST procedure defines T as 50 bits but using a $T = T_{max}$ of 30 bits will only increase the running time of an hour long search by 1 second, then it makes sense to use T of 30 bits. It will only extend the search a negligible amount of time while allowing the final algorithm (or next filter round algorithm) to additionally score all the hits between 30 and 50 bits, some of which may be above final (or next filter round) threshold.

INFERNAL uses a query-independent method for limiting filter thresholds by setting a single, minimally useful target survival fraction S_{min}. This approach is used instead of enforcing T_{max} with a single query-independent bit score threshold because, as mentioned earlier, bit score ranges for true homologs vary significantly between between different families. Survival fractions are based on E-values and are consequently more consistent across families.

More specifically, S_{min} is related to T_{max} as follows. The running time t of a filtered search is: $t = t_f + t_m * S$, where t_f and t_m are the time required to run the filter and the final algorithm, respectively, on the full target database and S is fraction of the database that survives the filter. The survival fraction S is a function of T: as T increases, S decreases, and vice versa. Because t is directly affected by S, one way to enforce a T_{max} is to use a single query-independent S_{min}, and converting it to a T_{max} for each query. This requires a way of converting between S and T, which is straightforward because E-values are available: $S = \frac{EH}{L}$, where E is the E-value for T using the filter scoring algorithm, L is the database size, and H is the average length of a surviving fraction of the database due to a filter hit above T.

The appropriate choice of S_{min} is likely to be highly dependent on the ratio of running times of the filter and final scoring algorithms. I investigate reasonable S_{min} values to use for HMM filtered searches based on empirical performance in a benchmark below.

4.4 Evaluation

I used the RFAM-based benchmark described in Chapter 3 to measure the speed, sensitivity and specificity of CM-based searches using different filtering methods. (What follows is a nearly identical description of the benchmark that appears in Chapter 3.) Briefly, the benchmark was constructed as follows. The sequences of the seed alignments of 503 RFAM (release 7) families were single linkage clustered by pairwise sequence identity, and separated into two clusters such that no sequence in one cluster is more than 60% identical to any sequence in the other. The larger of the two clusters was assigned as the query (preserving their original RFAM alignment and structure annotation), and the sequences in the smaller cluster were assigned as true positives in a test set. We required a minimum of five sequences in the query alignment. 51 RFAM families met these criteria, yielding 450 test sequences which were embedded at random positions in a 10 Mb "pseudogenome" generated by a 15-state fully connected hidden Markov model (HMM) trained by Baum-Welch expectation maximization [53] on genome sequence data from a wide variety of species. Each of the 51 query alignments was used to build a CM and search the pseudogenome in local mode, a single list of all hits for all families were collected and ranked, and true and false hits were defined (as described in Nawrocki and Eddy [189]).

The minimum error rate (MER) ("equivalence score") [206] was used as a measure of benchmark performance. The MER score is defined as the minimum sum of the false positives (negative hits above the threshold) and false negatives (true test sequences which have no positive hit above the threshold), at all possible choices of score threshold in the ranked list of all hits from the 51 searches. The MER score is a combined measure of sensitivity and specificity, where a lower MER score is better. We calculate two kinds of MER scores. For a *family-specific* MER score, we choose a different optimal threshold in each of the 51 ranked lists, and for a *summary* MER score, we choose a single optimal threshold in the master list of all hits. The summary MER score reflects the performance level for a large scale analysis of many families because it demands a single query-independent E-value reporting threshold for significance. The family-specific MER score indicates the performance that could be achieved with manual inspection and curation of the hits in each family to determine family specific score E-value thresholds. The design of the default filtering pipeline used by INFERNAL version 1.01 was based on these benchmark results, as discussed below.

Determining the best-performing final algorithm

I determined the most sensitive CM search strategy irrespective of speed to be used as the final algorithm in the filtering pipeline, and to serve as a best-case performance against which to judge the filtered searches. I tested the CM Inside and CYK algorithms, both with and without query-dependent bands (QDBs). For the banded runs I used a $\beta = 10^{-15}$ tail loss probability for QDB calculation that previous work has indicated sacrifices essentially zero sensitivity [189]. As shown in Table 4.3, using the banded Inside algorithm resulted in the lowest summary and family specific MER of the four methods tested (rows 1-4 in Table 4.3). Interestingly, banded Inside outperforms non-banded Inside (row 1 in Table 4.3); this is because enforcement of the bands eliminates about a dozen high scoring false positive hits that drive up the MERs. Based on this result, I defined banded Inside with $\beta = 10^{-15}$

as the final algorithm in the default filter pipeline.

Performance of QDB CYK filtering

I tested the performance of QDB CYK using various β values as the only filter in a single filter pipeline using the simple, query-independent filter thresholding technique of setting the QDB CYK E-value threshold as 100 times the final algorithm E-value threshold (rows 6-9 in Table 4.3). QDB CYK filtering with $\beta = 10^{-10}$ results in about a four-fold speedup with a negligible loss in sensitivity relative to a non-filtered run (row 3). Further, this strategy yields significantly better performance than running non-filtered CYK with identical $\beta = 10^{-10}$ (row 5), while only requiring about 10% longer to run. This clearly suggests it is more useful to use QDB CYK as a filter for Inside than as the final scoring algorithm as we did previously [189].

Performance of HMM filtering

Next, I addressed FST parameterization. What is the best value to use for the F parameter, which specifies the fraction of sequences allowed below the filter score threshold? The black solid points in Figure 4.3 shows the benchmark running time of FST calibrated HMM filtered searches versus MER for different values of F. The choice of F is a tradeoff of accuracy for speed. We chose a default of $F = 0.993$ as a reasonable value that obtains a speedup of about 25-fold with a minimal loss of accuracy (Figure 4.3 and Table 4.3, row 3 compared to 13).

How much does FST calibrated HMM filtering impact sensitivity and specificity? Tables 4.3, 4.4 and 4.5 demonstrate FST's impact on benchmark performance. Table 4.4 shows that the actual sensitivity (actual F) achieved by the filter on our benchmark is 0.924. The summary and family MER for an HMM filtered search using $F = 0.993$ and $S_{min} = 0.02$ are 144 and 134 (Table 4.3 row 10) up from 130 and 109 for a non-filtered search (row 3).

How does using FST to determine filter thresholds compare to using a query-independent target survival fraction S as a thresholding method? Figure 4.3 plots benchmark summary MER versus running time for different filtering strategies: FST with various F values and

		filtering with HMM				filtering with CM		post-filtering		summary	family	time
		algorithm	FST F	S_{min}	target S	algorithm	QDB β	algorithm	QDB β	MER	MER	(min/Mb/query)
	1	-	-	-	-	-	-	Inside	-	150	115	280.60
	2	-	-	-	-	-	-	CYK	-	156	133	102.16
	3	-	-	-	-	-	-	Inside	10^{-15}	130	109	89.13
	4	-	-	-	-	-	-	CYK	10^{-15}	153	130	30.60
	5	-	-	-	-	-	-	CYK	10^{-10}	154	132	21.97
	6	-	-	-	-	CYK	10^{-13}	Inside	10^{-15}	131	114	30.08
	7	-	-	-	-	CYK	10^{-10}	Inside	10^{-15}	130	114	24.24
	8	-	-	-	-	CYK	10^{-7}	Inside	10^{-15}	134	118	17.42
	9	-	-	-	-	CYK	10^{-4}	Inside	10^{-15}	142	127	10.18
	10	Forward	-	-	0.02	-	-	Inside	10^{-15}	160	149	0.95
	11	Forward	-	-	0.10	-	-	Inside	10^{-15}	156	142	3.16
	12	Forward	-	-	0.25	-	-	Inside	10^{-15}	149	131	7.46
	13	Forward	0.993	-	-	-	-	Inside	10^{-15}	145	135	3.73
	14	Forward	0.993	0.01	-	-	-	Inside	10^{-15}	144	134	3.84
	15	Forward	0.993	0.02	-	-	-	Inside	10^{-15}	143	133	3.99
	16	Forward	0.993	0.10	-	-	-	Inside	10^{-15}	143	132	5.64
	17	Forward	-	-	0.02	CYK	10^{-10}	Inside	10^{-15}	161	154	0.68
INFERNAL 1.01 default	18	Forward	0.993	0.02	-	CYK	10^{-10}	Inside	10^{-15}	143	134	1.26
	19	-	-	-	-	-	-	HMM Forward	-	214	204	0.39

Table 4.3: **Benchmark MER and timing statistics for different search strategies.** Each search strategy is defined by the algorithms and parameters used by zero, one or two filtering stages and a final post-filtering stage. Under "filtering with HMM": "algorithm" lists if an HMM filter is applied first ("Forward"), or not at all ("-"); "FST F" lists the target sensitivity F used for FST threshold calibration, or "-" if FST was not used; "S_{min}" is the minimum predicted survival fractions used to set filter thresholds (potentially overriding the FST calibrated thresholds); "target S" shows the single, target predicted survival fraction used for all models in non-FST HMM filtering strategies. Under "filtering with CM": "algorithm" lists if a CM "CYK" filter is applied (only on the surviving subsequences from the HMM filter if one was used) or not at all ("-"), and "QDB β" lists the tail loss probability used to calculate bands for the algorithm. Under "post-filtering": "algorithm" lists the final algorithm used for scoring subsequences that survive the <= 2 filtering stages; "QDB β" lists the tail loss probability for the band calculation for the final algorithm. The sensitivity and specificity of each strategy is summarized by "summary MER" and "family MER" as explained in the text. Lower MERs are better. "min/Mb/query" list minutes per Mb (1,000,000 residues) of search space per query model used to search. The benchmark contains 51 query models and 20 Mb of search space (both strands of the 10 Mb pseudogenome) as explained in the text.

target S thresholding for various S values. Target S thresholding is faster than FST for achieving MER values down to about 160, but FST is faster if lower MERs are desired. Tables 4.3, 4.4, and 4.5 also compare FST with target survival fraction target S methods.

Is FST robust to a wide range of final E-value thresholds (C)? With FST, the filter threshold increases as the final threshold increases (becomes more strict), increasing the filter's efficiency while theoretically maintaining the same level of sensitivity, F. Table 4.5 shows the effect of varying the final E-value threshold on the sensitivity and speed of FST calibrated HMM filters on the benchmark dataset. As E decreases, the sensitivity remains relatively constant while the speedup increases, until $E = 10^{-3}(1e-3)$ is reached, at which point sensitivity begins to decrease, suggesting FST is less reliable for stricter thresholds. Fortunately, enforcing $S_{min} = 0.02$ corrects this problem. This is because many FST calibrated thresholds for final thresholds $E < 10^{-3}$ correspond to $S < 0.02$, so enforcing S_{min} lowers the filter threshold and increases sensitivity.

Performance of the HMM and QDB CYK filter pipeline

Is it useful to combine a FST calibrated HMM filter and a QDB CYK filter? As mentioned above, FST calibrated HMM filters with $F = 0.993$ and $S_{min} = 0.02$ result in about a 25-fold speedup and QDB CYK filters with $\beta = 10^{-10}$ result in about a four-fold speedup. Combining these two filtering strategies by running the HMM first, searching the surviving fraction with QDB CYK, and using Inside only on the fraction that survives both, results in about a three-fold speedup relative to only using HMM filters with an acceptably small loss of accuracy (Figure 4.4 and rows 15 and 18 of Table 4.3). This strategy is about 70 times faster than the top performing strategy, non-filtered Inside search with $\beta = 10^{-15}$ at a small cost to sensitivity. And it is more than 200 times faster than non-banded Inside, while achieving a lower summary MER. These results were the motivation for making this two-stage filtering pipeline the default filtering strategy in INFERNAL version 1.01.

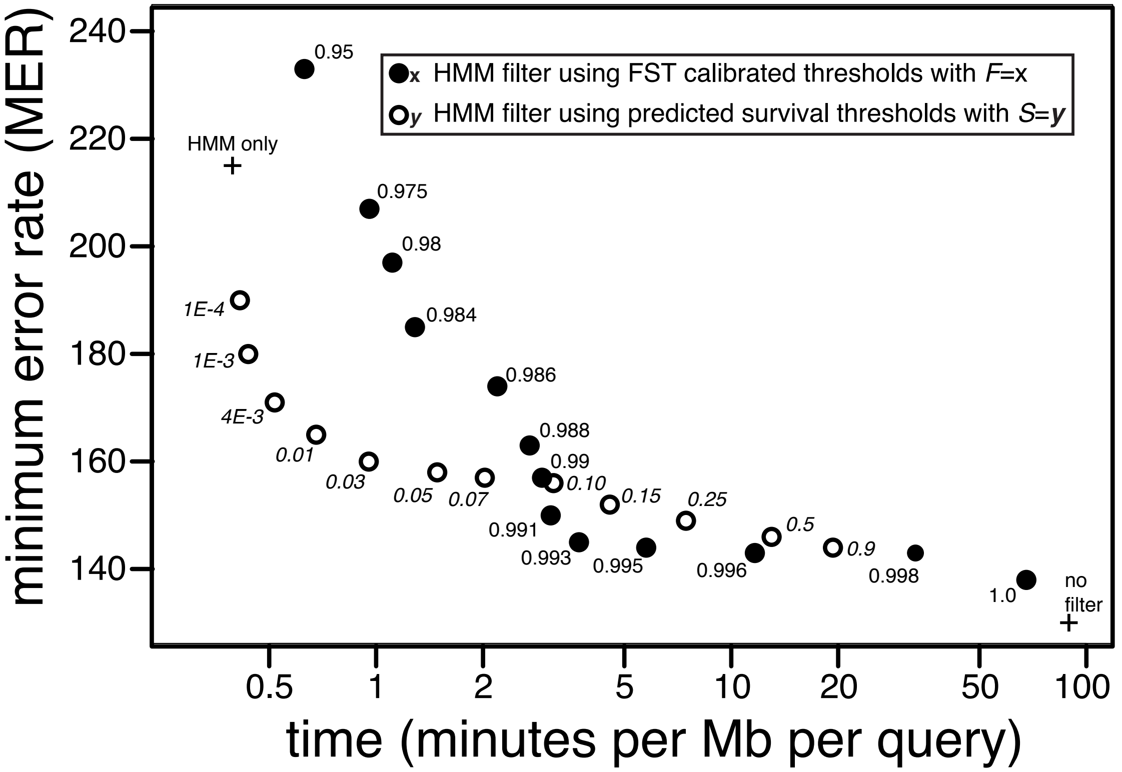


Figure 4.3: **MER versus time for the benchmark.** Solid black points show benchmark performance for HMM filtered searches using query-dependent FST calibrated filter thresholds with target sensitivity $F = x$, with x labeled per point. Open-circle points show benchmark performance for HMM filtered searches using a single, query-independent, target survival threshold of $S = y$, with y labeled per point. There are two additional "+" points: "HMM only": HMM Forward algorithm as the final scoring algorithm (with no filters); "no filter" Inside with QDB ($\beta = 10^{-15}$) as the final algorithm. For the FST searches $S_{min} = 0$. All searches performed with INFERNAL 1.01. Note that the x-axis is in log-scale.

Predicted survival fraction (S) range for FST HMM filter ($F = 0.993$, no S_{min})	# query	# test	non-filtered # found	FST HMM filtering ($F = 0.993$, no S_{min})		FST HMM filtering ($F = 0.993$, $S_{min} = 0.02$)		Non-FST HMM filtering single threshold ($S = 0.02$)	
				actual F	speedup	actual F	speedup	actual F	speedup
(no filter) $S = 1.0$	2	52	43	1.000	1.0	1.000	1.0	0.581	70.9
$1.0 > S >= 0.1$	11	98	76	0.987	10.6	0.987	10.6	0.974	79.3
$0.1 > S >= 1e-2$	17	165	135	0.919	52.8	0.919	51.0	0.911	88.6
$1e-2 > S >= 1e-3$	8	54	48	0.854	150.8	0.854	72.1	0.854	80.9
$1e-3 > S >= 1e-4$	7	53	31	0.807	185.0	0.871	103.0	0.871	121.2
$1e-4 > S >= 1e-5$	4	6	4	1.000	90.4	1.000	57.8	1.000	67.6
$1e-5 > S > 0$	2	22	4	0.750	265.5	0.750	121.6	0.750	143.6
all	51	450	341	0.924	23.9	0.930	22.3	0.871	93.4

Table 4.4: **Comparison of filter sensitivity and benchmark acceleration for queries with different FST predicted filter survival fractions.** The 51 query benchmark families were categorized based on the predicted survival fraction S of a FST filtered HMM benchmark search with final reporting threshold $E = 1$. FST was performed with $F = 0.993$ and no S_{min} value. Column 1 lists the survival fraction category; the first row "no filter $S = 1.0$" corresponds to queries for which FST indicates $S >= 1.0$ so the HMM filter is turned off. The next three columns list the number of query families ("# query"), total number of test sequences ("# test"), and number of the test sequences that the final algorithm scores with $E <= 1$ ("non-filtered # found"). The remaining six columns compare three filtering strategies: FST HMM filtering using $F = 0.993$ and no S_{min} value (this is row 13 in Table 4.3), FST HMM filtering with $F = 0.993$ and $S_{min} = 0.02$ (row 15 in Table 4.3), and non-FST filtering setting thresholds that give a predicted $S = 0.02$ (row 10 in Table 4.3). For each strategy: "actual F" lists the filter sensitivity per category, the fraction of the test sequences the final algorithm scores $E <= 1$ that also pass the filter score threshold and survive the filter; "speedup" lists the per-category acceleration of a filtered search versus a non-filtered search in the benchmark. Only HMM filters were used (no CYK filters). The final algorithm used was Inside with QDBs calculated with $\beta = 10^{-15}$.

final algorithm E-value reporting threshold (database size = 20 Mb)	corresponding database size for $E = 1$ threshold	non-filtered # found	FST HMM filtering ($F = 0.993$, no S_{min})		FST HMM filtering ($F = 0.993$, $S_{min} = 0.02$)		Non-FST HMM filtering single threshold ($S = 0.02$)	
			actual F	speedup	actual F	speedup	actual F	speedup
$E = 1e-5$	2 Tb	250	0.880	108.5	0.984	66.3	0.980	89.0
$E = 1e-4$	200 Gb	268	0.892	91.9	0.974	60.8	0.966	89.0
$E = 1e-3$	20 Gb	285	0.902	73.1	0.965	53.6	0.940	89.0
$E = 1e-2$	2 Gb	298	0.920	38.2	0.960	32.5	0.920	89.0
$E = 1e-1$	200 Mb	324	0.926	29.2	0.954	26.1	0.904	89.0
$E = 1$	20 Mb	341	0.924	23.9	0.930	22.3	0.871	89.0
$E = 10$	2 Mb	355	0.913	15.4	0.916	14.8	0.851	89.0
$E = 100$	200 Kb	368	0.910	4.9	0.913	4.8	0.834	89.0
$E = 1000$	20 Kb	391	0.910	2.9	0.910	2.9	0.800	89.0

Table 4.5: **Comparison of filter sensitivity and benchmark acceleration for different final algorithm reporting E-value thresholds.** Column 1 lists E, the final algorithm reporting E-value threshold in the benchmark (20 Mb, two strands of a 10 Mb pseudogenome). Column 2 lists the database size in which a score with E-value E from column 1 corresponds to $E = 1$. Column 3 lists the number of the 450 test sequences the final algorithm scores with an E-value $< E$ from column 1. The remaining six columns compare the same three filtering strategies as in Table 4.4 by filter sensitivity ("actual F") and acceleration of a filtered search versus a non-filtered search ("speedup"). Filter sensitivity is the fraction of test sequences the final algorithm scores with an E-value $< E$ from column 1 that also pass the filter score threshold and survive the filter. Only HMM filters were used (no CYK filters). The final algorithm used was Inside with QDBs calculated with $\beta = 10^{-15}$.

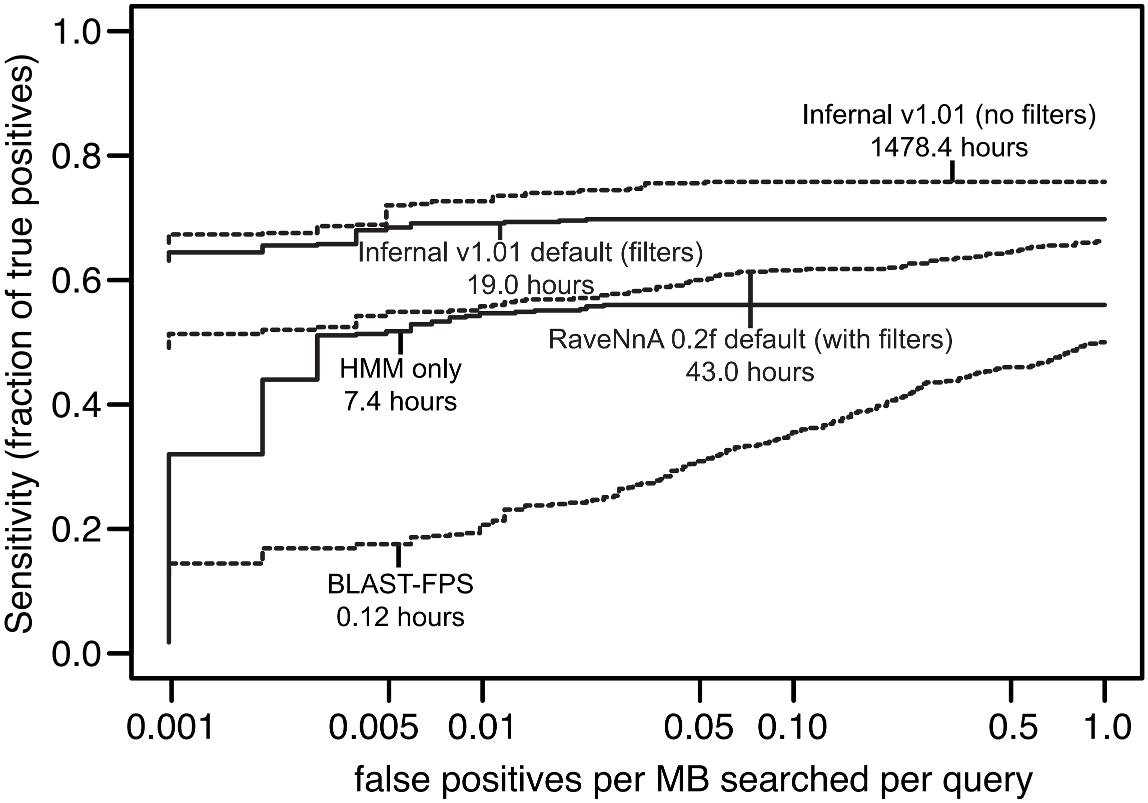


Figure 4.4: **ROC curves for the benchmark.** Plots are shown for RaveNnA 0.2f searches, family-pairwise-searches (FPS) with blastn, and for infernal 1.01 non-filtered CM searches, default filtered searches, and HMM only searches.

4.5 Implementation

INFERNAL's `cmcalibrate` program estimates HMM filter thresholds for achieving a pre-specified sensitivity F (0.993, by default) using the FST procedure, as described below. A set of these thresholds are then saved to the CM text file, and read by the `cmsearch` program. When a search is executed, the appropriate HMM filter threshold T is chosen based on the final score threshold being enforced by the search.

More specifically, the `cmcalibrate` program performs the following steps to estimate HMM filter thresholds for a pre-specified F:

1. Sample N sequences from the CM. By default $N = 10,000$, but it can be set by the user.

2. Score the N sequences using the CP9 HMM Forward algorithm, the CM CYK algorithm and the CM Inside algorithm.

3. For each CYK score C_i for $i = 1$ to $i_{max} = \text{floor}(0.90 * N)$, determine the N' sequences with CYK scores $>= C'$ and the corresponding N' Forward scores. Set T_i as the k^{th} ranked best Forward score, where $k = \text{ceiling}(F * N')$. Save (T_i, C_i) in a new list.

4. Repeat step 3 using Inside scores instead of CYK scores.

This procedure creates two lists of (T, C) pairs of size i_{max}, one for CYK and one for Inside. Each list is then pruned to only include a representative subset of pairs to avoid the necessity of storing all i_{max} pairs for use in `cmsearch`. The representative subset is defined such that no two C values C_1, C_2 ($C_1 < C_2$) with corresponding E-values E_1 and E_2 ($E_1 > E_2$) follow $E_2 - E_1 < (0.1 * E_1)$. This subset usually includes at most a few hundred points for the default N of $10,000$.

Figure 4.5 shows data for this procedure for three anecdotal RNA families using default values $N = 10,000$ and $F = 0.993$. Each small point represents a sequence, with x-coordinate equal to HMM score and y-coordinate equal to Inside score. The larger points are the representative subset of the (T, C) pairs. For each large (T, C) point, the fraction of small points with $y > C$ that have $x < T$ is $1 - F = 0.007$, these represent the 0.7% of sequences that a non-filtered search with final threshold C would report that a filtered search using filter threshold T will miss.

For an execution of `cmsearch` with final algorithm CYK or Inside with reporting threshold C', T from the CYK/Inside (T, C) pair in the CM file with the maximum $C < C'$ is selected and T is set as the HMM Forward filter threshold bit score. If T corresponds to a predicted survival fraction less than S_{min} (0.02 by default) then it is replaced with the T' that corresponds to S_{min} as described above in "Practical limits on filter thresholds". Additionally, if T corresponds to a predicted survival fraction S greater than a maximum value (by default 0.5), the HMM filter is turned off, and not used for the search.

In step 2 of the `cmcalibrate` procedure, the CYK and Inside scores are determined using an HMM banded version of the CYK and Inside search algorithms. Using HMM banding, which is described in Chapter 8, results in a significant speedup while very rarely affecting the score calculated by the algorithm by any appreciable amount (i.e. when there is a score difference, it is very often less than one bit, data not shown). Steps 3 and 4 and the definition of the representative subset are implemented in the `get_hmm_filter_cutoffs()` function in the `cmcalibrate.c` file of INFERNAL version 1.0 and 1.01.

This procedure could potentially be improved in several ways. First, F need not be prespecified. Instead, the lists of scores could simply be kept in a file and the FST procedure described in the text could be used to determine a T for any given pair of C *and* F. Also, the manner in which the (T, C) pairs are saved could be simplified, for example, by saving all pairs at one bit C increments across the observed range of C values. Investigating these modifications is left to future work.

By default, the QDB CYK filter thresholds are set in a query-independent manner by `cmsearch` as the bit score corresponding to 100 times the E-value of the final algorithm reporting threshold. This is true except when the predicted number of DP calculations required to run the final round algorithm on the fraction of the database surviving all filters is less than 3% the total number of DP calculations required for the entire search (filter

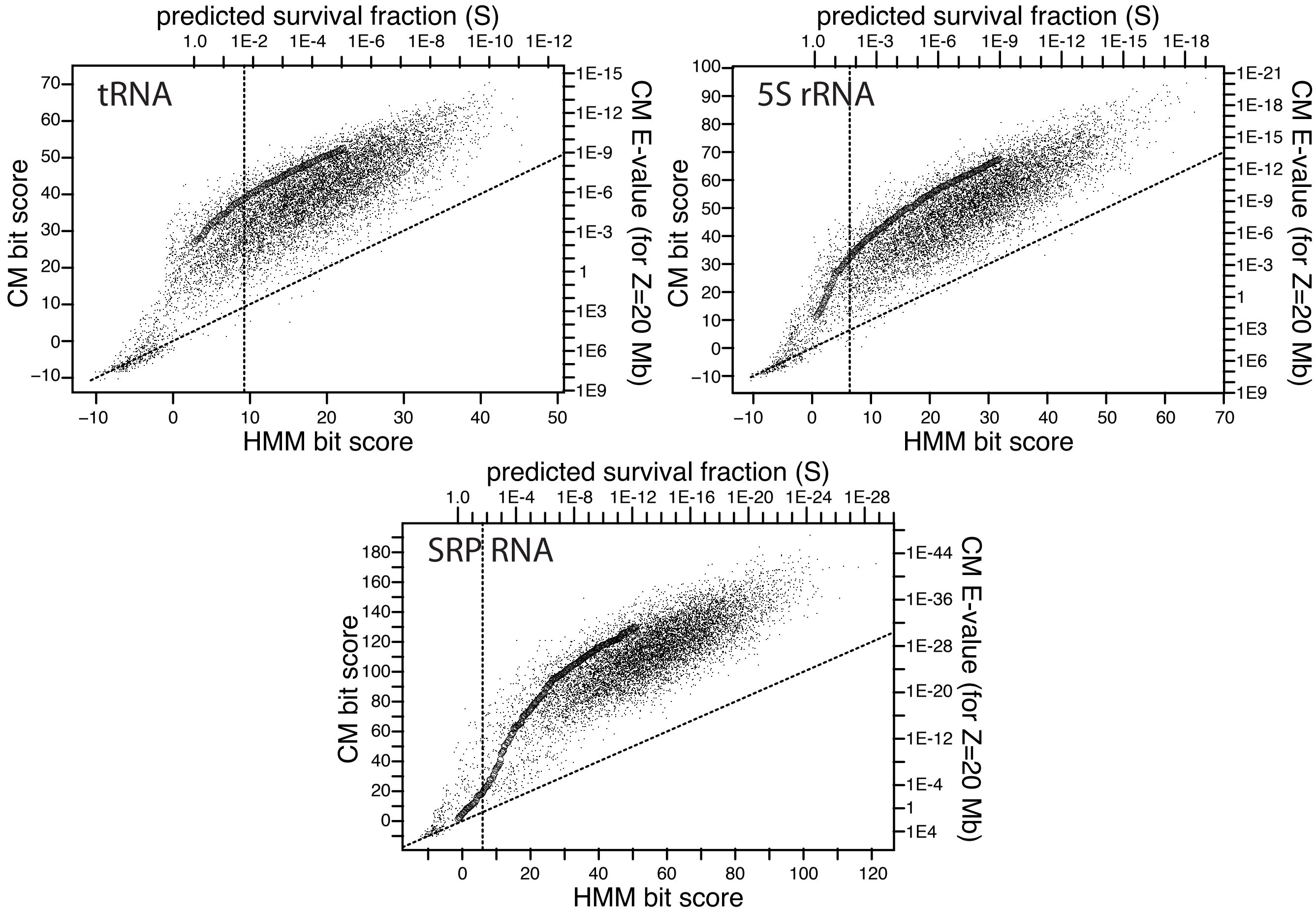


Figure 4.5: **CM Inside scores versus HMM Forward scores during FST calibration.** Complete data for the FST `cmcalibrate` calibration with $N = 10,000$ and $F = 0.993$ of three RFAM 9.1 families: 5S rRNA, tRNA, and RNase P (RF00001, RF00005, RF00011). Each sequence is represented as a small black point with x-coordinate equal to its HMM Forward score, and y-coordinate equal to its CM Inside score. Larger, open circle points indicate the representative set of saved filter survival threshold T and CM reporting score threshold C pairs saved to the CM file and used to determine thresholds during searching. The diagonal line is Forward=Inside score, and the vertical lines show the Forward scores that would yield a predicted survival fraction of 0.02 (the default S_{min} value). The CMs used for sampling and scoring, and HMMs used for scoring were locally configured as described in Chapter 1.

plus final calculations). In this case, the QDB CYK filter threshold is relaxed (made less strict) so that the final algorithm will perform exactly a predicted 3% of the required DP calculations. The reasoning explained above in "Practical limits on filter thresholds" applies here: making the QDB CYK filter threshold less strict in this case will only potentially increase sensitivity, while slowing down searches a small amount because the number of predicted calculations is increasing by at most 3%. The determination of filter thresholds is also explained, using example searches, in the INFERNAL user's guide [192].

4.6 Conclusion and future directions

As measured by our benchmark, the two-stage filter pipeline used by INFERNAL brings the average search time down 70-fold from about 90 to about 1.25 minutes per Mb when using the optimally performing QDB Inside with $\beta = 10^{-15}$ as the final algorithm (Table 4.3). The filters do sacrifice sensitivity as measured by the increase in the summary and family MERs by 13 and 25 respectively, but, temporarily at least, this is acceptable given the gain in speed. If the benchmark results hold for the general case, to run the 1371 RFAM release 9.1 families against the entire human genome would require about 20 CPU years (compared to 1500 CPU years for a non-filtered search), which means further acceleration remains an important goal of INFERNAL development.

There are several ways to make INFERNAL faster. One is to accelerate the current filters the HMM and QDB CYK algorithms. A new version (3.0) of the HMMER software package is in beta testing [62], which includes significantly faster HMM search algorithm implementations than those in INFERNAL 1.01. We plan to incorporate those implementations within INFERNAL for filtering. But the HMM filters are not the rate limiting step in CM searches - the time required to run the HMM filter on our benchmark is about one third the total time of the search (Table 4.3 rows 18 and 19). So the maximum we can gain from a faster implementation of the HMM filters, assuming consistent filter sensitivity, is about 33%. Alternatively, we could use different filters that are faster, more sensitive, or both. The structure-based filters introduced by Weinberg and Ruzzo [263] and Sun and

Buhler [247] are especially promising, because they offer greater speed than QDB CYK, INFERNAL's current structure-based filter. The FST approach could be used to set appropriate thresholds for other filtering strategies. Finally, we could attempt to write faster implementations of Inside, the best performing final algorithm. Ongoing work on HMMER 3 has suggested that optimizing the dynamic programming search algorithm implementations using single-instruction multiple data (SIMD) paralellism can yield significant speedups.

Chapter 5

Computational identification of functional RNA homologs in metagenomic data[1]

An important step in analyzing a metagenomic sequence dataset is identifying functional sequence elements. This is a prerequisite for determining important properties of the environment the sequence data were sampled from, such as the metabolic processes and organismal diversity present there. At least initially, functional sequence element identification is addressed computationally. One class of elements, functional noncoding RNA elements, are especially difficult to identify because they tend to be short, lack open reading frames, and sometimes evolve rapidly at the sequence level even while conserving structure integral to their function [55, 108, 128, 165, 248, 270].

Functional RNA elements include both RNA genes (genes transcribed into functional untranslated RNA) and cis-regulatory mRNA structures. RNA elements play many roles. Ribosomal RNAs (rRNAs) and transfer RNAs (tRNAs) are well known and universally present in all cellular life. Bacteria, archaea, and viruses, the organisms predominantly targeted by current metagenomics studies, also use numerous small RNA (sRNA) genes for translational and posttranslational regulation [96], as well as many cis-regulatory RNAs such

---

[1] This chapter was submitted to be eventually published as a book chapter in "Metagenomics: a new science at the interface of genomics, microbiology, and ecology" by American Society for Microbiology (ASM) press, edited by Claire Fraser-Liggett, Jacques Ravel and Jo Handelsman. The submitted version is reprinted here with the following modifications: two tables and one figure are omitted, they have been relocated to Chapter 1 as Tables 1.1 and 1.2 and Figure 1.9.

as riboswitches (structural RNAs that respond to binding small molecule metabolites and control expression of nearby genes [252, 272]). Archaea have numerous small nucleolar RNAs (snoRNAs) homologous to eukaryal snoRNAs that direct site-specific RNA methylation and pseudouridylation [7]. Many eukaryotes make extensive use of RNA regulatory mechanisms via pathways related to RNA interference (RNAi) and microRNAs (miRNAs) [4, 10], and these will become relevant to metagenomic studies as they begin to target eukaryotes. These are only a small list of the most abundant classes of functional RNA elements. There are many other examples: catalytic introns, eukaryotic spliceosomal RNAs, RNA components of ribonucleoprotein complexes including telomerase, ribonuclease P, and the signal recognition particle, and more.

It is striking that several of the large classes of RNAs just mentioned were either discovered recently (miRNAs, riboswitches) or have had their numbers greatly expanded by recent analyses (sRNAs, snoRNAs). This highlights the relative difficulty in discovering and analyzing functional RNA sequences, compared to more well-developed methodologies for discovering and analyzing protein coding sequences. It hints that other RNAs likely remain undiscovered [56].

In this chapter, we will discuss methods for computationally identifying homologs of known RNA elements, such as riboswitches and sRNA riboregulators. Another problem of great interest is to discover entirely *new* functional RNA elements by computational sequence analysis [165, 211, 257, 270], but as recent reviews have discussed, *de novo* RNA discovery ("genefinding") methods [6, 98, 180] currently have high false positive rates that are difficult to estimate statistically; these methods are unsuited to high-throughput analysis, and instead need to be used as screens that can be followed by experimental confirmation [44]. In contrast, RNA homology search programs are sufficiently reliable, and backed by sufficiently well-curated databases of known RNA sequence families, that automated large-scale computational metagenomic analyses are feasible.

5.1 Exploiting conserved structure in RNA similarity searches

Protein homology search by amino acid primary sequence comparison is powerful. At the amino acid level, BLASTP has no trouble detecting significant similarity down to about 25-30% amino acid sequence identity. Many protein coding regions conserve this level of similarity even across the deepest divergences in the tree of life amongst archaea, bacteria, and eukaryotes.

In contrast, RNA homology search by nucleotide primary sequence comparison is much less able to detect distant RNA homologies. BLASTN typically requires about 60-65% sequence identity to detect a statistically significant similarity for RNAs of typical length. Although some RNAs are very highly conserved over evolution (notably large and small subunit ribosomal RNAs, which are readily detected by sequence comparison in all species; the so-called human "ultraconserved" regions included regions of rRNA [11]), this is not the rule. Many functional RNA homologies are undetectable at the primary sequence level in cross-phylum comparisons (such as nematode/human or fly/human), because weakly or moderately conserved nucleic acid sequences can diverge to the 65% identity level in just a few tens of millions of years.

A striking example of this difference comes when searching for homologs of the components of some ribonucleoprotein (RNP) complexes. It is not uncommon to detect homologs of the protein components but not the RNA components of complexes such as the signal recognition particle, ribonuclease P, small nucleolar RNPs, and telomerase. The interpretation upon finding only the protein component is usually (and almost certainly correctly) that the RNP complex is present in the organism, but the RNA component(s) are too difficult to detect. For example, the probable presence of small nucleolar RNAs in archaea could be inferred from the presence of homologs of snoRNP protein components like fibrillarin well before snoRNA homologs were discovered [5, 199]. A similar situation can occur when identifying homologous cis-regulatory RNA elements (such as riboswitches) for clearly homologous coding genes.

Table 1.1[2] shows some specific anecdotal examples. These data are fairly typical of searching databases with protein versus RNA queries. They demonstrate two key points about the relative difficulty in detecting homologs of functional RNAs. First, notice that for the protein coding genes, the statistical significance of the similarity (the E-value) is always much better (lower and more significant) when comparing their amino acid sequences rather than when comparing their DNA sequences, highlighting the additional statistical power inherent in searches at the amino acid level. This is the reason for the recommended practice of always comparing protein sequences at the amino acid level [207]. Second, notice that RNA components are usually much shorter than the coding sequence of the protein components, further compromising statistical signal and the ability of primary sequence analysis (BLASTN) to resolve homologous relationships from background. (To enable reproduction of these results, the accessions for the sequence data used in these searches is provided in Table 1.2[3].)

What can be done about the weakness of primary sequence based methods for detecting functional RNAs? Some other source of statistical signal needs to be found for functional RNAs. Such a signal exists: many (though not all) functional RNAs conserve a distinctive RNA secondary structure.

Of course, proteins conserve structure more than sequence too. Remote homologies invisible to primary sequence analysis often become apparent when a protein's three-dimensional structure is solved. What makes RNA secondary structure constraints of particular utility for computational sequence analysis is their simplicity and relative contribution of statistical signal. RNA base-pairs induce strong pairwise correlations in RNA sequences. These correlations may be sufficiently obvious that they are apparent even to the naked eye (analogous to the obviousness of ORFs for coding gene analysis). RNA consensus secondary structures have been accurately inferred by manual "comparative sequence analysis" alone [107, 181, 202].

How much extra information does RNA secondary structure conservation contribute in

---

[2] This table appears in Chapter 1.

[3] This table appears in Chapter 1.

addition to primary sequence conservation? We can ask this question rigorously in the context of homology search applications, across a range of different types of RNAs. Figure 1.9[4] shows the average score of search models for about 100 RNA sequence families, comparing models of sequence conservation alone ("profile hidden Markov models", profile HMMs) to models of sequence plus RNA secondary structure conservation ("covariance models", CMs). These consensus models are discussed in more detail below; for the present point, their salient feature is that they are built from an input multiple alignment of homologous sequences, and they represent that alignment using a probabilistic position-specific scoring system.

The unit of score is a "bit" (essentially the same as BLAST "bit scores"), which is a measure of information content [166, 233]. Some intuition can be given for what a bit means, without much mathematics. A single perfectly conserved RNA residue (probability 1.0) contrasted to a uniform expected background (probability 0.25) is $\log_2 \frac{1.0}{0.25} = 2$ bits of information – you need to ask two yes/no questions to narrow four possibilities down to one, thus two "bits" (binary units) of information. A position where each residue occurs with equal probability (same as expected background) has zero bits of information. Imagine two positions that contain a covarying Watson-Crick base-pair in which each of the four possible base-pairs occurs with equal frequency $\frac{1}{4}$. In a sequence only model the two positions contribute zero bits of information, but in a structure/sequence model this pair contributes two bits of information from the pairwise correlation (the expected background in these columns is 0.0625 for each of the 16 possible base-pairs, but only 4 are observed with probability 0.25 each). In contrast, two columns that form a Watson-Crick base-pair that is perfectly conserved (a GC with probability 1.0 for example) always contribute four bits of information, regardless of whether they are modeled together as a pair ($\log_2 \frac{1.0}{0.0625} = 4$), or independently ($\log_2 \frac{1.0}{0.25} + \log_2 \frac{1.0}{0.25} = 4$). Thus, the best case for extracting useful sequence information from RNA secondary structure are covarying base-pairs that are individually not conserved in primary sequence at all. The more highly conserved the aligned RNA

---

[4] This figure appears in Chapter 1.

sequences are, the more primary sequence information content and less covariation will be seen.

Importantly, for local sequence alignment searches using probabilistic models, there is a direct, intuitive connection between the score in bits and the statistical significance (E-value) of a detected match [59]. Roughly speaking, every 3 or so bits of score improves the E-value by a factor of ten-fold (for high scores, the E-value is an exponential function of the bit score x; E is proportional to 2^{-x}). So, as a rule of thumb, extracting ten more bits of information for a homology search means shifting E-values by three orders of magnitude. This increase in resolution doesn't matter much if a sequence is already readily detected by primary sequence comparison (improving an already significant E-value of 10^{-30} to 10^{-33}, for example), but it becomes important when lifting a marginally insignificant E-value to significance (0.1 to 10^{-4}, for example).

Figure 1.9 shows the extra bits of information contributed by including RNA secondary structure in "typical" RNA search models. These models are all position-specific profiles built from alignments in the RFAM RNA families database, described below. There is substantial variation from family to family, but the extra information contributed by secondary structure is usually on the order of 10-20 bits or more, depending on the length and conservation of the alignment, which would be expected to improve E-values of homologs by about 3-6 orders of magnitude. This improvement can be seen in the results of the anecdotal searches of Table 1.1 comparing the E-values obtained by primary sequence BLASTN searches to INFERNAL, a sequence+secondary structure RNA homology search, as we will discuss in more detail below. The conclusion here is that while primary sequence is still the dominant source of information (at least for these particular "typical" searches; it is, of course, possible to imagine searching for RNAs with zero sequence information and only secondary structure information), adding secondary structure contributes enough information content that we can expect a structure+sequence method to resolve some homologs that were not quite resolvable by sequence analysis alone.

5.2 Infernal: software for RNA homology search and alignment

Computational methods that combine RNA secondary structure and sequence conservation information in a single consistent statistical model have been developed, based on probabilistic models called "stochastic context-free grammars" (SCFGs) [53, 57, 65, 226]. Dynamic programming algorithms exist for optimal alignment of SCFGs to target sequences, analogous to algorithms for sequence alignment except that SCFG algorithms are aligning by base-paired secondary structure in addition to sequence [53, 120, 135, 285]. A particular formulation of SCFGs, called *covariance models* (CMs), was developed specifically for automatic construction of statistical models from input RNA secondary structures or input multiple alignments annotated with consensus RNA structure. This technology is implemented in a freely available software package called INFERNAL (http://infernal.janelia.org).

A variety of other computational tools for RNA homology search exist besides INFERNAL (reviewed in [58, 128, 165, 270]). Some of the most popular tools are ERPIN [90], FASTR [287], RSMATCH [161], RNAMOTIF [167], RNATOPS [121], and PATSCAN [52]. INFERNAL is one of the most generally applicable tools, is the basis for a widely used RNA family database (RFAM; described below), and currently appears to be the best overall in performance according to published benchmarks [85]. Here we will restrict our discussion to INFERNAL.

To demonstrate how scoring structure increases statistical power for RNA homology search, we used INFERNAL to build CMs and perform searches for the single sequence/structure queries in Table 1.1 (the structures were obtained from the RFAM database, described below). As expected, modeling structure makes the target RNA more distinguishable from background, as evidenced by the decrease in E-values between BLASTN and CM searches of between three and thirteen orders of magnitude.

Figure 5.1 provides more detail for the cobalamin (B12) riboswitch example from Table 1.1. It shows the *Escherichia coli* query sequence and secondary structure, and the pattern of conservation in two different homologs found by a CM built from the *E. coli* query. Notice that although many of the residue substitutions between query and target are in the predicted loop regions, those that occur in a position that is base-paired are often accompanied by a compensatory change in the paired position to maintain a Watson-Crick or GU/UG pair. The extra information from the *E. coli* structure allows INFERNAL to find the homologous riboswitch in the *Acinetobacter baumannii* genome as the top scoring hit with a significant E-value of 3.7×10^{-5}, despite it sharing only 49% sequence identity with the *E.coli* riboswitch. The analogous search with BLASTN does not identify the riboswitch homology with a significant E-value (E=2.6).

CMs can be built from single RNAs, but they are most powerful when built from a multiple sequence alignment with consensus secondary structure annotation. CMs implement a position-specific ("profile") scoring system, where each consensus single-stranded position or base pair is represented by its own set of four or sixteen scores, and insertion/deletion scores are likewise specific to each point where an insertion or deletion can occur. Given enough aligned sequences, a position-specific profile model can learn which residues or base-pairs are highly conserved, what substitutions are tolerated by evolution, and where an RNA does and does not frequently tolerate insertion and deletion of sequence residues or structural domains. Given only a single RNA sequence (as in the examples in Table 1.1 and Figure 5.1), the CM scoring system reverts to a position-independent parameterization representing the averaged constraints on typical RNAs, essentially analogous to the use of score matrices in pairwise sequence alignment methods like BLAST.

CMs are probabilistic models, meaning that all the scoring parameters are probabilities rather than arbitrary scores and penalties. This helps in managing the complexity of setting a large number of parameters in an objective, automatic, and mathematically justified way; a consensus tRNA CM has about 1500 parameters and a consensus LSU rRNA CM has about 50,000 parameters that need to be determined. Using probabilities as parameters also helps in interpreting the significance of potential matches in a database search, and in calculating confidence values (posterior probabilities) associated with each residue in a proposed alignment. The use of probabilistic models for RNA structure/sequence analysis follows in the wake of similar techniques in primary sequence analysis, where score profiles

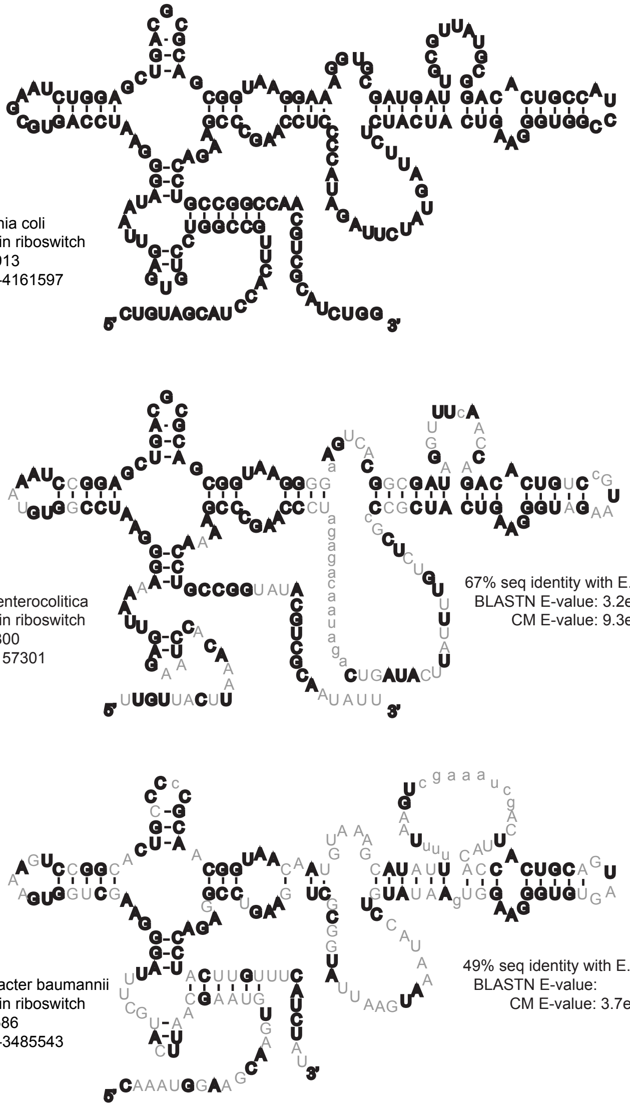


Figure 5.1: **Secondary structure of three cobalamin riboswitches.** Using the *E. coli* sequence as a query against their respective genomes, BLASTN detects the *Y. enterocolitica* cobalamin riboswitch with a significant E-value, but not the *A. baumannii* riboswitch. INFERNAL searches with a CM constructed from the *E. coli* sequence and structure (from the RFAM seed alignment for family RF00174 [89]) find both riboswitches with increased significance values. These example searches are listed in Table 1.1. Structures of the targets and percent identity figures were derived from the highest scoring CM alignment of each target to the query (*E. coli*). Sequence substitutions and insertions in the targets with respect to the query are shown in gray. Inserted residues with respect to the query are shown in lowercase. Base-Pairs in the RFAM annotated structure are connected by solid lines, except those that are not Watson-Crick, GU, or UG, which are connected by dotted lines. All riboswitches are immediately upstream (5'; within 100 residues) of *btuB* vitamin B12 transporter protein coding genes in their respective genomes.

(also called position-specific scoring matrices, PSSMs) have been made more powerful and consistent using probabilistic models called profile hidden Markov models (profile HMMs) [53, 142].

A CM can be used for a variety of alignment and search tasks. For example, very large numbers of RNA sequences can be aligned to a single RNA structure consensus with reasonable accuracy and efficiency: the RIBOSOMAL DATABASE PROJECT (RDP) now uses INFERNAL to produce alignments of hundreds of thousands of small subunit (SSU) ribosomal RNAs [40]. For sequence annotation, including metagenomic analysis, the main use of CMs is for homology search.

Because INFERNAL requires that the user provide a consensus RNA secondary structure for the query RNA, and because CMs are most powerful when models are built from multiple sequence alignments, a fair amount of work might be invested in carefully assembling a high-quality multiple sequence alignment annotated with a consensus structure. This investment may be feasible if one is only interested in sequence analysis of a particular RNA family, such as ribosomal RNA. However, if the goal is comprehensive high-throughput annotation of many different functional RNAs, for instance as part of analyzing a new metagenomic sequence dataset, it would be useful to have access to a large number of structure-annotated RNA alignments and pre-built CMs. Much as protein domain databases like PFAM and SMART have collected on the order of 10,000 protein domain sequence alignments for systematic profile HMM analysis [76, 151], there is a database called RFAM that has systematically collected RNA alignments and CMs [89].

5.3 Rfam: high-throughput RNA homology search and annotation

The RFAM database [89] is a curated and annotated collection of RNA sequence families, intended for the purpose of systematic, automated, high-throughput annotation of functional RNA elements in genomic and metagenomic sequence data. The current (9.1) version of RFAM contains 1372 families (http://rfam.sanger.ac.uk). Each RFAM family consists of three main components: a representative “seed” alignment, a covariance model (CM) built from the “seed” alignment, and a comprehensive “full” alignment.

The “seed” alignment is intended to be a small, stable, and curated alignment of representative members of the sequence family, annotated with a consensus RNA secondary structure. For example, the glycine riboswitch (RF000504; http://rfam.sanger.ac.uk/family/RF00504) is represented by an alignment of 53 RNAs.

The “full” alignment is intended to be comprehensive. It consists of an INFERNAL-generated structural alignment of all homologous RNAs detected by INFERNAL in a search, using the CM built from the “seed” alignment, of a composite DNA sequence database, RFAMSEQ, which now includes both genomic and metagenomic sequence data [89].

The alignments are useful for a variety of purposes, such as phylogenetic tree inference, examining the phylogenetic range over which a given RNA family occurs, or as a source of training data for other RNA structure analysis methods. For metagenomic analyses, the main application of INFERNAL and RFAM is homology search, and the main resource is the set of pre-built RFAM CMs.

The INFERNAL package (http://infernal.janelia.org) and RFAM CM files (http://rfam.sanger.ac.uk) can be freely downloaded and used to identify homologs of known functional RNAs in a metagenomics dataset. As an example of such an analysis, we performed CM searches of a previously published metagenomics dataset [251]. (A similar analysis of riboswitch occurrence in this metagenomic dataset using different search methodology has been published [136]; we compare the results below.) The dataset includes about 200,000 whole genome shotgun sequencing reads totaling about 230 Mb derived from samples of agricultural soil ($\sim$ 140 Mb, accession AAFX01000000) and three “whale fall” carcasses ($\sim$ 90 Mb, accessions AAFZ00000000, AAFY01000000, AAGA00000000). To simplify the analysis for our illustrative purposes here, we searched only for riboswitches, using the 15 RFAM 9.1 CMs of type 'cis reg; riboswitch' [89]. For comparison, we repeated the search with BLAST, using each individual sequence in the RFAM seed alignment as a BLAST query and combining the results to identify any significant matches [103]. Additionally, we performed searches with INFERNAL v1.0 run in a profile HMM mode, which ignores secondary structure and scores only primary sequence conservation. Comparison of the BLAST, profile HMM and CM search results illustrates the relative contribution of the two main differences between BLAST and CMs: the use of probabilistic profiles instead of pairwise comparisons (by comparing BLAST and HMM results), and scoring both sequence and RNA structure (by comparing CM and HMM results).

Table 5.1 includes the number of putative riboswitches (hits) with E-values less than 10^{-5} found for each family using each method. Also displayed are the number of hits detected by one method but not another for all six possible pairwise combinations of the three methods. Using the strict 10^{-5} E-value cutoff, INFERNAL CM searches found 135 total putative riboswitches in the soil and whale falls dataset; HMM searches found 102 (a subset of the 135); and BLAST found 50. Profile HMM searches detected 61 hits that BLAST did not, and CM searches detected 33 hits that profile HMMs did not, indicating that using profiles and additional scoring of structure both contribute significantly to an increased sensitivity of CMs over BLAST.

We can compare these results to the recently published results of a similar analysis of riboswitch occurrence in the same metagenomic dataset using different search methodology [136]. Kazanov et al. [136] used the pattern based search program RNA-PATTERN to identify candidates of 11 riboswitch families (8 of which we used in our analysis) in the same soil and whale falls data we analyzed. For the 8 families in common, their pattern based search detected 103 candidate riboswitches, compared to 125 identified by CM searches at a stringent threshold. RNA-PATTERN detected 14 candidates that CMs did not, and CMs detected 36 candidates that RNA-PATTERN did not. The largest differences were for the cobalamin family, for which CMs found 18 candidates undetected by RNA-PATTERN, and the glycine family, for which RNA-PATTERN found 10 candidates undetected by CMs using a CM E-value threshold of 10^{-5}. Six of these 10 are found by the glycine riboswitch CM, but with E-values just below the strict threshold, ranging between 10^{-3} and 10^{-5}. For the remaining four, we cannot distinguish whether these are missed by the CM, or whether they

			# seqs found			# different seqs found					
family	RFAM ID	avg % id	BLAST	HMM	CM	BLAST -HMM	BLAST -CM	HMM -BLAST	HMM -CM	CM -BLAST	CM -HMM
FMN	RF00050	72	8	9	9			1		1	
TPP	RF00059	56	8	23	35			15		27	12
SAM	RF00162	69	4	8	14	3		7		10	6
Purine	RF00167	56									
Lysine	RF00168	51		1	1			1		1	
Cobalamin	RF00174	54	20	43	47	2	2	25		29	4
glmS	RF00234	60		2	2			2		2	
Glycine	RF00504	54	7	8	17	3	1	4		11	9
SAM_alpha	RF00521	71	1	7	7			6		6	
PreQ1	RF00522	67									
SAM-IV	RF00634	73									
preQ1-II	RF01054	68									
MOCO_RNA_motif	RF01055	59			1					1	1
Mg_sensor	RF01056	78									
SAH	RF01057	63	2	1	2	1					1
total	-	-	50	102	135	9	3	61	0	88	33

Table 5.1: **Riboswitch search results.** "# seqs found": the number of hits receiving E-values better than (less than) 10^{-5} for each search method in the 230 Mb soil and whale falls dataset [251] for each RFAM 9.1 riboswitch family. "# different seqs found": number of hits detected by one method and not detected by another for each pairwise combination of methods, for example: "BLAST-HMM" for RF00162 is 3 because 3 hits detected by BLAST were undetected by the HMM search. A blank space in a cell indicates 0.

are false positive predictions by RNA-PATTERN and Kazanov et al. [136]; one disadvantage of pattern search programs is that they do not generally report any objective measure of the statistical significance of a match.

Can we trust that the statistically significant matches to the CM are really homologs, and that the increased numbers really reflect increased detection sensitivity? That is, based solely on the results of the demonstration experiment here, where we are simply counting the number of hits detected below some E-value threshold and asserting that these are all probable homologs, it is possible that INFERNAL is instead merely assigning incorrectly low E-values to nonhomologous sequences. One way to test the accuracy of any program's E-values is to search randomized nonhomologous sequence; one expects the top-scoring random match to have an E-value on the order of 1 (by definition of expectation value: the number of hits you expect to see in this database search with a score this high just by chance). (This sort of test is a useful control experiment to run whenever thinking of adopting any new search method.) In one recent experiment of ours [190], involving a benchmark of 51 CMs being searched against a 10 megabase synthetically generated target sequence, the highest nonhomologous hit had an E-value of 0.009, about what you'd expect from doing 51 independent searches ($1/51 = 0.019$) if E-values were accurate. An E-value threshold of 10^{-5} is if anything on the conservative side. Most importantly, an independent benchmark of a variety of RNA similarity search methods has been published [85], which generally found that CM based methods are the most sensitive and specific methods available.

5.4 Limitations of CMs

Now the fine print. Users applying INFERNAL and RFAM for metagenomics analysis should be aware of five important limitations of CM similarity search:

1. The principal drawback of CM methods is that they are slow. In the riboswitch example above, the fifteen CM searches took about 71 CPU hours (18 minutes on 250 processors), about 100 times longer than BLAST searches (45 minutes on one processor). Repeating this search using all 1372 RFAM 9.1 models would take roughly 3 CPU years (about 4 days on a 250 CPU cluster). Significant compute power (such as a moderate sized cluster) is required to do large scale analyses with CMs. INFERNAL is parallelized for use on clusters using the Message Passing Interface (MPI) [102].

Though still slow compared to BLAST, INFERNAL is much faster than it was just a few years ago. The current version (v1.0) [190] is about 100 times faster than version 0.55 [56]. The speedup is due to heuristics, including filtering [265, Nawrocki and Eddy, in preparation] and banded dynamic programming [189], which sacrifice a small amount of sensitivity for the increased speed. This sensitivity sacrifice, though small, disproportionately impacts remote homology detection [189, 190]. It may be worthwhile to switch off the heuristic speedups for smaller scale analyses if the requisite compute power is at hand. Conversely, if compute power is limiting, the heuristic speedup parameters can be tuned for greater acceleration at a greater cost in sensitivity [192]. Further acceleration remains a major goal of INFERNAL development.

Another computationally expensive step of CM similarity search is "calibrating" models in order to obtain E-values for search results, and to determine the appropriate filtering scheme for maximum speed without significant sensitivity loss. INFERNAL's `cmcalibrate` program must run several large computational simulations, and this takes several CPU hours for a typical sized CM. The CMs from the RFAM database come pre-calibrated, so RFAM users do not have to pay this cost, but any custom built models need to be calibrated.

2. A CM models only a single user-provided RNA consensus structure. Many RNA structures are inferred, rather than being determined by crystallographic or NMR methods, so secondary structure annotation may well be at least partially incorrect – especially in large collections like RFAM, where curation of a set of over 1300 consensus structures is challenging. Additionally, a single consensus structure is unable to properly capture the evolutionary variation observed amongst individual homologous secondary structures, except in a crude way (as structural deletions and insertions relative to the consensus). And finally, an assumption that an RNA adopts only a single secondary structure is only an approximation, as RNAs (like proteins) are sure to exist in an ensemble of different structures (perhaps bound and unbound to a protein or substrate). Riboswitches, for example, are a dramatic example of the function of an RNA depending on it adopting at least two distinct structural conformations.

3. Using a CM for non-structured RNAs is pointless. Many RNAs may not require a conserved structure for their function. For example, antisense regulatory RNAs that control gene expression simply by base-pairing to target mRNAs are acting as primary sequences, and they do not necessarily conserve any intramolecular secondary structure. Though CMs can model RNAs with no consensus base-pairs [192], it is more practical and appropriate to use profile HMMs rather than CMs, avoiding the CMs' computational costs.

4. CMs ignore some aspects of RNA structure. By their nature, CMs are only able to model a canonical secondary structure consisting of exclusively nested base-pairing relationships, meaning a set of base-pairs for which no two pairs "overlap" in sequence position (no two pairs between positions $i : j$ and $k : l$ exist such that $i < k < j < l$). This means CMs do not model RNA pseudoknots, base triples, nor most other contacts found in RNA tertiary structure. The goal of a CM is not to model RNA structure completely, but rather to harness as much additional structural information as possible for more accurate RNA search and alignment, while still allowing for reasonably efficient algorithms. Capturing yet more higher-order RNA structural information is possible, but it violates the constraints of SCFG-type probabilistic models and comes at a disproportionate cost in computational efficiency [222]. Other methods exist for RNA homology search that can model RNA pseudoknots, including ERPIN [90] and RNATOPS [121].

5. Truncated sequences present special issues. Metagenomic shotgun sequencing surveys produce sequence fragments from essentially random positions on a host genome. When a shotgun read overlaps a structural RNA element, residues involved in conserved base-pairs may be missing. An RNA structural alignment algorithm needs to anticipate and allow this sort of sequence truncation to be useful for shotgun sequence analysis. Until recently, the local alignment algorithms used for CMs and related SCFG-based RNA alignment methods looked for structural deletions (deletions that remove a stem or structural domain, respecting evolutionary base pairing constraints), but did not look for sequence truncations. INFERNAL now includes a new local alignment method, developed by Diana Kolbe, which more effectively deals with missing sequence data in shotgun reads [141]. As we write this, this feature is not yet incorporated in `cmsearch` for homology searches (it soon will be), but the new algorithm can be used for sequence alignment with INFERNAL's prototype `trcyk` program.

5.5 Conclusion

Compensatory base-pair changes in RNA sequence alignments are strikingly apparent even to the eye. The deeper the alignment (the more sequences known to conserve roughly the same structure), the more the RNA structure becomes obvious by sequence analysis alone. Robin Gutell and co-workers were able to predict the secondary structure of ribosomal RNA to greater than 98% accuracy per base-pair by essentially manual comparative analysis of careful rRNA alignments [107], and Francois Michel and Eric Westhof predicted the structure of group I intron catalytic introns in much the same way [181]. The automation of comparative RNA structure/sequence analysis is essentially the basis of algorithms that combine RNA secondary structure and sequence analysis to enable identification of more remote RNA homologs than primary sequence methods alone can achieve. These methods can be used to search metagenomics datasets for known families of RNAs using a combination of the INFERNAL software (http://infernal.janelia.org) and CMs from the RFAM database [89].

Chapter 6

Conclusion to RNA homology search

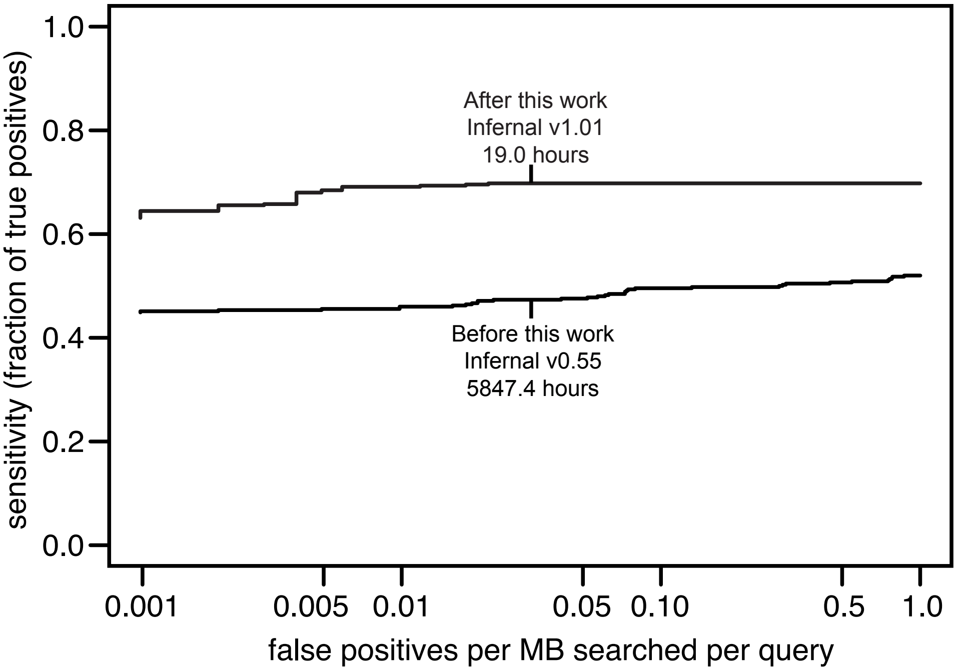


Figure 6.1: **Rfam benchmark ROC curves for Infernal v0.55 and v1.01.** The benchmark is explained in Chapter 3.

Figure 6.1 demonstrates the effect of my thesis work on the sensitivity, specificity and speed of CM homology search using INFERNAL.

The improved sensitivity and specificity is due to:

- Informative mixture Dirichlet priors for CM emissions (Chapter 2).
- Informative Dirichlet priors for CM transitions (Chapter 2).
- Entropy weighting parameterization strategy (Chapter 2).
- Log likelihood scoring of a target sequence given a model using the Inside algorithm (Chapter 5).

The roughly 300-fold acceleration is due to:

- Query-dependent banding (QDB) for the CYK and Inside algorithm (Chapters 2 and 5).
- Optimized implementations of the CYK and Inside algorithms. (Chapter 5).
- HMM filtering using a reimplementation of Weinberg/Ruzzo ML-heuristic HMMs [265] and the filter sensitivity targeting (FST) thresholding strategy. (Chapter 5).
- Combined HMM and QDB CYK filtering. (Chapter 5).

Part 2:

RNA alignment

Chapter 7

Introduction to small subunit ribosomal RNA alignment

Small subunit ribosomal RNA (SSU rRNA, or just SSU [1]) is a non-protein coding, structural RNA that is found in all cellular organisms. SSU forms the structural core of the small subunit of the ribosome. It interacts with the large subunit ribosomal RNA (LSU rRNA), ribosomal proteins, and transfer RNAs to decode messenger RNAs into amino acids and provide peptidyl transferase activity to form peptide bonds between adjacent amino acids during translation. Due to this central role in the vital process of translation, SSU's sequence and structure have been highly constrained by evolution. The constraint has been so strong that SSU gene sequences from organisms across the entire tree of life are highly similar and clearly homologous [32].

The high level of sequence similarity in SSU is exploited by environmental sequencing surveys that seek to characterize microbial diversity in various environments [122]. For more than twenty years, these studies have been performed for many environments and have led to repeated discoveries of new types of life [123, 155, 200]. In a typical environmental sequencing survey, SSU sequences are derived from a bulk nucleic acid sample from an

---

[1] Bacterial and archaeal SSU rRNA are also called 16S rRNA and eukaryotic SSU rRNA is called 18S rRNA, based on their sedimentation speed during centrifugation. Here, I will refer to all types of SSU rRNA as simply SSU.

environment and aligned with SSU sequences from known organisms with known (or at least trusted) positions on the tree of life. The alignment is then used as the basis for estimating a phylogenetic tree, which helps reveal the biodiversity of the environment by comparison of the placement of the environmental sequences with the known ones on the tree.

The scale of environmental survey studies is growing rapidly. A single study can generate thousands or tens of thousands of SSU sequences and currently there are about one million SSU sequences in GENBANK [12]. As these studies continue to grow in scale, computational tools that can rapidly and accurately align large numbers of SSU sequences will become more and more useful.

A focus of my thesis work has been to create a practical tool for generating large SSU alignments (up to millions of sequences). In this chapter, I introduce the history of SSU-based phylogenetic analyses, describe the current state-of-the-art tools for generating large alignments, and discuss the design and motivation of the alignment tool I have developed.

7.1 SSU and the tree of life

In 1977, Carl Woese working with George Fox at the University of Illinois at Urbana-Champaign was interested in studying the evolution of translation, which to him was "the central problem in the evolution of the cell" [273]. To do this he realized he needed a universal phylogenetic tree of life as a conceptual framework. He decided to use SSU because it existed in all cellular life, had evolved slowly enough to be comparable across all life, and was large enough to give a useful amount of data.

To infer the SSU tree, Woese employed the oligonucleotide cataloging technique for RNA sequencing [228] using the SSU of 13 organisms: ten prokaryotes and three eukaryotes [276]. In short, Woese digested the purified SSU with T_1 RNase and separated the resulting fragments with two-dimensional electrophoretic separation to create a separate oligonucleotide fingerprint for each organism's SSU. After determining the sequence of the oligonucleotides, he measured the sequence similarity between all possible pairs of the 13 sequences. The results clearly suggested that the 13 organisms were representatives of three separate phylogenetic clades. At that time it was universally accepted that all of life was divided into two primary phylogenetic groups, the prokaryotes and the eukaryotes, not three. More specifically, while the three eukaryotes clustered together and most of the prokaryotes clustered together (the blue-green bacteria, chloroplasts, gram-positive and gram-negative bacteria), the methanogenic "bacteria" formed a separate cluster that was as distinct from the other bacteria as it was from the eukaryotes. Woese and Fox proposed this as a new kingdom of life, the *archaebacteria*, which have since been renamed the *archaea*. The third domain uprooted convention and was very controversial. Woese spent years defending the three domain model [86]. Today, the existence of three separate domains is unchallenged.

In the next few years, Woese and his colleagues applied their oligonucleotide cataloging approach for specific clades including the *mycoplasmas* [279] and the purple bacteria [281]. They found that the approach had its limitations. While it was powerful enough to define bacterial *phyla* (one level lower in classification than kingdom), it had difficulty resolving the relationships between the subdivisions within phyla [275]. By the mid-1980s, Sanger DNA sequencing [227, 229] had matured enough to allow the determination of complete SSU sequences [105]. The burgeoning field of SSU-based phylogenetic analyses quickly switched from oligonucleotide cataloging to using full SSU sequences because they were more informative and statistically powerful.

7.2 Rapid culture-independent SSU sequence determination

By 1985, about 50 full or nearly full length SSU sequences had been determined [201]. The ability to determine an SSU sequence from an organism depended on being able to grow that organism in culture in a laboratory so that enough SSU could be isolated for sequencing. This limited SSU-based phylogenetic analyses to culturable organisms; a serious limitation because it was known that many (though no one knew how many) organisms could not be grown in a laboratory. In fact, at that point virtually all detailed microbiology knowledge in general, concerning metabolism, cell structure and physiology, had been gained from the study of culturable organisms. In 1985, researchers working in Norm Pace's lab at Indiana University pioneered a method for determining partial SSU sequences from bulk cellular RNA, advancing SSU sequencing to the world of unculturable microorganisms. This breakthrough would eventually lead to repeated dramatic discoveries of new microbial biodiversity that continue today [123].

Pace's method involved targeting so-called *universal* primer sites, 15-20 nt long regions of SSU that were conserved perfectly or nearly perfectly among the 50 existing sequences, with complementary oligonucleotides, and copying the intervening SSU region using reverse transcriptase [201]. Other groups soon began refining Pace's technique, including David Ward, Roland Weller [267] and Stephen Giovannoni [93]. Ward and Weller applied culture-independent SSU sequencing to look for novel microbes in a Yellowstone hot spring community, which had been relatively well-studied before culture-independent methods were developed [20]. The new methods revealed novel biodiversity. In one study they detected eight new microbes, doubling the number of known microbes in the hot spring community [260], and in another they discovered two new cyanobacteria and one new green nonsulfur bacteria [268]. Giovannoni's work was an early successful integration of Pace's method with Kary Mullis' contemporary improvements to the polymerase chain reaction (PCR) technique [184]. Giovannoni applied his PCR-based methods to the bacterioplankton of the Sargasso Sea, where he found two new types of bacteria, including *SAR11* which he later found to be the dominant organism in surface waters of the ocean and among the most abundant organisms on Earth [183].

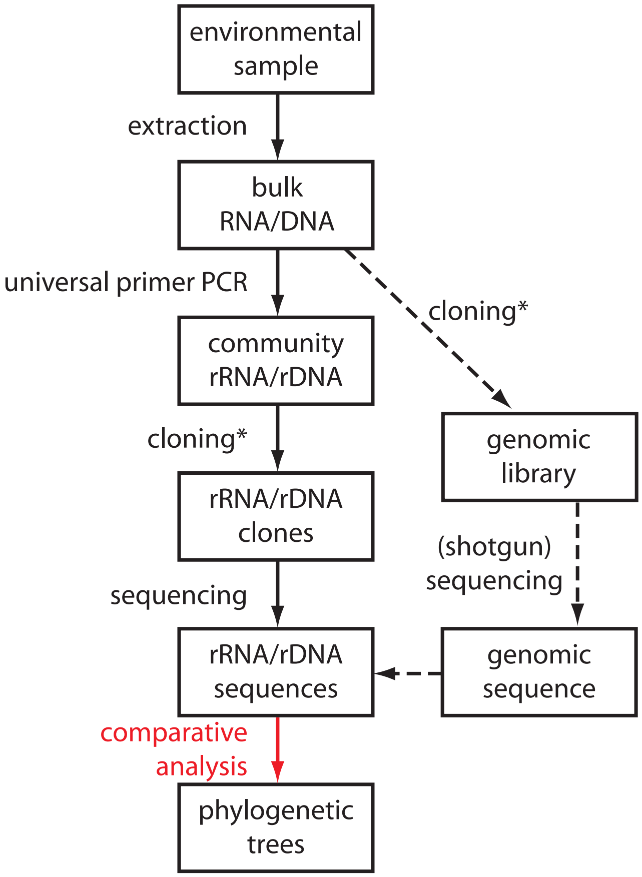
7.3 Environmental sequencing surveys

Giovannoni, Weller and Ward's studies in the late 1980s and early 1990s were some of the first *environmental sequencing surveys* that aimed to sample SSU sequences from organisms directly from their natural environment. Probably largely due to the exciting findings of the early studies, these types of surveys quickly became very popular. The basic methodology that Pace *et al.* began developing in the mid-1980s is summarized in Figure 7.1.

In the late 1990s and early 2000s, enabled by rapidly developing sequencing technologies, a new and more general brand of environmental surveys called *metagenomics* developed that did not specifically target SSU [109]. In metagenomics studies, an environmental sample of bulk DNA is sheared into fragments which are inserted into vectors for cloning and sequencing - typically using a shotgun-sequencing approach [251, 254, 256]. Some of these resulting sequence reads will be SSU, which are usually analyzed in much the same manner as in PCR-based studies to help reveal the organismal diversity in the sample (Figure 7.1, the dotted arrows). Additionally, because the genes of the sample environment's organisms are (more or less) randomly sampled, metagenomics provides a global view of the functional potential of an environment's microbes that an SSU-specific study cannot. Despite this advantage, and the growing popularity of metagenomics, SSU-specific studies are still widely performed, mainly due to the large number of available sequences and tools for SSU comparative analysis in public databases (discussed in more detail below). SSU is likely to continue to be an important phylogenetic marker in PCR- and metagenomics-based environmental surveys [250]. Because both of these types of surveys include the SSU comparative sequence analysis step most relevant to this work, I will group them together and refer to them as environmental surveys.

Environmental surveys have targeted hundreds of different environments and produced hundreds of thousands of SSU sequences. They differ markedly in the types of SSU sequences they target and the number of sequences they generate. Table 7.1 lists a few examples. While many focus on prokaryotes, targeting either archaea, bacteria, or both, a minority target eukaryotes as well. Of course, due to their general nature, metagenomics studies can generate sequences from all three domains. While environmental surveys vary widely in their scale, there has been a general upward trend in the number of sequences generated from these studies. This is reflected by the growth of the number of SSU sequences deposited in GENBANK [12] per year since 1996 (Figure 7.2).

Environmental surveys have continuously and profoundly expanded our understanding of microbial biodiversity on the planet. It is now clear that the vast majority of microorgan-



* the cloning step is bypassed by next-generation cyclic-array sequencing technologies

Figure 7.1: Flow chart of the key steps of environmental sequencing surveys. The dashed lines indicate steps in metagenomics analyses. Note that next-generation sequencing technologies bypass the cloning step.

environment(s)	domain(s)	# seqs	ref
cecal microbiota of mice	bacteria	5,088	[154]
human stomach	bacteria	1,833	[13]
hypersaline mat	bacteria	1,586	[155]
Sargasso sea	all 3	1,164	[256]
hydrothermal vents	eukarya	374	[66]
endolithic environment (pore space of rocks)	bacteria	342	[259]
soil & burrow casts of earthworms	archaea, bacteria	204	[87]
tidal flat sediment	archaea	90	[137]
salt marsh	eukarya	79	[241]

Table 7.1: **Sampling of SSU rRNA environmental surveys.** Studies were picked with a bias to demonstrate the diverse range of environments, number of sequences, and domains targeted.

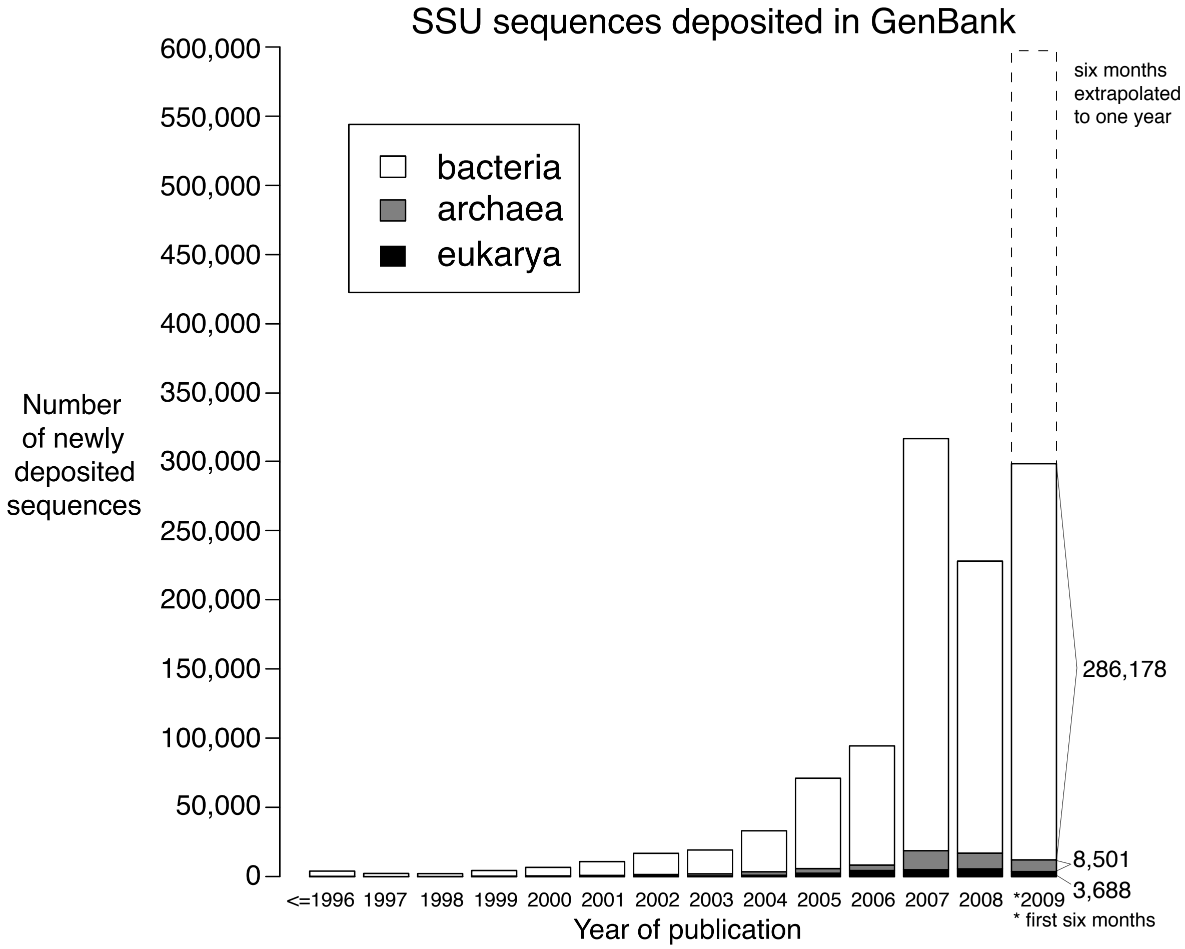


Figure 7.2: **Number of SSU rRNA sequences deposited in genbank per year since 1996.** Numbers were obtained from NCBI's Entrez Nucleotide website (http://www.ncbi.nlm.nih.gov/sites/entrez) with the following queries: archaea: "(SSU OR 16S OR small subunit) NOT(18S) AND (rRNA OR rDNA OR ribosomal RNA) AND (archae*) NOT (bacteri* OR eubacteri* OR eukary* OR eucary*);" bacteria: "(SSU OR 16S OR small subunit) NOT(18S) AND (rRNA OR rDNA OR ribosomal RNA) AND (bacteri* OR eubacteri*) NOT (archae* OR eukary* OR eucary*)"; eukarya: "(SSU OR 18S OR small subunit) NOT(16S) AND (rRNA OR rDNA OR ribosomal RNA) AND (eukary* OR eucary*) NOT (bacteri* OR eubacteri* OR archae*)". The queries were limited by publication date to the years shown.

isms are *un*culturable, (some estimates reach as high as 99% [239]). The pace of discovery has been rapid, and does not hint at slowing. Take, for example, the growth in the number of recognized bacterial phyla over the past twenty years. There were 12 in 1987 [275], 25 by 1997 [200], and 52 by 2003 [216]. In 2006, Ley et al. [155] defined 15 new candidate phyla in a single study of the Guerrero Negro hypersaline microbial mat, raising the number to 67 [155]. Not only does novel diversity continue to be discovered, but the rate of discovery is increasing.

Characterizing microbial communities

Finding novel life forms is exciting, but it is certainly not the only, or even the primary, motivation for environmental surveys. They help researchers better understand microbial communities living in various environments by identifying the organisms that comprise them and the metabolic processes present there (primarily in the case of metagenomics studies).

While understanding life in environments such as the burrow casts of earthworms [87] or the pore space of rocks [258] (Table 7.1) are mainly of interest to environmental microbiologists, other environments have been the focus of broader interest. In particular, medical microbiologists have been identifying the microorganisms living on and in the human body for over a decade [282], which have collectively been coined the human *microbiota*. SSU-based surveys have been conducted for the skin [88], vagina [124], mouth [188], and gut [13, 210].

The number of microbial species in the gut alone has been estimated at nearly 40,000 [83] and the number of cells making up a human's microbiota may outnumber their own by a factor of 10 [253]. Gastrointestinal diseases such as Crohn's disease, pouchitis and obesity have been associated with large-scale imbalances in the community structure of the gut microbiota [156]. The U.S. National Institutes of Health (NIH) Roadmap for medical research has recently begun the *Human Microbiome Project* (HMP), a five-year interdisciplinary initiative to characterize the microbiota and analyze its impact on health and disease [253]. The HMP, with a total budget of $115 million, will focus on the organisms living in and on the skin, nose, lungs, mouth, vagina, and gut.

7.4 Comparative SSU analysis for environmental surveys

For all environmental surveys, regardless of how sequences are obtained and how many are obtained, the final crucial step of "comparative analysis" (Figure 7.1) reveals the biodiversity in the sample. This is the step most relevant to the computational biologist, and will be the main focus of the remainder of this chapter and the next two.

Woese's association coefficient

In his seminal 1977 paper, Woese's comparative analysis step involved oligonucleotide cataloging [81, 276]. The SSU molecules are digested with T_1 RNase (which cleaves at `G` residues) producing short oligonucleotides of lengths up to about 20 residues. The oligonucleotides produced from a single SSU sequence define its catalog. The catalogs are then compared to each other to group the sequences phylogenetically using a metric called the *association coefficient*, S_{AB} defined as:

$$S_{AB} = \frac{2N_{AB}}{(N_A + N_B)}$$

N_A and N_B are the total sum of bases in distinct oligonucleotides (hexamers or larger) in the catalog for organism A and B, respectively. N_{AB} is the overlap: the number present both in A and B. This statistic, perhaps surprisingly, was powerful enough to convince Woese that his 13 SSU sequences composed three distinct domains of life. The cataloging approach is even powerful enough to define many bacterial phyla [279], but has difficulty resolving branching patterns within them [275].

Alignments and trees

As full-length sequencing of SSU became feasible in the late 1970s and early 1980s [21], Woese and his contemporaries first complemented and eventually replaced the oligonucleotide cataloging approach with more powerful phylogenetic inference techniques based on multiple sequence alignments. Though the specific methods and implementations have changed since then, present day SSU surveys still follow this basic paradigm, summarized in Figure 7.3.

A typical comparative analysis for an SSU survey currently consists of two key steps: the computation of an alignment of the newly generated sequences, and the computation of a phylogenetic tree based on that alignment. The alignment step typically and critically takes advantage of a pre-existing, manually curated (and so presumably highly accurate) reference alignment when aligning the new sequences. This provides valuable information to the alignment program about the expected sequence conservation and how it varies across the molecule, as discussed in detail below. Some alignment programs are able to take the conserved SSU secondary structure, as annotated in the reference alignment, into account as well. The input to the phylogenetic inference step is often a combined alignment of the new sequences and some or all of the reference sequences. The location of the new sequences on the tree in relation to the reference sequences (about which taxonomic information is known) reveals their identity, or at least which reference sequences they are most closely related to.

Figure 7.3 depicts alignment and phylogenetic inference as two independent steps, and indeed virtually all environmental surveys (including all listed in Table 7.1) treat them as such, using a separate program for each. In theory though, an alignment program would benefit from knowledge of the phylogeny of the sequences it is aligning, which suggests simultaneously inferring an alignment and a tree is more appropriate than doing each independently. The unification of alignment and phylogenetic inference techniques is an important goal in sequence analysis and is an active area of research [78, 116, 244], but important challenges remain. In particular, those methods are generally too computationally expensive to practically handle the scale of many SSU surveys (thousands of sequences of more than a thousand residues). In this work, I will treat alignment and phylogenetic inference as independent steps.

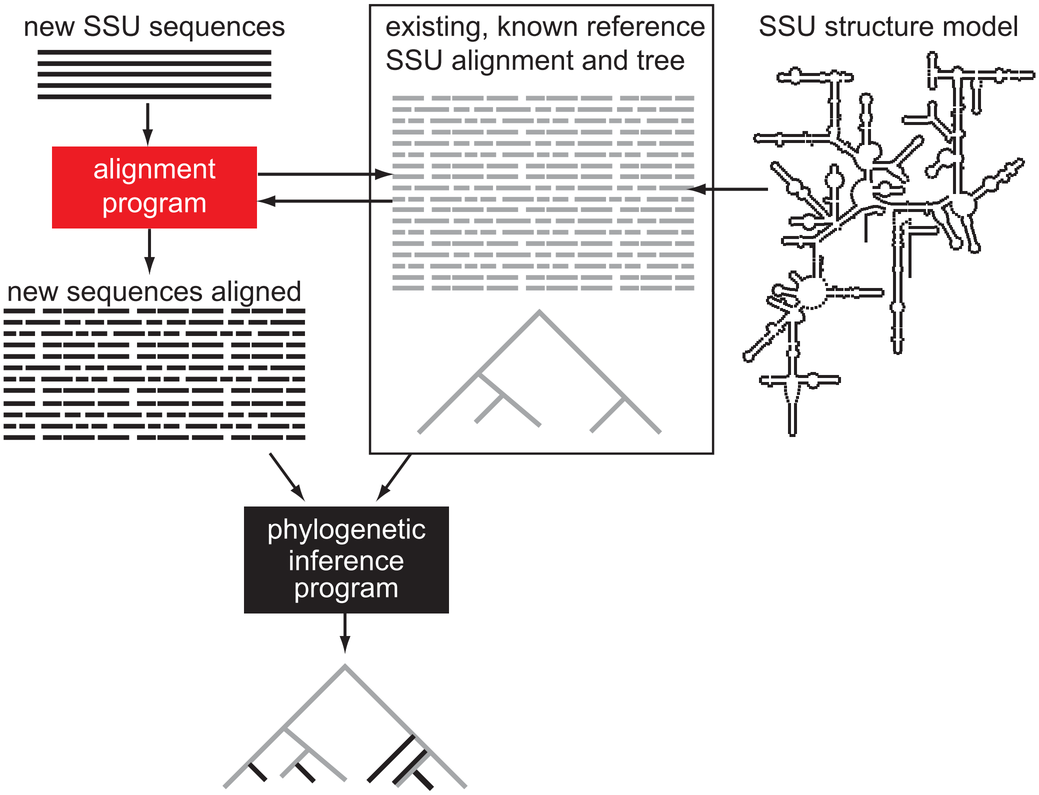

new SSU sequences
alignment program
new sequences aligned
existing, known reference SSU alignment and tree
SSU structure model
phylogenetic inference program

Figure 7.3: **Schematic of SSU sequence analysis for environmental surveys.** Two main computations are required, the alignment and the phylogenetic inference. The alignment step typically makes use of existing knowledge of SSU conservation and phylogenetic relationships in the form of a reference alignment. Some programs can utilize structural annotation in the reference alignment corresponding to the conserved secondary structure model of SSU. New sequences are added to the reference alignment by the alignment program and a tree building program calculates a tree estimating the evolutionary relationships of the aligned sequences. The placement of the reference sequences, for which taxonomic information is already known, provides a scaffold for classifying the organisms that the new sequences belong to.

Phylogenetic inference methods

Among the two steps, the main focus of this work is the alignment step. However, the alignment mainly serves as an intermediate in the comparative analysis. Its primary purpose is to be used to infer a tree that helps reveal the organismal diversity in the sample. This justifies a brief discussion of phylogenetic inference methods.

Methods to infer phylogenies can largely be divided into three groups [53]. *Maximum parsimony* methods seek to find a tree that implies the fewest evolutionary events (residue mutations, insertions and deletions) that explain the alignment. *Distance-matrix* based methods, such as the UPGMA and neighbor-joining algorithms, calculate a matrix of pairwise distances between all pairs of sequences in the alignment. Similar sequences are placed nearby on the tree, with the branch length separating them proportional to their pairwise distance. Finally, *probabilistic* methods calculate probabilities for possible trees based on an explicit model of evolution that includes probabilities for each possible mutation event (and sometimes insertions and deletions [221]). These methods are similar to *maximum parsimony* but allow varying mutation rates across positions of the alignment as well as in different parts of the tree.

A detailed discussion of these methods [72] is outside the scope of this work. What is most relevant to this work is that currently all of these methods implicitly assume that the input alignment is evolutionarily correct. In an evolutionarily correct alignment, all of the residues in a particular column are homologous, i.e. they have all descended from the same ancestral residue in the common ancestral sequence of all the sequences in the alignment.

Take the toy example in Figure 7.4. An alignment of three sequences has been used to infer the tree relating them. The tree could have been constructed using any of the three methods. (All three would very likely give this same tree. The tree *is* the maximum parsimony tree.) Assume the inferred tree is correct. A speciation event at the root conferred the ancestral sequence `UUACUG` to two descendants. In one of the descendants, the `A` mutated to a `C`. In the other descendant the `C` mutated to a `G`, and another speciation event occurred, after which the rightmost `U` mutated to a `C`. The alignment reflects these changes and so is

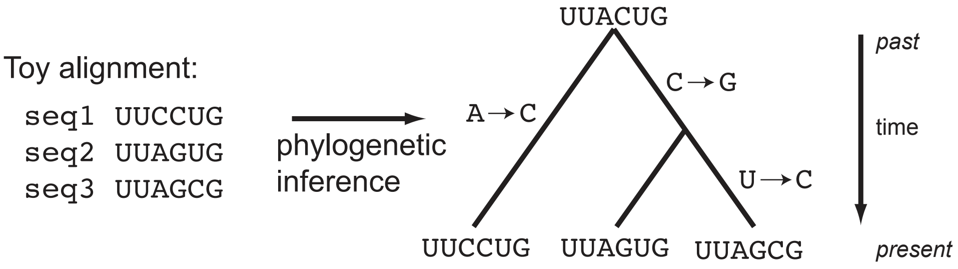


Figure 7.4: **Toy example of phylogenetic inference from an alignment.**

evolutionarily correct: within each column, all of the residues can be traced back to the same ancestral residue. However, if the alignment is incorrect and contains errors, the inference step is confounded, regardless of the method used. Maximum parsimony and maximum likelihood methods would attempt to build a tree explaining evolutionary events implicit in the alignment that did not occur. Distance-matrix methods would calculate inaccurate distances between any pair of sequences for which the pairwise alignment contained errors.

SSU alignments are, of course, much more complex than this toy example. Not only do they often include thousands of columns and sequences, but the sequences are different lengths, forcing the alignment to include gaps to signify insertions and deletions of residues. Alignment without gaps is trivial because there is no choice of alignment - the unaligned sequences *are* the aligned sequences. When gaps are allowed the alignment program must choose between many alternative alignments. (For two sequences of length N, the number of possible alignments that allow gaps in one of the two sequences is approximately $\frac{2^{2N}}{\sqrt{\pi N}}$ [53].) It is the job of the alignment program to determine the single best alignment based on a scoring system. Although it is intractable to enumerate and score all possible alignments, dynamic programming (DP) algorithms exist that can efficiently determine the optimal scoring alignment given a scoring system (without exhaustive enumeration). An example is the Needleman-Wunsch-Sellers algorithm introduced in Chapter 1 in the context of pairwise homology search. DP alignment algorithms are useful for creating multiple sequence alignments as well as for homology search, as discussed below.

Alignment masking

Some regions of SSU, such as the universal primer sites, are identical or nearly identical between the vast majority of SSU sequences and so are trivial to align correctly. But other, less well conserved regions can be more difficult to accurately align. Because alignment errors confound phylogenetic inference, it is a common practice to *mask* alignments by removing columns that are deemed to be ambiguously aligned prior to performing phylogenetic inference. Masking requires a method for judging the alignment ambiguity of different columns. The two most commonly used masks for SSU alignments were manually constructed by experts with extensive knowledge of the sequence and structural diversity of SSU across the tree of life.

David Lane's mask of SSU

In the late 1980s and early 1990s, David Lane, working with Norm Pace, defined an alignment mask for bacterial SSU alignments [145] based on his experience manually aligning about 300 SSU sequences (the complete set available at the time). Lane's mask became known as the *Lane mask* and has been commonly used for SSU analyses ever since, even as the number of available sequences and recognized biodiversity of the bacteria has greatly expanded. It is so well used that the term "Lane masking" is often used in place of "masking" in the field. The Lane mask is shown on the secondary structure of *Escherichia coli* in Figure 7.5.

Phil Hugenholtz's mask of SSU

In the late 1990s, Phil Hugenholtz defined 46 SSU alignment positions included by the Lane mask that he felt should be excluded, or removed, from alignments of sequences from all three domains. The Hugenholtz mask is included in the ARB software package (described below). It is shown on the secondary structure of *Escherichia coli* in Figure 7.6.

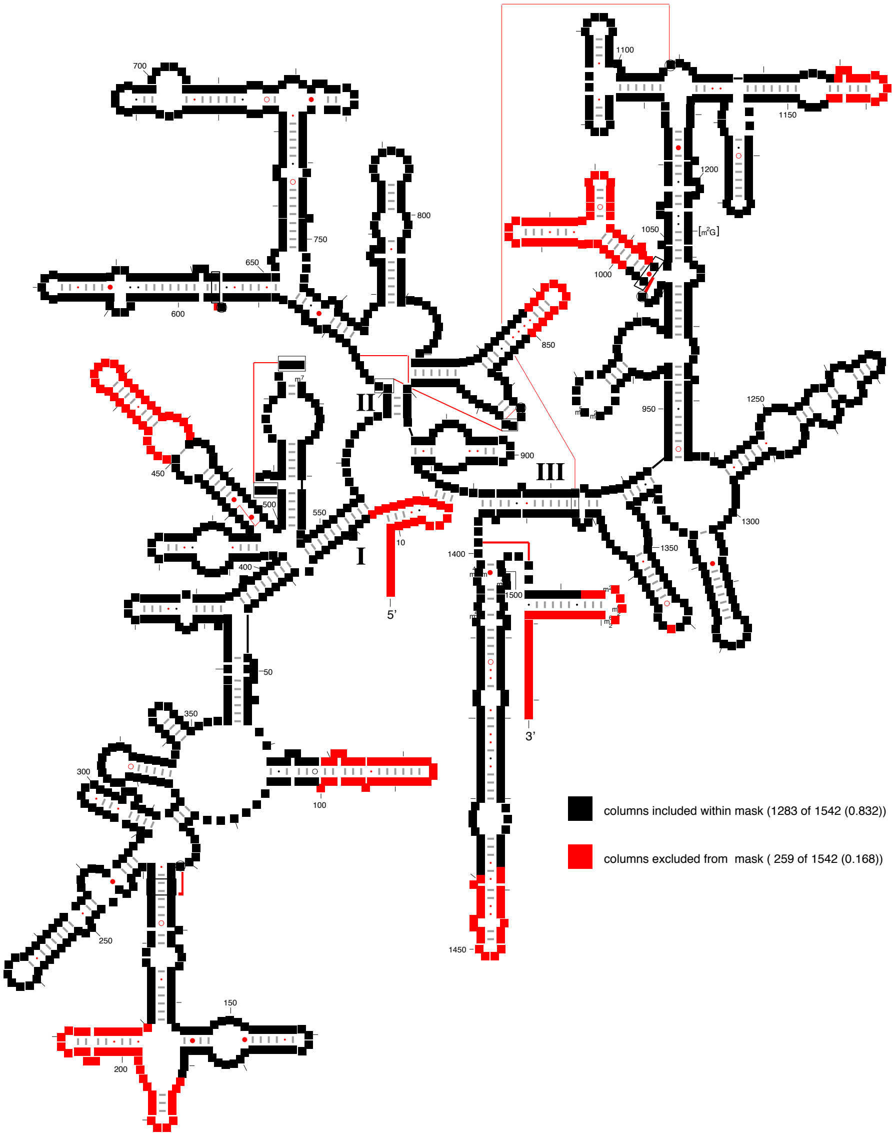


Figure 7.5: **David Lane's SSU alignment mask overlaid on the SSU secondary structure of *Escherichia coli*.** Black columns are included by the mask for phylogenetic analyses. Red columns are excluded by the mask for phylogenetic analyses. This figure was derived from the overlay of the Lane mask on the GREENGENES database's *Core Set* alignment of the *E. coli* sequence (GENBANK accession J01695) . It was generated using the SSU-ALIGN package described in Chapter 9. The secondary structure diagram layout is based on the CRW database [32].

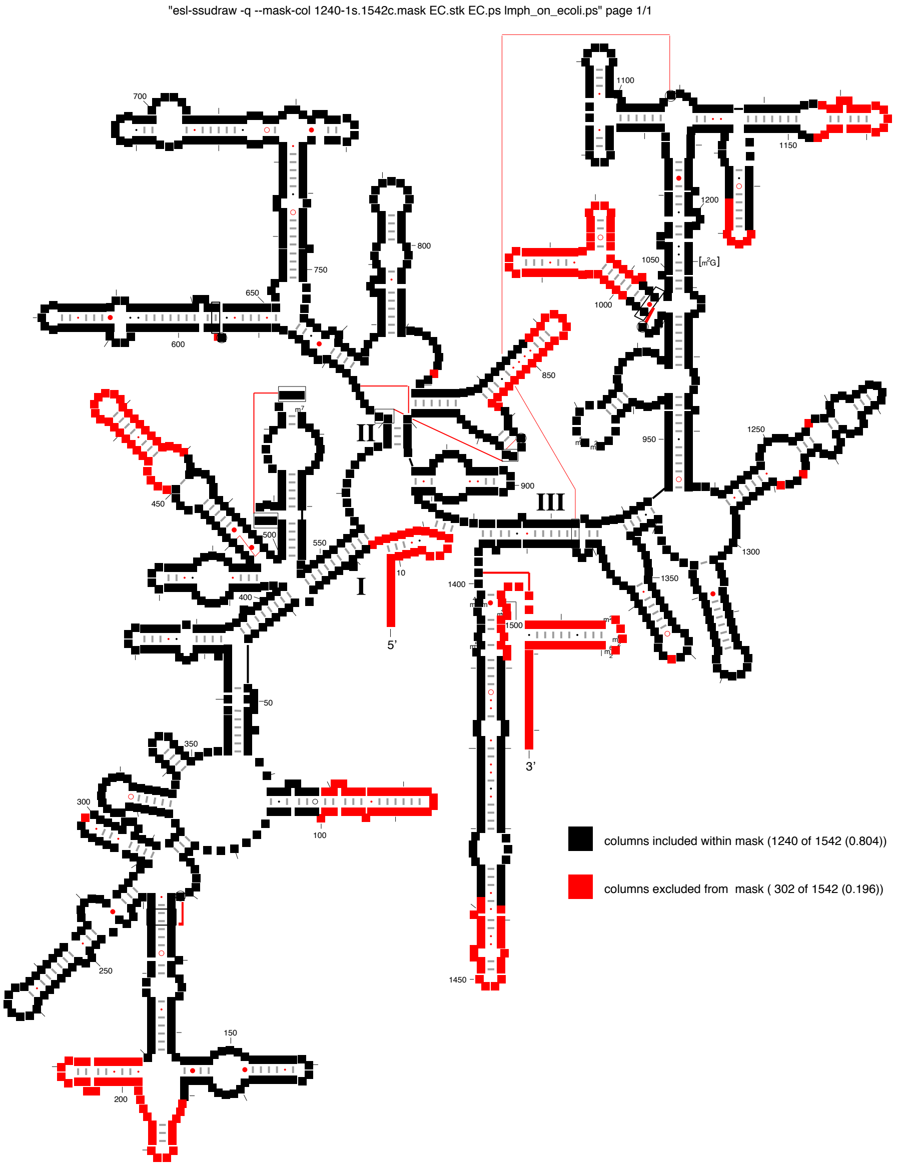

"esl-ssudraw -q --mask-col 1240-1s.1542c.mask EC.stk EC.ps lmph_on_ecoli.ps" page 1/1
700
1100
1150
800
750
650
1200
1050
[m²G]
1000
600
850
II
450
500
550
900
III
950
1250
1300
10
I
400
1400
1350
5'
1500
50
3'
350
300
100
■ columns included within mask (1240 of 1542 (0.804))
🟥 columns excluded from mask (302 of 1542 (0.196))
250
1450
150
200

Figure 7.6: **Phil Hugenholtz's SSU alignment mask overlaid on the SSU secondary structure of *Escherichia coli*.** Black columns are included by the mask for phylogenetic analyses. Red columns are excluded by the mask for phylogenetic analyses. This figure was derived from the overlay of the "LMPH" mask on the GREENGENES database's *Core Set* alignment of the *E. coli* sequence (GENBANK accession J01695) . It was generated using the SSU-ALIGN package described in Chapter 9. The secondary structure diagram layout is based on the CRW database [32].

Other uses of SSU alignments

Though most SSU analyses are performed in conjunction with environmental surveys, some are specifically aimed at resolving phylogenies. These include the first SSU analyses by Woese [276, 279, 281] as well more recent ones by molecular systematicists, including an examination of bumble-bee phylogeny [31], anaplasma phylogeny [152] and protostome phylogeny [175], to name just a few. These studies are often based on SSU alignments as well as other alignments, such as LSU rRNA and highly conserved protein sequences.

Additionally, SSU alignments have other uses besides serving as input to phylogenetic inference programs. For example, they have been instrumental in developing and refining the existing canonical models of SSU structure [32, 278, 280], which can lead to important functional insights. Also, SSU alignments aid primer design for SSU surveys by facilitating identification of conserved sequence regions specific to a given phylogenetic clade [8]. Finally, SSU alignments provide useful datasets for training RNA sequence analysis programs; the RIBOSUM RNA scoring matrices used by the RSEARCH program described in Chapter 1 were derived from SSU and LSU rRNA alignments.

7.5 SSU alignment methods

Due to the prevalence and scale of SSU-based environmental surveys, several databases dedicated to SSU sequence analysis have been developed. Nearly all of these databases use their own alignment tool for aligning the SSU sequences they contain. These tools differ mainly by alignment strategy. This section briefly discusses these different strategies, and the next section provides more detail about the the most widely used databases and their respective alignment tools.

The gold standard: manual alignment

Despite all of the advances in computational methods for sequence alignment in the past 30 years, the most reliable alignments are still manually created by expert curators. While a computer program is often very good at getting an alignment almost right, if accuracy is of paramount importance then it is still necessary for an expert to check over the output alignment. One of the reasons human experts outperform computers is that a human can easily take into account extra information unavailable to most computer programs. A good example is the conserved secondary structure of SSU, which most alignment programs are ignorant of. Higher-order structural contacts (tertiary interactions) are another example - no existing alignment programs take these into account. Further, a human has access to databases of existing alignments with sequences covering the entire tree of life and, more importantly, the capacity to intelligently mine that data as needed. The curator can extract similar sequences as needed when computing the alignment, essentially performing simultaneous phylogenetic inference and alignment, which current automated methods have difficulty with.

The problem with manual alignment is that it is time consuming. When only a few dozen SSU sequences existed in the 1980s, they could all be aligned manually in a reasonable amount of time. This was still true in the early 1990s when several hundred sequences were available [145]. Today, with several hundred thousand sequences being generated per year (Figure 7.2), it is clearly impractical. For this reason, nearly all SSU surveys use an automated alignment computer program to create alignments.

There are several different computer programs that are used. The main similarity between them all is that they take advantage of a manually created reference, or *seed*, alignment when aligning new *target* sequences. The main difference between the programs is the specific manner in which the seed alignment is used. There are two classes of programs. *Profile*-based programs align new sequences to a statistical model (a profile) that represents the diversity in the entire seed alignment. *Nearest-neighbor* programs use a small, carefully selected subset of the seed sequences when aligning new sequences (Figure 7.7).

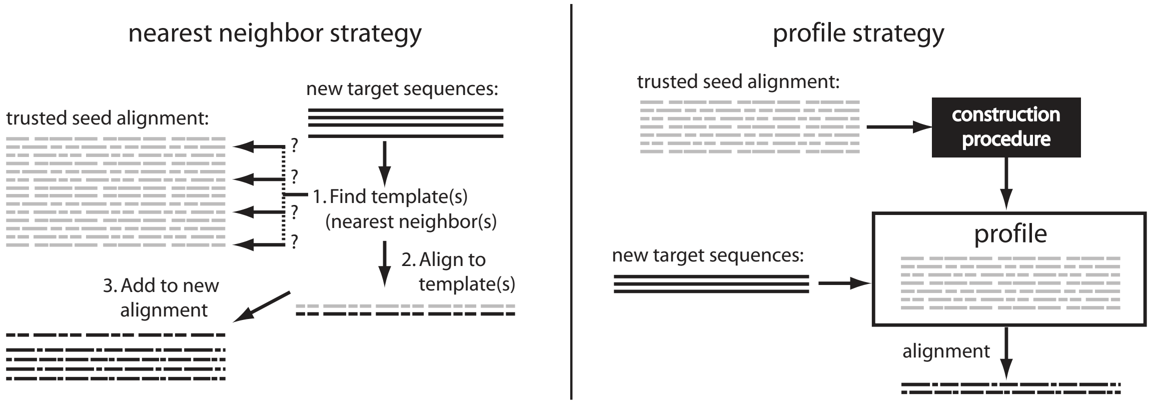


Figure 7.7: Schematic of nearest-neighbor and profile based alignment strategies.

Nearest-neighbor strategy

Given a new target sequence to align, the nearest-neighbor strategy proceeds through two steps. First, one or more *template* sequences, or nearest-neighbors, are selected from the seed alignment. These are the seed sequences most similar to the target by some criterion (for example: most matching 7-mers (oligonucleotides of length 7)). The second step is the calculation of the alignment of the target sequence to the template(s). The non-template seed sequences are ignored in the alignment step.

Importantly, if only one template sequence is chosen then the alignment is a simple pairwise alignment to that template. In this case, the program has no information regarding the varying levels of expected conservation, or the likelihood of insertions or deletions, at different positions of the alignment. (Profile-based alignment programs, however, as described next, do have access to this information.) This is information that an expert curator would almost certainly take into account when computing the alignment.

Of course if the template is 100% identical to the target, the alignment is trivial and the method used is irrelevant. However, as the identity between template and target decreases the reliability of a pairwise alignment strategy decreases.

The existing SSU nearest-neighbor tools differ in the number of template sequences they use and in the specific scoring system used to calculate an alignment as discussed in more detail below. All of these tools use only primary sequence information to calculate an alignment, i.e. they do not explicitly score how well the proposed alignment agrees with a model of SSU secondary structure. However, there is nothing inherent to the nearest-neighbor strategy that prevents it from modeling structure.

Profile strategy

An alternative to nearest-neighbor based approaches is to align a sequence to a statistical model called a *profile* that is built from a multiple sequence alignment of a representative set of sequences (the seed alignment). Profiles are routinely used for homology search as discussed in Chapter 1, for which they are generally considered among the most powerful tools available. Homology search with profiles requires scoring sequences by aligning them to the profile. Because of this, it is trivially simple to modify profile homology search programs to create multiple alignments, and the widely used HMMER, SAM and INFERNAL packages are able to create alignments as well as perform searches.

Profile-based alignment is used by some large and popular non-SSU sequence databases including PFAM [76], RFAM [89], and SMART [151]. These databases use the "seed-full" strategy for building and maintaining multiple alignments using profiles. A small set of typically 50 or so representative sequences are chosen and aligned with manual curation to create a *seed* alignment. A profile is built from the seed alignment and used to align all other examples of the sequence family (potentially found in a database search using the profile) to create a *full* alignment. If the seed sequences are indeed representative of the sequence diversity in the family, the model is typically able to accurately align the target sequences. A single SSU database, the RDP database, uses profile based alignment to a seed, as discussed below.

Though presented as two distinct classes, profile and nearest-neighbor based methods can also be viewed as two extremes on a continuum. As the number of template sequences used by a nearest-neighbor based approach increases, the method becomes increasingly similar to a profile strategy. By using more than one template, the program has position-specific information. For example, an `A` in the target sequence should align with a higher score to a position that is 100% `A` in the template sequences, than to a position that is 25% `A`, `C`, `G`, and `U` in the template sequences.

Important considerations regarding alignment strategies

1. Error propagation. Errors in the seed alignment are likely to propagate during the alignment of target sequences. In the nearest-neighbor approach, an error in the alignment of template sequence x is likely to propagate on to the alignment of any new sequence y that uses x as a template.

2. Running time for nearest-neighbor template selection scales with the size of the seed alignment. Finding the appropriate template sequence requires some type of comparison between the new sequence and each candidate template sequences, so as the number of candidate templates increases, so does the number of computations required to pick the templates. For a profile, no such step is required, once the profile is built, each sequence is independently aligned directly to it.

3. Aligning novel sequences. With the nearest-neighbor strategy, as the similarity between the target and template sequence(s) decrease, the probability of alignment errors in the target sequence alignment increase. Similarly for profiles, as the target sequence becomes increasingly different from all of the seed sequences, it becomes more difficult for the profile to accurately align the sequence. However, a profile is more general than any nearest-neighbor approach because it encapsulates the diversity of the entire seed alignment, and so may be better able to accurately align novel sequences. Reliable alignment of novel sequences is crucial because they are the interesting sequences in SSU surveys (the more divergent, the more interesting) and are continually being discovered. During manual alignment, an expert curator would expend disproportionate effort when aligning novel sequences, taking structure into account where necessary.

These considerations have implications on the desired number of sequences in the seed alignment, especially for the nearest-neighbor based methods. Considerations #1 and #2 suggest that the number should be low, to make the construction of a manually curated seed alignment with minimal errors feasible and to limit the time required to define the templates. However, consideration #3 argues that the seed alignment should contain a large number of sequences (or at least a sufficiently dense representative set) to minimize the probability that any target sequence is significantly different from all the seed sequences.

In practice, the size of seed alignments used by nearest-neighbor based and profile based SSU alignment tools differ dramatically (Table 7.2). The nearest-neighbor based approaches for SSU alignment all use seeds with thousands of sequences, while the only existing SSU profile based database (RDP, described below) uses two profiles built from seeds of about 500 and 80 sequences respectively. The profile-based PFAM and RFAM databases include some alignments of hundreds of thousands of sequences created with profiles that were built from seeds of a few hundred sequences or less.

If the alignment accuracy of the two strategies is comparable, this presents a clear and important advantage of profile-based methods. Manually constructing a highly refined, accurate seed of a hundred or so sequences is easier than constructing one with thousands of sequences.

Structural SSU alignment using profile SCFGs

Unlike for nearest-neighbor based methods, there are existing profile-based SSU alignment tools that can explicitly take into account conserved secondary structure during alignment. Stochastic context-free grammars (SCFGs) are probabilistic models well-suited to modeling the well-nested structure and sequence conservation of RNAs. The INFERNAL package implements profile SCFGs called *covariance models* (CMs, introduced in Chapter 1) that model the consensus structure and sequence of a particular RNA family [190]. The RNACAD package is another implementation of profile SCFGs by Michael Brown and David Haussler [24].

Profile SCFGs are probabilistic models that can directly calculate confidence estimates of the alignment ambiguity given the model for each aligned residue in output alignments.

These confidence estimates could be used to automatically determine alignment-specific masks for removing ambiguously aligned regions prior to phylogenetic inference.

However, SCFG-based alignment is much more computationally expensive than primary sequence-based alignment. This is especially true for INFERNAL. In 2001, prior to development of version 0.55 of INFERNAL, alignment of a single SSU sequence required more than 22 Gb of RAM, making it infeasible on modern computers. Sean Eddy solved the memory problem with version 0.55 by extending the Myers-Miller linear memory dynamic programming trick [186] to CMs, reducing the required RAM to 67 Mb. However, version 0.55 still requires about 15 minutes to align a single SSU sequence. RNACAD uses constraints from a first pass sequence-only based alignment to accelerate alignment, and requires about 30 seconds to align a single SSU sequence [24]. In Chapter 8, I describe the application of an acceleration technique like Brown's to INFERNAL, which reduces SSU alignment time to about 1 second per sequence (timings are for a single Intel Xeon 3.0 Ghz processor).

7.6 Dedicated SSU databases and alignment tools

The most widely used and cited SSU databases and the alignment programs they use are listed in Table 7.2. I will briefly describe each database and the alignment technique it uses below. Then I will discuss my motivation for developing a new alignment program based on CMs.

Many of these databases provide useful tools other than alignment programs that are helpful to researchers doing SSU analysis, including alignment-independent classification of sequences, design of primers, and quality checking of aligned sequences. These are listed but not explained in the descriptions below.

Arb

ARB is a freely available software package consisting of a set of interacting tools for maintaining and analyzing a database of sequence data [164]. It is a general tool that can be used for any nucleic acid or protein family, but it is most well-used and known for SSU anal-

database	ARB	CRW	GREENGENES	SILVA	RDP
latest ref	[164]	[32]	[47]	[214]	[40]
year of incept	~1994	2000	2005	2007	1991
date of 1^{st} pub	Feb 2004	Jan 2002	July 2006	Oct 2007	Apr 1991
total # citations	1512	568	283	83	4738
avg # cites/year	288.0	76.6	97.0	49.8	262.0
# seed seqs	N/A	?*	4,938	51,601	508 (bac); 79 (arc)
# total seqs	N/A	38,723	397,006	868,390	920,643
alignment tool	FAST ALIGNER	?*	NAST [46]	SINA	INFERNAL [190] (prev. RNACAD) [24]
alignment tool availability	free to download	not available	free via web server	free via web server (<= 300 seqs)	free to download
alignment strategy	NN	NN	NN	NN	profile
# templates	1	?*	1	up to 40	N/A
manual curation?	possible	sometimes	no	no	no
archaea	N/A	+	+	+	+
bacteria	N/A	+	+	+	+
eukarya	N/A	+		+	

Table 7.2: **Summary of SSU rRNA sequence databases and alignment strategies.** The five most popular, actively maintained SSU databases are listed. More details on databases and alignment strategies are in the text. Sequence counts pertain solely to SSU sequences. Alignment strategy: "NN" stands for nearest-neighbor based alignment. "Manual curation" refers to the master alignments, not the seeds, which were all manually curated. Citation counts are from Google Scholar (http://scholar.google.com) searches performed June 30, 2009. GREENGENES citation count is sum of both [47] (database publication) and [46] (alignment tool publication). Average citations per year were calculated by dividing total number by number of full months since the first publication for the database. RDP citations are sums from 13 separate publications [37–40, 146, 168–173, 197, 198] The ARB column has "N/A" in many rows because it is a software package for creating a personal database as opposed to a centralized database. (*) For CRW, details on the size of the seed alignments, number of NN templates, and alignment tool are unclear from [32].

ysis. ARB includes phylogenetic inference tools, an alignment tool, and visualization tools for manually checking and refining the alignment based on a consensus secondary structure model of SSU [32] as well as an existing database of aligned sequences including taxonomic information, to which new sequences can be added by the user.

The ARB FAST ALIGNER is used to add new sequences imported by the program to a pre-existing seed alignment using a nearest-neighbor based approach. For each new target sequence to be aligned, the single template sequence from the seed that is most similar to the target is chosen. A pairwise alignment is computed and used as the basis for merging the target into the seed alignment. The specifics of the alignment algorithm and the template selection algorithm seem to be unpublished.

Greengenes

The GREENGENES website and database is maintained by a team of researchers at the Lawrence Berkeley National Laboratory headed by Gary L. Andersen [47]. Each periodic update of the database includes an alignment of all currently available full length, or nearly full length (> 1250 residues) bacterial and archaeal sequences from GENBANK [12]. In addition to being in aligned form, the GREENGENES data provide four important advantages over their GENBANK versions: screening for artefactual chimeric sequences, standardization of description fields for author's annotations, assignment of taxonomy from several independent expert curators, and easy integration with ARB. With each update, an ARB compatible database file is generated, allowing ARB users to keep their local databases up to date.

With each database update, the NAST alignment program (Nearest Alignment Space Termination) [46] is used to align any new GENBANK SSU sequences deposited since the previous update. Like ARB's FAST ALIGNER, NAST uses a nearest-neighbor based approach to perform pairwise alignment of target sequences to a single template sequence. The specific template for each target is selected from the seed alignment, called the *Core Set*, as the sequence with the most matching 7-mers with the target. The Core Set contains roughly 5,000 sequences, about 4,800 of which are bacterial and 200 are archaeal. NAST always returns aligned target sequences in the pre-specified width of the seed alignment: 7,682 columns. If an alignment of a target implies an insertion of a gap outside of the 7,682 format, a local misalignment is introduced to prevent from adding the gap column. This is to allow consistent annotation of position-dependent features such as primer positions, column masks and secondary structure features by maintaining exactly 7,682 columns. NAST is freely available for use through a web server, but not for download.

Comparative RNA Website

A group at the University of Texas at Austin headed by Robin Gutell has maintained the COMPARATIVE RNA WEBSITE (CRW) since 2000. CRW contains sequences, alignments, structures, and conservation information for 5S, 5.8S, SSU, and LSU ribosomal RNAs as well as self-splicing group I and group II introns and tRNAs. Gutell worked together with Carl Woese, Harry Noller and others to create one of the first SSU secondary structure models in 1980 based on chemical modification, nuclease susceptibility, and comparative analysis to identify covarying positions that were plausible base-pairing interactions [278]. Since then he has continued to work on SSU comparative analysis and has updated and validated the structural models as new sequences and crystal structures have become available [107]. Due largely to Gutell's expertise, CRW stands out as the database with the most trustworthy SSU structural information. It includes several hundred expertly predicted secondary structures of different SSU sequences spanning all three domains of life. The CRW SSU structural models have been used by the RDP database as a consensus structure for building profile SCFGs as described below.

The sequences in the CRW alignments represent all major branches on the tree of life [32], but are not synchronized to include all sequences from a major sequence database like GREENGENES, SILVA, and RDP. Though the algorithmic details are unclear, the description of their alignment procedure suggests that a nearest-neighbor based approach is used for most sequences, but that regions of the alignment which occur in variable sequence regions of the SSU comparative model are manually checked and/or revised prior to acceptance into the database. Extra care is taken when aligning novel sequences. Expert manual structure prediction of individual sequences is performed in conjunction with the alignment, with each exercise informing the other [32]. The CRW alignment program is not publicly available for download or use through a website.

Silva

The SILVA database from the Microbial Genomics Group at the Max Planck Institute for Marine Microbiology in Bremen, Germany, maintains quality-checked alignments of SSU and LSU rRNA, associated taxonomic information and other annotations compatible with ARB. SILVA uses the SINA alignment program to compute two types of alignments: a *Parc* alignment of all SSU or LSU sequences from the EMBL database (of the same release number) greater than 300 residues, and a *Ref* alignment with a subset of the *Parc* sequences. To be included in *Ref*, a sequence must be assigned a quality score above a minimum threshold and be greater than a minimum length of 1200 (for Bacterial and Eukaryotic SSU), 900 (for Archaeal SSU). The quality scores are assigned based on the alignment as described below.

Given a target sequence to align, SINA first chooses up to 40 template sequences that are most similar to the target using a suffix tree data structure. The template sequences are chosen from a seed alignment of 51,601 aligned SSU sequences from all three domains that contains 46,000 alignment columns (this alignment width maintains consistency with the *ssu_jan04.arb* database released by the ARB project in 2004). Because the "quality of the final datasets critically depends on the quality of the seed alignments", the seed was extensively cross checked by multiple expert curators, and "any sequences that could not be unambiguously aligned were removed from the seed" [214]. Once the templates are chosen, they are transferred to a partial-order graph [148] and a variant of the Needleman-Wunsch [193] alignment algorithm is used to align them with affine gap penalties. Having the templates organized in a partial graph allows the alignment to switch between the possible template sequences for different regions of the alignment, depending on which template includes the highest scoring match to a particular region of the target. It is unclear how the algorithm guarantees a width of exactly 46,000 columns. (If the alignment between templates and the target is shorter (fewer than 46,000 columns) then it is trivial to expand the alignment by adding gaps, but if the alignment width exceeds 46,000 columns the solution is less obvious.)

Following alignment, a quality score is calculated for each target based on a *variability statistic* and a *base-pair score*. The variability statistic measures the similarity of the target to its most similar template sequence. The assumption here is that sequences that were highly identical to a template will be correctly aligned. The base-pair score reflects the agreement of the implicit base-pairs in the aligned target sequence with Gutell's secondary structure model from CRW [32]. (Note that the structure model was *not* used to calculate the alignment, just to assess its quality.) The variability and base-pair scores are normalized and combined into a quality score of 0 to 100. Sequences with quality scores below 50 are removed from the *Ref* datasets.

The SINA aligner is available for use through a web server, but not available for download. As of this writing, the maximum number of sequences that can be aligned at once using the web server is 300.

Ribosomal Database Project

The Ribosomal Database Project (RDP) was the first large SSU database. It was started in 1991 by Gary Olsen, Niels Larsen and Carl Woese [197] at the University of Illinois. In December 1997, the database moved to its current location at the Center for Microbial Ecology at Michigan State University, where it is headed by Jim Cole. Woese and Olsen were affiliated with the database (co-authors on publications) until 1999 [171] and 2001 [173], respectively.

RDP currently contains the most aligned SSU sequences of any database (Table 7.2). It is updated monthly from the International Nucleotide Sequence Database Collaboration (INSDC, which includes GENBANK, DDBJ and EMBL). In addition to alignments, RDP includes tools for chimera checking, alignment-independent or alignment-dependent classification of sequences, tree building, and even an assignment generator that creates lesson plans related to SSU sequence analysis for professors [40].

RDP's alignment strategy is unique amongst SSU databases in several ways. First, RDP uses an independently developed general alignment tool that was not developed specifically for the database. From 2000 until 2008, RDP used the RNACAD software, a profile SCFG-based RNA modeling software package applicable to any RNA family, developed by Michael Brown and David Haussler at the University of California, Santa Cruz [24]. In the middle of 2008, RDP switched to using INFERNAL [190], another profile SCFG implementation developed by Sean Eddy, myself and Diana Kolbe. INFERNAL adoption was facilitated by the alignment acceleration techniques described in Chapter 8 of this work. Secondly, RDP is the only database that uses a profile-based alignment method instead of a nearest-neighbor based approach. Finally, by using profile SCFGs to compute alignments, it is the only database that incorporates scoring of both sequence conservation and structure conservation.

When it used RNACAD, RDP only maintained bacterial alignments. The bacterial seed alignment used to build the profile for RNACAD alignment consisted of just 34 sequences and the consensus structure was based on the secondary structure model of Gutell as of 2000 [107], which is an updated version of the 1980 structure published by Woese, Gutell, Noller and others [278]. Currently, RDP includes two profiles, a separate archaeal and bacterial model, built and used with INFERNAL. The archaeal model was built from a seed of about 80 sequences, and the bacterial model was built from a seed of about 500 sequences. The consensus secondary structure used for the model is still based on that of Gutell and colleagues [32].

7.7 Developing a new SSU alignment tool

In 2005, I decided to create a new program for large-scale SSU alignment as part of my thesis project. This decision was based on an analysis of existing SSU databases and their alignment tools combined with the recognition of the explosive growth rate in the scale and popularity of SSU surveys (Figure 7.2). (Note that in 2005 RDP was not yet using INFERNAL.) The proposed program had the following design goals:

1. **Scalable and fast** - to potentially create alignments of millions of sequences.
2. **Accurate** - by scoring both the conserved sequence and structure of SSU.
3. **Profile-based** - so as not to require the large manually curated seed alignments of thousands of sequences that nearest neighbor based methods typically use.
4. **Cover all three domains** - using separate seed alignments and corresponding profiles for archaeal, bacterial and eukaryotic SSU sequences that could potentially be refined or split into more specific alignments (that covered a tighter phylogenetic range) by users.
5. **Able to generate alignment-specific masks** - to prune ambiguously aligned regions of the alignment.
6. **Freely available, extensible and documented** - to help promote wide use in the community.

I decided to use CMs as the basis for the program because the current version of INFERNAL, 0.55, already met nearly all of these goals. It is profile-based, scales well, is freely available, and uses structure to compute its alignments. To extend it to achieve all of my goals would involve constructing the seed alignments to achieve goal 4 and addressing the key problem of the slow speed of CM alignment algorithms.

I have accelerated CM alignment using a banded dynamic programming technique (described in Chapter 8). The adoption of INFERNAL by RDP in 2008 served as an important indication that fast CM-based alignment of SSU was indeed useful to the community. However, INFERNAL as used by RDP does not meet design goals 4 or 5, nor does it contain specific documentation to help users create SSU alignments. I decided to complete these goals and create a new program called SSU-ALIGN that requires and uses INFERNAL for alignment but also contains SSU seed alignments and profiles for all three domains, a User's Guide devoted to SSU analyses, and the ability to probabilistically mask alignments. The SSU-ALIGN program is described in Chapter 9.

Chapter 8

HMM banding for faster structural RNA alignment

The standard CM CYK dynamic programming (DP) algorithm calculates the optimal alignment of a target sequence to a CM [53]. Unfortunately, the algorithm is computationally expensive. The time required to run CYK scales with $N^3 \log(N)$ and the required memory scales with $O(N^3)$ for a model of N consensus sequence positions. Aligning a single SSU sequence requires about 15 minutes (as a single thread on a 3.0 GHz Intel Xeon) and over 20 Gb of RAM. The memory complexity problem was solved in 2002 through implementation of a divide-and-conquer version of CYK (similar to the Hirshberg or Myers/Miller algorithm), which reduced the memory required for SSU alignment to about 60 Mb [57]. But the time complexity problem still remains, prohibiting the practical use of CMs for large-scale SSU alignment.

8.1 Faster alignment using banded dynamic programming

A common way to accelerate DP algorithms is by constraining the DP matrix using *bands* derived from a faster, heuristic alignment of the target sequence (Chapter 1 and Figure 1.6). During the DP recursion, cells outside the pre-calculated bands are ignored to save time.

Banded DP approaches include BLAST [3], FASTA [208] and LAGAN [25] among others. Chapter 2 of this work describes a banded DP approach for accelerating the CM CYK search algorithm called query-dependent banding (QDB). The bands are calculated using only the probabilities of the query model and are applied to constrain the CYK DP recursion. QDB provides a modest average speedup of about four-fold, but the acceleration increases with family size and is about 30-fold for SSU. However, there is an alternative banding strategy that can lead to even greater acceleration.

In 2000, Michael Brown, a recent graduate from David Haussler's group at UC Santa Cruz, published a banded DP method for accelerating profile SCFG alignment using bands derived from a first-pass profile HMM alignment of the target sequence [24]. Brown's technique uses the Forward and Backward HMM algorithms to determine the posterior probability that each HMM state emits (aligns to) each residue of the target sequence given the parameters of the HMM [53]. Given these posterior probabilities, a band is determined for each HMM state defining the range of target sequence positions that align to the state with probability above some minimum threshold. The bands are then transferred from each HMM state to its analogous state in the profile SCFG and enforced during subsequent SCFG alignment. Brown implemented his banding technique in the program RNACAD and demonstrated its utility for SSU alignment [24]. Since that initial publication in 2000, no further development of the program has taken place. RNACAD was used to align SSU sequences by the RDP database from 2000 until 2008.

I have reimplemented Brown's HMM banding technique in INFERNAL as described below. The resulting acceleration versus standard CYK is nearly 2000-fold for SSU with a negligible affect on alignment accuracy (Table 8.2). In 2008, RDP switched to using INFERNAL for SSU alignment, partly because it is about 25-fold faster than RNACAD [40]. The description of HMM banding in this chapter corresponds to the implementation in INFERNAL versions 1.0 and 1.01 (the most current as of this writing).

8.2 HMM banded alignment in Infernal

HMM banded alignment of a target sequence with INFERNAL consists of four steps:

1. Calculate the posterior probability that each target residue aligns to each HMM state using the Forward and Backward algorithms.
2. Determine the band of possible residues that could align to each HMM state that excludes a given amount of probability mass.
3. Transfer the HMM bands onto the CM CYK dynamic programming matrix.
4. Align the target sequence to the CM using a banded CYK algorithm that skips computations outside the bands.

The result is the optimal alignment of the target that is consistent with the bands. The goal of HMM banded alignment is to derive bands that constrain, and thus accelerate, the CM alignment as much as possible while still containing the globally optimal alignment that the non-banded CYK algorithm would find. The approach therefore depends on an appropriately constructed HMM for the posterior probability calculation and resulting band determination. INFERNAL uses a specific type of profile HMMs called *CM Plan 9 (CP9) HMMs* that are closely related to Weinberg and Ruzzo [265]'s *maximum-likelihood (ML) heuristic HMMs.* I will delay discussion of the construction and parameterization of CP9 HMMs until after a more detailed explanation of the four steps of HMM banded alignment. For now, the relevant details are that CP9s are composed of a series of N nodes, one for each consensus position modeled by the CM, with one match, insert, and delete state per node, and that a mapping exists between corresponding HMM states and CM states based on the consensus positions they correspond to. In this discussion, k^M, k^I and k^D are used to refer to the HMM match, insert and delete state, respectively, of HMM node k, and k^* is used generically to refer to any state in node k.

Step 1. Calculate the posterior probability that each target residue aligns to each HMM state using the Forward and Backward algorithms.

INFERNAL includes implementations of the standard Forward and Backward HMM algorithms [53] adapted for the CP9 profile HMM architecture (Figure 8.2). Forward and Backward are DP algorithms that recursively calculate $F_{k^*}(s)$ and $B_{k^*}(s)$, respectively, for all states k^* of the model and all L residues $x_1..x_s..x_L$ of target sequence x. $F_{k^*}(s)$ is the summed probability of all alignments of the target subsequence $x_1..x_s$ to the model up to and including the alignment of x_s to state k^* of the model. $B_{k^*}(s)$ is the summed probability of all alignments of the target subsequence $x_s..x_L$ that include the alignment of x_s to state k of the model [53].

The values in fully calculated Forward and Backward DP matrices are then used to derive the posterior probability $P(\pi_s = k^*)$ that each target residue s aligns to each state of the model ($* = M$, I, or D) [53]. This step requires knowing $P(x)$, the summed probability of all full alignments of x to the model, which can be calculated as the sum of Forward scores for position L over all possible end states of the model (in this case, $P(x) = F_{N^M}(L) + F_{N^D}(L) + F_{N^I}(L)$). The meaning of these posterior probabilities is similar for emitting states (matches and inserts), but is different for delete states:

equation	meaning
$P(\pi_s = k^M) = F_{k^M} + B_{k^M} - P(x)$	the probability that residue x_s was emitted from (aligns to) state k^M
$P(\pi_s = k^I) = F_{k^I} + B_{k^I} - P(x)$	the probability that residue x_s was emitted from (aligns to) state k^I
$P(\pi_s = k^D) = F_{k^D} + B_{k^D} - P(x)$	the probability that residue x_s was the last emitted residue (aligned residue) when state k^D was entered

Step 2. Determine the band of possible residues that could align to each HMM state that excludes a given amount of probability mass.

The posterior probabilities are then used to define bands of sequence positions, smin(k^*)..smax(k^*), that have a non-negligible probability of aligning to each state k^*. Here, we define a parameter τ (set by default as 10^{-7}) as the threshold for the negligible probability mass that is allowed outside each band. The values of smin(k^*) and smax(k^*) are determined such that the cumulative probability in the left and right tails of the posterior probability distribution is less than $\frac{\tau}{2}$:

$$\sum_{s=1}^{\text{smin}(k^*)-1} P(\pi_s = k^*) < \frac{\tau}{2},$$

$$\sum_{s=\text{smax}(k^*)+1}^{L} P(\pi_s = k^*) < \frac{\tau}{2}.$$

This step is very similar to the determination of the query-dependent d bands (dmin(v)..dmax(v)) for the QDB algorithm described in chapter 2. However, with QDB, the summed probability mass of all possible d values for each CM state is guaranteed to be 1.0. There is no such guarantee here. The sums are constrained as follows:

$$0.0 <= \sum_{s=1}^{L} P(\pi_s = k^*) = P(k^*) <= 1.0$$

Importantly, this means that $P(k^*)$ can be less than τ (or even $\tau/2$), in which case smax(k^*) < smin(k^*), and there are 0 sequence positions within the band. This is a special case in which the the probability of using state k^* in the alignment (for *any* residue) is below τ and thus negligible by our definition. This case may allow all DP recursions for the CM state that maps to k^* to be ignored during calculation of the banded CM alignment in step 4.

Alternatively, a normalization step can be added before the bands are determined:

$$P({\pi_s}^n = k^*) = \frac{P(\pi_s = k^*)}{P(k^*)} \tag{8.1}$$

This guarantees that $(\sum_{s=1}^{L} P({\pi_s}^n = k^*)) = 1.0$. If these ${\pi_s}^n$ values are used in place of the π_s values during the calculation of smin(k^*) and smax(k^*), all states are guaranteed to include a band of at least 1 residue. Bands defined in this way will be looser in general (include more sequence positions) than those defined by the non-normalized method. Looser bands lead to slower alignments, but potentially increase the chance that the optimal CM alignment will be found in step 5. Both versions are implemented in INFERNAL, with the non-normalized version being the default because it yields greater speedups while nearly always resulting in the identical alignment as the normalized version as shown in the benchmark results later in this chapter.[1]

Step 3. Transfer the HMM bands onto the CM CYK dynamic programming matrix.

The HMM sequence bands smin..smax are then transferred onto the three-dimensional CYK matrix α to facilitate banded CYK alignment in step 4. During CYK alignment, the value in $\alpha_v(j,d)$ is the log probability of the parse subtree rooted at CM state v generating (aligning to) the target subsequence $x_i..x_j$, where $i = j - d + 1$. Unlike an HMM matrix DP cell, which corresponds to just one sequence position, an α DP cell corresponds to *two* sequence positions: i and j. HMM bands are transferred onto the CYK matrix by defining i and j bands for each CM state v. An HMM sequence band for k^* corresponds to either a band on i or a band on j for the CM state v that k^* maps to. A CM state v and HMM state k^* map to each other if they either: emit to, insert after, or delete the same consensus position. (CM and HMM state mapping is described in more detail later.) Transferring an HMM sequence band to a CM i or j band simply means copying its values, for example:

$$\text{imin}(v) = \text{smin}(k^*) \qquad \text{imax}(v) = \text{smax}(k^*)$$

---

[1] The normalized version is enabled using the command-line option `--sums` to the program `cmalign`.

node type	state type	i band	j band
MATP	MP	k^M	k^M
MATP	ML	k^M	k^D
MATP	MR	k^D	k^M
MATP	D	k^D	k^D
MATP	IL	k^I	-
MATP	IR	-	k^I
MATL	ML	k^M	-
MATL	D	k^D	-
MATL	IL	k^I	-
MATR	MR	-	k^M
MATR	D	-	k^I
MATR	IR	-	k^D

node type	state type	i band	j band
BEGL	S	-	-
BEGR	S	-	-
BEGR	IL	k^I	-
ROOT	S	-	-
ROOT	IL	k^I	-
ROOT	IR	-	k^I
END	E	-	-
BIF	B	-	-

Table 8.1: **Transfer of HMM sequence bands to CM i and j bands.** The HMM state type whose sequence band is copied to define each CM state type's i and j band is shown. For example, a `MATP_ML`'s i band is copied from the HMM match state (k^M) that maps to it, and its j band is copied from the HMM delete state (k^D) that maps to it. A "-" indicates that the CM state type's i or j band is not set by directly copying an HMM's s band, but rather based on other CM state's bands as mentioned in the text.

Whether the sequence band is copied to be either the i or j band of the CM state v it maps to depends on the type of CM state v is, as shown in Table 8.1.

Transferring each HMM state's sequence band to its mapping CM state's i or j band defines many but not all of the CM state's bands (Table 8.1). Some CM states which do not map to any HMM state (as discussed later) will have undetermined i and j bands. These bands are determined based on other CM state's bands that were copied from HMM bands. For example, a CM `BEGL_S` state v does not explicitly map to any HMM state. Its bands are set as the minimal width bands that will encompass all possible parse subtrees in the set of states C_v that it can transition to:

$$\text{imin}(v) = \min_{y \in C_v} \text{imin}(y), \qquad \text{imax}(v) = \max_{y \in C_v} \text{imax}(y),$$

$$\text{jmin}(v) = \min_{y \in C_v} \text{jmin}(y), \qquad \text{jmax}(v) = \max_{y \in C_v} \text{jmax}(y).$$

Additionally, CM states that map to only one HMM state will have either undetermined i or j bands. These bands are determined based on nearby states as well. For example, if a `MATL` node is immediately followed by a `MATP` node, the `MATL_ML` state's j band is set as the j band from the `MATP_MP` state in the next node.[2]

The final step is to convert the i bands to j-dependent d bands using the formula: $d = j - i + 1$, so they can be easily enforced on the CYK matrix. For example, if state v has its bands defined as:

$$\begin{aligned} \text{imin}(v) &= 2, \qquad \text{imax}(v) = 3, \\ \text{jmin}(v) &= 4, \qquad \text{jmax}(v) = 5. \end{aligned}$$

The i bands are replaced by two new v and j-dependent d bands, $\text{hdmin}(v, j)..\text{hdmax}(v, j)$, one for each j in v's j band:

$$\begin{aligned} \text{hdmin}(v, 4) &= 3, \qquad \text{hdmax}(v, 4) = 4, \\ \text{hdmin}(v, 5) &= 3, \qquad \text{hdmax}(v, 5) = 4. \end{aligned}$$

Step 4. Align the target sequence to the CM using a banded CM CYK algorithm that skips computations outside the bands.

Given j bands and j-dependent d bands for the CM, a banded version of the standard CM CYK alignment algorithm can be executed that only performs the DP recursion for matrix cells within the bands, ignoring those outside the bands. This algorithm gives the optimal alignment that is consistent with the bands. The HMM banded CYK algorithm is given below using notation introduced in Chapter 2:

---

[2] I was unable to develop an elegant algorithm for setting these undetermined bands that makes them as tight as possible without decreasing accuracy. In my implementation, each state type is handled in a special way. See the code for details (`hmmband.c:cp9_HMM2ijBands()`).

Initialization (impose bands): for $v = M - 1$ down to 0, $j = 0$ to L:

for $d = 0$ to $\text{hdmin}(v, j) - 1$ $\quad \alpha_v(j, d) = -\infty$;

for $d = (\text{hdmax}(v, j) + 1)$ to L $\quad \alpha_v(j, d) = -\infty$.

Initialization at $d = 0$: for $v = M - 1$ down to 0, $j = 0$ to L:

if $\text{hdmin}(v, j) <= 0 <= \text{hdmax}(v, j)$:

$v =$ end state (E): $\quad \alpha_v(j, 0) = 0$;

$v =$ bifurcation (B): $\quad \alpha_v(j, 0) = \alpha_y(j, 0) + \alpha_z(j, 0)$;

$v =$ delete or start (D, S): $\quad \alpha_v(j, 0) = \max_{y \in C_v}[\alpha_y(j, 0) + \log t_v(y)]$;

else ($v = P, L, R$): $\quad \alpha_v(j, 0) = -\infty$.

Recursion: for $v = M - 1$ down to 0, $j = \text{jmin}(v)$ to $\text{jmax}(v)$,

$d = \max(1, \text{hdmin}(v, j))$ to $\text{hdmax}(v, j)$:

$v = E$: $\quad \alpha_v(j, d) = -\infty$;

$v = B$: $\quad \text{kmin} = \max(\text{hdmin}(z, j), (d - \text{hdmax}(y, j - k)), (j - \text{jmax}(y)))$,

$\text{kmax} = \min(\text{hdmax}(z, j), (d - \text{hdmin}(y, j - k)), (j - \text{jmin}(y)))$,

$\alpha_v(j, d) = \max_{\text{kmin} \leq k \leq \text{kmax}}[\alpha_y(j - k, d - k) + \alpha_z(j, k)]$;

$v = D, S$: $\quad \alpha_v(j, d) = \max_{y \in C_v}[\alpha_y(j, d) + \log t_v(y)]$;

else ($v = P, L, R$): $\quad \alpha_v(j, d) = \max_{y \in C_v}[\alpha_y(j - \Delta_v^R, d - (\Delta_v^L + \Delta_v^R)) + \log t_v(y)]$

$+ \log e_v(x_i, x_j)$.

When the algorithm completes, the value in $\alpha_0(1, L)$ contains the bit score for the optimal alignment of the full sequence $x_1..x_L$ that is consistent with the bands. As with other DP alignment algorithms, such as Smith-Waterman [236] or the Viterbi HMM algorithm [53], the actual alignment itself can be obtained by tracing back through the DP matrix, following the path of transitions that led to the optimal score.

Implementation of HMM banded CYK

Some details of the implemented version of the HMM banded CYK algorithm in INFERNAL differ from the simple version given above.[3] The most important difference pertains to the memory efficiency of the algorithm. As it is described above, the HMM banded CYK algorithm requires access to a full three-dimensional α DP matrix of size $M \times L \times L$ cells for a CM of M states and target sequence of length L. For large RNAs, the size of this matrix can be prohibitively large, requiring close to 23 Gb of RAM for SSU, as noted in [57]. The initialization step sets all cells outside the bands to $-\infty$. Most of these cells are never accessed again and are completely irrelevant to the alignment calculation. Empirically, for most alignments, the vast majority (often more than 99%) of the total DP cells lie outside the bands and so initialization can require a large fraction of the total running time. For improved memory and time efficiency, the implemented version only allocates matrix cells within the bands. This complicates an exact description of the implemented version relative to the simpler one above, but does not change its logic, which is why the simpler version is include here.

The implemented version must keep track of the indexing of a DP cell within the now much smaller α matrix, which depends on the size of the j band for the corresponding state, and the d band for the corresponding state and j index. Additionally, when updating a cell for state v in α based on the value in a cell for state y (the final two lines of the simple algorithm above), it is not sufficient to ensure that the cell for state v is within the bands (which the simple algorithm above *does* ensure), but it must also be known that the state y cell is within the bands (which the simple algorithm above *does not* ensure), because if not the state y cell does not exist in the matrix[4]

---

[3] See the `fast_cyk_align_hb()` function in the file `cm_dpalign.c`.

[4] The need to check that the state y cell is valid could be obviated by carefully allocating the matrix such that it only includes cells within the bands *and* any cell that may possibly be accessed during the banded DP recursion. I have not implemented such a version of the algorithm.

8.3 Comparison of HMM and query-dependent banding

The HMM banding and query-dependent banding (QDB, Chapter 2) strategies both enforce constraints on the α CYK matrix. They differ in how the bands are derived, the dimension(s) of the matrix the bands constrain, and their relative utility for the two applications of CYK - database search and sequence alignment.

QDBs are determined independent of the target sequence and constrain the d dimension of α - the subsequence lengths that can align to each subtree of the model. For example, QDBs limit the length of a consensus hairpin loop (Figure 1.1). In constrast, HMM bands are derived from an HMM alignment and so are sequence-dependent. They constrain not only the length of a given hairpin loop (the d dimension), but also its location in the target sequence (the j dimension). This means (given equal band widths) that HMM banding excludes more of the matrix than QDB and may lead to faster alignments. The width of the HMM bands depend on the sequence conservation between the target and the query model. Bands for targets with higher sequence conservation will have more well-resolved HMM alignments (with tighter posterior probability distributions) and consequently tighter bands. In contrast, QDBs are sequence-independent and do not vary with sequence conservation. For this reason HMM banding is more well suited to the alignment of homologous sequences, and QDB is more well suited to searching databases, in which the vast majority of the target sequence is nonhomologous to the model. The efficacy of QDB and HMM banding using two alignment benchmarks is discussed in the next section.

The HMM banded CYK alignment algorithm is very similar to the QDB CYK database search algorithm from the “QDB algorithm” section of chapter 2. The most obvious difference, mentioned above is the enforcement of bands on the j dimension of the matrix as well as the d dimension. Another difference is that in the HMM banded version the d dimension bands are dependent on not only the state v, but also on j. A more subtle difference is the way in which the calculation proceeds. In the HMM banded version the nesting order of the recursion iterates first over v, in the outermost loop, than over j, and finally over d in the innermost loop. In the QDB search version the ordering of the v and j loop are inverted because the target sequence in searches are often long, e.g. chromosomes, and so iterating over j first is more efficient [53].

QDB CYK can be described as a special case of HMM banded CYK alignment, for which the bands are set as follows. First, the j bands are unrestrictive: $\text{jmin}(v) = 1$ and $\text{jmax}(v) = L$ for all v. Second, the j-dependent d bands are equal to the j-independent d bands from the QDB band calculation algorithm: $\text{hdmin}(v, j) = \text{dmin}(v)$ and $\text{hdmax}(v, j) = \text{dmax}(v)$ for all j.

8.4 Benchmarking

To evaluate the performance of INFERNAL's HMM banded alignment strategy I compared its running time and accuracy to non-banded CYK alignment, QDB CYK alignment and HMM alignment with the Viterbi algorithm. I used three test datasets for this comparison.

The first dataset has previously been used to test structural RNA alignment accuracy by Kolbe and Eddy [141]. It includes sequence data from two RNA families: bacterial RNase P RNA and bacterial SSU rRNA. Each family is represented by a manually curated structural alignment that is presumed correct. Kolbe and Eddy [141] created training and testing subsets of the curated alignments such that no training/testing sequence pair is more than 60% identical for RNase P or 82% identical for SSU. RNase P has 28 training and 15 test sequences and SSU has 101 training and 51 test sequences. I will refer to this dataset as Kolbe09.

To test HMM and CM alignment using the Kolbe09 dataset, a profile is built from the training alignment and used to align the test sequences. Accuracy of the predicted test alignment is measured relative to the manually curated alignment as the fraction of correctly aligned residues in the predicted alignment. Residues from columns of the manually curated alignment that were defined as consensus during CM or HMM construction must occur in the same column in the predicted alignment to be considered correct. Residues from non-consensus columns that lie between two bordering consensus columns must occur between the same two bordering consensus columns in the predicted alignment to be considered correct. In [141], the authors were interested in testing alignment accuracy on partial RNA sequences so their test involved fragments of the test sequences. Here, I am interested in general alignment accuracy, so the full length sequences were used.

I constructed a second, more comprehensive dataset that includes 100 sequences emitted from each of the 1372 RFAM version 9.1 CMs, a SSU rRNA and a LSU rRNA CM (built from alignments from CRW [32]) using INFERNAL's `cmemit` program. Each sequence was realigned to the model it was emitted from using various alignment strategies and running times and accuracy were measured. The accuracy metric calculation is the same as for Kolbe09 but, in the absence of a trusted manually curated alignment, the optimal alignment determined using non-banded CYK is considered the "correct" one. I will refer to this as the Emit dataset.

Finally, a third dataset was constructed with another 1374 sets of 100 sequences, but this time the sequences were generated randomly (using a single-state HMM with emission probability 0.25 for each `A`, `C`, `G`, and `U`). One set was sampled for each RFAM 9.1, SSU or LSU model. The length distribution of each model's sequence set is the same as the length distribution for the corresponding set in the Emit dataset. Each set was realigned to its corresponding model and evaluated in the same manner as in the Emit set. I will refer to this as the Random dataset.

The three test datasets complement each other. In the Kolbe09 dataset, the availability of a (presumed) correct alignment enables the test of what benefit, if any, modeling structure has on accuracy, by comparing CM and HMM results. Further, I can measure the extent to which that benefit is retained using the various banded strategies. For the Emit and Random datasets, no trusted, correct alignments are available, but the larger set of families cover a much broader range of family sizes and structures, so these dataset measure the general utility of the different banded strategies. The Random set offers a unique challenge to HMM banding because no homology exists between query and targets.

Results

The results on the modified Kolbe09 benchmark are included in Table 8.2. In general, HMM banded CYK performs very well compared with non-banded CYK, and matches or outperforms QDB accuracy in all cases but one, while achieving more than 10-fold greater acceleration. Using $\tau = 10^{-7}$ and without normalizing posteriors, HMM banded CYK is able to find the globally optimal alignment for both RNase P and SSU roughly 100-fold and 2000-fold faster, respectively, than non-banded CYK. The values in the row pertaining to this strategy, which is used by default in INFERNAL's `cmalign` program, are in bold-faced type.

Table 8.3 shows the accuracy of non-normalized HMM banded CYK with $\tau = 10^{-7}$ and QDB using $\beta = 10^{-7}$ on the Emit and Random datasets. The accuracy of the two methods is high. They perform very similarly on the Emit dataset. QDB does better on the Random dataset.

Figure 8.1 shows the empirical running time of various alignment strategies on the Emit and Random benchmark datasets. These data show that HMM banded alignment is significantly faster than QDB across the wide range of different models from RFAM. This is true for both datasets, although the speed gap is smaller for the Random set. This indicates that for random sequences the HMM bands are wider than they are for the "homologous" emitted sequences, which is expected because HMM alignments of random sequence should be less well-defined (have lower posterior probabilities) than those of sequences emitted from the model. The HMM banded strategy is the only strategy for which the timings for the Emit and Random datasets are noticeably different. This reflects the fact that the non-banded CYK, QDB and HMM Viterbi algorithms are all sequence-independent, while HMM bands are computed based on the target sequence.

Entropy weighting and alignment accuracy

As noted in [141], the entropy weighting model parameterization strategy described in Chapters 1 and 2 does not significantly improve performance on the Kolbe09 alignment

alignment algorithm	prob mass excluded	normalized?	RNase P accuracy vs opt	RNase P accuracy vs ref	RNase P time (sec/seq)	RNase P speedup	SSU rRNA accuracy vs opt	SSU rRNA accuracy vs ref	SSU rRNA time (sec/seq)	SSU rRNA speedup
non-banded CYK	-	-	1.000	0.843	8.710	1.0	1.000	0.981	1321.535	1.0
QDB CYK	10^{-11}	-	1.000	0.843	1.170	2.2	1.000	0.981	66.586	19.9
QDB CYK	10^{-7}	-	0.974	0.833	0.926	2.8	1.000	0.981	45.402	29.1
QDB CYK	10^{-3}	-	0.929	0.810	0.533	4.8	0.990	0.972	21.486	61.5
HMM banded CYK	10^{-11}	yes	1.000	0.843	0.043	59.9	1.000	0.981	0.970	1362.1
HMM banded CYK	10^{-7}	yes	1.000	0.843	0.027	94.7	1.000	0.981	0.916	1442.3
HMM banded CYK	10^{-3}	yes	0.983	0.841	0.018	143.6	1.000	0.981	0.883	1496.7
HMM banded CYK	10^{-11}	no	1.000	0.843	0.034	74.7	1.000	0.981	0.761	1737.5
HMM banded CYK	10^{-7}	**no**	**1.000**	**0.843**	**0.021**	**121.0**	**1.000**	**0.981**	**0.719**	**1839.0**
HMM banded CYK	10^{-3}	no	0.968	0.836	0.014	184.0	0.999	0.981	0.695	1900.7
HMM Viterbi	-	-	0.820	0.787	0.002	1306.5	0.972	0.966	0.080	16641.6

Table 8.2: **Banded CYK performance on the modified Kolbe09 benchmark.** "prob mass excluded" is the probability allowed outside each band: the β parameter in QDB or τ parameter in HMM banding. "normalized?" only pertains to HMM banding, if "yes" HMM posterior probabilities were normalized as discussed in the text. "vs opt" gives alignment accuracy compared with the optimal alignment found by non-banded CYK. "vs ref" gives alignment accuracy compared with the reference alignment. The row with bold-faced values pertains to the default parameters used by INFERNAL 1.0's `cmalign` program. Alignment accuracy is calculated as described in the text and [141].

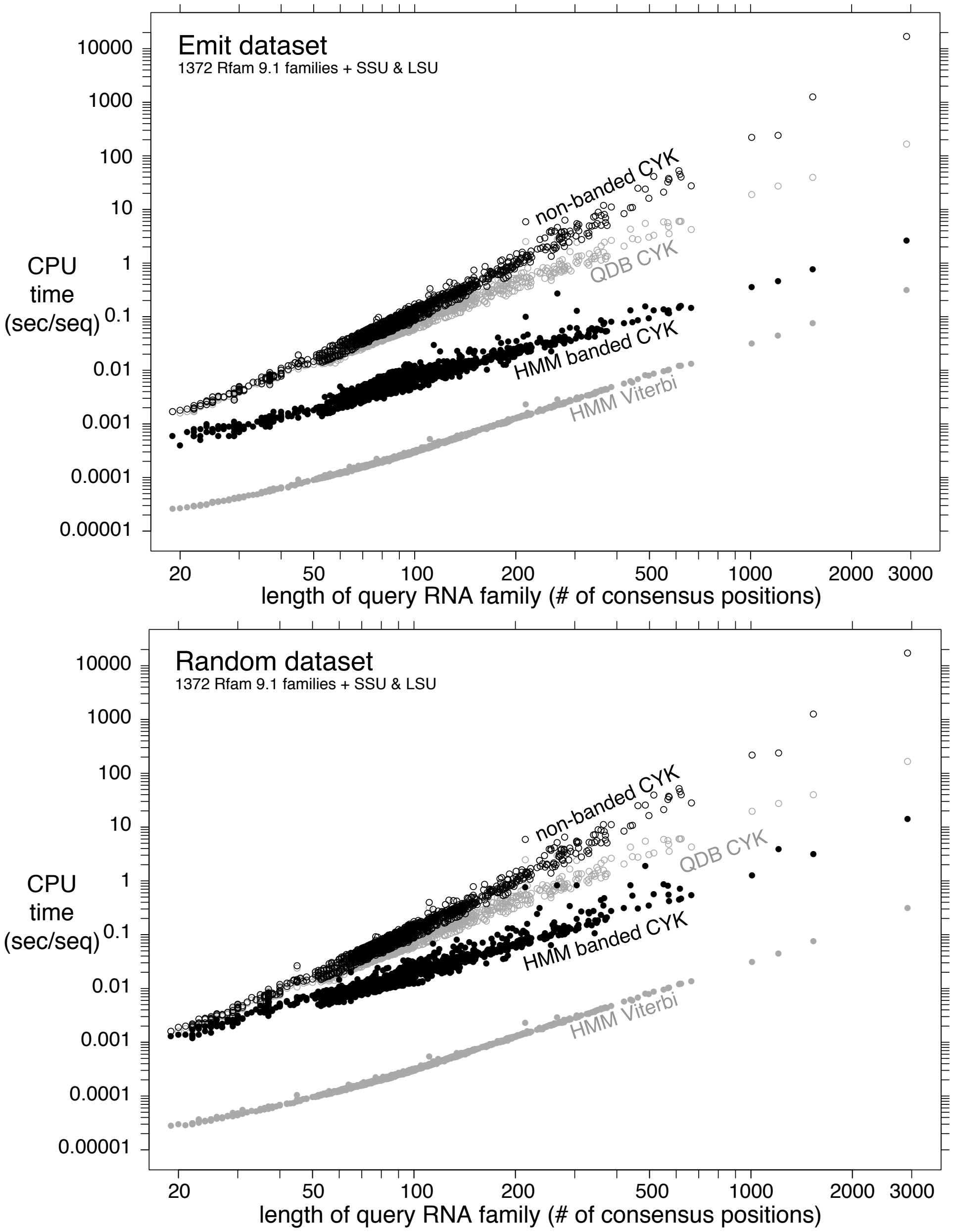


Figure 8.1: **Empirical run times of different alignment algorithms.** The average time required to align a target sequences with each of the 1372 9.1 CMs, an SSU rRNA and a LSU rRNA CM, is shown as a point, plotted on a log-log graph as a function of the consensus model length. Open black circles are non-banded CYK. Open gray circles are QDB CYK with $\beta = 10^{-7}$. Filled black circles are non-normalized HMM banded CYK with $\tau = 10^{-7}$. Filled gray circles are CP9 HMM Viterbi. For the Emit dataset, targets were sampled from the model. For the Random set they were randomly generated as described in the text. INFERNAL version 1.01 was used for all alignments. The SSU and LSU rRNA models were built from alignments from the CRW database [32].

	QDB CYK $\beta = 10^{-7}$	HMM banded CYK $\tau = 10^{-7}$ non-normalized
Average accuracy on the Emit dataset	0.99913	0.99913
Average accuracy on the Random dataset	0.99401	0.99079

Table 8.3: **Average alignment accuracy on the Emit and Random datasets for QDB CYK and HMM banded CYK.**

benchmark relative to standard parameterization using Dirichlet priors. For SSU, accuracy actually decreases slightly when entropy weighted models are used. For this reason, the Kolbe09 results in Table 8.2 are for alignments using models built without entropy weighting. For comparison, when using entropy-weighted models the alignment accuracy is 0.976 and 0.848 for SSU and RNase P, respectively, using parameters corresponding to the bold-faced row in Table 8.2 (in which the corresponding accuracies are 0.981 and 0.843). In contrast, the Emit and Random benchmarks were performed using entropy weighted models. This was done partly because RFAM computes its "full" alignments this way and also so that the generated sequences in the Emit set would be relatively difficult to align (entropy weighting makes the model less specific).

Timing analysis of individual steps of HMM banded alignment

To determine the amount of CPU time required for each step of alignment, I created a modified version of the `cmalign` program which reports the timings for each step. I used this program to investigate HMM banded SSU alignment with the Kolbe09 dataset with $\tau = 10^{-7}$ using both the non-normalized and normalized posterior strategies. The observed timings are listed in Table 8.4. For both strategies, the definition of the bands requires more than half the total time. Interestingly, most of the time difference between the non-normalized and normalized posterior strategies is due to the band calculation step. This is because in this step the normalization strategy must compute the $P(k^*)$ values from equation 8.1, while these values are not needed and not computed by the non-normalized strategy.

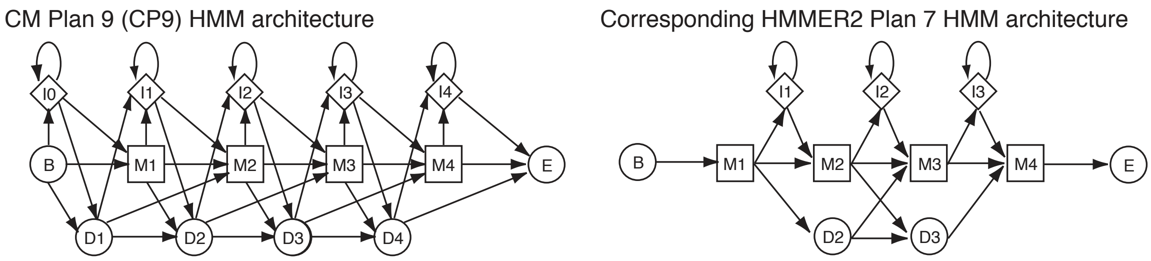
		$\tau = 10^{-7}$ non-normalized		$\tau = 10^{-7}$ normalized	
step(s)	description	sec/seq	% of total	sec/seq	% of total
1	HMM Forward/Backward	0.494	68.6%	0.492	53.7%
2&3	band calculation	0.142	19.7%	0.313	34.2%
4	HMM banded CYK	0.085	11.7%	0.110	12.1%

Table 8.4: **Timings for steps of HMM banded alignment on the modified Kolbe09 benchmark.** Timings are shown for HMM banded alignment with $\tau = 10^{-7}$ using non-normalized posteriors and normalized posteriors.

8.5 Constructing an HMM that is maximally similar to a CM

I now return to the problem of constructing and parameterizing the HMM used for banding. It is important that this HMM is constructed and parameterized to be as similar to the CM as possible because the less difference between the two models, the less likely it is that bands derived from an HMM alignment of a target will obscure the globally optimal CM alignment. Of course, an HMM is incapable of modeling the consensus structure of a CM, but this is the only necessary difference between the two models.

Weinberg and Ruzzo introduced a method for constructing and parameterizing a profile HMM, called a *maximum-likelihood (ML) heuristic HMM*, that is maximally similar to a CM [265, 266]. The method is implemented in the RAVENNA package. I have implemented a very similar type of HMMs called *CM Plan 9 (CP9) HMMs* in INFERNAL. The difference between ML-heuristic HMMs and CP9s is the number of states in the model; whereas ML-heuristic HMMs include two states for each state of a corresponding CM, CP9s usually include only one, except in the case of HMM states that correspond to CM states in `MATP`


CM Plan 9 (CP9) HMM architecture

Corresponding HMMER2 Plan 7 HMM architecture

Figure 8.2: **CM Plan 9 HMM and HMMER2 Plan 7 HMM architectures.** Each model contains four nodes. Match, insert, and delete states are named as “Mx”, “Ix” and “Dx” respectively with “x” indicating the node index. For sake of comparison, the Plan 7 HMM is depicted in global configuration for alignment, instead of a local configuration for search, as it is in the HMMER2 User’s Guide [63].

nodes, as noted below.

CP9 HMMs were specifically designed to be similar to the Plan 7 (P7) HMMs of the HMMER package [62] (which is also freely available from the Eddy lab), so that I could easily port current and future optimized implementations of HMM alignment and search algorithms from HMMER to INFERNAL. Figure 8.2 shows the CP9 architecture compared with the HMMER 2 plan 7 architecture. Note that CP9s contain two additional transitions per node: from a delete state to the insert state of the same node, and from the insert state to the delete state of the next node.[5] Also, CP9s have extra states: an insert state before the first node and after the last node (`I0` and `I4`) and delete states in the first and last node (`D1` and `D4`). All of these CP9 additions are necessary to model an analogous state or transition in a CM. For example, the `I0` state corresponds to a CM’s `ROOT_IL` state which models inserts before the first consensus position. Also, the CM architecture includes transitions between insertions and deletions (Figure 1.12).

Mapping CP9 HMM states and CM states

Given a CM for a sequence family with N consensus positions, the CP9 HMM architecture dictates the placement of states and transitions in a corresponding N-node CP9 HMM. In order to parameterize the HMM to be maximally similar to the CM, its states must first

---

[5] This is the reason for the *7* and *9* in the names, P7s have 7 transitions out of each node, CP9s have 9.

be mapped to their corresponding CM states. This mapping is also necessary to transfer HMM bands onto the CM CYK DP matrix in step 3. The mapping is mainly achieved using three simple rules:

1. Match states that emit to the same consensus position map to each other.
2. Insert states that insert after (3' of) the same consensus position map to each other.
3. Delete states that delete (cause a gap in) the same consensus position map to each other.

These three rules map nearly all HMM states to exactly 1 CM state, and nearly all CM states to exactly 1 HMM state, with a few exceptions. Some HMM match states will map to more than CM state. For example, any consensus position modeled by a `MATP` node will have two match states that emit to that position (`MATP_MP` and either `MATP_ML` or `MATP_MR`), so HMM states modeling these positions will map to two CM states. Additionally some CM states will map to more than one HMM state. For example, `MATP_MP` match states emit two base-paired residues to two separate consensus positions, and thus map to the two HMM match states that also emit to those positions. Finally, some CM states do not map to any HMM states. These include `BEGL_S`, `BEGR_S`, `END_E` and `BIF_B` states which are only necessary in the CM for modeling structure and so do not correspond to any HMM state.

Following the enforcement of the three rules, two additional steps are taken to complete the mapping. First, the still unmapped CM `ROOT_S` state, from which all alignments begin, is mapped to the analogous HMM `B` state. The final step involves the subset of HMM insert states that map to two CM insert states. These cases only occur because of a design flaw that causes an ambiguity in the CM grammar allowing a pair of distinct CM insert states to emit after the same consensus position. One pair of these insert states exists for each `END` node in a CM. For each pair, one of the two insert states is always immediately prior to an `END_E` state in the CM state graph. (The CM from Figure 1.12 contains two pairs of these states, one in guide tree nodes 12 and 13 and one in nodes 22 and 23. For more detail, see [192] which shows the state graph for these nodes.) Because grammar ambiguities are generally undesirable, and especially so when using CYK-like DP algorithms that find maximum likelihood solutions [50, 91], this CM ambiguity has been removed in practice within INFERNAL by *detaching* all insert states immediately prior to `END_E` states. Detachment simply involves setting all transition probabilities into the state to 0.0 so it is never used in a parse tree (alignment). The other state in each pair is not modified. The detached inserts are not necessary to model by an HMM, so they are not mapped to any HMM state. This leaves any HMM state that was mapped to two CM inserts now mapped to only one (the unmodified, non-detached one of the pair).

CM Plan 9 HMM parameterization

Using a mapping of CM states to CP9 HMM states, INFERNAL uses Weinberg's ML-heuristic HMM parameterization procedure [266] to set the CP9 emissions and transitions. Emissions are relatively straightforward to set. For HMM match and insert states that map to a single CM state, emission probabilities are set by simply copying that CM state's emission probabilities. The remaining emitting HMM states all model a base-paired position in the CM consensus structure and map to a `MATP_MP` and either a `MATP_ML` or `MATP_MR` state. For each such HMM state k^M mapping to CM states v_1 and v_2, the emission probability $e_{k^M}(x)$ for each RNA base $x \in \{A, C, G, U\}$ is set as:

$$e_{k^M}(x) = \frac{\psi(v_1)}{\psi(v_1) + \psi(v_2)} * e_{v_1}(x) + \frac{\psi(v_2)}{\psi(v_1) + \psi(v_2)} * e_{v_2}(x)$$

Where $\psi(v_1)$ and $\psi(v_2)$ are the expected number of times v_1 and v_2 are entered in a CM parse tree. In the above equation, the `MATP_MP` probabilities ($e_{v_1}(x)$) are calculated by marginalizing the 16 possible base-pair emissions for the appropriate consensus position modeled by v_1 (i.e. the position corresponding to either the left or right half of the consensus base-pair).

The calculation of the HMM transition probabilities is less straightforward because some HMM transitions do not have an analogous transition in the CM, and vice versa. For example, an HMM transition between the match states modeling consensus positions 14 and 15 of the example in Figure 1.12 does not correspond to any single CM transition. INFERNAL implements Weinberg's method for defining HMM transitions for these cases [266].

Chapter 9

SSU-align: a tool for structural alignment of SSU rRNA sequences

SSU-ALIGN is a freely available, open source software program for creating large-scale alignments of small subunit ribosomal RNA (SSU) using covariance models (CMs). The archaeal, bacterial, and eukaryotic (nuclear) SSU CMs in SSU-ALIGN are built from structural seed alignments derived from Robin Gutell's Comparative RNA Website (CRW) [32]. The following features distinguish SSU-ALIGN from other SSU alignment programs (reviewed in chapter 7):

- **Structural alignments are calculated at roughly one second per sequence.**

  CM alignment takes into account conserved sequence and well-nested structure. In the past, the slow speed of CM alignment has prevented their application to SSU alignment, but the sequence-based banding approach described in chapter 8 yields roughly a 1000-fold speedup, making large-scale SSU CM alignment feasible.

- **Masks for pruning ambiguously aligned columns are automatically generated.**

  CMs are probabilistic models that allow calculation of the posterior probability that each sequence residue belongs in each column of the output alignment given the model.

Regions of low posterior probabilities are indicative of high alignment ambiguity and should be pruned away (masked out) prior to phylogenetic inference. SSU-ALIGN can automatically detect and remove these columns for any alignment it generates.

- **Accurate alignments can be computed using seed alignments of less than one hundred sequences.**

  CMs built from small seed alignments can achieve comparable accuracy to nearest-neighbor based alignment methods that use seed (reference) alignments of thousands or tens of thousands of sequences. This reduces the level of manual curation necessary for creating useful seeds, making it easier to extend SSU-ALIGN by adding new seeds and corresponding CMs that cover specific phylogenetic ranges. For example, a *Firmicute*-specific seed and model could be constructed to create large alignments for phylogenetic analysis of the *Firmicute* bacterial phyla.

9.1 Aligning SSU sequences with SSU-align

SSU-ALIGN takes target sequences and a CM file with $N >= 1$ SSU models as input and proceeds through two stages to generate structurally annotated alignments (Figure 9.1). The CM file may contain one or more SSU models. The default CM file supplied with SSU-ALIGN version 0.1 contains three SSU CMs: an archaeal model, a bacterial model, and a eukaryotic model each built from structural alignments derived from CRW [32], as described later in this chapter.

In the first stage, each target sequence is scored with each of the models based only on sequence conservation. This is done with a profile HMM derived from each CM, which is significantly faster than using the CM. The model whose profile HMM gives the highest score to each sequence is defined as the *best-matching* model for that sequence. If this highest score is not above a predefined threshold, the sequence is discarded and not evaluated further. The boundaries of the best-matching HMM alignment are used to truncate each target sequence if the alignment does not span the entire target length. In stage 2, each surviving, and possibly truncated, sequence is aligned to its best-matching model, this time using the CM which scores both sequence and conserved structure. Up to N alignments are created, one for each model that was the best-match to at least one target sequence.

SSU-ALIGN can generate masks for the alignments it creates based on the posterior probabilities for each residue in the alignment. These masks can then be used to remove ambiguously aligned columns of the alignment prior to using phylogenetic inference tools. A User's Guide is supplied with SSU-ALIGN that explains how to use the program for creating and masking alignments.

9.2 Automated probabilistic alignment masking

The goal of masking is to identify and remove columns containing residues that are ambiguously aligned and therefore likely to contain errors prior to using the alignment for phylogenetic inference. Two commonly used SSU masks were determined manually by David Lane and Phil Hugenholtz (Figures 7.5 and 7.6) based on expert knowledge and extensive experience with SSU alignments (Chapter 7). Alignment posterior probabilities from probabilistic models offer an alternative, objective way of evaluating alignment ambiguity (high posterior probability means low ambiguity and vice versa) and creating masks for any alignment.

The HMM banded alignment technique described in Chapter 8 exploits posterior probabilities calculated from the Forward and Backward algorithms in HMM alignments. Inside and Outside, the SCFG analogs of the Forward and Backward, can be used to calculate posterior probabilities for CM alignment [53]. These algorithms were not implemented in INFERNAL prior to this work because they are about two-fold slower than the (already slow) CYK algorithm and because the Myers-Miller linear memory trick that made CM SSU alignment practical (requiring 60 Mb instead of 20 Gb of RAM) had only been applied to CYK [57]. HMM banding drastically reduces the memory requirement for CM alignment, because only DP cells within the bands need be allocated, making it possible to run Inside and Outside on SSU.

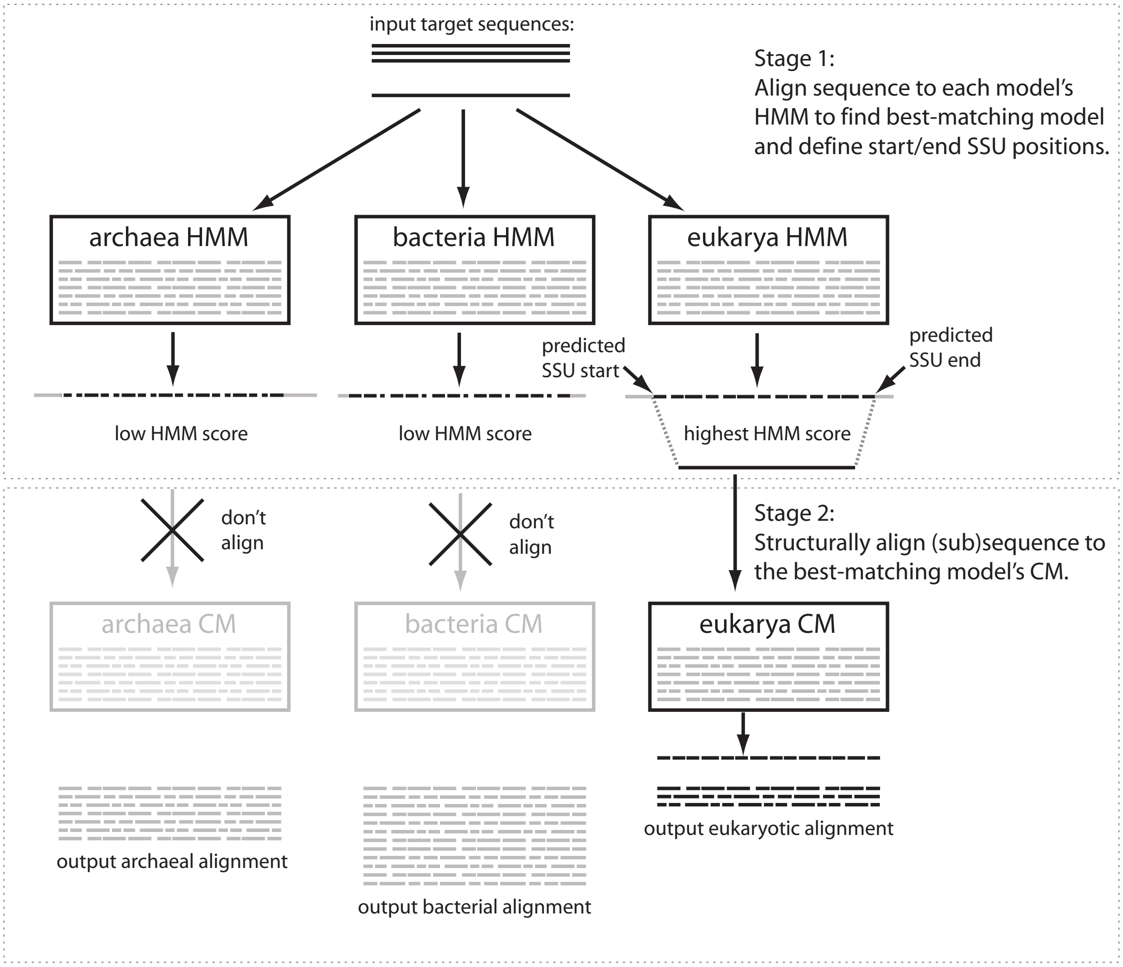

input target sequences:
Stage 1:
Align sequence to each model's HMM to find best-matching model and define start/end SSU positions.
archaea HMM
bacteria HMM
eukarya HMM
predicted SSU start
predicted SSU end
low HMM score
low HMM score
highest HMM score
don't align
don't align
Stage 2:
Structurally align (sub)sequence to the best-matching model's CM.
archaea CM
bacteria CM
eukarya CM
output archaeal alignment
output bacterial alignment
output eukaryotic alignment

Figure 9.1: **Schematic of the SSU-align alignment pipeline.** Unaligned target sequences are input to the program. In stage 1, each sequence is independently aligned using only primary sequence scoring to each of N HMMs, one built from each model in the input CM file. The model whose HMM alignment has the maximum bit score is the "best-matching" model for that sequence, in this example "eukarya" is the best-matching model. In stage 2, the unaligned (sub)sequence from the best-matching model's HMM alignment (potentially with some sequence trimmed off the ends) is aligned to the best-matching model's CM which scores both sequence and conserved secondary structure. The CM aligned target sequence is added to that model's output alignment. After all target sequences are processed, the program has output up to N new structural alignments, one for each model that was the best-matching model for at least 1 target sequence.

Alignment ambiguity and length heterogeneity

Alignment ambiguity often arises in regions of alignments with low sequence conservation where insertions and deletions are common, corresponding to regions of the molecule that exhibit high length heterogeneity across different species. In such cases, it is often difficult to determine the correct alignment because alternative alignments seem plausible. Take for example the CM alignment of the `GUAU` subsequence of the *Desulfovibrio desulfuricans* SSU sequence to a loop region depicted in Figure 9.2. The reference (consensus) sequence for the loop (`AUUCAAC`) differs from `GUAU` in both sequence and length. Consequently, the CM alignment for this loop is not well defined, and two alternative alignments are given posterior probabilities above 0.35. In contrast, the surrounding helix region, for which higher sequence similarity exists between the two sequences, is aligned confidently with high posterior probabilities.

Inserted columns should always be excluded during masking

Importantly, any CM alignment mask should automatically exclude every insert column of the alignment. This is because profile probabilistic models like CMs do not actually align inserted residues (residues aligning to insert states) between different sequences, but rather simply insert them between the appropriate consensus columns in the alignment. This means that sequence residues appearing in the same insert alignment columns are not aligned with respect to each other and consequently should be removed prior to phylogenetic analysis which assumes aligned residues are homologous.

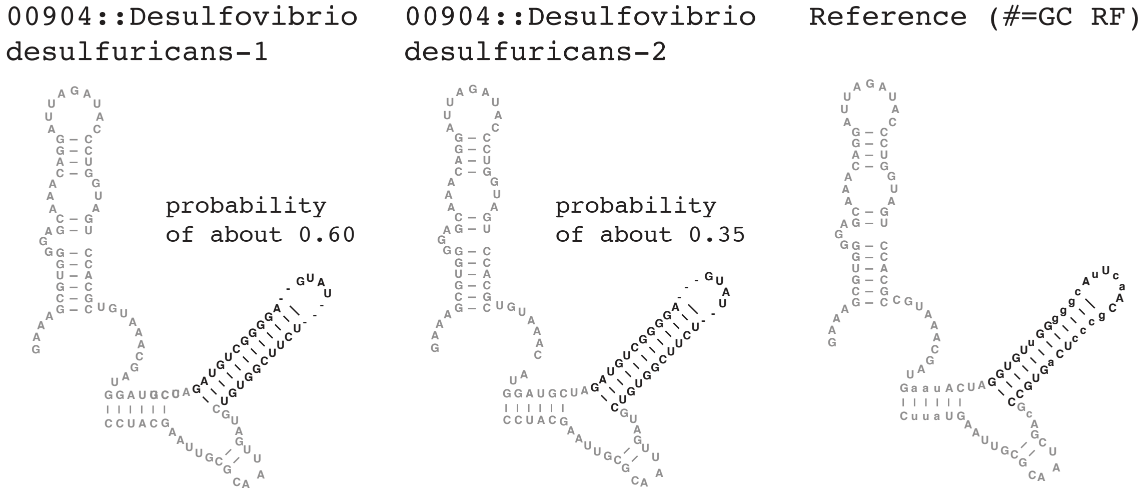
Benchmarking probabilistic masking

I decided on a simple method for defining masks that requires that a given fraction x of the residues in an aligned consensus (non-insert) column have a posterior probability above a minimum threshold y to be included (not pruned away) by the mask. I tested this approach with different x and y values on the SSU alignment test from the Kolbe09 benchmark [141] described in Chapter 8. Briefly, the test consists of building a CM from an aligned subset

```
00904::Desulfovibrio_desulfuricans-1                    GAUGUCGGGGA--GUAU---UCUUCGGUGUC
#=GR 00904::Desulfovibrio_desulfuricans-1 POSTX.        99999999999--5665---89999999999
#=GR 00904::Desulfovibrio_desulfuricans-1 POST.X        99999999996--8009---79999999999

00904::Desulfovibrio_desulfuricans-2                    GAUGUCGGGGA---GUAU--UCUUCGGUGUC
#=GR 00904::Desulfovibrio_desulfuricans-2 POSTX.        99999999999---3333--89999999999
#=GR 00904::Desulfovibrio_desulfuricans-2 POST.X        99999999996---6665--79999999999

#=GC SS_cons                                            <<<<<<<<<<<.......>>>>>>>>>>>
#=GC RF                                                 GGUGUuGGgggcAuUcaACgcccUCaGUGCC
```



00904::Desulfovibrio desulfuricans-1

00904::Desulfovibrio desulfuricans-2

Reference (#=GC RF)

Figure 9.2: **Example of alignment ambiguity in a hairpin loop.** Top: An alignment fragment of two different alignments of the *Desulfovibrio desulfuricans* (sequence accession `M34113`) sequence from the SSU-ALIGN bacterial seed alignment for the region between consensus columns 861 and 881. Each alignment is annotated with its posterior probability in the `#=GR POSTX.` and `#=GR POST.X` rows. For example, the third `A` in the first aligned sequence has a posterior probability of 0.60 of being aligned in its current position. The probability this `A` aligns in the next position over, as it does in the second alignment, is 0.36. There is a probability of $1.0 - 0.60 - 0.36 = 0.04$ that the `A` does not align at either of these two positions (not shown). The `#=GC SS_cons` and `#=GC RF` rows correspond to the consensus secondary structure and sequence respectively. The alignments were created using `cmalign` to align this sequence to the bacterial CM with the `--sample` option. Bottom: The secondary structures corresponding to the two possible alignments of the *D. desulfuricans* and the reference alignment. Residues in the actual alignment are black. Residues surrounding the alignment fragment are gray.

of 101 training sequences from a *gold standard* CRW alignment and aligning a separate subset of 51 test sequences, where the training and testing sets have been defined such that no train/test pair is more than 82% identical. Accuracy is measured by the similarity of the CM alignment of test sequences to the original CRW alignment (for more details, see Chapter 8 or [141]). The purposefully low sequence similarity between the training and test set sequences are meant to increase the difficulty of the benchmark.

The effect of masking using several different combinations of x and y values on the alignment accuracy and coverage of CM alignment on the Kolbe09 benchmark is summarized in Table 9.1. Coverage is defined as the fraction of residues in the test alignment that are included by the mask. Accuracy is the fraction of residues included by the mask which are correctly aligned (as defined in Chapter 8 and [141]). The results indicate a trade-off between coverage and accuracy: coverage decreases but accuracy increases as x and y increase, because the mask becomes more stringent, requiring a larger fraction of more confidently aligned residues in included columns. There is no clear best-performing combination of x and y. SSU-ALIGN uses 0.95 for both x and y as the default, a strategy which attains 99.74% accuracy and 85.37% coverage on the benchmark.

A separate validation of the automated masking strategy, apart from benchmarking is via comparison to the manually created masks of David Lane and Phil Hugenholtz discussed in Chapter 7. To enable the comparison, I used the SSU-ALIGN bacterial CM to realign each of the bacterial seed sequences and masked the resulting alignment based on posterior probabilities using x and y values of 0.95. The resulting mask overlaid on the SSU-ALIGN bacterial consensus secondary structure model is shown next to David Lane's mask on the *E. coli* SSU structure in Figure 9.3. In general, the same regions are excluded by both masks. The SSU-ALIGN mask excludes significantly fewer positions than does the Lane mask. This is to be expected and does not suggest the SSU-ALIGN is in any way "better" than the Lane mask. The SSU-ALIGN mask was derived from an alignment consisting only of the seed sequences the model was parameterized from, which the model should be able to align with high confidence. The key point here is that the SSU-ALIGN mask is different from the Lane

x (fraction of sequences with pp above y)

y	0.800		0.900		0.925		0.950		0.975		0.990	
	cov	acc	cov	acc	cov	acc	cov	acc	cov	acc	cov	acc
0.800	0.9576	0.9897	0.9367	0.9928	0.9297	0.9935	0.9253	0.9939	0.9088	0.9949	0.9014	0.9952
0.900	0.9186	0.9946	0.9095	0.9950	0.9008	0.9955	0.8940	0.9959	0.8774	0.9965	0.8634	0.9971
0.925	0.9111	0.9950	0.8869	0.9961	0.8797	0.9963	0.8724	0.9968	0.8533	0.9973	0.8441	0.9975
0.950	0.8987	0.9958	0.8691	0.9969	0.8599	0.9972	**0.8537**	**0.9974**	0.8396	0.9977	0.8214	0.9978
0.975	0.8626	0.9972	0.8404	0.9977	0.8354	0.9979	0.8234	0.9981	0.7954	0.9982	0.7740	0.9985
0.990	0.7720	0.9981	0.7424	0.9985	0.7209	0.9986	0.6904	0.9988	0.6420	0.9990	0.5969	0.9990

Table 9.1: **Effect of different masking strategies on CM alignment accuracy and coverage on the Kolbe09 SSU alignment benchmark.** Alignments were performed using INFERNAL 1.01's `cmalign` program with the `--cyk` flag, which performs HMM banded CYK alignment with $\tau = 10^{-7}$. Masks were defined as follows: any consensus alignment column in which more than x fraction of the aligned residues have a posterior probability of at least y is included by the mask. All other columns are excluded (not counted). "cov" (coverage) is the fraction of residues in the test alignment that are included by the mask. "acc" (accuracy) is the fraction of residues included by the mask which are correctly aligned. The values cooresponding to the SSU-ALIGN default banding strategy $x = 0.95$ and $y = 0.95$ are in bold-faced type. The Kolbe09 benchmark and its definition of alignment correctness are described in more detail in Chapter 8 and in [141]. Masks were computed with the `ssu-mask` program.

mask because it is specific to the alignment it was created for. The ability to automatically construct alignment specific masks is an important feature of SSU-ALIGN that distinguishes it from other alignment tools.

The automatically masked bacterial alignment nicely demonstrates the tight relationship between length heterogeneity and alignment ambiguity as measured by posterior probabilities. The positions excluded from the mask are tightly correlated with positions of the alignment that include at least some gaps (deletions) in consensus positions corresponding to length heterogeneity between different sequences. The effect can be seen clearly in Figure 9.4 which displays the frequency of deletions of each consensus column on a secondary structure diagram of SSU. Positions excluded from the automated mask appear as open circles. Note that the circles occur in clusters that are almost always adjacent to a region with at least a few deletions.

9.3 Implementation

The SSU-ALIGN package includes the INFERNAL software package [192] (written in C), Sean Eddy's EASEL sequence analysis library (also written in C), a perl script called `ssu-align` and two additional C executable programs, `ssu-mask` and `ssu-draw`. The `ssu-align` script orchestrates the use of the INFERNAL's `cmsearch` and `cmalign` programs and EASEL's `esl-sfetch` program. The input CM file must have been created prior to running `ssu-align`, by INFERNAL's `cmbuild` program. An SSU CM file is provided with the program, but users can also build their own. Stage 1 HMM scoring is performed by `cmsearch`. Sequence truncation, when necessary, is performed by `esl-sfetch`. Stage 2 CM alignment of targets to their best-matching models is performed by `cmalign`. Alignment masking and structure diagram creation can be performed post-alignment following completion of the `ssu-align` script by the `ssu-mask` and `ssu-draw` programs.

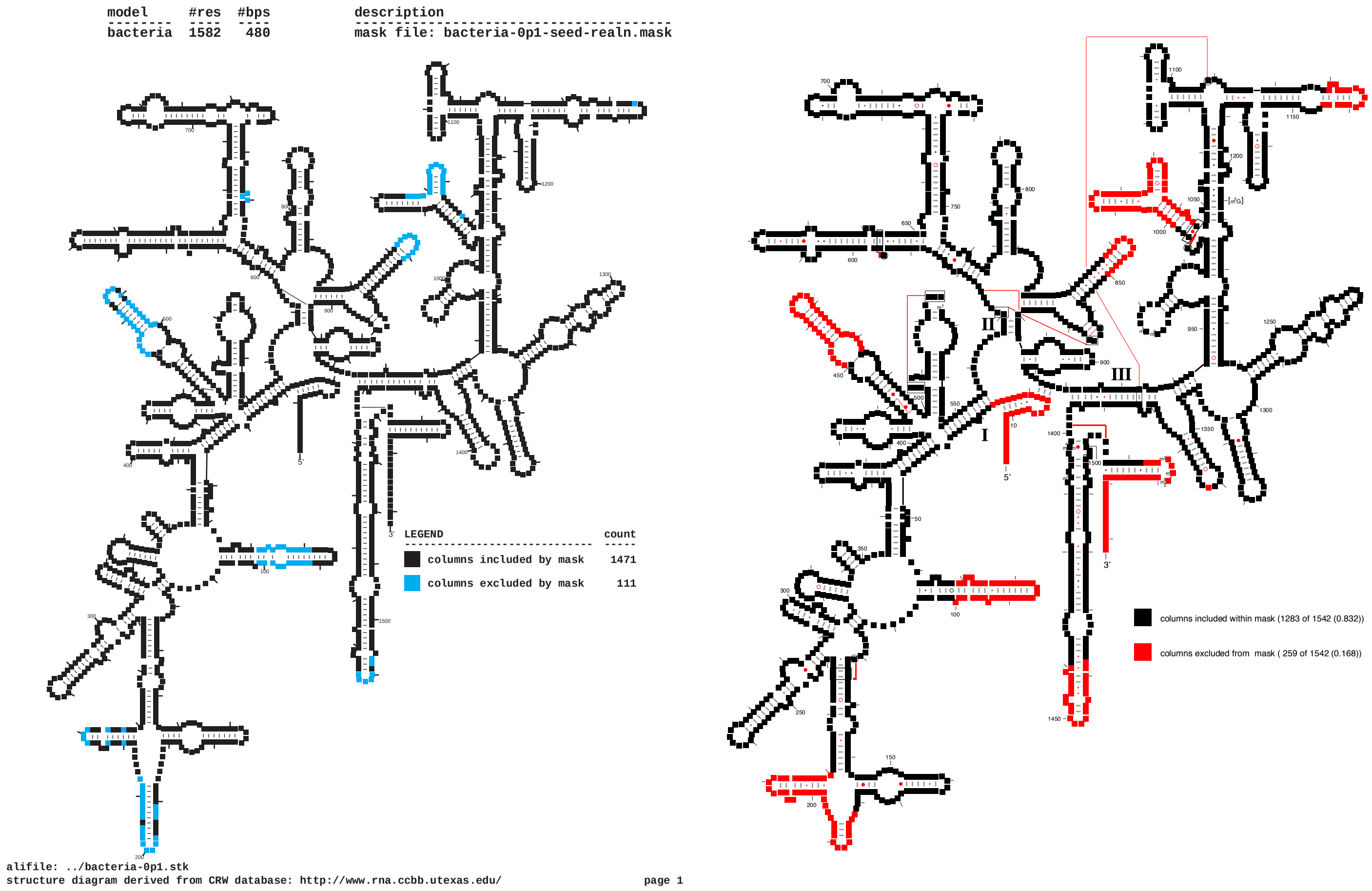
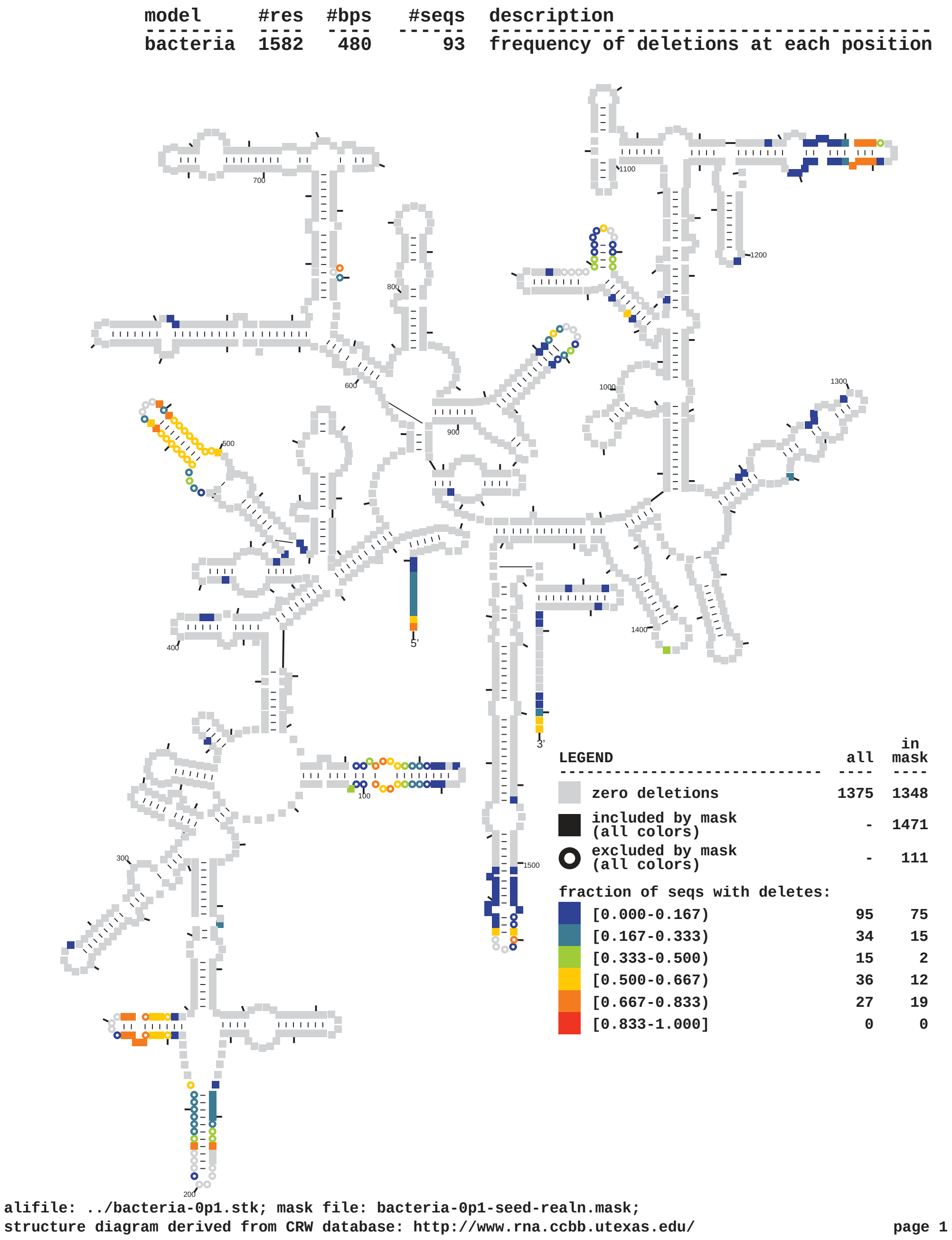

model #res #bps
bacteria 1582 480
description
mask file: bacteria-0p1-seed-realn.mask
LEGEND
count
columns included by mask 1471
columns excluded by mask 111
alifile: ../bacteria-0p1.stk
structure diagram derived from CRW database: http://www.rna.ccbb.utexas.edu/
page 1
5'
3'
I
II
III
columns included within mask (1283 of 1542 (0.832))
columns excluded from mask (259 of 1542 (0.168))

Figure 9.3: **Similarity of an SSU-align automatically constructed mask (left) and David Lane's SSU alignment mask (right).** ***Escherichia coli***. Black columns are included by the mask for phylogenetic analyses. Blue (left) or red (right) columns are excluded by the mask for phylogenetic analyses. The SSU-ALIGN mask was derived automatically based on a realignment of the bacterial seed sequences as explained in the text. The right diagram was derived from the overlay of the "LMPH" mask on the GREENGENES database's *Core Set* alignment of the *E. coli* sequence (GENBANK accession J01695). Both diagrams were generated using the `ssu-align` program included in the SSU-ALIGN package

.



```
model      #res  #bps   #seqs  description
--------   ----  ----   -----  -------------------------------------
bacteria   1582   480      93  frequency of deletions at each position
```

LEGEND	all	in mask
▨ zero deletions	1375	1348
■ included by mask (all colors)	-	1471
○ excluded by mask (all colors)	-	111
fraction of seqs with deletes:		
■ [0.000-0.167)	95	75
■ [0.167-0.333)	34	15
■ [0.333-0.500)	15	2
■ [0.500-0.667)	36	12
■ [0.667-0.833)	27	19
■ [0.833-1.000]	0	0

alifile: ../bacteria-0p1.stk; mask file: bacteria-0p1-seed-realn.mask;
structure diagram derived from CRW database: http://www.rna.ccbb.utexas.edu/ page 1

Figure 9.4: **Secondary structure diagram displaying frequency of deletions of each consensus position in a masked bacterial alignment of the SSU seed sequences.** Statistics correspond to a CM alignment of the bacterial seed sequences using the SSU-ALIGN bacterial CM. Consensus positions excluded from the mask (denoted as circles) are those for which more than 5% of the residues have a posterior probability below 0.95, as explained in the text. This diagram was generated using the `ssu-draw` program included in the SSU-ALIGN package.

9.4 SSU-align's SSU rRNA sequence and structure models

CMs model both the conserved sequence and secondary structure of an RNA family. Constructing CMs requires as input a multiple sequence alignment with well-nested consensus secondary structure annotation. An important question in the design of the SSU-ALIGN program was where to obtain these alignments from. I decided to use the COMPARATIVE RNA WEBSITE (CRW) [32] as the source because it has the largest amount of high quality structural data of any of the SSU databases (see Chapter 7).

The structure models used by CRW are based on nearly thirty years of comparative analysis. The first secondary structures of SSU were created in the early 1980s, by Carl Woese, Harry Noller, Robin Gutell and others using comparative sequence analysis to identify covarying positions indicative of structural relationships such as base-pairs [196, 278, 280]. Since then, Gutell and his colleagues have continued to refine those models. Their comparative approach was validated in 2000, when the crystal structure of the small subunit of *Thermus thermophilus* was solved [271] and 97% of the predicted base-pairs in the then current bacterial secondary structure model were confirmed [107].

Over the past twenty years, Gutell and coworkers have constructed SSU alignments of thousands of sequences using a combination of automated techniques and manual curation. As part of this process, they have singled out novel (phylogenetically distinct) SSU sequences and manually predicted their secondary structures. The alignment and structural data is publicly available in the CRW database [32].

SSU secondary structure data from the Comparative RNA Website

CRW includes separate SSU alignments for archaea, bacteria, chloroplasts, eukarya (nuclear), and mitochondria. I concentrated only on the archaeal, bacterial, and eukaryotic alignments for the initial version of SSU-ALIGN. Unfortunately, the CRW alignments are not structurally annotated, so they cannot be used directly to build CMs. However, CRW curators have predicted structures for a subset of the sequences in each alignment. To create structure annotated alignments I mapped the structural data onto the alignments through a series of steps as described below. Statistics on the CRW alignments and structural data as of May 13, 2009 are given below:

family	# of aligned sequences		# of structures
	primary	seed	
archaea	788	132	25
bacteria	35998	1266	231
eukarya	1937	N/A	259

The *primary* alignments are the largest, most complete alignments in CRW. The *seed* alignments are smaller and "highly refined". Because CM alignment accuracy is highly dependent on the quality of the seed alignment used to build the model, I decided to concentrate on the seed alignments for archaea and bacteria and the primary alignment for the eukarya (because no eukaryotic seed exists).

In an effort to ensure high accuracy, I decided not to use the full CRW alignments as my seed alignments but rather to use subsets of the alignments containing only the sequences for which individual structure predictions exist. There were two main reasons for this. First, the sequences that were chosen for individual structure prediction were those that represented the major phylogenetic groups and "reveal the major forms of sequence and structure conservation and variation" [32]. This suggests they would constitute a good seed alignment from which to build a profile. (A good seed alignment should be representative of the family and generally does not benefit from redundancy [53].) Secondly, after predicting an individual structure, the CRW curators use the structure to revise the larger alignments, which suggests that these particular sequences are the most reliably aligned because they received the most expert attention.

Defining consensus structures from individual structures

Obtaining the consensus structure annotated alignments needed to build CMs required combining the individual structural data and the alignments. One simple approach would be to define a single individual structure x as the consensus structure and impose it on the entire alignment. However, this is not ideal because it means the consensus *will not* include structural features that are absent from x but present in other individual structures, and *will* include structural features that may be unique to x, or at least uncommon in other individual structures. A better approach is to combine or average the individual structures in a reasonable way to determine the consensus structure, and then impose it on the alignment. The procedure I used for deriving consensus structure annotated seed alignments from the CRW data is shown in Figure 9.5.

The first step was to extract the aligned sequences that matched to the individual structures from the master alignments. An individual structure sequence i and an aligned sequence a qualified as match if: a and i both had the same sequence accession, and the unaligned sequence of a was either identical to i or an exact subsequence of i. Not all of the individual structures i had a matching aligned sequence a per these criteria. The number of matches per alignment is shown below:

family	# of aligned sequences	# of structures	# of matches (overlap)
archaea	132	25	23
bacteria	1266	231	95
eukarya	1937	259	148

This defines three sets of aligned sequences in which each sequence has its own predicted structure. CMs can only model well-nested base-pairing interactions, so all pseudoknotted base-pairs were removed from the individual structures. A well-nested structure is a set of base-pairs for which no two pairs between positions i:j and k:l exist such that $i < k < j < l$. I used the program KNOTTED2NESTED.PY by Smit et al. [235] to remove pseudoknots using the `-m OSP` option which maximizes the number of base-pairs in the resulting nested structure.

From this set, any sequences with more than 5 ambiguous bases were removed because

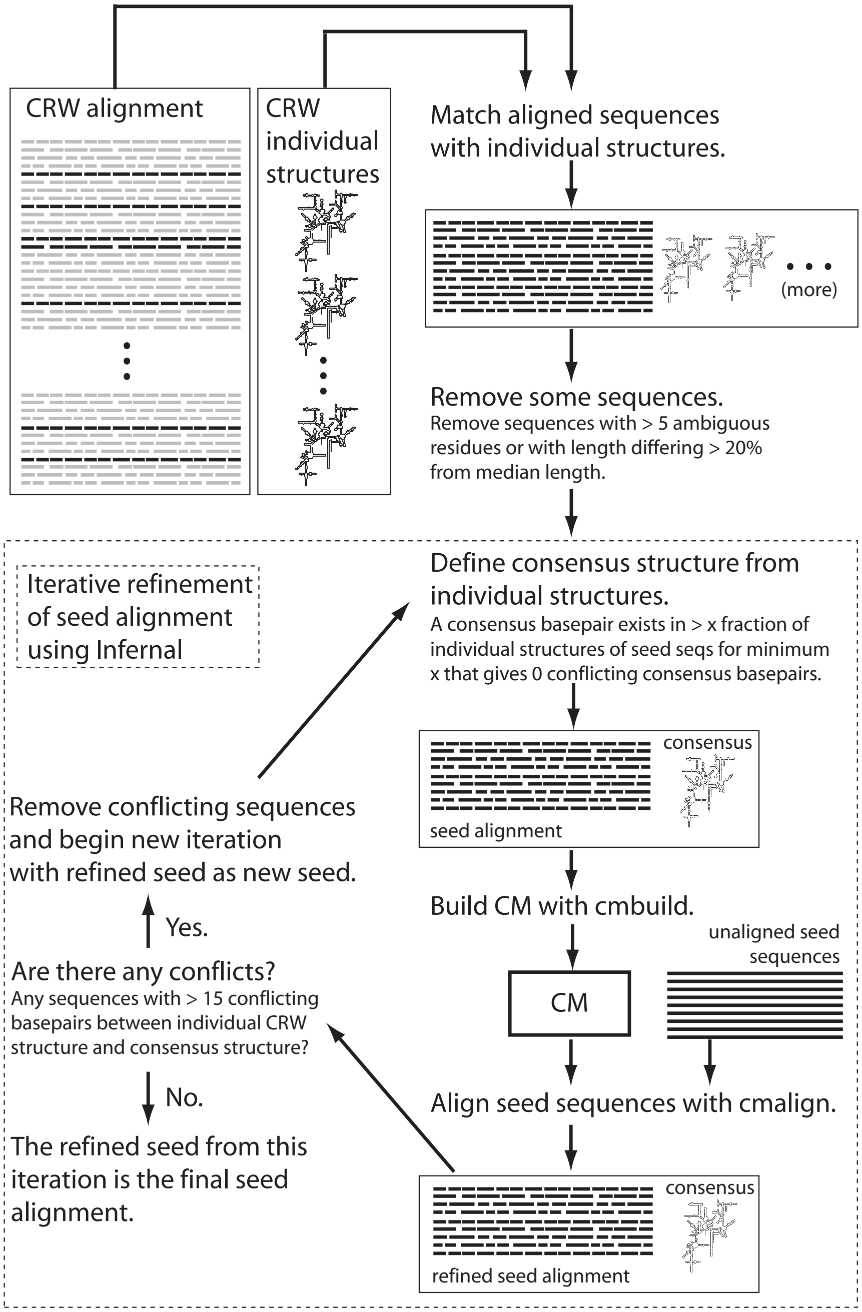

CRW alignment
CRW individual structures
Match aligned sequences with individual structures.
• • • (more)
Remove some sequences.
Remove sequences with > 5 ambiguous residues or with length differing > 20% from median length.
Iterative refinement of seed alignment using Infernal
Define consensus structure from individual structures.
A consensus basepair exists in > x fraction of individual structures of seed seqs for minimum x that gives 0 conflicting consensus basepairs.
consensus
seed alignment
Build CM with cmbuild.
unaligned seed sequences
CM
Align seed sequences with cmalign.
consensus
refined seed alignment
Remove conflicting sequences and begin new iteration with refined seed as new seed.
Yes.
Are there any conflicts?
Any sequences with > 15 conflicting basepairs between individual CRW structure and consensus structure?
No.
The refined seed from this iteration is the final seed alignment.

Figure 9.5: **The procedure for converting SSU alignments and individual structures from crw to seed alignments for SSU-align**. CRW individual structures only exist for a small subset of the sequences from the CRW alignments (the black sequences among the majority of gray ones). Each conversion step is explained in more detail in the text. Figure 9.6 demonstrates an example of conflicting base-pairs. Table 9.2 includes alignment statistics and number of sequences surviving each step for each of the three seed alignments derived from CRW.

ambiguous bases in a seed alignment inject noise into the parameters of a CM (see equation 1.4). Additionally, sequences less than 80%, or more than 120% the median length of the alignment were removed.

At this stage, base-pairing *conflicts* between the aligned individual structures were identified. A conflict exists between two base-pairs in different structures, one between alignment columns i and j and the other between columns j and k, if $i = k$ and $j \neq l$, or $j = l$ and $i \neq k$. An example of two conflicting base-pairs is shown in Figure 9.6. Conflicting base-pairs are problematic because a consensus base-pair between columns i and j in a CM seed alignment is assumed to exist (or be deleted) between the residues in i and j in *all* sequences of the alignment. Columns involved in conflicting base-pairs violate this assumption by specifying that residues from the same column are involved in different base-pairs in different sequences.

```
                                      .........i....j.......
00560::Xylella_fastidiosa             GCAGGGGACCUUAGGGCCUUGU
#=GS 00560::Xylella_fastidiosa SS     <<<<<<..<<....>>>>>>>>
00018::Thermomicrobium_roseum         GGCGCA--G-GCGAC-UGUGCU
#=GS 00018::Thermomicrobium_roseum SS <<<<<<..<.....>.>>>>>>
                                      ........k.....l.......
```

Figure 9.6: **Example of conflicting base-pairs between two aligned individual SSU structure predictions from CRW.** The individual base-pair between aligned columns i and j in *Xylella fastidiosa* (sequence accession `M34115`) conflicts with the *Thermomicrobium roseum* (accession `AE003861`) base-pair between columns k and l as defined in the text (because $j = l$ and $i \neq k$).

Next, I removed conflicts using an iterative alignment refinement procedure that eliminates sequences with more than 15 conflicts after each iteration. The alignments at each stage are determined using a CM. The initial consensus structure used to build the CM for the first iteration was defined as the set of consensus base pairs between alignment positions i and j that exist as paired in more than x fraction of the individual structures. The value for x was determined as the minimum value for which there were no conflicting base-pairs in the consensus set.

This provided me with an initial alignment that I then iteratively refined using INFER-

			cons	#	avg #	avg #	initial sequence removal			
model	stage	# seqs	struct x	cons bps	indiv. bps	conflict bps	total	ambig	short	long
archaea	CRW matches	23	0.170	472	456.74	1.30	0	0	0	0
archaea	post-initial removal	23	0.170	472	456.74	1.30				
archaea	1st refinement	23	0.130	474	456.74	0.52				
bacteria	CRW matches	95	0.210	480	460.91	1.06	2	2	0	0
bacteria	post-initial removal	93	0.200	480	460.82	1.05				
bacteria	1st refinement	93	0.210	480	460.82	1.28				
eukarya	CRW matches	148	0.420	422	466.25	12.94	42	11	33	7
eukarya	post-initial removal	106	0.440	442	487.50	16.04				
eukarya	1st refinment	106	0.410	448	487.50	8.71				
eukarya	2nd refinement	89	0.440	448	487.73	5.49				

Table 9.2: **Statistics on the conversion of CRW data to seed alignments for SSU-align** For each of the three models, statistics for the alignments at each stage of the CRW conversion process are shown. "CRW matches" alignments are subsets of the CRW alignments for matching sequences and individual structures. "post-initial removal" alignments have had sequences more than 120% the median length ("long" column), less than 80% the median length ("short" column), or with more than 5 ambiguous bases ("ambig" column) removed. The remaining rows are for alignments following each round of the iterative refinement process using INFERNAL. More details on the CRW conversion are in the text.

NAL. Each iteration consists of a build step, an alignment step, and a sequence removal step. First, a CM is built from the current alignment (in iteration 1 this is a subset of the CRW alignment). Then all of the seed sequences are aligned to the CM to generate a new alignment. The individual structures are mapped onto the new alignment and a new consensus structure is derived as described above. Any sequence with more than 15 base-pair conflicts between its individual structure and the new consensus structure are removed from the seed. This procedure continues until 0 sequences are removed in the final step. The alignment generated during the final iteration became the seed alignment I used for SSU-ALIGN.

Table 9.2 lists the number of sequences removed at each stage of the procedure and statistics on base-pair conflicts. The three final seed alignments used to create the SSU-ALIGN models are summarized in Table 9.3.

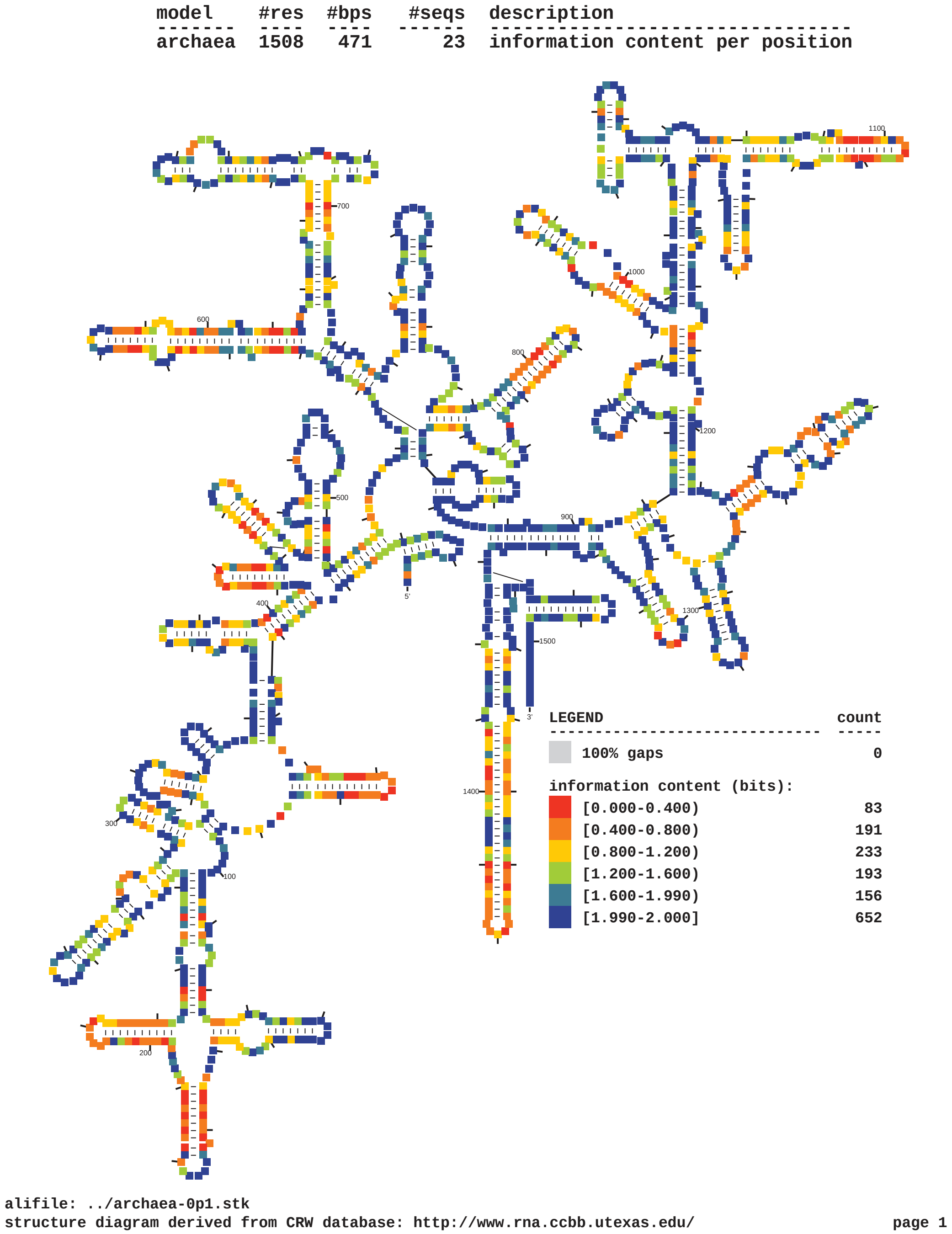
The remainder of this chapter includes 12 secondary structure diagrams displaying various statistics per consensus column of the three seed alignments, with figure numbers as

model name	number of sequences	consensus length	alignment length	number of base-pairs	average sequence length	average pairwise identity
archaea	23	1508	1563	471	1485	81%
bacteria	93	1582	1689	480	1527	80%
eukarya	89	1881	2652	448	1800	79%

Table 9.3: **Statistics of the three seed alignments used by SSU-align.** These are the three alignments resulting from the CRW conversion depicted in Figure 9.5 and described in the text.

indicated below:

statistic	archaea	bacteria	eukarya
information content	Figure 9.7	Figure 9.11	Figure 9.15
extra information from structure	Figure 9.8	Figure 9.12	Figure 9.16
frequency of deletions	Figure 9.9	Figure 9.13	Figure 9.17
frequency of insertions	Figure 9.10	Figure 9.14	Figure 9.18



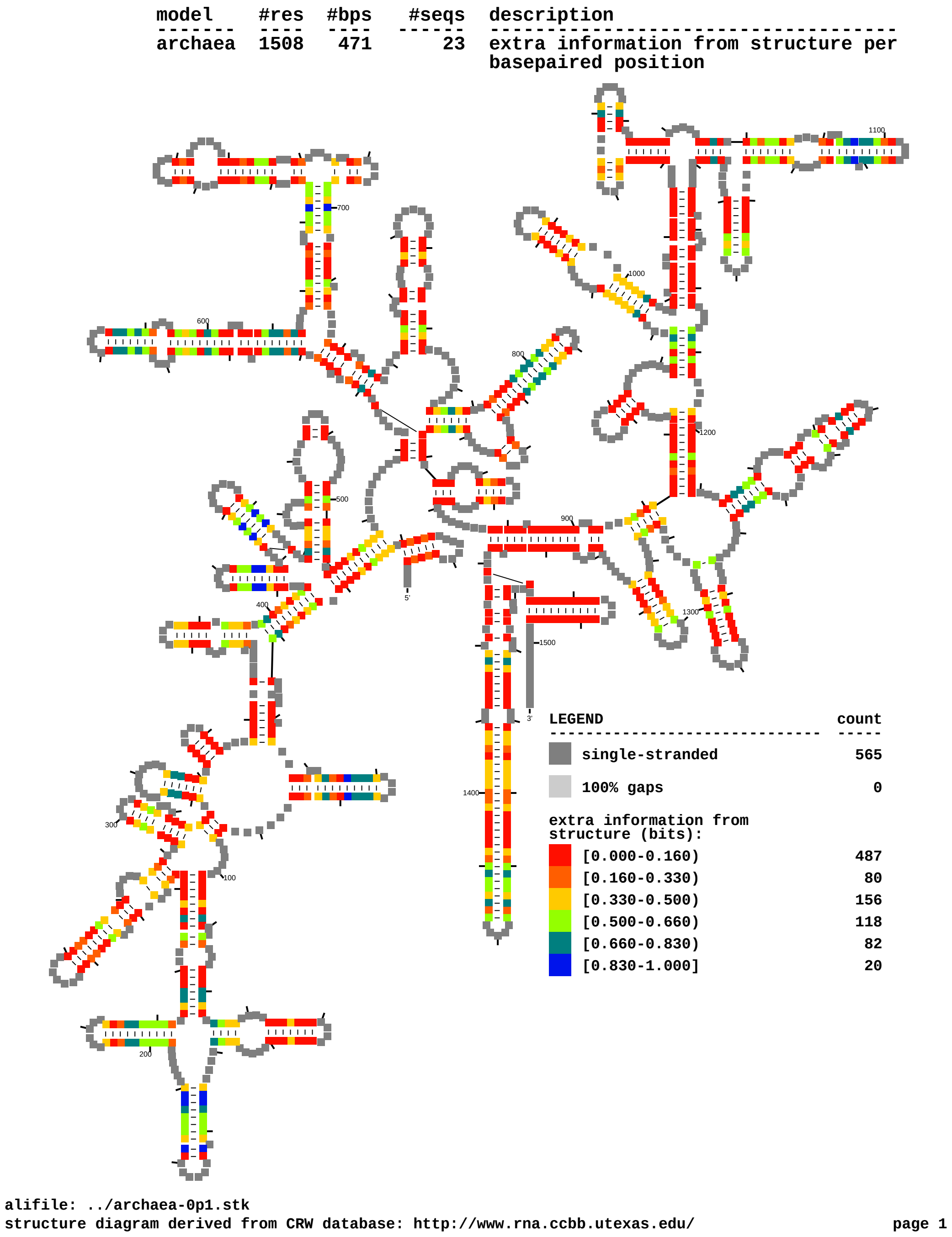
```
model    #res  #bps   #seqs  description
-------  ----  ----  ------  ---------------------------------
archaea  1508   471      23  information content per position
```

```
LEGEND                              count
------------------------------      -----
⬜ 100% gaps                            0

information content (bits):
🟥 [0.000-0.400)                       83
🟧 [0.400-0.800)                      191
🟨 [0.800-1.200)                      233
🟩 [1.200-1.600)                      193
■ [1.600-1.990)                       156
🟦 [1.990-2.000]                      652
```

```
alifile: ../archaea-0p1.stk
structure diagram derived from CRW database: http://www.rna.ccbb.utexas.edu/                  page 1
```

Figure 9.7: **Secondary structure diagram displaying primary sequence information content per consensus position of the archaeal SSU seed alignment.** Statistics correspond to the SSU-ALIGN seed alignment derived from the CRW database [32] as described in the text. This diagram was generated using the `ssu-draw` program included in the SSU-ALIGN package.



```
model     #res  #bps   #seqs  description
-------   ----  ----   -----  -----------------------------------
archaea   1508   471      23  extra information from structure per
                              basepaired position
```

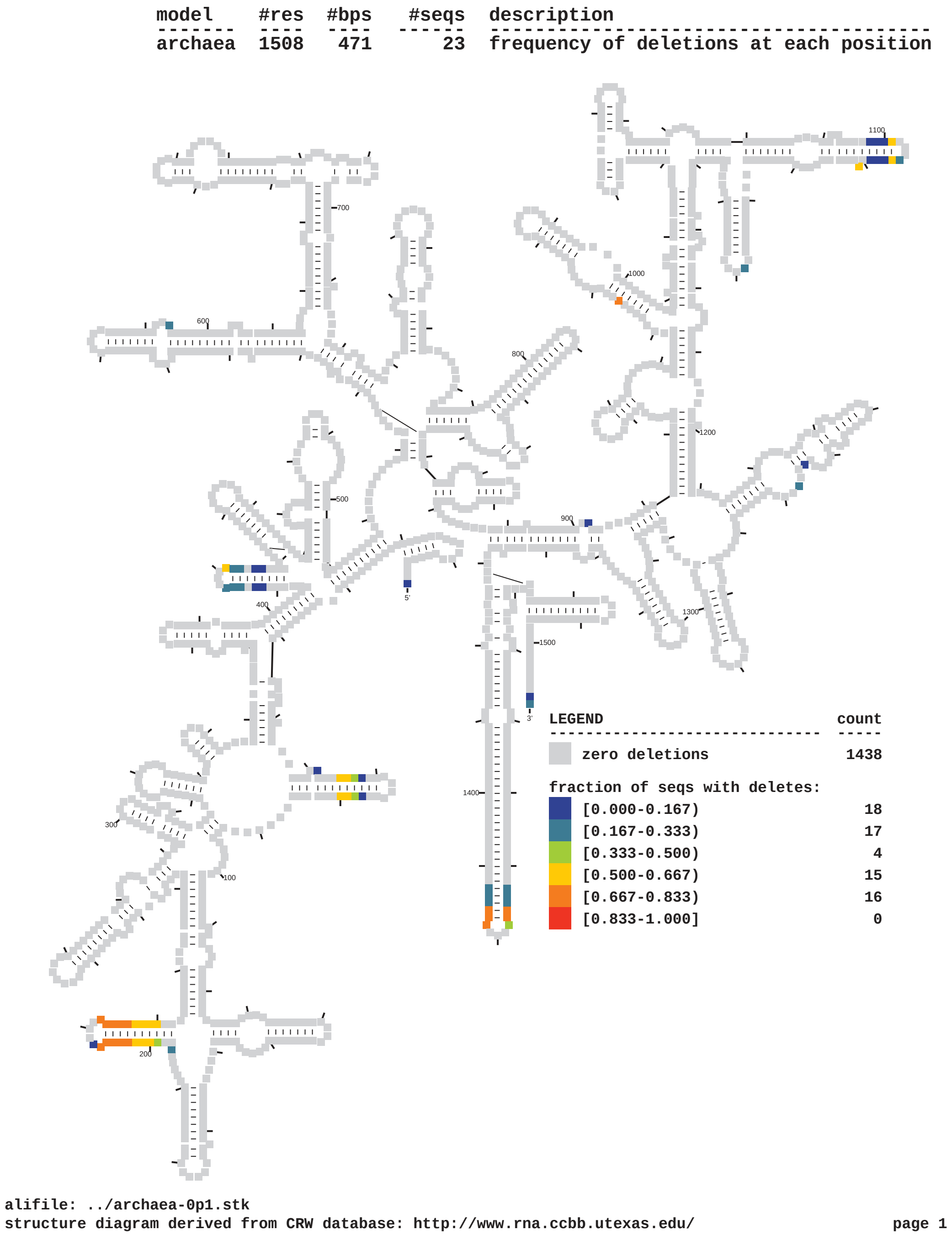
```
LEGEND                                   count
-----------------------------            -----
■ single-stranded                          565

▨ 100% gaps                                  0

extra information from
structure (bits):
🟥 [0.000-0.160)                           487
🟧 [0.160-0.330)                            80
🟨 [0.330-0.500)                           156
🟩 [0.500-0.660)                           118
■ [0.660-0.830)                             82
🟦 [0.830-1.000]                            20
```

```
alifile: ../archaea-0p1.stk
structure diagram derived from CRW database: http://www.rna.ccbb.utexas.edu/            page 1
```

Figure 9.8: **Secondary structure diagram displaying extra information from conserved structure per consensus position of the archaeal SSU seed alignment.** Statistics correspond to the SSU-ALIGN seed alignment derived from the CRW database [32] as described in the text. This diagram was generated using the `ssu-draw` program included in the SSU-ALIGN package.



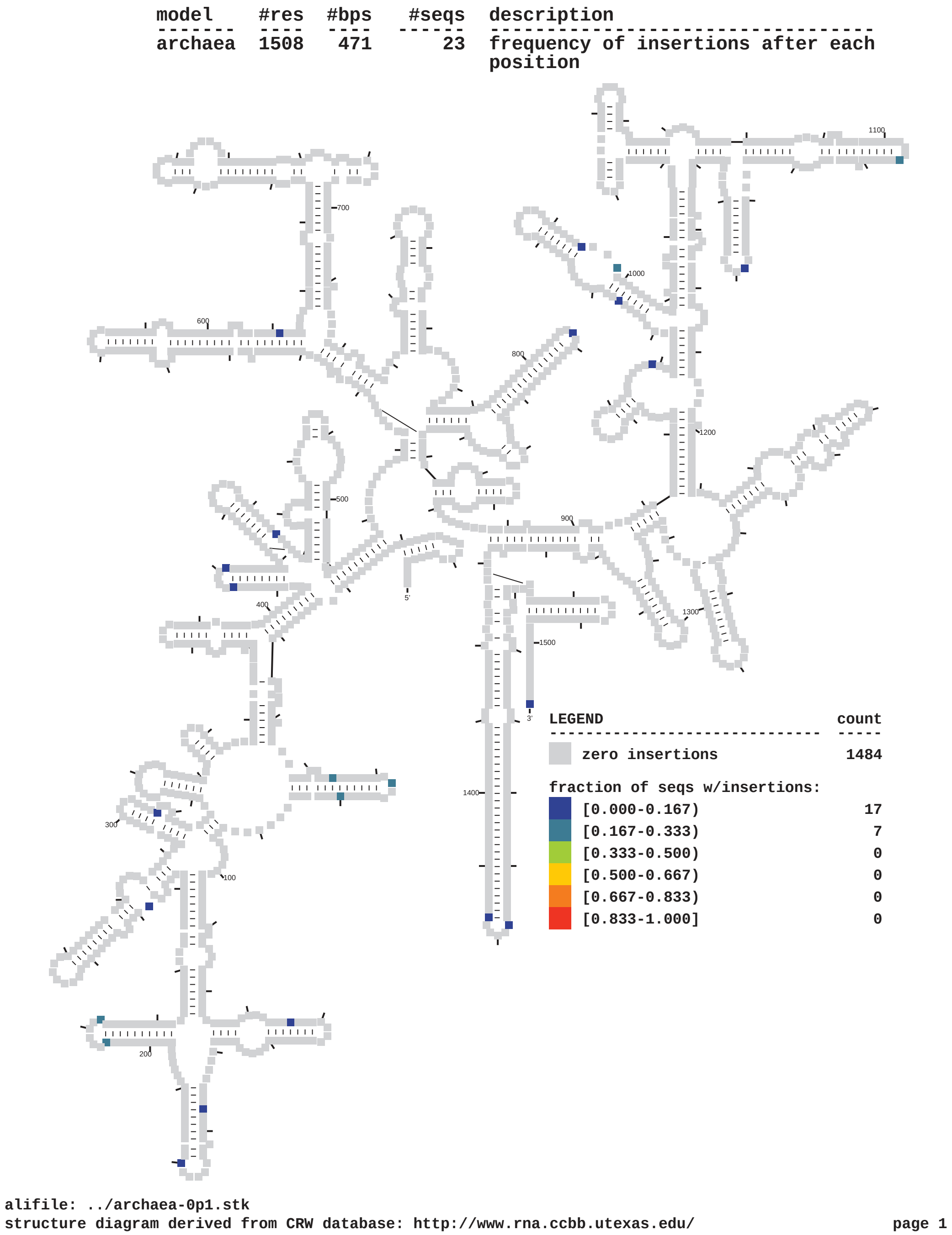
```
model    #res  #bps   #seqs  description
-------  ----  ----  ------  ----------------------------------------
archaea  1508   471      23  frequency of deletions at each position
```

```
LEGEND                                    count
------------------------------            -----
▨ zero deletions                           1438

fraction of seqs with deletes:
■ [0.000-0.167)                              18
■ [0.167-0.333)                              17
■ [0.333-0.500)                               4
■ [0.500-0.667)                              15
■ [0.667-0.833)                              16
■ [0.833-1.000]                               0
```

```
alifile: ../archaea-0p1.stk
structure diagram derived from CRW database: http://www.rna.ccbb.utexas.edu/                page 1
```

Figure 9.9: **Secondary structure diagram displaying frequency of deletions per consensus position of the archaeal SSU seed alignment.** Statistics correspond to the SSU-ALIGN seed alignment derived from the CRW database [32] as described in the text. This diagram was generated using the `ssu-draw` program included in the SSU-ALIGN package.



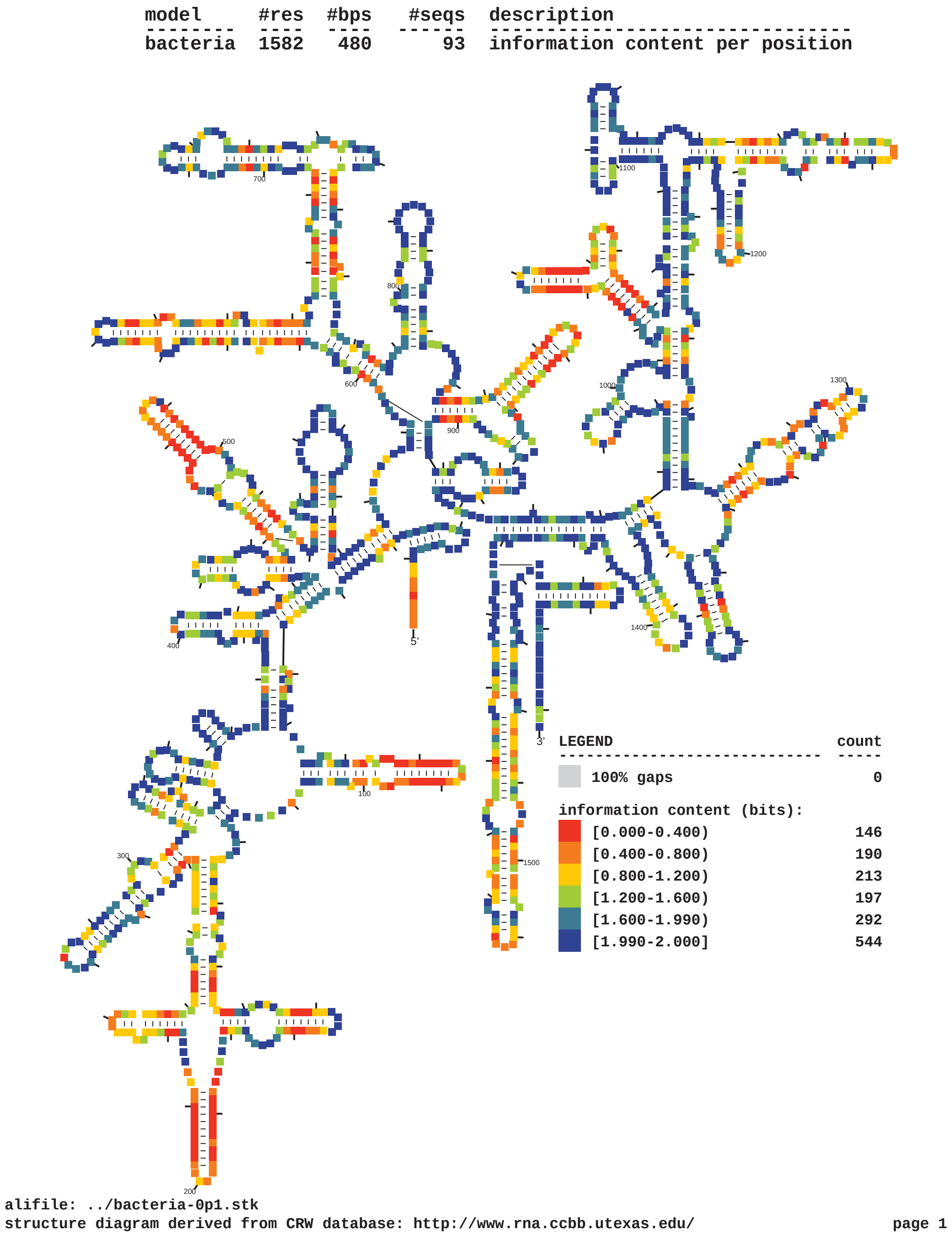
```
model    #res  #bps   #seqs  description
-------  ----  ----  ------  -----------------------------------
archaea  1508   471      23  frequency of insertions after each
                             position
```

```
LEGEND                                  count
-----------------------------           -----
⬜ zero insertions                        1484

fraction of seqs w/insertions:
🟦 [0.000-0.167)                            17
🟦 [0.167-0.333)                             7
🟩 [0.333-0.500)                             0
🟨 [0.500-0.667)                             0
🟧 [0.667-0.833)                             0
🟥 [0.833-1.000]                             0
```

```
alifile: ../archaea-0p1.stk
structure diagram derived from CRW database: http://www.rna.ccbb.utexas.edu/          page 1
```

Figure 9.10: **Secondary structure diagram displaying frequency of insertions after each consensus position in the archaeal SSU seed alignment.** Statistics correspond to the SSU-ALIGN seed alignment derived from the CRW database [32] as described in the text. This diagram was generated using the `ssu-draw` program included in the SSU-ALIGN package.



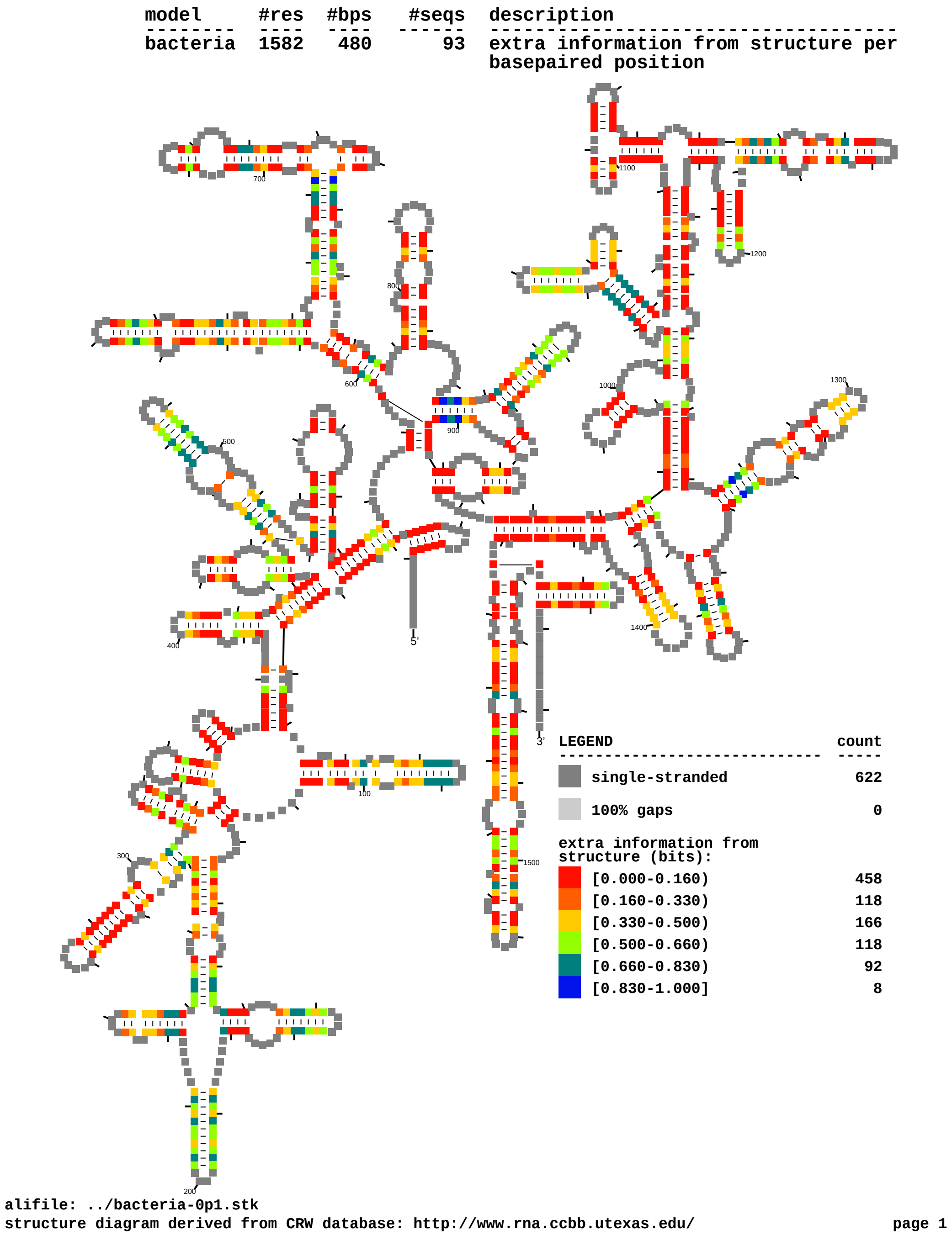
```
model      #res   #bps    #seqs   description
--------   ----   ----   ------   ----------------------------------
bacteria   1582    480       93   information content per position
```

```
LEGEND                                      count
-----------------------------               -----
▨  100% gaps                                    0

information content (bits):
🟥  [0.000-0.400)                              146
🟧  [0.400-0.800)                              190
🟨  [0.800-1.200)                              213
🟩  [1.200-1.600)                              197
🟦  [1.600-1.990)                              292
■  [1.990-2.000]                              544
```

```
alifile: ../bacteria-0p1.stk
structure diagram derived from CRW database: http://www.rna.ccbb.utexas.edu/                  page 1
```

Figure 9.11: **Secondary structure diagram displaying primary sequence information content per consensus position of the bacterial SSU seed alignment.** Statistics correspond to the SSU-ALIGN seed alignment derived from the CRW database [32] as described in the text. This diagram was generated using the `ssu-draw` program included in the SSU-ALIGN package.

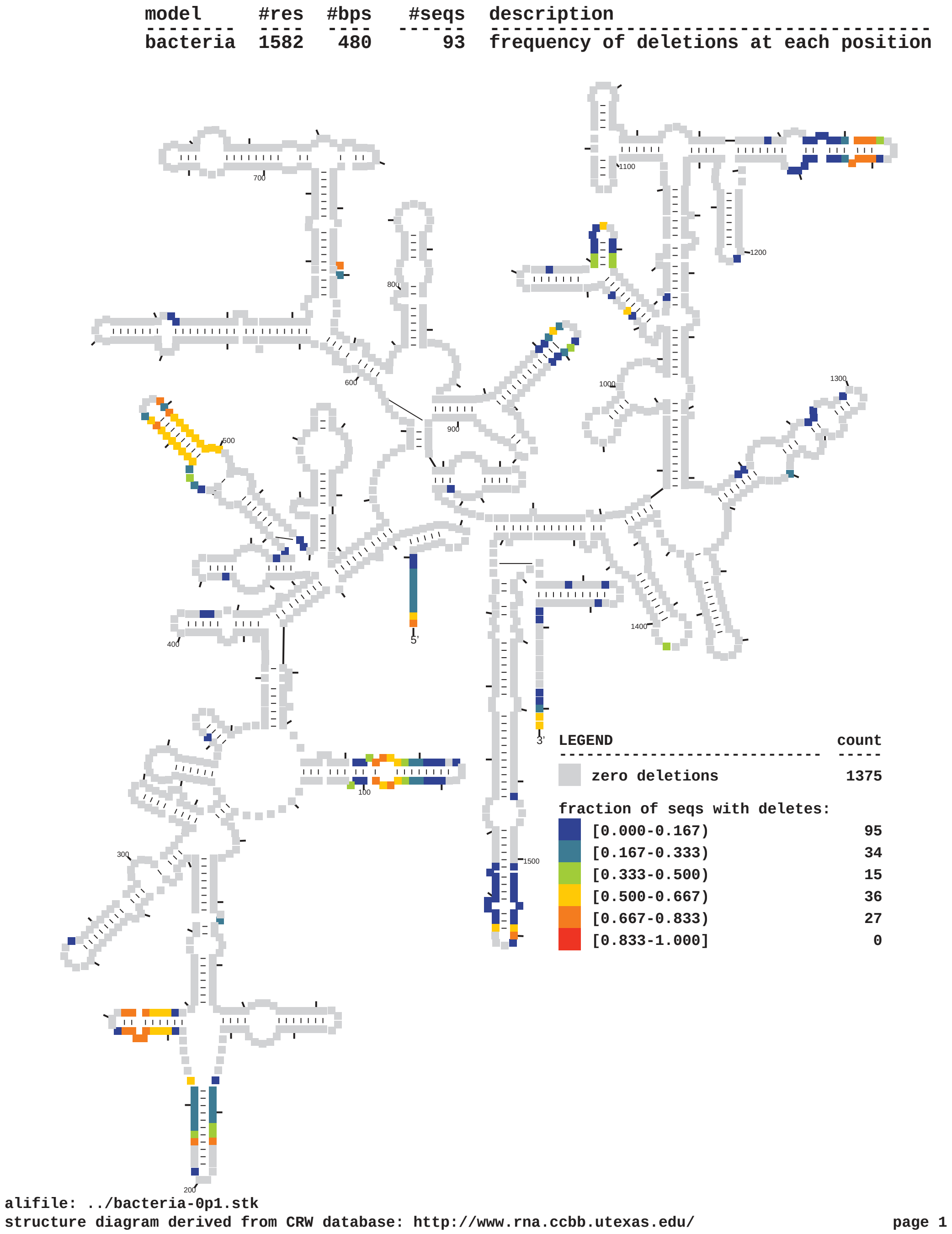


```
model      #res   #bps    #seqs   description
---------  ----   ----   ------   -----------------------------------
bacteria   1582    480       93   extra information from structure per
                                  basepaired position
```

```
LEGEND                                  count
----------------------------            -----
▇ single-stranded                         622
▇ 100% gaps                                 0
extra information from
structure (bits):
🟥 [0.000-0.160)                           458
🟧 [0.160-0.330)                           118
🟨 [0.330-0.500)                           166
🟩 [0.500-0.660)                           118
▇ [0.660-0.830)                            92
🟦 [0.830-1.000]                             8
```

```
alifile: ../bacteria-0p1.stk
structure diagram derived from CRW database: http://www.rna.ccbb.utexas.edu/                page 1
```

Figure 9.12: **Secondary structure diagram displaying extra information from conserved structure per consensus position of the bacterial SSU seed alignment.** Statistics correspond to the SSU-ALIGN seed alignment derived from the CRW database [32] as described in the text. This diagram was generated using the `ssu-draw` program included in the SSU-ALIGN package.



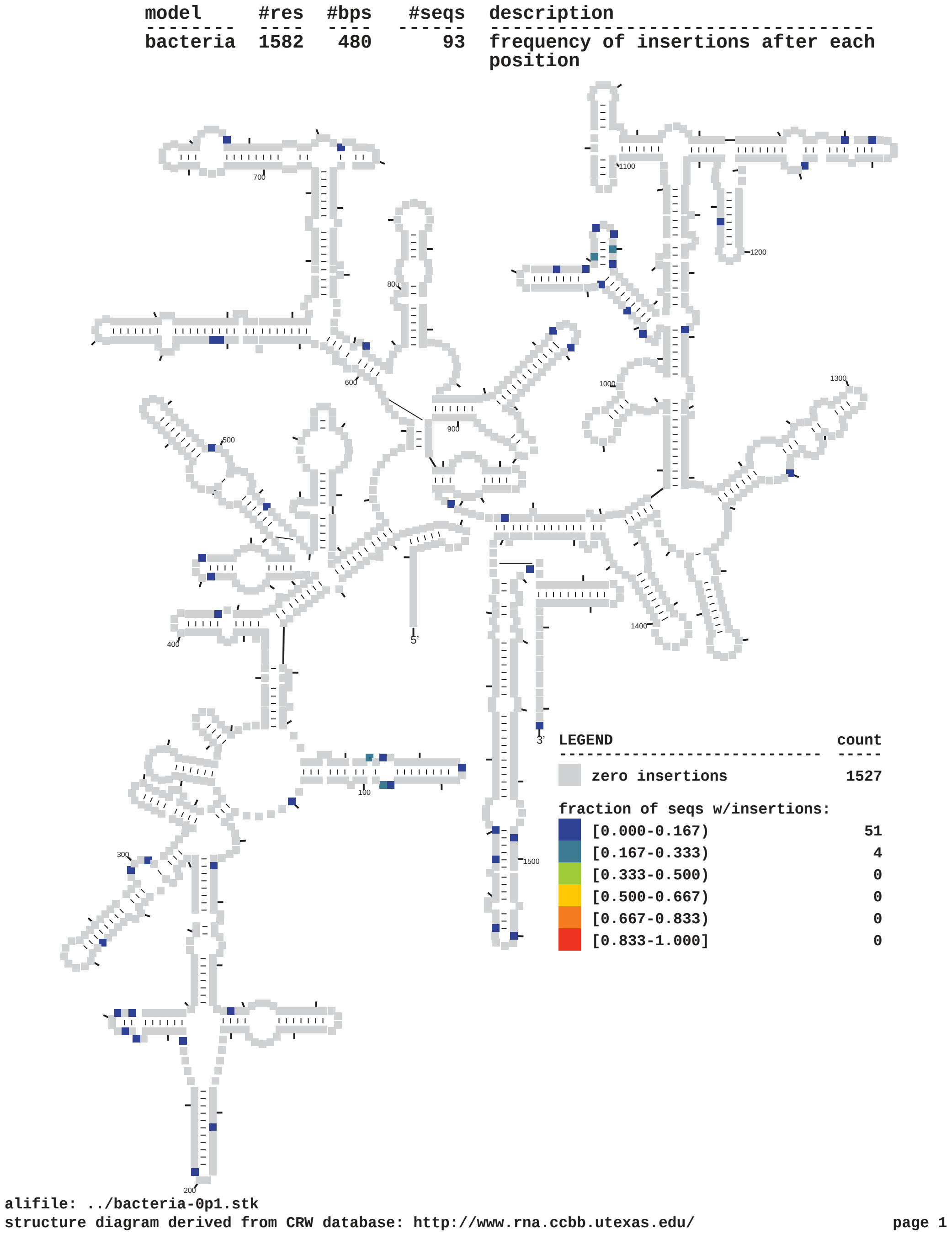
```
model       #res   #bps    #seqs   description
---------   ----   ----   ------   ----------------------------------------
bacteria    1582    480       93   frequency of deletions at each position
```

```
LEGEND                              count
------------------------------      -----
⬜ zero deletions                    1375

fraction of seqs with deletes:
🟦 [0.000-0.167)                       95
🟦 [0.167-0.333)                       34
🟩 [0.333-0.500)                       15
🟨 [0.500-0.667)                       36
🟧 [0.667-0.833)                       27
🟥 [0.833-1.000]                        0
```

```
alifile: ../bacteria-0p1.stk
structure diagram derived from CRW database: http://www.rna.ccbb.utexas.edu/                page 1
```

Figure 9.13: **Secondary structure diagram displaying frequency of deletions per consensus position of the bacterial SSU seed alignment.** Statistics correspond to the SSU-ALIGN seed alignment derived from the CRW database [32] as described in the text. This diagram was generated using the `ssu-draw` program included in the SSU-ALIGN package.

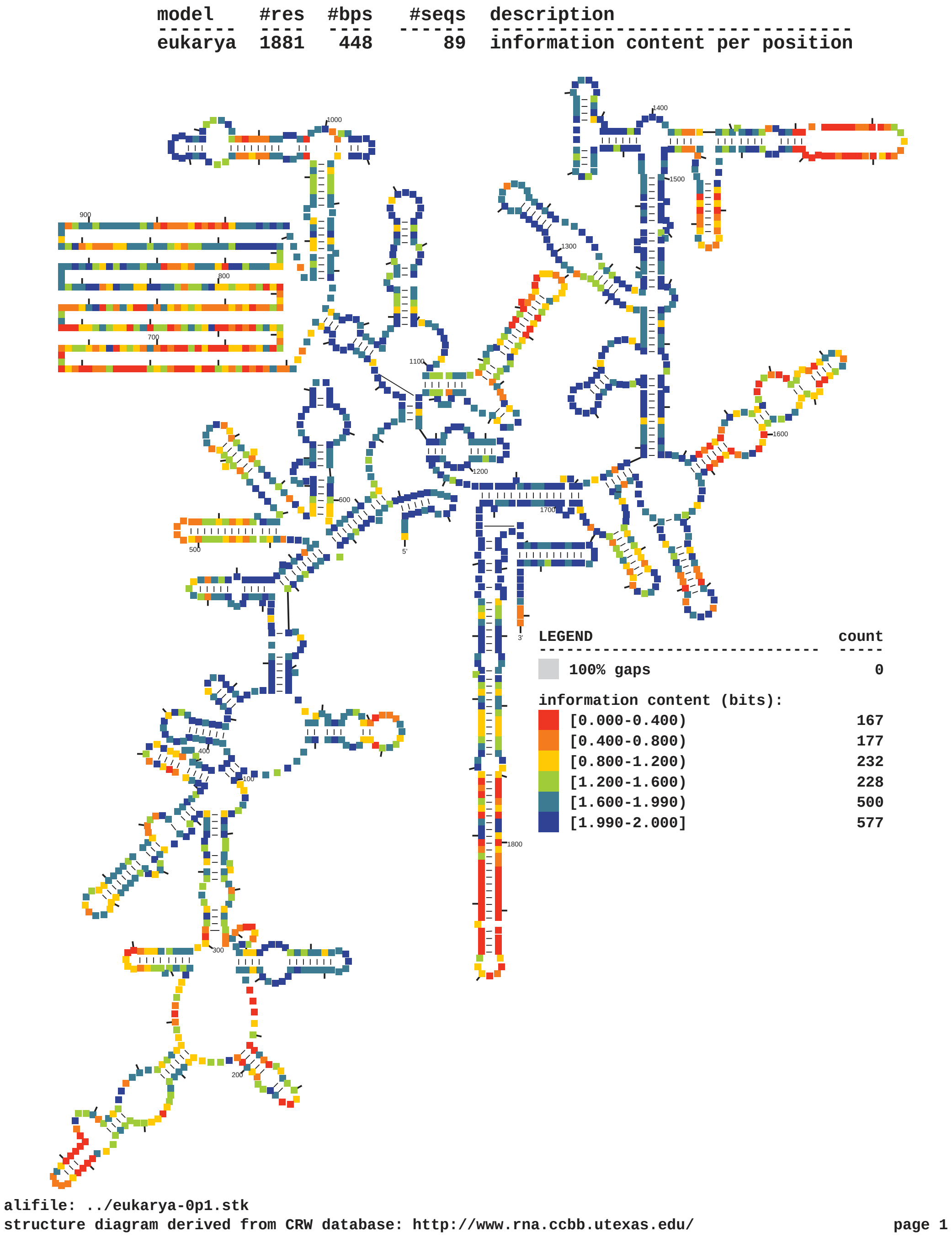


model	#res	#bps	#seqs	description
bacteria	1582	480	93	frequency of insertions after each position

LEGEND		count
⬜	zero insertions	1527
fraction of seqs w/insertions:		
🟦	[0.000-0.167)	51
🟦	[0.167-0.333)	4
🟩	[0.333-0.500)	0
🟨	[0.500-0.667)	0
🟧	[0.667-0.833)	0
🟥	[0.833-1.000]	0

alifile: ../bacteria-0p1.stk
structure diagram derived from CRW database: http://www.rna.ccbb.utexas.edu/ page 1

Figure 9.14: **Secondary structure diagram displaying frequency of insertions after each consensus position in the bacterial SSU seed alignment.** Statistics correspond to the SSU-ALIGN seed alignment derived from the CRW database [32] as described in the text. This diagram was generated using the `ssu-draw` program included in the SSU-ALIGN package.



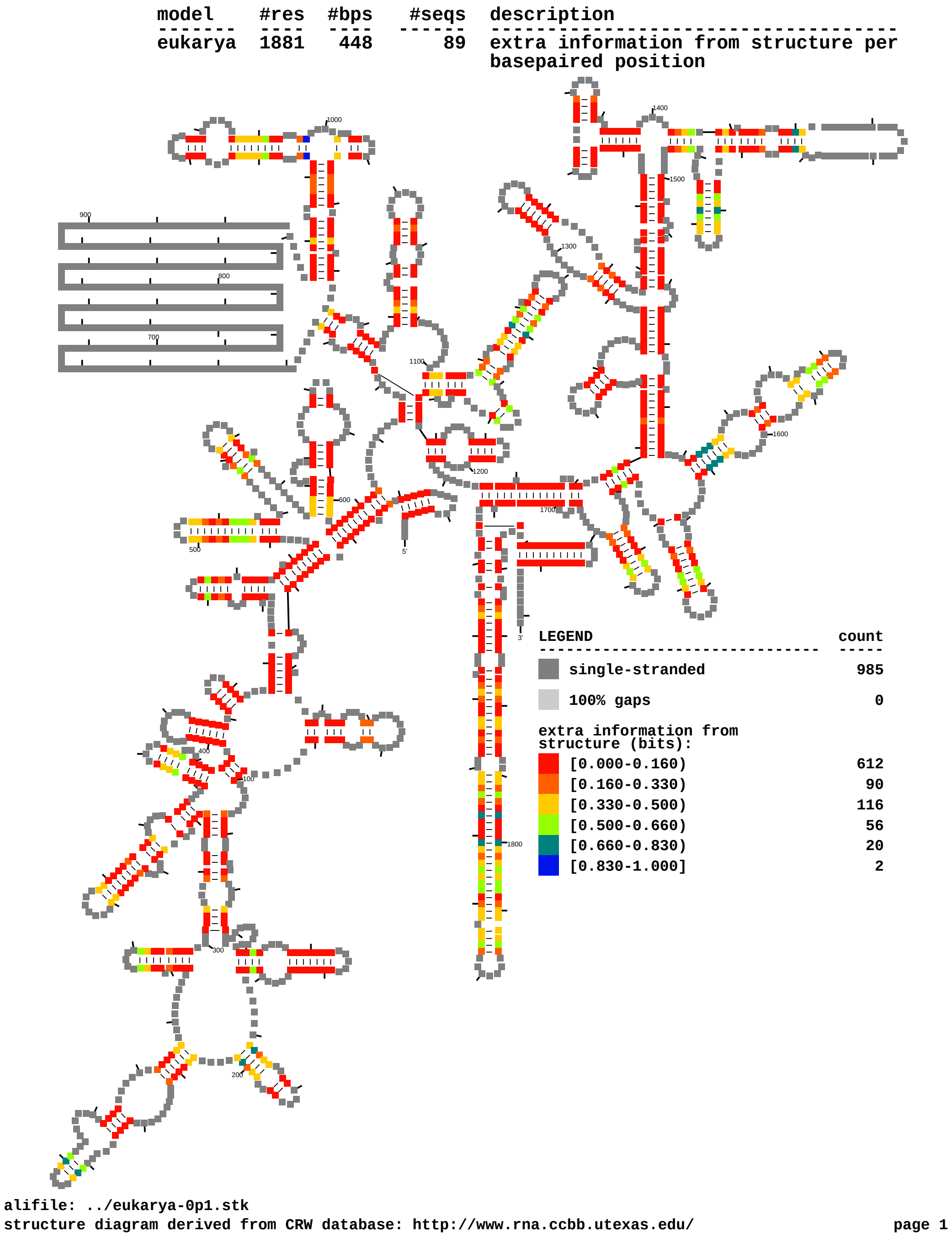
```
model     #res   #bps    #seqs   description
-------   ----   ----   ------   ---------------------------------
eukarya   1881    448       89   information content per position
```

```
LEGEND                                  count
-------------------------------         -----
⬜ 100% gaps                                0

information content (bits):
🟥 [0.000-0.400)                          167
🟧 [0.400-0.800)                          177
🟨 [0.800-1.200)                          232
🟩 [1.200-1.600)                          228
🟦 [1.600-1.990)                          500
🟦 [1.990-2.000]                          577
```

```
alifile: ../eukarya-0p1.stk
structure diagram derived from CRW database: http://www.rna.ccbb.utexas.edu/              page 1
```

Figure 9.15: **Secondary structure diagram displaying primary sequence information content per consensus position of the eukaryotic SSU seed alignment.** Statistics correspond to the SSU-ALIGN seed alignment derived from the CRW database [32] as described in the text. This diagram was generated using the `ssu-draw` program included in the SSU-ALIGN package.



```
model     #res   #bps    #seqs   description
-------   ----   ----   ------   ------------------------------------
eukarya   1881    448       89   extra information from structure per
                                 basepaired position
```

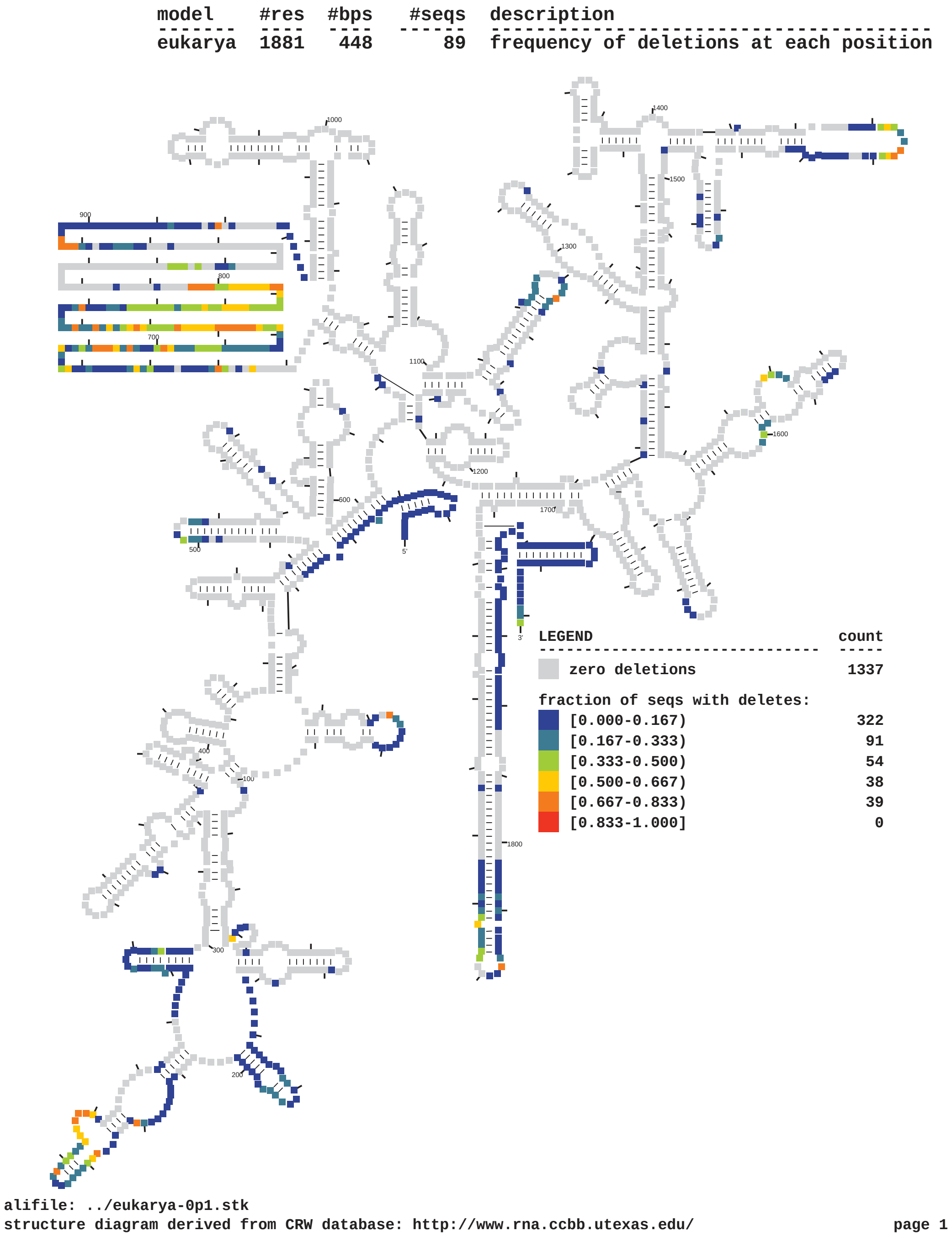
```
LEGEND                                   count
-------------------------------          -----
■ single-stranded                          985

▨ 100% gaps                                  0

extra information from
structure (bits):
🟥 [0.000-0.160)                           612
🟧 [0.160-0.330)                            90
🟨 [0.330-0.500)                           116
🟩 [0.500-0.660)                            56
🟩 [0.660-0.830)                            20
🟦 [0.830-1.000]                             2
```

```
alifile: ../eukarya-0p1.stk
structure diagram derived from CRW database: http://www.rna.ccbb.utexas.edu/                 page 1
```

Figure 9.16: **Secondary structure diagram displaying extra information from conserved structure per consensus position of the eukaryotic SSU seed alignment.** Statistics correspond to the SSU-ALIGN seed alignment derived from the CRW database [32] as described in the text. This diagram was generated using the `ssu-draw` program included in the SSU-ALIGN package.



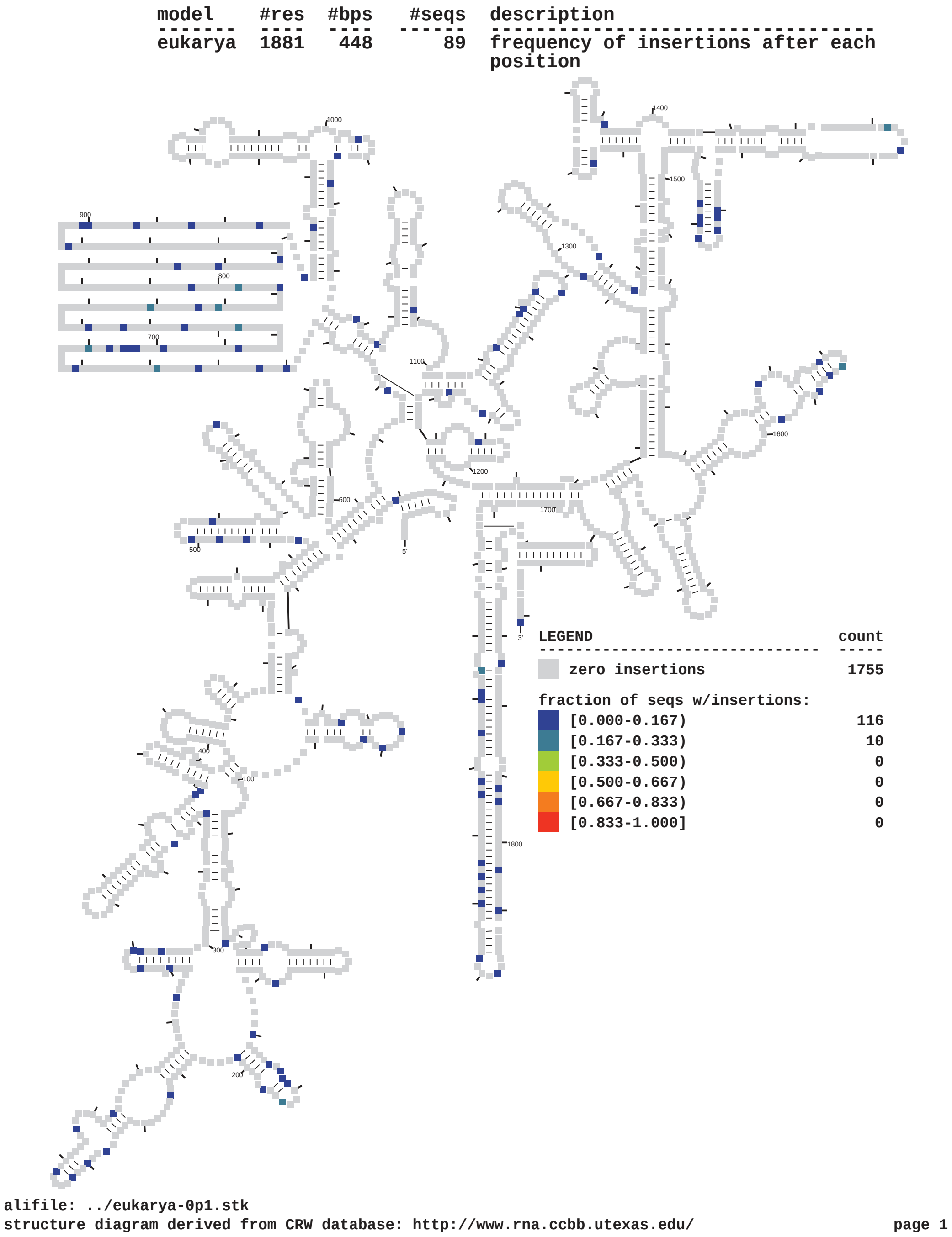
```
model     #res   #bps    #seqs   description
-------   ----   ----   ------   ---------------------------------------
eukarya   1881    448       89   frequency of deletions at each position
```

```
LEGEND                                  count
--------------------------------        -----
⬜ zero deletions                         1337

fraction of seqs with deletes:
🟦 [0.000-0.167)                           322
🟦 [0.167-0.333)                            91
🟩 [0.333-0.500)                            54
🟨 [0.500-0.667)                            38
🟧 [0.667-0.833)                            39
🟥 [0.833-1.000]                             0
```

```
alifile: ../eukarya-0p1.stk
structure diagram derived from CRW database: http://www.rna.ccbb.utexas.edu/                page 1
```

Figure 9.17: **Secondary structure diagram displaying frequency of deletions per consensus position of the eukaryotic SSU seed alignment.** Statistics correspond to the SSU-ALIGN seed alignment derived from the CRW database [32] as described in the text. This diagram was generated using the `ssu-draw` program included in the SSU-ALIGN package.



```
model     #res   #bps    #seqs   description
-------   ----   ----   ------   -----------------------------------
eukarya   1881    448       89   frequency of insertions after each
                                 position
```

```
LEGEND                                    count
------------------------------            -----
■ zero insertions                          1755

fraction of seqs w/insertions:
■ [0.000-0.167)                             116
■ [0.167-0.333)                              10
■ [0.333-0.500)                               0
■ [0.500-0.667)                               0
■ [0.667-0.833)                               0
■ [0.833-1.000]                               0
```

```
alifile: ../eukarya-0p1.stk
structure diagram derived from CRW database: http://www.rna.ccbb.utexas.edu/                page 1
```

Figure 9.18: **Secondary structure diagram displaying frequency of insertions after each consensus position in the eukaryotic SSU seed alignment.** Statistics correspond to the SSU-ALIGN seed alignment derived from the CRW database [32] as described in the text. This diagram was generated using the `ssu-draw` program included in the SSU-ALIGN package.

Bibliography

[1] S. F. Altschul. Amino acid substitution matrices from an information theoretic perspective. *J. Mol. Biol.*, 219:555–565, 1991.

[2] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215:403–410, 1990.

[3] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucl. Acids Res.*, 25:3389–3402, 1997.

[4] V. Ambros. The functions of animal microRNAs. *Nature*, 431:350–355, 2004.

[5] K. A. Amiri. Fibrillarin-like proteins occur in the domain archaea. *J. Bacteriol.*, 176: 2124–2127, 1994.

[6] T. Babak, B. J. Blencowe, and T. R. Hughes. Considerations in the identification of functional RNA structural elements in genomic alignments. *BMC Bioinformatics*, 8: 33, 2007.

[7] J. P. Bachellerie, J. Cavaille, and A. Hüttenhofer. The expanding snoRNA world. *Biochimie*, 84:775–790, 2002.

[8] G. C. Baker, J. J. Smith, and D. A. Cowan. Review and re-analysis of domain-specific 16S primers. *J Microbiol Methods.*, 55:541–555, 2003.

[9] N. Ban, P. Nissen, J. Hansen, P. B. Moore, and T. A. Steitz. The complete atomic structure of the large ribosomal subunit at 2.4 å resolution. *Science*, 289:905–920, 2000.

[10] D. P. Bartel. MicroRNAs: genomics, biogenesis, mechanism, and function. *Cell*, 116: 281–297, 2004.

[11] G. Bejerano, M. Pheasant, I. Makunin, S. Stephen, W. J. Kent, J. S. Mattick, and D. Haussler. Ultraconserved elements in the human genome. *Science*, 304:1321–1325, 2004.

[12] D. A. Benson, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, and D. L. Wheeler. GenBank. *Nucleic Acids Res.*, 34:D16–D20, 2006.

[13] E. M. Bik, P. B. Eckburg, S. R. Gill, K. E. Nelson, E. A. Purdom, F. Francois, G. Perez-Perez, M. J. Blaser, and D. A. Relman. Molecular analysis of the bacterial microbiota in the human stomach. *Proc Natl Acad Sci U S A.*, 103:732–737, 2006.

[14] E. H. Blackburn. Telomerase. In R. F. Gesteland, T. R. Cech, and J. F. Atkins, editors, *The RNA World, Second Edition*, pages 609–635. Cold Spring Harbor Laboratory Press, New York, 1999.

[15] R. K. Bradley, L. Pachter, and I. Holmes. Specific alignment of structured RNA: stochastic grammars and sequence annealing. *Bioinformatics.*, 24:2677–2683, 2008.

[16] R. R. Breaker. Riboswitches and the RNA world. In R. F. Gesteland, T. R. Cech, and J. F. Atkins, editors, *The RNA World, Third Edition*, pages 89–107. Cold Spring Harbor Laboratory Press, New York, 2006.

[17] S. Brenner, F. Jacob, and M. Meselson. An unstable intermediate carrying information from genes to ribosomes for protein synthesis. *Nature*, 190:576–581, 1961.

[18] S. E. Brenner, C. Chothia, and T. J. P. Hubbard. Assessing sequence comparison

methods with reliable structurally identified distant evolutionary relationships. *Proc. Natl. Acad. Sci. USA*, 95:6073–6078, 1998.

[19] M. R. Brent. Genome annotation past, present, and future: how to define an ORF at each locus. *Genome Res.*, 15:1777–1786, 2005.

[20] T. D. Brock. Life at high temperatures. evolutionary, ecological, and biochemical significance of organisms living in hot springs is discussed. *Science.*, 158:1012–1019, 1967.

[21] J. Brosius, M. L. Palmer, P. J. Kennedy, and H. F. Noller. Complete nucleotide sequence of a 16S ribosomal RNA gene from escherichia coli. *Proc Natl Acad Sci U S A.*, 75:4801–4805, 1978.

[22] J. W. Brown. The ribonuclease P database. *Nucl. Acids Res.*, 27:314, 1999.

[23] M. Brown, R. Hughey, A. Krogh, I. S. Mian, K. Sjolander, and D. Haussler. Using dirichlet mixture priors to derive hidden Markov models for protein families. In L. Hunter, D. Searls, and J. Shavlik, editors, *Proceedings of the First International Conference on Intelligent Systems for Molecular Biology*, pages 47–55, Menlo Park, CA, 1993. AAAI.

[24] M. P. Brown. Small subunit ribosomal RNA modeling using stochastic context-free grammars. *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, 8:57–66, 2000.

[25] M. Brudno, C. B. Do, G. M. Cooper, M. F. Kim, E. D. Davydov, NISC Comparative Sequencing Program, E. Green, A. Sidow, and S. Batzoglou. LAGAN and multi-LAGAN: efficient tools for large-scale multiple alignment of genomic DNA. *Genome Res.*, 13:721–731, 2003.

[26] P. Bucher and K. Hofmann. A sequence similarity search algorithm based on a probabilistic interpretation of an alignment scoring system. *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, 4:44–51, 1996.

[27] C. B. Burge and S. Karlin. Finding the genes in genomic DNA. *Curr. Opin. Struct. Biol.*, 8:346–354, 1998.

[28] C. B. Burge, T. Tuschl, and P. A. Sharp. Splicing of precursors to mRNAs by the spliceosomes. In R. F. Gesteland, T. R. Cech, and J. F. Atkins, editors, *The RNA World, Second Edition*, pages 525–560. Cold Spring Harbor Laboratory Press, Cold Spring Harbor, New York, 1999.

[29] A. B. Burgin and N. R. Pace. Mapping the active site of ribonuclease P RNA using a substrate containing a photoaffinity agent. *EMBO J.*, 9:4111–4118, 1990.

[30] N. Bushati and S. Cohen. microRNA functions. *Annu Rev Cell Dev Biol.*, 23:175–205, 2007.

[31] S. A. Cameron, H. M. Hines, and P. H. Williams. A comprehensive phylogeny of the bumble bees (*Bombus*). *Biological Journal of the Linnean Society*, 91:161–188, 2007.

[32] J. J. Cannone, S. Subramanian, M. N. Schnare, J. R. Collett, L. M. D'Souza, Y. Du, B. Feng, N. Lin, L. V. Madabusi, K. M. Müller, N. Pande, Z. Shang, N. Yu, and R. R. Gutell. The Comparative RNA Web (CRW) Site: an online database of comparative sequence and structure information for ribosomal, intron, and other RNAs. *BMC Bioinformatics*, 3:2, 2002.

[33] J. L. Chen and C. W. Greider. An emerging consensus for telomerase RNA structure. *Proc Natl Acad Sci U S A.*, 101:14683–14684, 2004.

[34] J. L. Chen and N. R. Pace. Identification of the universally conserved core of ribonuclease P RNA. *RNA.*, 3:557–560, 1997.

[35] Noam Chomsky. Three models for the description of language. *IRE Transact. Information Theory*, 2:113–124, 1956.

[36] Noam Chomsky. On certain formal properties of grammars. *Information and Control*, 2:137–167, 1959.

[37] J. R. Cole, B. Chai, T. L. Marsh, R. J. Farris, Q. Wang, S. A. Kulam, S. Chandra, D. M. McGarrell, T. M. Schmidt, G. M. Garrity, and J. M. Tiedje. The ribosomal database project (RDP-II): Previewing a new autoaligner that allows regular updates and the new prokaryotic taxonomy. *Nucl. Acids Res.*, 31:442–443, 2003.

[38] J. R. Cole, B. Chai, R. J. Farris, Q. Wang, S. A. Kulam, D. M. McGarrell, G. M. Garrity, and J. M. Tiedje. The ribosomal database project (RDP-II): sequences and tools for high-throughput rRNA analysis. *Nucleic Acids Res.*, 33:D294–D296, 2005.

[39] J. R. Cole, B. Chai, R. J. Farris, Q. Wang, A. S. Kulam-Syed-Mohideen, D. M. McGarrell, A. M. Bandela, E. Cardenas, G. M. Garrity, and J. M. Tiedje. The ribosomal database project (RDP-II): introducing myRDP space and quality controlled public data. *Nucleic Acids Res.*, 35:D169–D172, 2007.

[40] 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. *Nucl. Acids Res.*, 37:D141–D145, 2009.

[41] F. F. Costa. Non-coding RNAs: lost in translation? *Gene.*, 386:1–10, 2007.

[42] F. H. C. Crick. On protein synthesis. *Symp. Soc. Exp. Biol.*, 12:138–163, 1958.

[43] M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt. A model of evolutionary change in proteins. In M. O. Dayhoff, editor, *Atlas of Protein Sequence and Structure*, pages 345–352. National Biomedical Research Foundation, Washington DC, 1978.

[44] C. del Val, E. Rivas, O. Torres-Quesada, N. Toro, and J. I. Jiménez-Zurdo. Identification of differentially expressed small non-coding RNAs in the legume endosymbiont *Sinorhizobium meliloti* by comparative genomics. *Mol. Microbiol.*, 66:1080–1091, 2007.

[45] N. Delihas and S. Forst. MicF: an antisense RNA gene involved in response of *Escherichia Coli* to global stress factors. *J Mol Biol.*, 313:1–12, 2001.

[46] T. Z. DeSantis, P. Hugenholtz, K. Keller, E. L. Brodie, N. Larsen, Y. M. Piceno, R. Phan, and G. L. Andersen. NAST: A multiple sequence alignment server for comparative analysis of 16S rRNA genes. *Nucleic Acid Res.*, 34:W394–399, 2006.

[47] 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.*, 72:5069–5072, 2006.

[48] R. F. Doolittle. Some reflections on the early days of sequence searching. *J Mol Med.*, 75:239–241, 1997.

[49] P. Doty, H. Boedtker, J. R. Fresco, R. Haselkorn, and M. Litt. Secondary structure in ribonucleic acids. *Proc Natl Acad Sci U S A.*, 45:482–499, 1959.

[50] R. D. Dowell and S. R. Eddy. Evaluation of several lightweight stochastic context-free grammars for RNA secondary structure prediction. *BMC Bioinformatics*, 5:71, 2004.

[51] R. D. Dowell and S. R. Eddy. Efficient pairwise RNA structure prediction and alignment using sequence alignment constraints. *BMC Bioinformatics*, 7:400, 2006.

[52] M. Dsouza, N. Larsen, and R. Overbeek. Searching for patterns in genomic data. *Trends Genet.*, 13:497–498, 1997.

[53] R. Durbin, S. R. Eddy, A. Krogh, and G. J. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* Cambridge University Press, Cambridge UK, 1998. ISBN 0521629713.

[54] S. R. Eddy. Where did the BLOSUM62 alignment score matrix come from? *Nat. Biotechnol.*, 22:1035–1036, 2004.

[55] S. R. Eddy. Non-coding RNA genes and the modern RNA world. *Nat. Rev. Genet.*, 2:919–929, 2001.

[56] S. R. Eddy. Computational genomics of noncoding RNA genes. *Cell*, 109:137–140, 2002.

[57] S. R. Eddy. A memory-efficient dynamic programming algorithm for optimal alignment of a sequence to an RNA secondary structure. *BMC Bioinformatics*, 3:18, 2002.

[58] S. R. Eddy. Computational analysis of RNAs. *Cold Spring Harbor Symp. Quant. Biol.*, 71:117–128, 2006.

[59] S. R. Eddy. A probabilistic model of local sequence alignment that simplifies statistical significance estimation. *PLoS Comput. Biol.*, 4:e1000069, 2008.

[60] S. R. Eddy. Profile hidden Markov models. *Bioinformatics*, 14:755–763, 1998.

[61] S. R. Eddy. Cove - fast pattern searching for RNA secondary structures. [ftp://selab.janelia.org/pub/software/cove/], 1996.

[62] S. R. Eddy. HMMER - biosequence analysis using profile hidden Markov models. [http://hmmer.janelia.org/], 2008.

[63] S. R. Eddy. The HMMER2 user's guide. [http://hmmer.janelia.org/], 2003.

[64] S. R. Eddy. Rnabob - fast pattern searching for RNA secondary structures. [ftp://selab.janelia.org/pub/software/rnabob/], 2005.

[65] S. R. Eddy and R. Durbin. RNA sequence analysis using covariance models. *Nucl. Acids Res.*, 22:2079–2088, 1994.

[66] V. P. Edgcomb, D. T. Kysela, A. Teske, A. de Vera Gomez, and M. L. Sogin. Benthic eukaryotic diversity in the guaymas basin hydrothermal vent environment. *Proc Natl Acad Sci U S A.*, 99:7658–7662, 2002.

[67] S. C. Elgin and S. I. Grewal. Heterochromatin: Silence is golden. *Curr Biol.*, 13: R895–R898, 2003.

[68] G. L. Eliceiri. Small nucleolar RNAs. *Cell Mol. Life Sci.*, 56:22–31, 1999.

[69] V. A. Erdmann, M. Z. Barciszewska, M. Symanski, A. Hochberg, N. de Groot, and J. Barciszewski. The non-coding RNAs as riboregulators. *Nucl. Acids Res.*, 29:189–193, 2001.

[70] M. D. Ermolaeva, H. G. Khalak, O. White, H. O. Smith, and S. L. Salzberg. Prediction of transcription terminators in bacterial genomes. *J Mol Biol.*, 301:27–33, 2000.

[71] D. Evans, S. M. Marquez, and N. R. Pace. RNase P: Interface of the RNA and protein worlds. *Trends Biochem Sci.*, 31:333–341, 2006.

[72] J. Felsenstein. *Inferring Phylogenies.* Sianuer Associates, Sunderland, Massachussetts, 2nd edition, 2003. ISBN 0878931775.

[73] D. F. Feng and R. F. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J Mol Evol.*, 25:351–360, 1987.

[74] Gwennaele A. Fichant and Christian Burks. Identifying potential tRNA genes in genomic DNA sequences. *J. Mol. Biol.*, 220:659–671, 1991.

[75] R. D. Finn, J. Mistry, B. Schuster-Bockler, S. Griffiths-Jones, V. Hollich, T. Lassmann, S. Moxon, M. Marshall, A. Khanna, R. Durbin, S. R. Eddy, E. L. Sonnhammer, and A. Bateman. Pfam: Clans, web tools and services. *Nucl. Acids Res.*, 34: D247–D251, 2006.

[76] R. D. Finn, J. Tate, J. Mistry, P. C. Coggill, S. J. Sammut, H.-R. Hotz, G. Ceric, K. Forslund, S. R. Eddy, E. L. L. Sonnhammer, and A. Bateman. The Pfam protein families database. *Nucl. Acids Res.*, 36:D281–D288, 2008.

[77] R. D. Fleischmann, M. D. Adams, O. White, R. A. Clayton, E. F. Kirkness, A. R. Kerlavage, C. J. Bult, J. F. Tomb, B. A. Dougherty, J. M. Merrick, K. McKenney, G. Sutton, W. FitzHugh, C. Fields, J. D. Gocayne, J. Scott, R. Shirley, L. I. Liu, A. Glodek, J. M. Kelley, J. F. Weidman, C. A. Phillips, T. Spriggs, E. Hedblom, M. D. Cotton, T. R. Utterback, M. C. Hanna, D. T. Nguyen, D. M. Saudek, R. C. Brandon,

L. D. Fine, J. L. Fritchman, J. L. Fuhrmann, N. S. M. Geoghagen, C. L. Gnehm, L. A. McDonald, K. V. Small, C. M. Fraser, H. O. Smith, and J. C. Venter. Whole-genome random sequencing and assembly of *Haemophilus influenzae* Rd. *Science*, 269:496–512, 1995.

[78] R. Fleissner, D. Metzler, and A. von Haeseler. Simultaneous statistical multiple alignment and phylogeny reconstruction. *Syst Biol.*, 54:548–561, 2005.

[79] A. C. Forster and R. H. Symons. Self-cleavage of plus and minus RNAs of a virusoid and a structural model for the active sites. *Cell.*, 49:211–220, 1987.

[80] A. C. Forster and R. H. Symons. Self-cleavage of virusoid RNA is performed by the proposed 55-nucleotide active site. *Cell.*, 50:9–16, 1987.

[81] G. E. Fox, K. R. Pechman, and C. R. Woese. Comparative cataloging of 16S ribosomal ribonucleic acid: Molecular approach to procaryotic systematics. *Int J Syst Bacteriol*, 27:44–57, 1977.

[82] George E. Fox and Carl R. Woese. 5S RNA secondary structure. *Nature*, 256:505–507, 1975.

[83] D. N. Frank and N. R. Pace. Gastrointestinal microbiology enters the metagenomics era. *Curr Opin Gastroenterol.*, 24:4–10, 2008.

[84] D. N. Frank and N. R. Pace. Ribonuclease P: Unity and diversity in a tRNA processing ribozyme. *Annu Rev Biochem.*, 67:153–180, 1998.

[85] E. K. Freyhult, J. P. Bollback, and P. P. Gardner. Exploring genomic dark matter: A critical assessment of the performance of homology search methods on noncoding RNA. *Genome Res.*, 17:117–125, 2007.

[86] T. Friend. *The Third Domain: The Untold Story of Archaea and the Future of Biotechnology*. Joseph Henry Press, Washington, DC, 2007.

[87] M. A. Furlong, D. R. Singleton, D. C. Coleman, and W. B. Whitman. Molecular and culture-based analyses of prokaryotic communities from an agricultural soil and the burrows and casts of the earthworm *Lumbricus rubellus*. *Appl Environ Microbiol.*, 68: 1265–1279, 2002.

[88] Z. Gao, C. H. Tseng, Z. Pei, and M. J. Blaser. Molecular analysis of human forearm superficial skin bacterial biota. *Proc Natl Acad Sci U S A.*, 104:2927–2932, 2007.

[89] P. P. Gardner, J. Daub, J. G. Tate, E. P. Nawrocki, D. L. Kolbe, S. Lindgreen, A. C. Wilkinson, R. D. Finn, S. Griffiths-Jones, S. R. Eddy, and A. Bateman. Rfam: Updates to the RNA families database. *Nucl. Acids Res.*, 37:D136–D140, 2009.

[90] D. Gautheret and A. Lambert. Direct RNA motif definition and identification from multiple sequence alignments using secondary structure profiles. *J. Mol. Biol.*, 313: 1003–1011, 2001.

[91] R. Giegerich. Explaining and controlling ambiguity in dynamic programming. In R. Giancarlo and D. Sankoff, editors, *Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching*, number 1848, pages 46–59, Montréal, Canada, 2000. Springer-Verlag, Berlin.

[92] W. Gilbert. The RNA world. *Nature*, 319:618, 1986.

[93] S. J. Giovannoni, T. B. Britschgi, C. L. Moyer, and K. G. Field. Genetic diversity in sargasso sea bacterioplankton. *Nature.*, 345:60–63, 1990.

[94] O. Gotoh. An improved algorithm for matching biological sequences. *J. Mol. Biol.*, 162:705–708, 1982.

[95] S. Gottesman. The small RNA regulators of *Escherichia coli*: Roles and mechanisms. *Annu. Rev. Microbiol.*, 5:303–328, 2004.

[96] S. Gottesman. Micros for microbes: Non-coding regulatory RNAs in bacteria. *Trends Genet.*, 7:399–404, 2005.

[97] Michael Gribskov, Andrew D. McLachlan, and David Eisenberg. Profile analysis: Detection of distantly related proteins. *Proc. Natl. Acad. Sci. USA*, 84:4355–4358, 1987.

[98] S. Griffiths-Jones. Annotating noncoding RNA genes. *Annu. Rev. Genomics Hum. Genet.*, 8:279–298, 2007.

[99] S. Griffiths-Jones, A. Bateman, M. Marshall, A. Khanna, and S. R. Eddy. Rfam: an RNA family database. *Nucl. Acids Res.*, 31:439–441, 2003.

[100] S. Griffiths-Jones, S. Moxon, M. Marshall, A. Khanna, S. R. Eddy, and A. Bateman. Rfam: Annotating non-coding RNAs in complete genomes. *Nucl. Acids Res.*, 33: D121–D141, 2005.

[101] G. Grillo, F. Licciulli, S. Liuni, E. Sbis, and G. Pesole. PatSearch: a program for the detection of patterns and structural motifs in nucleotide sequences. *Nucleic Acids Res.*, 31:3608–3612, 2003.

[102] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Computing*, 22: 789–828, 1996.

[103] W. N. Grundy. Homology detection via family pairwise search. *J. Comput. Biol.*, 5: 479–491, 1998.

[104] C. Guerrier-Takada, K. Gardiner, T. Marsh, N. Pace, and S. Altman. The RNA moiety of ribonuclease P is the catalytic subunit of the enzyme. *Cell*, 35:849–857, 1983.

[105] R. Gupta, J. M. Lanter, and C. R. Woese. Sequence of the 16S ribosomal RNA from halobacterium volcanii, an archaebacterium. *Science.*, 221:656–659, 1983.

[106] S. K. Gupta, J. Kececioglu, and A. A. Schaffer. Improving the practical space and time

efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment. *J. Cell. Biol.*, 2:459–472, 1995.

[107] R. R. Gutell, J. C. Lee, and J. J. Cannone. The accuracy of ribosomal RNA comparative structure models. *Curr. Opin. Struct. Biol.*, 12:301–310, 2002.

[108] C. Hammann and E. Westhof. Searching genomes for ribozymes and riboswitches. *Genome Biol.*, 8:210, 2007.

[109] J. Handelsman, M. R. Rondon, S. F. Brady, J. Clardy, and R. M. Goodman. Molecular biological access to the chemistry of unknown soil microbes: a new frontier for natural products. *Chem Biol.*, 5:R245–R249, 1998.

[110] Steven Henikoff and Jorja G. Henikoff. Automated assembly of protein blocks for database searching. *Nucl. Acids Res.*, 19:6565–6572, 1991.

[111] Steven Henikoff and Jorja G. Henikoff. Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. USA*, 89:10915–10919, 1992.

[112] T. M. Henkin. Riboswitch RNAs: using RNA to sense cellular metabolism. *Genes Dev.*, 22:3383–3390, 2008.

[113] M. W. Hentze and L. C. Khn. Molecular control of vertebrate iron metabolism: mRNA-based regulatory circuits operated by iron, nitric oxide, and oxidative stress. *Proc Natl Acad Sci U S A.*, 93:8175–8182, 1996.

[114] H. A. Heus and A. Pardi. Structural features that give rise to the unusual stability of RNA hairpins containing GNRA loops. *Science*, 253:191–194, 1991.

[115] R. W. Holley, J. Apgar, G. A. Everett, J. T. Madison, M. Marquisee, S. H. Merrill, J. R. Penswick, and A. Zamir. Structure of a ribonucleic acid. *Science*, 14:1462–1465, 1965.

[116] I. Holmes. A probabilistic model for the evolution of RNA structure. *BMC Bioinformatics.*, 5:166, 2004.

[117] I. Holmes. Accelerated probabilistic inference of RNA structure evolution. *BMC Bioinformatics*, 6:73, 2005.

[118] I. Holmes. Using evolutionary expectation maximization to estimate indel rates. *Bioinformatics.*, 21:2294–2300, 2005.

[119] I. Holmes and W. J. Bruno. Evolutionary HMMs: a bayesian approach to multiple alignment. *Bioinformatics.*, 17:803–820, 2001.

[120] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, Massachusetts, 1979.

[121] Z. Huang, Y. Wu, J. Robertson, L. Feng, R. Malmberg, and L. Cai. Fast and accurate search for non-coding RNA pseudoknot structures in genomes. *Bioinformatics*, 24: 2281–2287, 2008.

[122] P. Hugenholtz. Exploring prokaryotic diversity in the genomic era. *Genome Biol.*, 3: 1–8, 2002.

[123] P. Hugenholtz, B. M. Goebel, and N. R. Pace. Impact of culture-independent studies on the emerging phylogenetic view of bacterial diversity. *J Bacteriol.*, 180:4765–4774, 1998.

[124] R. W. Hyman, M. Fukushima, L. Diamond, J. Kumm, L. C. Giudice, and R. W. Davis. Microbes on the human vaginal epithelium. *Proc Natl Acad Sci U S A.*, 102: 7952–7957, 2005.

[125] B. D. James, G. J. Olsen, J. S. Liu, and N. R. Pace. The secondary structure of ribonuclease P RNA, the catalytic element of a ribonucleoprotein enzyme. *Cell.*, 52: 19–26, 1988.

[126] E. T. Jaynes. *Probability Theory: The Logic of Science.* Available from http://bayes.wustl.edu., 1998.

[127] S. Johnson. *Remote Protein Homology Detection Using Hidden Markov Models*. PhD thesis, Washington University School of Medicine, 2006.

[128] F. Jossinet, T. E. Ludwig, and E. Westhof. RNA structure: Bioinformatic analysis. *Curr. Opin. Microbiol.*, 10:279–285, 2007.

[129] S. Karlin and S. F. Altschul. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proc. Natl. Acad. Sci. USA*, 87:2264–2268, 1990.

[130] S. Karlin and S. F. Altschul. Applications and statistics for multiple high-scoring segments in molecular sequences. *Proc. Natl. Acad. Sci. USA*, 90:5873–5877, 1993.

[131] K. Karplus. Evaluating regularizers for estimating distributions of amino acids. In C. Rawlings, D. Clark, R. Altman, L. Hunter, T. Lengauer, and S. Wodak, editors, *Proceedings of the Third International Conference on Intelligent Systems in Molecular Biology*, pages 188–196, Menlo Park, CA, 1995. AAAI Press.

[132] K. Karplus, K. Sjolander, C. Barrett, M. Cline, D. Haussler, R. Hughey, L. Holm, and C. Sander. Predicting protein structure using hidden Markov models. *Proteins*, 1 (Suppl.):134–139, 1997.

[133] K. Karplus, C. Barrett, and R. Hughey. Hidden Markov models for detecting remote protein homologies. *Bioinformatics*, 14:846–856, 1998.

[134] K. Karplus, R. Karchin, G. Shackelford, and R. Hughey. Calibrating E-values for hidden Markov models using reverse-sequence null models. *Bioinformatics*, 21:4107–4115, 2005.

[135] T. Kasami. An efficient recognition and syntax algorithm for context-free algorithms. Technical Report AFCRL-65-758, Air Force Cambridge Research Lab, Bedford, Mass., 1965.

[136] M. D. Kazanov, A. G. Vitreschak, and M. S. Gelfand. Abundance and functional diversity of riboswitches in microbial communities. *BMC Genomics*, 8:347, 2007.

[137] B. S. Kim, H. M. Oh, H. Kang, and J. Chun. Archaeal diversity in tidal flat sediment as revealed by 16S rDNA analysis. *J Microbiol.*, 43:144–151, 2005.

[138] R. J. Klein and S. R. Eddy. RSEARCH: finding homologs of single structured RNA sequences. *BMC Bioinformatics*, 4:44, 2003.

[139] B. Knudsen and J. Hein. Pfold: RNA secondary structure prediction using stochastic context-free grammars. *Nucl. Acids Res.*, 31:3423–3428, 2003.

[140] B. Knudsen and J. Hein. RNA secondary structure prediction using stochastic context-free grammars and evolutionary history. *Bioinformatics*, 15:446–454, 1999.

[141] D. L. Kolbe and S. R. Eddy. Local RNA structure alignment with incomplete sequence. *Bioinformatics*, 25:1236–1243, 2009.

[142] A. Krogh, M. Brown, I. S. Mian, K. Sjölander, and D. Haussler. Hidden Markov models in computational biology: Applications to protein modeling. *J. Mol. Biol.*, 235:1501–1531, 1994.

[143] K. Kruger, P. J. Grabowski, A. J. Zaug, J. Sands, D. E. Gottschling, and T. R. Cech. Self-splicing RNA: Autoexcision and autocyclization of the ribosomal RNA intervening sequence of *Tetrahymena*. *Cell*, 31:147–157, 1982.

[144] E. C. Lai, P. Tomancak, R. W. Williams, and G. M. Rubin. Computational identification of *Drosophila* microRNA genes. *Genome Biol.*, 4:R42, 2003.

[145] D. J. Lane. 16S/23S rRNA sequencing. In E. Stackebrandt and M. Goodfellow, editors, *Nucleic acid techniques in bacterial systematics*, pages 115–175. John Wiley and Sons, New York, 1991.

[146] Niels Larsen and Christian Zwieb. The signal recognition particle database (SRPDB). *Nucl. Acids Res.*, 21:3019–3020, 1993.

[147] D. Laslett and B. Canback. ARAGORN, a program to detect tRNA genes and tmRNA genes in nucleotide sequences. *Nucl. Acids Res.*, 32:11–16, 2004.

[148] C. Lee, C. Grasso, and M. F. Sharlow. Multiple sequence alignment using partial order graphs. *Bioinformatics.*, 18:452–464, 2002.

[149] R. Lee, R. Feinbaum, and V. Ambros. A short history of a short RNA. *Cell*, S116: S89–S92, 2004.

[150] A. Lescoute, N. B. Leontis, C. Massire, and E. Westhof. Recurrent structural RNA motifs, isostericity matrices and sequence alignments. *Nucleic Acids Res.*, 33:2395–2409, 2005.

[151] I. Letunic, T. Doerks, and P. Bork. SMART 6: Recent updates and new developments. *Nucl. Acids Res.*, 37:D229–D232, 2009.

[152] A. E. Lew, K. R. Gale, C. M. Minchin, V. Shkap, and D. T. de Wall. Phylogenetic analysis of the erythrocytic anaplasma species based on 16S rDNA and GroEL (HSP60) sequences of a. marginale, a. centrale, and a. ovis and the specific detection of a. centrale vaccine strain. *Vet Microbiol.*, 92:145–160, 2003.

[153] R. Lewin. Surprising discovery with a small RNA. *Science*, 218:777–778, 1982.

[154] R. E. Ley, F. Backhed, P. Turnbaugh, C. A. Lozupone, R. D. Knight, and J. I. Gordon. Obesity alters gut microbial ecology. *Proc Natl Acad Sci U S A.*, 102:11070–11075, 2005.

[155] R. E. Ley, J. K. Harris, J. Wilcox, J. R. Spear, S. R. Miller, B. M. Bebout, J. A. Maresca, D. A. Bryant, M. L. Sogin, and N. R. Pace. Unexpected diversity and complexity of the guerrero negro hypersaline microbial mat. *Appl Environ Microbiol.*, 72:3685–3695, 2006.

[156] R. E. Ley, P. J. Turnbaugh, S. Klein, and J. I. Gordon. Microbial ecology: Human gut microbes associated with obesity. *Nature.*, 444:1022–1023, 2006.

[157] L. P. Lim, M. E. Glasner, S. Yekta, C. B. Burge, and D. P. Bartel. Vertebrate microRNA genes. *Science.*, 299:1540, 2003.

[158] E. Lindahl and A. Elofsson. Identification of related proteins on family, superfamily and fold level. *J Mol Biol.*, 295:613–625, 2000.

[159] K. Liolios, K. Mavromatis, N. Tavernarakis, and N. C. Kyrpides. The genomes on line database (GOLD) in 2007: Status of genomic and metagenomic projects and their associated metadata. *Nucleic Acids Res.*, 36:D475–D479, 2008.

[160] Z. Lippman and R. Martienssen. The role of RNA interference in heterochromatic silencing. *Nature.*, 431:364–370, 2004.

[161] J. Liu, J. T. Wang, J. Hu, and B. Tian. A method for aligning RNA secondary structures and its application to RNA motif detection. *BMC Bioinformatics*, 6:89, 2005.

[162] T. M. Lowe and S. R. Eddy. tRNAscan-SE: A program for improved detection of transfer RNA genes in genomic sequence. *Nucl. Acids Res.*, 25:955–964, 1997.

[163] T. M. Lowe and S. R. Eddy. A computational screen for methylation guide snoRNAs in yeast. *Science*, 283:1168–1171, 1999.

[164] W. Ludwig, O. Strunk, R. Westram, L. Richter, H. Meier, , A. Buchner, T. Lai, S. Steppi, G. Jobb, W. Forster, I. Brettske, S. Gerber, A. W. Ginhart, O. Gross, S. Grumann, S. Hermann, R. Jost, A. Konig, T. Liss, R. Lussmann, M. May, B. Nonhoff, B. Reichel, R. Strehlow, A. Stamatakis, N. Stuckmann, A. Vilbig, M. Lenke, T. Ludwig, A. Bode, and K. H. Schleifer. ARB: a software environment for sequence data. *Nucleic Acids Res.*, 32:1363–1371, 2004.

[165] A. Machado-Lima, H. A. del Portillo, and A. M. Durham. Computational methods in noncoding RNA research. *J. Math. Biol.*, 56:15–49, 2008.

[166] D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press, Cambridge, UK, 2003. ISBN 0521642981.

[167] T. J. Macke, D. J. Ecker, R. R. Gutell, D. Gautheret, D. A. Case, and R. Sampath. RNAMotif, an RNA secondary structure definition and search algorithm. *NAR*, 29: 4724–4735, 2001.

[168] B. L. Maidak, N. Larsen, M. J. McCaughey, R. Overbeek, G. J. Olsen, K. Fogel, J. Blandy, and C. R. Woese. The ribosomal database project. *Nucleic Acids Res.*, 22: 3485–3487, 1994.

[169] B. L. Maidak, G. J. Olsen, N. Larsen, R. Overbeek, M. J. McCaughey, and C. R. Woese. The ribosomal database project (RDP). *Nucleic Acids Res.*, 24:82–85, 1996.

[170] B. L. Maidak, G. J. Olsen, N. Larsen, R. Overbeek, M. J. McCaughey, and C. R. Woese. The RDP (ribosomal database project). *Nucleic Acids Res.*, 25:109–111, 1997.

[171] B. L. Maidak, J. R. Cole, Jr C. T. Parker, G. M. Garrity, N. Larsen, B. Li, T. G. Lilburn, M. J. McCaughey, G. J. Olsen, R. Overbeek, S. Pramanik, T. M. Schmidt, J. M. Tiedje, and C. R. Woese. A new version of the RDP (Ribosomal Database Project). *Nucl. Acids Res.*, 27:171–173, 1999.

[172] B. L. Maidak, J. R. Cole, T. G. Lilburn, , P. R. Saxman, J. M. Stredwick, G. M. Garrity, B. Li, G. J. Olsen, S. Pramanik, T. M. Schmidt, and J. M. Tiedje. The RDP (ribosomal database project) continues. *Nucleic Acids Res.*, 28:173–174, 2000.

[173] B. L. Maidak, J. R. Cole, T. G. Lilburn, , P. R. Saxman, R. J. Farris, G. M. Garrity, G. J. Olsen, T. M. Schmidt, and J. M. Tiedje. The RDP-II (ribosomal database project). *Nucleic Acids Res.*, 29:173–174, 2001.

[174] N. Majdalani, C. Cunning, D. Sledjeski, T. Elliott, and S. Gottesman. DsrA RNA regulates translation of RpoS message by an anti-antisense mechanism, independent

of its action as an antisilencer of transcription. *Proc Natl Acad Sci U S A.*, 95: 12462–12467, 1998.

[175] J. Mallatt and C. J. Winchell. Testing the new animal phylogeny: First use of combined large-subunit and small-subunit rRNA gene sequences to classify the protostomes. *Mol Biol Evol.*, 19:289–301, 2002.

[176] M. Mandal and R. R. Breaker. Gene regulation by riboswitches. *Nat Rev Mol Cell Biol.*, 5:451–463, 2004.

[177] D. H. Mathews and D. H. Turner. Dynalign: an algorithm for finding the secondary structure common to two RNA sequences. *J. Mol. Biol.*, 317:191–203, 2002.

[178] D. H. Mathews, J. Sabina, M. Zuker, and D. H. Turner. Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *J Mol Biol.*, 288:911–940, 1999.

[179] G. Meister and T. Tuschl. Mechanisms of gene silencing by double-stranded RNA. *Nature.*, 431:343–349, 2004.

[180] I. M. Meyer. A practical guide to the art of RNA gene prediction. *Brief. Bioinform.*, 8:396–414, 2007.

[181] F. Michel and E. Westhof. Modelling of the three-dimensional architecture of group I catalytic introns based on comparative sequence analysis. *J. Mol. Biol.*, 216:585–610, 1990.

[182] A. A. Michels, V. T. Nguyen, A. Fraldi, V. Labas, M. Edwards, F. Bonnet, L. Lania, and O. Bensaude. MAQ1 and 7SK RNA interact with CDK9/cyclin t complexes in a transcription-dependent manner. *Mol Cell Biol.*, 23:4859–4869, 2003.

[183] R. M. Morris, M. S. Rapp, S. A. Connon, K. L. Vergin, W. A. Siebold, C. A. Carlson, and S. J. Giovannoni. SAR11 clade dominates ocean surface bacterioplankton communities. *Nature.*, 420:806–810, 2002.

[184] K. B. Mullis and F. A. Faloona. Specific synthesis of DNA in vitro via a polymerase-catalyzed chain reaction. *Methods Enzymol.*, 155:335–350, 1987.

[185] M. Muramatsu, J. L. Hodnett, and H. Busch. Base composition of fractions of nuclear and nucleolar ribonucleic acid obtained by sedimentation and chromatography. *J Biol Chem*, 241:1544–1550, 1966.

[186] Eugene W. Myers and Webb Miller. Optimal alignments in linear space. *Comput. Applic. Biosci.*, 4(1):11–17, 1988.

[187] A. Nahvi, N. Sudarsan, M. S. Ebert, X. Zou, K. L. Brown, and R. R. Breaker. Genetic control by a metabolite binding mRNA. *Chem Biol.*, 9:1043, 2002.

[188] I. Nasidze, D. Quinque, J. Li, M. Li, K. Tang, and M. Stoneking. Comparative analysis of human saliva microbiome diversity by barcoded pyrosequencing and cloning approaches. *Anal Biochem.*, 391:64–68, 2009.

[189] E. P. Nawrocki and S. R. Eddy. Query-dependent banding (QDB) for faster RNA similarity searches. *PLoS Comput. Biol.*, 3:e56, 2007.

[190] E. P. Nawrocki, D. L. Kolbe, and S. R. Eddy. Infernal 1.0: Inference of RNA alignments. *Bioinformatics*, 25:1335–1337, 2009.

[191] E. P. Nawrocki, D. L. Kolbe, and S. R. Eddy. Infernal - inference of RNA secondary structure alignments. [http://infernal.janelia.org/], 2009.

[192] E. P. Nawrocki, D. L. Kolbe, and S. R. Eddy. The Infernal user's guide. [http://infernal.janelia.org/], 2009.

[193] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48:443–453, 1970.

[194] J. Neyman and E. S. Pearson. On the problem of the most efficient tests of statistical hypotheses. *Phil. Trans. Royal Soc. London A*, 231:289–337, 1933.

[195] P. Nissen, J. Hansen, N. Ban, P. B. Moore, and T. A. Steitz. The structural basis of ribosome activity in peptide bond synthesis. *Science.*, 289:920–930, 2000.

[196] Harry F. Noller and Carl R. Woese. Secondary structure of 16S ribosomal RNA. *Science*, 212:403–411, 1981.

[197] G. J. Olsen, N. Larsen, and C. R. Woese. The ribosomal RNA database project. *Nucleic Acids Res.*, 19:2017–2021, 1991.

[198] G. J. Olsen, R. Overbeek, N. Larsen, T. L. Marsh, M. J. McCaughey, M. A. Maciukenas, W. M. Kuan, T. J. Macke, Y. Xing, and C. R. Woese. The ribosomal database project. *Nucleic Acids Res.*, 20:2199–2200, 1992.

[199] A. D. Omer, T. M. Lowe, A. G. Russell, H. Ebhardt, S. R. Eddy, and P. P. Dennis. Homologs of small nucleolar RNAs in Archaea. *Science*, 288:517–522, 2000.

[200] N. R. Pace. A molecular view of microbial diversity and the biosphere. *Science.*, 276: 734–740, 1997.

[201] N. R. Pace, D. A. Stahl, D. J. Lane, and G. J. Olsen. Analyzing natural microbial populations by rRNA sequences. *ASM News*, 51:4–12, 1985.

[202] N. R. Pace, B. C. Thomas, and C. R. Woese. Probing RNA structure, function and history by comparative analysis. In R. F. Gesteland and J. F. Atkins, editors, *The RNA World*, pages 113–142. Cold Spring Harbor Laboratory Press, New York, 1993.

[203] J. Park, K. Karplus, C. Barrett, R. Hughey, D. Haussler, T. Hubbard, and C. Chothia. Sequence comparisons using multiple sequences detect three times as many remote homologues as pairwise methods. *J. Mol. Biol.*, 284:1201–1210, 1998.

[204] Angelo Pavesi, Franco Conterlo, Angelo Bolchi, Giorgio Dieci, and Simone Ottonello. Identification of new eukaryotic tRNA genes in genomic DNA databases by a multistep weight matrix analysis of transcriptional control regions. *Nucl. Acids Res.*, 22:1247–1256, 1994.

[205] W. R. Pearson. Searching protein sequence libraries: Comparison of the sensitivity and selectivity of the smith-waterman and FASTA algorithms. *Genomics.*, 11:635–650, 1991.

[206] W. R. Pearson. Comparison of methods for searching protein sequence databases. *Protein Sci.*, 4:1145–1160, 1995.

[207] W. R. Pearson. Effective protein sequence comparison. *Meth. Enzymol.*, 266:227–258, 1996.

[208] W. R. Pearson and D. J. Lipman. Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA*, 85:2444–2448, 1988.

[209] J. S. Pedersen, G. Bejerano, A. Siepel, K. Rosenbloom, K. Lindblad-Toh, E. S. Lander, J. Kent, W. Miller, and D. Haussler. Identification and classification of conserved RNA secondary structures in the human genome. *PLoS Comput. Biol.*, 2:e33, 2006.

[210] Z. Pei, E. J. Bini, L. Yang, M. Zhou, F. Francois, and M. J. Blaser. Bacterial biota in the human distal esophagus. *Proc Natl Acad Sci U S A.*, 101:4250–4255, 2004.

[211] C. Pichon and B. Felden. Small RNA gene identification and mRNA target predictions in bacteria. *Bioinformatics*, 24:2807–2813, 2008.

[212] T. Powers and H. F. Noller. A functional pseudoknot in 16S ribosomal RNA. *EMBO J.*, 10:2203–2214, 1991.

[213] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, 1993. ISBN 0521431085.

[214] E. Pruesse, C. Quast, K. Knittel, B. M. Fuchs, W. Ludwig, J. Peplies, and F. Ö. Glockner. SILVA: a comprehensive online resource for quality checked and aligned ribosomal RNA sequence data compatible with ARB. *Nucleic Acids Res.*, 35:7188–7196, 2007.

[215] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, 77:257–286, 1989.

[216] M. S. Rappe and S. J. Giovannoni. The uncultured microbial majority. *Annu Rev Microbiol.*, 57:369–394, 2003.

[217] J. Reeder, J. Reeder, and R. Giegerich. Locomotif: From graphical motif description to RNA motif search. *Bioinformatics.*, 23:i392–i400, 2007.

[218] M. Regalia, M. A. Rosenblad, and T. Samuelsson. Prediction of signal recognition particle RNA genes. *Nucl. Acids Res.*, 30:3368–3377, 2002.

[219] E. Rivas. Evolutionary models for insertions and deletions in a probabilistic modeling framework. *BMC Bioinformatics.*, 6:63, 2005.

[220] E. Rivas and S. R. Eddy. Noncoding RNA gene detection using comparative sequence analysis. *BMC Bioinformatics*, 2:8, 2001.

[221] E. Rivas and S. R. Eddy. Probabilistic phylogenetic inference with insertions and deletions. *PLoS Comput. Biol.*, 4:e1000172, 2008.

[222] E. Rivas and S. R. Eddy. A dynamic programming algorithm for RNA structure prediction including pseudoknots. *J. Mol. Biol.*, 285:2053–2068, 1999.

[223] D. A. Rodionov, A. G. Vitreschak, A. A. Mironov, and M. S. Gelfand. Regulation of lysine biosynthesis and transport genes in bacteria: yet another RNA riboswitch? *Nucleic Acids Res.*, 31:6748–6757, 2003.

[224] Y. Sakakibara, M. Brown, R. Hughey, I. S. Mian, K. Sjölander, R. C. Underwood, and D. Haussler. The application of stochastic context-free grammars to folding, aligning and modeling homologous RNA sequences. unpublished manuscript, 1994.

[225] Y. Sakakibara, M. Brown, R. Hughey, I. S. Mian, K. Sjölander, R. C. Underwood, and D. Haussler. Stochastic context-free grammars for tRNA modeling. *Nucl. Acids Res.*, 22:5112–5120, 1994.

[226] Y. Sakakibara, M. Brown, R. C. Underwood, I. S. Mian, and D. Haussler. Stochastic context-free grammars for modeling RNA. In Lawrence Hunter, editor, *Proceedings of the Twenty-Seventh Annual Hawaii International Conference on System Sciences: Biotechnology Computing*, volume V, pages 284–293, Los Alamitos, CA, 1994. IEEE Computer Society Press.

[227] F. Sanger and A. R. Coulson. A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase. *J Mol Biol.*, 94:441–448, 1975.

[228] F. Sanger, G. G. Brownlee, and B. G. Barrell. A two-dimensional fractionation procedure for radioactive nucleotides. *J Mol Biol.*, 13:373–398, 1965.

[229] F. Sanger, S. Nicklen, and A. R. Coulson. DNA sequencing with chain-terminating inhibitors. *Proc Natl Acad Sci U S A.*, 74:5463–5467, 1977.

[230] D. Sankoff. Simultaneous solution of the RNA folding, alignment, and protosequence problems. *SIAM J. Appl. Math.*, 45:810–825, 1985.

[231] P. Schattner, S. Barberan-Soler, and T. M. Lowe. A computational screen for mammalian pseudouridylation guide H/ACA RNAs. *RNA*, 12:15–25, 2006.

[232] P. Sellers. On the theory and computation of evolutionary distances. *SIAM J. Appl. Math.*, 26:787–793, 1974.

[233] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423,623–656, 1948.

[234] K. Sjölander, K. Karplus, M. Brown, R. Hughey, A. Krogh, I. S. Mian, and D. Haussler. Dirichlet mixtures: A method for improving detection of weak but significant protein sequence homology. *Comput. Applic. Biosci.*, 12:327–345, 1996.

[235] S. Smit, K. Rother, J. Heringa, and R. Knight. From knotted to nested RNA structures: a variety of computational methods for pseudoknot removal. *RNA*, 14:410–416, 2008.

[236] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *J. Mol. Biol.*, 147:195–197, 1981.

[237] E. L. L. Sonnhammer, S. R. Eddy, E. Birney, A. Bateman, and R. Durbin. Pfam: Multiple sequence alignments and HMM-profiles of protein domains. *Nucl. Acids Res.*, 26:320–322, 1998.

[238] G. Srinivasan, C. M. James, and J. A. Krzycki. Pyrrolysine encoded by UAG in archaea: Charging of a UAG-decoding specialized tRNA. *Science*, 296:1459–1462, 2002.

[239] J. T. Staley and A. Konopka. Measurement of in situ activities of nonphotosynthetic microorganisms in aquatic and terrestrial habitats. *Annu Rev Microbiol.*, 39:321–346, 1985.

[240] G. Stefani and F. J. Slack. Small non-coding RNAs in animal development. *Nat Rev Mol Cell Biol.*, 9:219–230, 2008.

[241] T. Stoeck and S. Epstein. Novel eukaryotic lineages inferred from small-subunit rRNA analyses of oxygen-depleted marine environments. *Appl Environ Microbiol.*, 69:2657–2663, 2003.

[242] G. Storz. An expanding universe of noncoding RNAs. *Science*, 296:1260–1263, 2002.

[243] G. Storz, S. Altuvia, and K. M. Wassarman. An abundance of RNA regulators. *Annu. Rev. Biochem.*, 74:199–217, 2005.

[244] M. A. Suchard and B. D. Redelings. BAli-phy: Simultaneous bayesian inference of alignment and phylogeny. *Bioinformatics.*, 22:2047–2048, 2006.

[245] N. Sudarsan, J. E. Barrick, and R. R. Breaker. Metabolite-binding RNA domains are present in the genes of eukaryotes. *RNA*, 9:644–647, 2003.

[246] N. Sudarsan, J. K. Wickiser, S. Nakamura, M. S. Ebert, and R. R. Breaker. An mRNA structure in bacteria that controls gene expression by binding lysine. *Genes Dev.*, 17:2688–2697, 2003.

[247] Y. Sun and J. Buhler. Designing secondary structure profiles for fast ncRNA identification. In *Proc. Computational Systems Bioinformatics*, volume 7, pages 145–156, 2008.

[248] M. Szymanski, M. Z. Barciszewska, M. Zywicki, and J. Barciszewski. Noncoding RNA transcripts. *J. Appl. Genet.*, 44:1–19, 2003.

[249] J. D. Thompson, D. G. Higgins, and T. J. Gibson. CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties, and weight matrix choice. *Nucl. Acids Res.*, 22:4673–4680, 1994.

[250] S. G. Tringe and P. Hugenholtz. A renaissance for the pioneering 16S rRNA gene. *Curr Opin Microbiol.*, 11:442–446, 2008.

[251] S. G. Tringe, C. von Mering, A. Kobayashi, A. A. Salamov, K. Chen, H. W. Chang, M. Podar, J. M. Short, E. J. Mathur, J. C. Detter, P. Bork, P. Hugenholtz, and E. M. Rubin. Comparative metagenomics of microbial communities. *Science*, 308:554–557, 2005.

[252] B. J. Tucker and R. R. Breaker. Riboswitches as versatile gene control elements. *Curr. Opin. Struct. Biol.*, 15:342–348, 2005.

[253] P. J. Turnbaugh, R. E. Ley, M. Hamady, C. M. Fraser-Liggett, R. Knight, and J. I. Gordon. The human microbiome project. *Nature.*, 449:804–810, 2007.

[254] G. W. Tyson, J. Chapman, P. Hugenholtz, E. E. Allen, R. J. Ram, P. M. Richardson, V. V. Solovyev, E. M. Rubin, D. S. Rokhsar, and J. F. Banfield. Community structure and metabolism through reconstruction of microbial genomes from the environment. *Nature.*, 428:37–43, 2004.

[255] G. Varani and I. Tinoco Jr. RNA structure and NMR spectroscopy. *Q Rev Biophys.*, 24:479–532, 1991.

[256] J. C. Venter, K. Remington, J. F. Heidelberg, A. L. Halpern, D. Rusch, J. A. Eisen, D. Wu, I. Paulsen, K. E. Nelson, W. Nelson, D. E. Fouts, S. Levy, A. H. Knap, M. W. Lomas, K. Nealson, O. White, J. Peterson, J. Hoffman, R. Parsons, H. Baden-Tillson, C. Pfannkoch, Y. H. Rogers, and H. O. Smith. Environmental genome shotgun sequencing of the Sargasso Sea. *Science*, 304:66–74, 2004.

[257] J. Vogel and C. M. Sharma. How to find small non-coding RNAs in bacteria. *Biol. Chem.*, 386:1219–1238, 2005.

[258] J. J. Walker and N. R. Pace. Phylogenetic composition of rocky mountain endolithic microbial ecosystems. *Appl Environ Microbiol.*, 73:3497–3504, 2007.

[259] J. J. Walker, J. R. Spear, and N. R. Pace. Geobiology of a microbial endolithic community in the yellowstone geothermal environment. *Nature.*, 434:1011–1014, 2005.

[260] D. M. Ward, R. Weller, and M. M. Bateson. 16S rRNA sequences reveal numerous uncultured microorganisms in a natural community. *Nature.*, 345:63–65, 1990.

[261] K. M. Wassarman and G. Storz. 6S RNA regulates *E. coli* RNA polymerase activity. *Cell*, 101:613–623, 2000.

[262] H. C. Watson and J. C. Kendrew. The amino-acid sequence of sperm whale myoglobin. comparison between the amino-acid sequences of sperm whale myoglobin and of human hemoglobin. *Nature.*, 190:670–672, 1961.

[263] Z. Weinberg and W. L. Ruzzo. Exploiting conserved structure for faster annotation of non-coding RNAs without loss of accuracy. *Bioinformatics*, 20 Suppl. 1:I334–I341, 2004.

[264] Z. Weinberg and W. L. Ruzzo. Faster genome annotation of non-coding RNA families without loss of accuracy. *RECOMB '04*, pages 243–251, 2004.

[265] Z. Weinberg and W. L. Ruzzo. Sequence-based heuristics for faster annotation of non-coding RNA families. *Bioinformatics*, 22:35–39, 2006.

[266] Zasha Weinberg. *Accurate annotation of non-coding RNAs in practical time.* PhD thesis, University of Washington, 2005.

[267] R. Weller and D. M. Ward. Selective recovery of 16S rRNA sequences from natural microbial communities in the form of cDNA. *Appl Environ Microbiol.*, 55:1818–1822, 1989.

[268] R. Weller, M. M. Bateson, B. K. Heimbuch, E. D. Kopczynski, and D. M. Ward. Uncultivated cyanobacteria, chloroflexus-like inhabitants, and spirochete-like inhabitants of a hot spring microbial mat. *Appl Environ Microbiol.*, 58:3964–3969, 1992.

[269] P. J. Wilderman, N. A. Sowa, D. J. FitzGerald, P. C. FitzGerald, S. Gottesman, U. A. Ochsner, and M. L. Vasil. Identification of tandem duplicate regulatory small RNAs in pseudomonas aeruginosa involved in iron homeostasis. *Proc Natl Acad Sci U S A.*, 101:9792–9797, 2004.

[270] S. Will, K. Reiche, I. L. Hofacker, P. F. Stadler, and R. Backofen. Inferring noncoding RNA families and classes by means of genome-scale structure-based clustering. *PLoS Comput. Biol.*, 3:e65, 2007.

[271] B. T. Wimberly, D. E. Brodersen, , R. J. Morgan-Warren, A. P. Carter, C. Vonrhein, T. Hartsch, and V. Ramakrishnan. Structure of the 30s ribosomal subunit. *Nature.*, 407:327–339, 2000.

[272] W. C. Winkler. Riboswitches and the role of noncoding RNAs in bacterial metabolic control. *Curr. Opin. Chem. Biol.*, 9:594–602, 2005.

[273] C. R. Woese. The archaeal concept and the world it lives in: A retrospective. *Photosynth Res.*, 80:361–372, 2004.

[274] C. R. Woese. *The Genetic Code: The Molecular Basis for Genetic Expression.* Harpers Collins Publishers Inc., New York, 1968.

[275] C. R. Woese. Bacterial evolution. *Microbial Rev.*, 51:221–71, 1987.

[276] C. R. Woese and G. E. Fox. Phylogenetic structure of the prokaryotic domain: the primary kingdoms. *Proc Natl Acad Sci U S A.*, 74:5088–5090, 1977.

[277] C. R. Woese and R. R. Gutell. Evidence for several higher order structural elements in ribosomal RNA. *Proc Natl Acad Sci U S A.*, 86:3119–3122, 1989.

[278] C. R. Woese, L. J. Magrum, R. Gupta, R. B. Siegel, D. A. Stahl, J. Kop, N. Crawford, J. Brosius, R. Gutell, J. J. Hogan, and H. F. Noller. Secondary structure model for bacterial 16S ribosomal RNA: phylogenetic, enzymatic and chemical evidence. *Nucleic Acids Res.*, 10:2275–93, 1980.

[279] C. R. Woese, J. Maniloff, and L. B. Zablan. Phylogenetic analysis of the mycoplasmas. *Proc Natl Acad Sci U S A.*, 77:494–498, 1980.

[280] C. R. Woese, R. Gutell, R. Gupta, and H. F. Noller. Detailed analysis of the higher-order structure of 16S-like ribosomal ribonucleic acids. *Microbiol Rev.*, 47:621–669, 1983.

[281] C. R. Woese, E. Stackebrandt, W. G. Weisburg, B. J. Paster, M. T. Madigan, V. J. Fowler, C. M. Hahn, P. Blanz, R. Gupta, K. H. Nealson, and G. E. Fox. The phylogeny of purple bacteria: the alpha subdivision. *Syst Appl Microbiol.*, 5:315–326, 1984.

[282] P. C. Woo, S. K. Lau, J. L. Teng, H. Tse, and K. Y. Yuen. Then and now: use of 16S rDNA gene sequencing for bacterial identification and discovery of novel bacteria in clinical microbiology laboratories. *Clin Microbiol Infect.*, 14:908–934, 2008.

[283] J. Wuyts, P. De Rijk, Y. Van de Peer, T. Winkelmans, and R. De Wachter. The European large subunit ribosomal RNA database. *Nucl. Acids Res.*, 29:175–177, 2001.

[284] J. Wuyts, Y. Van de Peer, T. Winkelmans, and R. De Wachter. The European database on small subunit ribosomal RNA. *Nucl. Acids Res.*, 30:183–185, 2002.

[285] D. H. Younger. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10:189–208, 1967.

[286] M. M. Yusupov, G. Z. Yusupova, A. Baucom, K. Lieberman, T. N. Earnest, J. H. Cate, and H. F. Noller. Crystal structure of the ribosome at 5.5 a resolution. *Science.*, 292:883–896, 2001.

[287] S. Zhang, B. Haas, E. Eskin, and V. Bafna. Searching genomes for noncoding RNA using FastR. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, 2:366–379, 2005.

[288] S. Zhang, I. Borovok, Y. Aharonowitz, R. Sharan, and V. Bafna. A sequence-based filtering method for ncRNA identification and its application to searching for riboswitch elements. *Bioinformatics*, 22:e557–e565, 2006.

Vita

Eric Paul Nawrocki

Date of Birth	July 19, 1980
Place of Birth	Columbia, MD USA
Degrees	B.S. Computer Science, May 2003 B.S. Biology, May 2003 Ph.D. Computational Biology, December 2009
Publications	Nawrocki EP, Kolbe DL, Eddy SR (2009). Infernal 1.0: inference of RNA alignments. *Bioinformatics* 25(10):1335-1337. Gardner PP, Daub J, Tate JG, Nawrocki EP, Kolbe DL, Lindgreen S, Wilkinson AC, Finn RD, Griffiths-Jones S, Eddy SR, Bateman A (2009). Rfam: updates to the RNA families database. *Nucleic Acids Res.* 37:D136-D140. Nawrocki EP, Eddy SR (2007) Query-dependent banding (QDB) for faster RNA similarity searches. *PLoS Comput Biol* 3(3):e56.

August 2009