EXTRACTING SIGNAL FROM NOISE IN BIOLOGICAL DATA: EVALUATIONS AND APPLICATIONS OF TEXT MINING AND SEQUENCE COEVOLUTION

by

J. GREGORY CAPORASO

B.S., Computer Science, University of Colorado at Boulder, 2001

B.A., Biochemistry, University of Colorado at Boulder, 2004

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Biochemistry and Molecular Genetics

2009

This thesis for the Doctor of Philosophy degree by

J. Gregory Caporaso

has been approved for the

Department of Biochemistry and Molecular Genetics

by

Mair Churchill, Chair

Lawrence Hunter, Advisor

Jeffrey Kieft

Robin Knight

David Pollock

Date _____

Caporaso, J. Gregory (Ph.D., Biochemistry)

Extracting signal from noise in biological data: Evaluations and applications of text mining and sequence coevolution

Thesis directed by Professor Lawrence Hunter.

As the quantity of biological data continues to expand, it is the role of the computational biologist to develop new methods and tools to efficiently and accurately translate biological data into biological knowledge. Focusing on biomedical literature and biological sequences, this dissertation is about techniques for learning more from biological data.

The early chapters address biomedical text mining, and specifically the problem of automatically compiling data on protein point mutations from biomedical literature. Protein point mutations and substitutions are central in many areas of biomedical research, including human disease, biodiversity, and protein structure/function relationships. Mutation databases exist to centralize known information, but are frequently expensive to compile. An automated approach is presented for developing high-performance text mining systems and is applied to develop MutationFinder, a tool that scans text and extracts descriptions of point mutations into structured formats. Manual and automated approaches for annotating mutations are then compared, resulting in the conclusion that combining automatic and manual annotation tools may be the best approach to develop comprehensive and accurate biomedical databases.

The later chapters focus on identifying pairs of coevolving positions in proteins. Just as macroscopic structures like the bills of hummingbirds and the corolla tubes of flowering plants coevolve, it is expected that interacting positions within and between proteins also coevolve to maintain highly specific interactions. If true, coevolutionary signals should be detectable in multiple sequence alignments and may contain information on intramolecular or intermolecular interactions between amino acid residues. An analysis of coevolution

algorithms leads to the surprising conclusion that algorithms that do not incorporate phylogeny can match the performance of those that do incorporate phylogeny. A coevolution algorithm is then applied to predict interactions between component proteins of the Type VI Secretion System (T6SS), leading to a new model of the T6SS.

Additional contributions of this work include two open-source software projects, MutationFinder and the PyCogent coevolution module. These high-quality software tools allow for the reproduction, application, and expansion of the work presented in this dissertation: text mining for point mutations, and detecting coevolution of biological sequences.

The form and content of this abstract are approved. I recommend its publication.

Approved: Lawrence Hunter

To my parents, Barbara and Jim Caporaso.

ACKNOWLEDGMENTS

My family, friends, and colleagues have been amazingly supportive of my work over the last five years. I would first like to thank my soon-to-be wife, Ingrid Muja, who from the beginning of my graduate school career has been supportive of my goals and provided endless encouragement. I would also like to begin by thanking my parents Barbara and Jim Caporaso, my grandmother Mary Caporaso, and my brother Kevin Caporaso, for their support, encouragement, and confidence. This dissertation would not exist without them.

I would next like to thank my thesis advisor Larry Hunter. Although I did not enter graduate school via Larry's program, the Computational Bioscience Program (CBP), he took a chance on accepting me as a trainee, and thereby provided me with the means and guidance to write a computational biology dissertation. Larry instills a sense of purpose, ethics, and enjoyment in his research and teaching, and I hope to carry that over into my own career. Next, I would like to thank Rob Knight. Rob advised my undergraduate computational biology research projects, and served on my thesis committee. It was on Rob's suggestion that I initially became interested in pursuing graduate school and a research career. Rob has shared his excitement for biological research with me, and has constantly challenged and encouraged me along the way. I would also like to thank the remaining members of my thesis committee, Mair Churchill (my committee chair), Jeff Kieft, and David Pollock. My thesis committee has shaped my research and this dissertation, and by being tough on me, they have made me a better scientist.

I would next like to thank present and past members of the Hunter lab, Kevin Cohen, Bill Baumgartner, Helen Johnson, Anis Karimpour-Fard, Karin Verspoor, Zhiyong Lu, Philip Ogren, Mike Bada, Hannah Tipney, Sonia Leach, and Hyunmin Kim; and present and past members of the Knight lab, Sandra Smit, Jeremy Widmann, Micah Hamady, Cathy Lozupone, Mike Robeson, Daniel McDonald, and Jesse Zaneveld. I have collaborated on

many projects, some big and some small, with these colleagues, and they have been first in line to give feedback on my new ideas and work. Further, they have been my friends. I would additionally like to thank Kathy Thomas, the department administrator for the CBP, and David Farrell, the systems administrator for the CBP, for their endless help with the many technical difficulties along the way.

In the following sections, I acknowledge my collaborators and co-authors on the specific projects presented in this dissertation.

Chapter I: I would like to thank Helen Johnson for proof-reading this chapter, and providing me with useful feedback on the presentation of the material and the content of the chapter.

Chapter II: I would like to acknowledge Bill Baumgartner, David Randolph, Kevin Cohen, and Larry Hunter, my co-authors on the initial presentation of this work. I would like to thank Jim Martin for helpful discussion and guidance in the initial stages of this project. Additionally, I would like to thank Zhiyong Lu for providing GeneRIFs describing protein transport events, and the four anonymous reviewers of the original article for their helpful comments and suggestions during the preparation of this article.

Chapter III: I would like to acknowledge Nita Deshpande, Lynne Fink, Philip Bourne, Kevin Cohen, and Larry Hunter, my co-authors on the initial presentation of this work. I would additionally like to acknowledge Sue Brozowski for evaluating the MutationFinder normalization of PDB mutation fields, Jeffrey Haemer, Kristina Williams, and Bill Baumgartner for proof-reading and helpful discussion, and the four anonymous reviewers of the original article for their insightful feedback.

Chapter IV: I would like to acknowledge Sandra Smit, Brett Easton, Larry Hunter, Gavin Huttley, and Rob Knight, my co-authors on the original presentation of this work. I would also like to thank Karen Meyer-Arendt for work on the Statistical Coupling Analysis implementation, Micah Hamady for work on the Myosin alignment, and Massimo Buvoli

for suggesting the Myosin rod domain as a subject for coevolutionary analysis.

Chapter VI: I would like to thank my collaborators on this project, Amy Dear and Marcelo Sousa, of the Sousa biochemistry lab at CU Boulder; and Rob Knight, the PI on the computational aspects of this project. Amy has provided me with guidance in compiling information and data on the T6SS, created the two figures presented in this chapter, and helped with proof-reading of this chapter. I would additionally like to thank Sandra Smit for her help with proof-reading this chapter.

Appendix A: I would like to thank Bill Baumgartner, David Randolph, Kevin Cohen, and Larry Hunter, my co-authors on the initial presentation of this work. I would also like to thank Sonia Leach, Hyunmin Kim and Jim Martin for helpful comments and discussion during system and corpus development and testing; and the two anonymous reviewers for their useful suggestions during the preparation of the original article, and reviewer 2, in particular, for providing the example mutation-like cell line we mention in *Section 5*: T98G.

Appendix B: I would like to thank Massimo Buvoli for suggesting this project, and helping with my studies of myosin. I would additionally like to thank Micah Hamady and Rob Knight for openly sharing the information in the MyoMapr database before its publication, and for providing assistance in working with the data therein.

TABLE OF CONTENTS

APPENDIX

LIST OF TABLES

TABLE

LIST OF FIGURES

FIGURE

CHAPTER I

INTRODUCTION

Finding a needle in a haystack can be straight-forward if you have the right tool for the job. By using a big magnet (or, more destructively, a match), separating needles and hay is a simple task. This dissertation is about developing, testing, and applying tools to extract important information (needles) from background noise (hay), and specifically working with the deluge of biomedical data that researchers are currently faced with to more efficiently translate biomedical knowledge to advances in human health and our understanding of biological systems.

The two types of biological information that I have focused on over the course of my Ph.D. are biomedical literature and biological sequences. Primary biomedical literature is the central means by which biologists share their findings and introduce new information. The quantity of biomedical literature, measured by the number of citations in Medline, is increasing exponentially [1] making it impossible for researchers to keep up with new developments in their field. High-quality, fully automated techniques for retrieving, summarizing, and populating databases directly from literature are important now and will only become more necessary as the biomedical literature continues to grow. The quantity of biological sequence data is also growing exponentially [2], and new techniques for extracting information from sequences will increase the rate of biological discovery. Determining the structures of macromolecules and identifying protein-protein interactions are two extremely important areas of molecular biology and biochemistry, and are both areas where computational techniques seem able to assist. I expect that information about protein-protein interactions and protein (and RNA) structure can be extracted from the evolutionary information present in multiple sequence alignments. Understanding how to accurately and efficiently extract that information is a central aspect of this dissertation.

In this chapter, I introduce concepts that are central to the subsequent chapters. Many of the topics discussed in this chapter can (and do) fill texts larger than the entirety of this dissertation. For space considerations, I therefore try to briefly introduce topics here and provide references to materials that I have found helpful over the course of my studies. I conclude with a description of what can be found in each of the subsequent chapters of my dissertation.

1.1 A brief introduction to proteins

Much of this dissertation is focused on the analysis of protein sequence data. Proteins are biological macromolecules, composed of a linear chain of amino acid residues linked by peptide bonds. The *sequence* of a protein, also known as its primary structure, refers to the linear order of amino acid residues from the amino- (or N-) terminus to the carboxy- (or C-) terminus of the protein chain. Protein sequences are encoded by DNA; the processes of transcription of DNA to RNA by RNA polymerases and subsequent translation of RNA to protein via ribosomes are central to all known cellular life on Earth.

Twenty amino acids form the set of protein building blocks in most species. Each amino acid is encoded by one or more codons (trinucleotides) in a coding region of DNA, and the mapping of 61 codons to the twenty amino acids and 3 codons to stop-translation signals, is referred to as the genetic code[1]. (My early research, which was published during the course of my Ph.D. studies but which is not included in this dissertation, focused on the evolution of the genetic code [3, 4]).

Proteins are central to all known life on Earth, functioning as enzymes (e.g. polymerases), structural components (e.g. actin), signaling molecules (e.g. insulin), transcription factors (e.g. p53), and receptors (e.g. dopamine receptor), among other roles. Mu-

[1]Note there are variants of the canonical (or human nuclear) genetic code. One example is the vertebrate mitochondrial genetic code, which contains differences in the codons associated with isoleucine, methionine, tryptophan, and stop-translation.

tations in protein coding regions of DNA (introduced in Section 1.1.2) are associated with many genetic diseases, including cancer (e.g. p53 and BRCA1), cardiomyopathies (e.g. myosin and actin), cystic fibrosis (e.g. CFTR), and many, many others. Proteins and protein mutations are therefore of great interest in biology and medicine, and studying protein mutations and substitutions is a powerful way to understand the function and malfunction of proteins.

1.1.1 Protein sequences

As mentioned above, most proteins are composed of twenty amino acid building blocks. When protein sequences are represented computationally, they are typically presented as a string of characters, where each character is a one-letter abbreviation for an amino acid residue. Sequences are presented in order from the N-terminus (corresponding to the 5' end of an mRNA) to the C-terminus (corresponding to the 3' end of an mRNA). For example, the first five positions of the human beta hemoglobin protein sequence (gi:4504349) would be represented as MVHLT, indicating that the first through fifth positions in order are methionine, valine, histidine, leucine, and threonine.

There are several different techniques for sequencing a protein, or determining its ordered amino acid composition. The three methods that are primarily used, as of this writing, are automated Edman degradation (discussed in basic biochemistry texts, including [5]), mass spectroscopy (reviewed in [6, 7]), and prediction from DNA or mRNA sequences (if known). Because these are complex topics which are not central to this dissertation (although the derived data is central), I refer the reader to the cited reviews for in-depth coverage.

1.1.2 Protein mutations

Mutations occur at the DNA level in three commonly occurring variations: point mutations, insertions, and deletions. Point mutations in protein coding genes occur when one

nucleotide is replaced with another nucleotide, and this change can be categorized as silent, missense, or nonsense. Silent mutations occur when the nucleotide substitution results in a change from one codon to a synonymous codon due to the degeneracy of the genetic code (i.e., the many-to-one mapping from codons to amino acids). Missense mutations occur when a codon is changed to a codon for another amino acid. Nonsense mutations occur when an amino acid codon is replaced with a stop-termination codon. Deletion mutations occur when one or more nucleotides are omitted from a DNA sequence. Because amino acids are encoded by non-overlapping groups of three adjacent nucleotides, deletion mutations which occur in multiples of other than three nucleotides result in frame-shift mutations, and are frequently not tolerated. Conversely, insertion mutations occur when one or more nucleotides are inserted into a DNA sequence, and also must occur in multiples of three to avoid frame-shift mutations.

1.1.3 Protein evolution

For evolution to occur, three components must be present: heredity, variation, and selective pressure. The (germ-line) genetic material of an organism is replicated and passed to the subsequent generations, fulfilling the heredity component of evolution. Variation can be introduced into genetic material through mutation, and selection acts on the phenotypic consequences of mutations to determine which will be tolerated or rejected.

Mutation occurs at the DNA level, frequently during DNA replication, and selection typically occurs at the protein level (although any mutations associated with a phenotype, including mutations in non-coding RNAs or regulatory regions of DNA, can be targeted by selection). Variations which make an organism more likely to produce offspring will typically propagate through the population, while those which make an organism less likely to produce offspring will typically disappear[2]. Errors that occur during the transcription

[2]'Typically' is a key word here. Forces other than natural selection, such as genetic drift, are also important in determining which variations will exist in subsequent generations.

of a gene or during the translation of a corresponding protein are not inheritable, and are therefore not considered mutations. This section focuses on DNA mutations in protein-coding genes.

All mutations to protein-coding genes do not have equal effects. Due to the degeneracy of the genetic code, many mutations are synonymous, meaning that they do not result in an amino acid change. For example, the codons AAA and AAG both code for lysine in the canonical genetic code. A mutation from AAA to AAG is therefore a synonymous mutation because both codons translate to lysine. Non-synonymous mutations, or mutations that do result in a different amino acid in the protein sequence, can be either deleterious, neutral, or beneficial to the organism which incurred the mutation. The vast majority are either deleterious or neutral, and are either selected against or tolerated. In the rare cases of beneficial mutations, when the mutation imparts a selective advantage to the organism, that mutation can spread though the population and eventually become dominant. When this happens, the mutation is said to have resulted in a substitution event.

Because each amino acid side chain has different physicochemical properties, and because substitution between a pair of amino acids can sometime require more than one DNA mutation, the rate of substitution between different amino acids is not equal. Substitution matrices have been derived, frequently based on empirical data, to quantify the differential substitution rates between amino acids. The pioneering work in this area was done by Dayhoff and colleagues [8], resulting in the PAM family of substitution matrices.

Gene and protein evolution is a vast area of research. A basic introduction is provided in most biochemistry and molecular biology textbooks (e.g. [9, 10]). Some additional materials that I have found useful in this area, although focused on bioinformatic methods rather than biological concepts, include the works of Joseph Felsenstein compiled in *Inferring Phylogenies* [11], and *Biological Sequence Analysis* [12]. An approachable introduction to the on-going controversies and debates over the processes involved in evolution is presented

in Chapter II of *Sex and Death: An Introduction to the Philosophy of Biology* [13].

1.1.4 Protein structures

Protein function is an emergent property of protein structure, which refers to the conformation adopted by a protein under its normal physiological conditions. Protein structure is determined by the sequence of the protein, any changes made to the amino acids following translation (i.e. post-translational modifications), and the protein's environment. Understanding the structure of a protein is a key aspect of understanding how and why it functions in a certain way, and also holds the key to selectively disrupting a protein's function in rational drug design.

Protein structures are typically viewed hierarchically. Primary structure refers to a protein's sequence. Secondary structure refers to interactions between amino acid residues close in primary structure, and results in familiar elements such as alpha helices and beta sheets. Tertiary structure refers to interactions between amino acid residues distant in primary structure, and is responsible for different protein folds. Finally, quaternary structure refers to interactions between different protein chains, such as the alpha and beta hemoglobin chains of which two of each are required to form a functional hemoglobin protein.

To properly function, proteins fold to their native conformation. The forces that guide protein folding and maintain the folded state of a protein are hydrogen-bonding interactions, hydrophobic interactions, salt-bridge (ionic) interactions, and relatively few covalent interactions. Determining the folded structure of a protein is a complex task. The two dominant methods of determining protein structure are X-ray crystallography and nuclear magnetic resonance (NMR). A good introduction to protein structure, including techniques for determining protein structure can be found in Chapter VI of *Lehninger: Principles of Biochemistry* [14].

1.1.5 Mutant protein structures

Solving the structure of wild-type and mutant variants of a protein is a common technique for understanding the role of a particular amino acid residue in a protein's structure. With both structures in hand, and especially in conjunction with functional data, it becomes possible to deduce how a protein performs its specific function. The Protein Data Bank (PDB) [15, 16], the central repository for protein structures, is populated in large part with mutant protein structures. Mutant proteins and the PDB are central to Chapter III, and mutant protein structures are revisited there.

For an example of a comparison between mutant and wild-type protein structures, see the PDB structure 2QLE and the associated article [17]. In this study, the authors crystallize a Ser205Val GFP variant to understand how the variant is capable of fluorescence when this mutation is thought to disrupt the proton transfer chain required for fluorescence. The crystal structure suggests changes in the conformation of nearby residues, which appear to compensate for the Ser205Val mutation by providing an alternative proton transport pathway.

1.1.6 Protein mutation databases

The importance of protein mutation data in biology and medicine can be illustrated by the many centralized protein mutation databases which exist for different purposes. The Human Genome Variation Society maintains a list of mutation databases[3] which, as of this writing, contains hundreds of independent databases. This list includes many databases which catalogue mutations in human proteins, including the Human Gene Mutation Database (HGMD) [18], Online Mendelian Inheritance in Man (OMIM) [19], and MutDB [20]. These represent examples of species-specific databases. The GPCRDB [21] and NucleaRDB [22] are examples of protein-family-specific databases which compile information, including mu-

[3] http://www.hgvs.org/dblist/dblist.html

tations, on G-protein coupled receptors and nuclear hormone receptors, respectively. The HIV Drug resistance database is a HIV-specific database which includes curated mutations known to be associated with differential sensitivity to HIV drugs [23]. Further illustrating the importance of mutation data, a forth-coming special issue of *BMC Bioinformatics* will focus exclusively on the theme: 'Annotation, Interpretation and Management of Mutations'.

1.1.7 Relevant dissertation sections

Protein point mutations are a unifying aspect of this thesis. The text mining chapters (II and III) focus on the problem of identifying and extracting descriptions of point mutations from biomedical literature. The sequence analysis chapters (IV, V and VI) focus on identifying correlated substitution events from alignments of protein sequences, with the hope that these can provide insight into the function of the positions involved.

1.2 Biomedical text mining

It is frequently noted that biological researchers are unable to keep pace with the rate of publication of biomedical literature pertaining to their field [1, 24–26]. Biomedical text mining techniques are aimed at addressing this issue by automating the extraction of important data from free text (e.g., biomedical journal articles), and using that data in downstream applications such as database annotation. Strong evidence for the necessity of text mining and other automated methods of genome and proteome annotation assistance comes from recent work by Baumgartner *et al.* [27], who found that the current rates of biological database curation will not provide full coverage of even just the currently sequenced genomes in the foreseeable future.

The text mining work presented herein is focused on a specific concept recognition task: extracting descriptions of point mutations from free text into a structured format. This was an attractive problem to address because several groups had already done work in this area, (for example, Horn *et al.* [28] and Rebholz-Schuhmann *et al.* [29], both reviewed in Section

2.1.2), but there was much room for improvement. Additionally, defining the task and test data was relatively straight-forward. Finally, there were no publicly available software tools to identify protein mutations in free text, so the possibility of creating a useful and usable tool existed.

The following sections describe the general task of concept recognition and the specific problem of point mutation recognition addressed in this dissertation. Other works that I have published in biomedical text mining but that are not included in this dissertation, specifically on applying concept recognition in automated document classification and question-answering, are Caporaso *et al.* [30], Caporaso *et al.* [31], and Baumgartner *et al.* [32].

1.2.1 Concept recognition

Concept recognition is the process of associating free text with semantic concepts, and is the subject of much current work in biomedical language processing [32–36]. By annotating free text with concepts, it becomes possible to more accurately search and compute using text.

A familiar example of concept recognition in free text is incorporated in the Google search engine. If a user searches for the term *80305*, the first result returned (as of this writing) is a map of Boulder, Colorado. This is because the search engine has recognized that the text *80305* is an instance of the concept *U.S. zip code*, and infers that the user may be interested in a map of Boulder. Concept recognition is challenging largely because the text alone is often not enough to disambiguate the concept. In this example, *80305* could also refer to the gene identifier for the human protein TRABD, which is also returned as a top Google hit.

As an example in the biomedical domain, a category of concept recognizers are protein name recognizers (e.g., [34, 35]). Given free text, protein name recognizers can be applied to associate mentions of proteins in the text with the concept *protein*. This makes it possible for

downstream applications to request, for example, only documents which contain a mention of a protein.

Two approaches commonly employed to build concept recognition systems are dictionary-based and rule-based approaches. In a dictionary-based approach, the system is given a list of terms that are considered to be instances of the target concept. This could be a list of american zip codes or protein names (constructed from, for example, the set of protein names and synonyms in UniProt [37]). When the system is applied to input text, each term in the text[4] is looked up in the dictionary. If a term matches, it is tagged as being an instance of that concept. A rule-based approach attempts to match text to a rule (or set of rules) provided to the system, frequently in the form of regular expressions. For the zip code example, a rule would match a block of text containing five consecutive digits preceded by a space but with no intervening spaces, and followed either by a space or a dash and four additional digits followed by a space. For protein recognition, a purely rule-based system is probably not reasonable due to the diverse styles of protein names, ranging from English words to combinations of letters, digits, and symbols.

Dictionary-based concept recognition systems have the disadvantage of needing to be updated as frequently as new instances of the concept are introduced into language. The language of biomedical text is constantly expanding, and concept recognition dictionaries for biomedical text therefore require frequent updates. Rule-based systems, on the other hand, frequently require many person-hours of manual rule or regular expression development: a time-consuming, error-prone, and tedious task. Both types of systems are frequently plagued by spurious matches, usually resulting from language ambiguity.

[4]The meaning of a *term* in the text is typically defined on a task-specific basis, but frequently would be a set of white-space delimited characters.

1.2.2 Point mutation recognition

The specific concept recognition task discussed in this dissertation is *point mutation recognition*, defined here as the process of extracting a relationship between three entities from free text: the position where a substitution occurred in a biological sequence (i.e., a nucleic acid or polypeptide sequence), and the wild-type and mutant entities (i.e., bases or residues). I refer to point mutation recognizers as concept recognition systems, and point out that point mutation recognition is similar both to named-entity recognition and to information extraction. Point mutation mentions in abbreviated formats (e.g., *'the A42G mutant,'* Table 2.3a) look like data typically targeted by named entity recognizers, while mentions in 'natural language' formats (e.g., *'alanine 42 replaced with glycine,'* Table 2.3c-e) look like data typically targeted by information extraction (IE) systems. Jackson and Mouliner [38] note that IE systems attempt to fill in *forms* that describe events involving several entities. All concepts recognized by point mutation recognizers are exactly that. Even mutation mentions in the abbreviated formats, although they look very much like named entities, are references to events that relate three entities: a pre-event state (the wild-type entity), a post-event state (the mutant entity), and an event location (the sequence position).

1.2.3 Relevant dissertation sections

Chapter II is a method-development chapter which discusses a rapid approach for generating regular expressions to drive concept recognition systems. This approach was applied to develop the open-source software package, MutationFinder, which is presented in Appendix I. This work was originally presented in Caporaso *et al.* [39, 40]. Since many manually curated mutation databases already exist (see Section 1.1.6), after MutationFinder had been developed it was possible to perform an evaluation of how automated annotation compared to manual annotation of protein mutations. This evaluation is the subject of Chapter III,

and was originally presented in Caporaso *et al.* [41].

1.3 Protein sequence analysis

My dissertation changes direction beginning with Chapter IV, as I begin presenting work based on biological sequence analysis. I now describe two relevant sequence analysis techniques: multiple sequence alignment and detecting sequence coevolution. Both of these techniques, in the contexts presented here, are based on the study of protein evolution and I therefore begin this section with an introduction to protein evolution.

1.3.1 Multiple sequence alignment

Multiple sequence alignment refers to the task of aligning protein sequences in a table where the rows represent protein sequences and the columns represent sequence positions. The columns represent a shared feature of the corresponding position, such as homology (i.e., all protein positions in a single column were derived from a single ancestral protein sequence position) or structural similarity (i.e., positions are involved in the same structural motif, such as an alpha helix). Because the applications of multiple sequence alignment presented in this dissertation focus on evolutionary relationships of sequences, all multiple sequence alignments described herein were performed to align homologous sequence positions.

Because a set of homologous proteins will very often contain sequences of different lengths (arising from insertion or deletion events since the ancestral sequence), a gap character is introduced into the sequence representations in alignments (usually the - character). When a gap is present in an alignment column, it means that according to the hypothesized alignment, some sequences have either had an insertion or deletion at the corresponding alignment position. (From the alignment alone, it is not possible to tell whether an insertion or deletion occurred.)

Historically the most commonly used tool for multiple sequence alignment is ClustalW [42]. The performance of ClustalW is now considered to be inferior to more modern multiple

sequence alignment techniques, such as MUSCLE [43] and MAFFT [44]. Many multiple sequence alignments were generated for the studies presented in this dissertation, and unless otherwise noted, they were generated with MUSCLE. MUSCLE was chosen because it frequently performs among the top methods when evaluated against the BAliBASE benchmark data [45], and it is available in a convenient implementation from its author. A good review of multiple sequence alignment techniques is presented by Wallace *et al.* [46], and a recent comparison of methods is presented by Essoussi *et al.* [47].

1.3.2 Sequence covariation and molecular coevolution

Most methods of protein sequence analysis make the simplifying assumption that positions in a protein evolve independently of one another, meaning that substitutions at one sequence position do not effect either the rate of substitution or the phenotypically-acceptable amino acids at other positions. However, we know that proteins fold and adopt structures based on interactions between amino acid residues so it is obvious that this assumption is invalid. The latter chapters of this dissertation address non-independent evolution of pairs of positions within and between proteins.

When the identity of the amino acid at one position in a protein has an effect on which amino acids can be present at another protein position, those positions are evolving in a non-independent manner and are said to be coevolving. Knowledge of positions that coevolve in biological sequences has been applied to predict structures of RNAs [48–50] and proteins [51–54]; to predict intermolecular interactions [53, 55]; to identify functionally important regions of molecules [56, 57]; and to identify energetic pathways through molecules [58, 59]. Accurately and comprehensively identifying coevolving positions from sequence data alone is therefore a heavily sought-after goal, with the potential to advance these important research areas.

Probably the most successful application of sequence covariation to date has been predicting secondary structure of non-coding RNAs (ncRNAs). For example, Freyhult *et al.*

[48] applied Mutual Information (MI, introduced in Section 4.2.1.1) to identify base pairing positions in Rfam alignments. In this study they found that MI was useful for predicting the base pairing positions in tRNA, and additionally could be applied to detect pseudoknots.

In an example of using sequence coevolution to identify and understand the role of non-conserved but nevertheless important positions in a protein, Süel *et al.* [59] identified a network of positions which covary with one another in the GPCR family using Statistical Coupling Analysis (SCA, introduced in Section 4.2.1.4). They hypothesize that this network of positions represents an 'energetic pathway' in GPCRs through which allosteric changes are propagated. These 'statistically coupled' positions correlate with previous biochemical data on positions involved in allosteric interactions in GPCRs, suggesting that sequence covariation can be applied to identify non-conserved but functionally significant positions and interactions in proteins.

Most methods which attempt to identify coevolving protein positions begin with multiple sequence alignments of homologous proteins and attempt to identify alignment columns with correlated substitution patterns. This is a very tricky task however because, as is frequently noted, correlation does not imply causation: correlated alignment columns are multiply realized, arising from different underlying causes. As noted by Waddell *et al.* [60], "correlated evolution is what is detected, whereas coevolution is the hypothesized cause when others (such as correlation due to the phylogeny) are discounted." Other causes of coevolution include phylogenetic effects and random covariation, and great care must be exercised to avoid describing a pair of positions as coevolving when they merely covary for other, less interesting reasons. In this text, I discuss detecting *covariation* to infer *coevolution*. Chapter IV introduces sequence covariation and coevolution in detail, and I therefore defer further discussion at this point.

1.3.3 Relevant dissertation sections

Chapter IV presents an evaluation of coevolution detection algorithms on protein alpha helices. This study, originally presented in [61], has shed light on the relative performances of coevolution detection algorithms which explicitly incorporate the phylogeny of the input sequences with methods that do not incorporate the phylogeny. Chapter V is a follow-up study to Chapter IV; there I present in further detail my preferred method for detecting covariation, joint-entropy-normalized mutual information, and several pre- and post-processing steps that I apply in coevolutionary analyses. Chapter VI presents an application of coevolution detection methods to make predictions about protein-protein interactions.

1.4 Software

Publishing software along with analyses is extremely important for computational biologists. It is an essential step in performing reproducible experiments. Providing well-documented source code is the equivalent of publishing a detailed methods section in traditional biological research articles: it provides the opportunity for other scientists to follow an analysis exactly as it was performed, and modify or improve upon a protocol in future studies. My dissertation work has lead to the open-source publication of two software packages: MutationFinder and the PyCogent coevolution module.

1.4.1 MutationFinder

MutationFinder is a stand-alone, open-source, rule-based system for extracting point mutation mentions from text. I am the primary developer of this package. The MutationFinder package also contains a high-quality gold standard data set, and a scoring script for mutation extraction systems. Executables, source code, and unit tests are available in Python, Perl, and Java. MutationFinder can be used as a stand-alone script, or im-

ported into other Python, Perl, or Java applications. MutationFinder is available via its project webpage at http://mutationfinder.sourceforge.net. The system details are presented in Appendix I.

1.4.2 PyCogent coevolution module

PyCogent, or the PYthon COmparative GENomics Toolkit, is an open-source bioinformatics package which I have contributed to since 2003. My largest contribution to date is the coevolution module, consisting of 2453 lines of code plus 4793 lines of unit test code[5], which implements five coevolution detection algorithms with a common interface and support code for performing coevolutionary analyses. PyCogent is available via its project webpage at http://pycogent.sourceforge.net.

1.5 Dissertation goals

My initial thesis goal began with advancing protein point mutation recognition to allow for accurate and comprehensive extraction of protein point mutation mentions from biomedical literature into structured formats. Advances in this area would allow for automatic generation of comprehensive and accurate mutation databases, and other downstream applications such as improved accuracy in retrieval of mutation literature through concept-based information retrieval. Then, by evaluating the data in the resulting protein-specific mutation databases in the context of multiple sequence alignments of homologous protein sequences, I would identify compensated pathogenic deviations (CPDs), or mutations that were deleterious in one species, but where the mutant residue was found as wild-type in another species. Using coevolutionary analyses, I could then test the hypothesis that when CPDs were present in a sequence, the compensated position could be identified by mining biomedical literature, and the compensating position could be identified through coevolutionary analysis.

[5]These statistics refer to revision 191 of PyCogent.

My goal of advancing automated protein point mutation recognition was met and resulted in the publication of MutationFinder and three peer-reviewed articles [39–41]. These studies are presented in Chapters II and III.

My search for CPDs focused on searching for compensating mutations in the myosin heavy chain, using manually compiled mutations in the MyoMapr database [62] to identify the compensated mutations, and sequence covariation analysis to identify the compensating positions. Sequence covariation did not result in hypotheses about potentially compensating positions, and the negative results of that study are presented in Appendix II. My work with sequence covariation did, however, lead to other interesting studies which are discussed more thoroughly in this dissertation. After an in-depth analysis of the performance of nine covariation algorithms on real (i.e., non-simulated) biological sequence data (published as [61] and presented in Chapter IV), I applied a top-performing algorithm to identify intermolecular interactions between components of the Type VI Secretion System (presented in Chapter VI). I now briefly describe the details of each dissertation chapter.

1.5.1 Chapter II

In Chapter II, I present an approach for automatically developing collections of regular expressions to drive high-performance concept recognition systems with minimal human interaction. I applied this approach to develop MutationFinder, a system for automatically extracting mentions of point mutations from text. MutationFinder achieves performance equivalent to or better than manually developed mutation recognition systems, and further, the generation of its 759 patterns has required only 5.5 expert-hours.

In this chapter, I also discuss the development and evaluation of the human-annotated gold-standard corpus created to test MutationFinder. This corpus contains 1515 complete point mutation mentions annotated in 813 Medline abstracts, and is publicly available as part of the MutationFinder package for use in testing future point mutation extraction systems.

1.5.2 Chapter III

Biomedical text mining and other automated techniques are beginning to achieve performance which suggests that they could be applied to aid database curators. However, few studies have evaluated how these systems might work in practice. In Chapter III I focus on the problem of annotating mutations in Protein Data Bank (PDB) entries. I evaluate the relationship between the performance of two automated techniques, a text-mining-based approach (MutationFinder) and an alignment-based approach, using intrinsic versus extrinsic evaluations. I find that high performance on gold standard data (an intrinsic evaluation) does not necessarily translate to high performance for database annotation (an extrinsic evaluation). I show that this is in part a result of lack of access to the full text of journal articles, which appears to be critical for comprehensive database annotation by text mining. Additionally, I evaluate the accuracy and completeness of manually annotated mutation data in the PDB, and find that it is far from perfect. I conclude that currently the most cost-effective and reliable approach for database annotation might incorporate manual and automatic annotation methods. This study was performed in collaboration with the director of the PDB, and the ideas and techniques presented in this chapter are actively being explored as possibilities to enhance the PDB's structure deposition and annotation process.

1.5.3 Chapter IV

In Chapter IV I change focus to detecting coevolving positions in protein sequences from multiple sequence alignments. Identifying coevolving positions in protein sequences has diverse applications ranging from understanding and predicting the structure of single molecules to generating proteome-wide predictions of interactions. Algorithms for detecting coevolving positions can be classified into two categories: tree-aware, which incorporate knowledge of phylogeny, and tree-ignorant, which do not. Tree-ignorant methods are fre-

quently orders of magnitude faster, but are widely held to be insufficiently accurate because of a confounding of shared ancestry with coevolution. In this chapter, I tested the hypothesis that by using a null distribution that appropriately controls for the shared-ancestry signal, tree-ignorant methods would exhibit equivalent statistical power to tree-aware methods. Using a novel t-test transformation of coevolution metrics, four tree-aware and five tree-ignorant coevolution algorithms were systematically compared, applying them to myoglobin and myosin alignments. I further considered the influence of sequence recoding using reduced-state amino acid alphabets, a common tactic employed in coevolutionary analyses to improve both statistical and computational performance.

Consistent with my hypothesis, the transformed tree-ignorant metrics (particularly Mutual Information) often outperformed the tree-aware metrics. Examination of the effect of recoding suggested that charge-based alphabets were generally superior for identifying the stabilizing interactions in alpha helices. Performance was not always improved by recoding however, indicating that the choice of alphabet is critical.

The results suggest that t-test transformation of tree-ignorant metrics can be sufficient to control for patterns arising from shared ancestry, therefore allowing the faster tree-ignorant methods to be applied with more confidence. This finding opens the possibility of applying coevolution algorithms in large (possibly genome-wide) analyses of sequence coevolution that previously would have required too much compute time.

1.5.4 Chapter V

Chapter V is a bridge between the comparison studies presented in Chapter IV and the application of coevolutionary analyses presented in Chapter VI and Appendix II. In this chapter, I present a follow-up study where I attempt to optimize the joint-entropy-normalized mutual information (NMI) metric for applications of coevolutionary analysis with pre-processing and post-processing steps. The NMI metric is presented in greater detail in this chapter, and NMI plus the filtering steps presented here form the bases of the

studies presented in Chapter VI and Appendix II.

1.5.5 Chapter VI

Just as amino acid residues within a protein do not act alone to create a protein's function, individual proteins rarely act alone. Most proteins are involved in pathways or complexes which result in an emergent phenotype. Prediction of physical and function protein-protein interactions is an active area of research in computational biology, and one which is theoretically possible to address through sequence covariation. In Chapter VI, I attempt to detect coevolving positions between pairs of proteins suspected to be involved in the recently discovered Type VI Secretion System (T6SS) complex to make predictions about which pairs are involved in physical interactions with one another. These predictions are now being tested biochemically in attempt to construct a model of the T6SS complex.

CHAPTER II

PATTERN DEVELOPMENT FOR CONCEPT RECOGNITION SYSTEMS

2.1 Background

The development of accurate and comprehensive text mining and biomedical language processing systems typically requires many person-hours of manual work, usually in the form of annotating corpora for machine-learning-based methods, or developing rules for rule-based methods. Methods that accelerate the development of useful systems have the potential to facilitate access to data currently available only as free text in the biomedical literature. In this chapter I investigate two problems: first, I test the hypothesis that automatic methods can be used to learn patterns of sufficient quantity and quality to match or out-perform a manually-built rule set; and second, I aim to develop a high-quality corpora of mutation mentions with good inter-annotator agreement to serve as a gold standard for mutation extraction systems.

The work in this chapter focuses on MutationFinder, a high-performance system for extracting mentions of point mutations from text, and a high-quality corpus containing 1515 hand-annotated mutation mentions in 813 MEDLINE abstracts. MutationFinder relies on a collection of regular expressions to extract mutation mentions from text. A unique feature of this system and its development process is that its top-performing collection of regular expressions was generated by an almost fully automated process, requiring only 5.5 hours of human interaction, and that it outperforms a baseline system which used only manually constructed regular expressions. The MutationFinder corpus is unique in that it is larger (in terms of number of mutations and abstracts annotated) and more reliable (because of its development around clearly defined annotation guidelines, and evaluation by interannotator

[6]The work presented in this chapter was originally presented in: *Rapid Pattern Development for Concept Recognition Systems: Application to Point Mutations* (Journal of Bioinformatics and Computational Biology, V5 N6 Pg. 1233-1259).

agreement) than existing mutation corpora. Corpus annotation required 54 person-hours, outside of system development.

The MutationFinder software and corpus were initially presented in [39], and are summarized Appendix I. The subject of this chapter is a generalizable methodology for designing high-performance, rule-based, biomedical concept recognition (CR) systems with minimal human effort. Using MutationFinder as a case-study, the utility of this methodology for streamlining CR system development is demonstrated. Additionally a description of a manually curated corpus containing mutation mentions, and the annotation process and guidelines used in its construction and evaluation, are presented. MutationFinder, the complete corpus, and the corpus annotation guidelines are publicly available at http://mutationfinder.sourceforge.net.

2.1.1 Evaluating text mining systems

Text mining systems are frequently evaluated in terms of *precision*, *recall*, and *F-measure* on test data to evaluate the degree to which the system matches a pre-defined gold standard data set. Precision (P), related to specificity, is defined as the proportion of all positives returned by the system which are true. Recall (R), or sensitivity, is defined as the proportion of all actual positives which are identified by the system. F-measure is a summary statistic defined as the harmonic mean of precision and recall. Where TP refers to *true positives*, FP refers to *false positives*, and FN *false negatives*, precision, recall, and F-measure are calculated as:

$$Precision = \frac{TP}{TP + FP} \tag{2.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.2}$$

$$F - measure = \frac{2 \times P \times R}{P + R} \tag{2.3}$$

2.1.2 Prior art

Point mutation recognition systems have been the subject of several recent publications [28, 29, 39, 63–65]. MuteXt [28], MEMA [29], and Mutation GraB [63] attempt to extract mentions of mutations paired with a specific gene or gene product from input texts. OSIRIS [64] is a web-based information retrieval system for compiling mutation literature using a concept-driven, mutation-recognition approach. MutationMiner [65] is a system which uses mutation data from literature to annotate protein structures.

The MuteXt system, when tested against a manually compiled database of GPCR mutations, was reported to achieve a precision of 87.9% and a recall of 49.5%. MEMA, when judged on 100 human-annotated abstracts, was reported to achieve a precision of 99% and a recall of 35% for extracting gene-mutation pairs, and a precision of 98% and a recall of 75% when extracting mutations only (i.e., when no attempt is made to connect them with a specific gene or protein). Mutation GraB , when tested on 295 unseen, manually annotated articles, was reported to achieve a mean precision of 74% and a mean recall of 81%. When extracting mutations alone (and tested on the MEMA corpus), Mutation GraB was reported to achieve a precision of 97.7% and a recall of 77.3%. Because the testing schemes (i.e., test corpus and computed performance metrics) differed for each system, these values are not directly comparable, except for the comparison between Mutation GraB and MEMA. The reported performances of these systems is summarized in Table 2.1.

2.1.3 Automatic pattern generation for concept recognition

Several previous attempts [66–72] have been made to generate patterns for concept recognition systems automatically. The AutoSlog/AutoSlog-TS systems [66, 67] are classic early examples, and the approach of Hovy and his colleagues [68, 69] have been quite influ-

Table 2.1: Performance of prior systems.
The reported performances of prior mutation extraction systems. Note that because systems were tested on different corpora, the performances are not directly comparable.

System	P	R	F
MuteXt	0.88	0.50	0.64
MEMA	0.99	0.35	0.85
MutationGraB	0.74	0.81	0.77
MutationGrab (on MEMA corpus)	0.98	0.77	0.86

ential in recent years. These systems differ from ours in that they either require annotated corpora [66], hand-classified documents [67], or multiple rounds of iteration and mandatory tuning [68, 69].

Recently, there has also been domain-specific work on pattern-learning for biomedical texts [70–72]. These approaches differ from ours in that they typically overgenerate considerably, producing large numbers of patterns whose utility is often underevaluated and questionable.

2.1.4 MutationFinder

MutationFinder [39] focuses on extracting mentions of point mutations without trying to connect them with their gene or protein source. While at first glance the fact that MutationFinder does not attempt to associate mutations with their gene or gene product might appear to be a limitation, we deliberately chose this approach in order to create modular software. Rather than simultaneously attacking three unsolved problems (i.e., extracting mutation mentions, extracting gene/protein names, and associating the appropriate pairs), MutationFinder's design is based on the UNIX philosophy of software development: 'Write programs that do one thing and do it well.' [73] Our approach gives MutationFinder users the ability to combine the output of MutationFinder with the output of, for example, the currently top-performing protein name identifier (Hakenberg *et al.* [34], as determined by the BioCreative 2 evaluation). Rather than being locked into a single problem or approach,

MutationFinder users may mix and match independent tools. Source code modifications are not required to, for example, use an alternative protein-name-recognition technique.

Like the earlier mutation recognition systems [28, 29, 63], MutationFinder applies a set of regular expressions to identify mutation mentions in input texts. Our currently-top-performing collection of regular expressions results in a precision of 98.4% and recall of 81.9% when extracting mutation mentions from completely blind test data. (We cannot make a direct comparison of our performance with those of the earlier systems, but the publication of our gold standard data set and a script for scoring the output of mutation recognition systems makes direct comparisons of mutation recognition systems readily performable for the first time.)

Unlike the earlier systems, the 759 regular expressions driving MutationFinder were automatically generated using a generalizable bootstrap approach that is the primary subject of this chapter. This approach minimizes the time investment typically associated with developing high-performance rule-based systems in the form of manual annotation or pattern construction. Only 5.5 hours were required by a domain expert (Caporaso) to refine an initial collection of patterns into the input for an automated regular expression generator.

We also developed a gold standard data set, which is an additional subject of this chapter, at a cost of approximately 54 person-hours split among three annotators. It was split into development data and blind test data, and used for optimizing and evaluating MutationFinder. While extremely helpful for system optimization and testing, a human-annotated gold standard data set *is not a prerequisite* for developing high-performance concept recognizers by our method.

2.2 Performance metrics

Systems were compared in terms of precision, recall, and F-measure on three separate metrics: *Extracted Mentions, Normalized Mutations,* and *Document Retrieval*. When cal-

culating *Extracted Mentions*, each mention of a mutation (including duplicates) must be identified. When calculating *Normalized Mutations*, on the other hand, a perfect score can be achieved if each of the mutations that is mentioned in text is extracted at least once, even if some mentions are missed. Last, *Document Retrieval* measures a system's ability to recognize if *any* mutations have been mentioned in a document. *Extracted Mentions* is the strictest metric in terms of recall, and is perhaps the most discriminative for judging the performance of the information extraction system (since each missed mutation mention is scored as a false negative). *Normalized Mutations* is the best measure of a system's utility in terms of document-level entity recognition, whereas *Document Retrieval* measures a system's utility in terms of document-level information retrieval.

2.3 Algorithm and application

Our approach for achieving high-performance concept recognition is based on a six-step process for regular expression generation. Like Webclopedia [68] and the Huang *et al.* [71] text alignment technique, we do not rely on annotated training data. Annotated development and test data are useful for system optimization and testing, but are not required for rule set development. The six steps, followed by an in-depth discussion with examples (see Table 2.3 and Sections 2.3.1 through 2.3.6), are:

1. Automatically compile a collection of *raw patterns*, or patterns likely to represent the information to be extracted. This step aims to achieve high recall while paying little attention to precision.

2. Refine the collection of raw patterns to eliminate false positives. This step is intended to address the precision ignored in the first step.

3. Process the patterns to assign terms from a semantic grammer to the entities to be extracted. In the case of mutation extaction, the semantic classes to assign are wild-

type residue (`WRES`), mutant residue (`MRES`), and sequence position (`SPOS`). We refer to the semantically annotated patterns resulting from this step as *mutation patterns*.

4. Generate regular expressions from the mutation patterns resulting from step 3.

5. Perform *post hoc* analyses to optimize system precision — this does not require human-annotated development data. Optionally, optimize system recall on development data. General changes to the system can be tested, or in-depth error analyses can be performed to inform design and implementation details.

6. Optionally[7], test the system on previously unseen test data. It is critical that this data be different from the development data.

Table 2.2 details the amount of time devoted to each step. Using MutationFinder as a case study, the subsequent sections demonstrate the development of a high-performance rule set for concept recognition using our methodology.

Table 2.2: Person-hours required in each system development step.
*Steps 5.2 and 6 required time for corpus development. These steps are optional, and therefore aren't included in the total time for system development. Step numbers correspond with subsection numbers in Section 2.3.

Step	**Person-hours**
1: Raw pattern compilation	0.0
2: Raw pattern refinement	2.0
3: Mutation pattern generation	2.5
4: Regular expression generation	0.0
5.1: System optimization: precision	1.0
5.2: System optimization: recall (optional)	18.0*
6: System testing (optional)	36.0*
Total required time	5.5

[7]While optional for system development, this step is, however, required if one wishes to publish performance data on their system.

2.3.1 Step 1: Raw pattern compilation

To bootstrap the system, a simple rule is applied to identify blocks of text that were likely to describe point mutations. These text blocks were then converted into *raw patterns*. Example input texts and the resulting raw patterns are presented in Table 2.3 and discussed throughout this section. This step requires no human interaction.

2.3.1.1 Bootstrap rule

Development of the bootstrap rule for automatically generating raw patterns was informed by domain knowledge. The bootstrap rule stated that a block of text might describe a point mutation if it:

1. contained at least two mentions of amino acid residues,

2. mentioned at least one positive integer in numeric characters, and

3. did not span a sentence break.

This rule was implemented by splitting input text on periods and applying two regular expressions to each resulting 'sentence.' (While more effective methods exist for splitting sentences [74], a review of the raw patterns, and an error analysis following system development, suggested that this simplistic approach was acceptable for this application.) Two regular expressions were applied to determine if a sentence matched the bootstrap rule. First, an amino acid pattern (ignoring case) matched mentions of amino acids in their three-letter abbreviations or full names. This was required to match twice. Second, a sequence position pattern, designed to identify mentions of specific positions in a gene or protein, matched numerically-expressed positive integers less than 10,000. This was required to match once.

Table 2.3: Example output from steps 1-4.

Source PubMed identifiers for rows f and g are 11210138 and 2198714, respectively. All other examples are hypothetical instances of common patterns. For space, the output of *Step 4* is presented only for *row a*; all regular expressions are available as supplementary material. The regular expression used to match residues is abbreviated with `RES_PATTERN` for clarity, but is available as supplementary material. Line breaks in all texts, patterns, and the regular expression are present for clarity. Patterns eliminated in *Step 2* contain cross-references to the section describing the triggered rule. `RES` and `POS` signify undifferentiated mentions of possible amino acid residues and sequence positions, respectively; `WRES`, `MRES`, and `SPOS` signify wild-type residue, mutant residue, and sequence positions (respectively) referring to the mutation mentioned. Step numbers correspond with subsection numbers in Section 2.3.

	Step 1.1 result: Text containing mentions identified (Section 2.3.1.1)	**Step 1.2 result:** Raw patterns generated (Section 2.3.1.2)	**Step 2 result:** Raw patterns refined (Section 2.3.2)	**Step 3 result:** Mutation patterns generated (Section 2.3.3)
a	*'the **Ala64Gly** mutation'*	`RESPOSRES`	`RESPOSRES` *(no change)*	`WRESSPOSMRES`
b	*'the **Ala64—>Gly**'*	`RESPOS-->RES`	`RESPOS-->RES` *(no change)*	`WRESSPOS-->MRES`
c	*'we mutated **Ala64** to **glycine***'	`RESPOS to RES`	`RESPOS to RES` *(no change)*	`WRESSPOS to MRES`
d	*'we mutated **Ala64** to **glycine**, **leucine**, and **valine**'*	`RES-POS to RES, RES, and RES`	`RES-POS to RES, RES, and RES` *(no change)*	`WRES-SPOS to MRES`
				`WRES-SPOS to RES, MRES`
				`WRES-SPOS to RES, RES, and MRES`
e	*'**alanine** was mutated to **glycine** at positions **64** and **72**'*	`RES was mutated to RES at positions POS and POS`	`RES was mutated to RES at positions POS and POS` *(no change)*	`WRES was mutated to MRES at positions SPOS`
				`WRES was mutated to MRES at positions POS and SPOS`
f	*'the catalytic residues (**Asp325**,**Glu354**, and **Asp421**) are necessary'*	`RESPOS, RESPOS, and RESPOS`	false positive, manually eliminated (Section 2.3.2.2)	*not applicable*
g	*'A **pro**-memoria on the occasion of the **200**th anniversary of **his** death'*	`RES-memoria on the occasion of the POSth anniversary of RES`	false positive, automatically eliminated (one MEDLINE occurrence) (Section 2.3.2.1)	*not applicable*
h	*'the **Ala64Gly** and **Ala72Thr** mutations'*	`RESPOSRES and RESPOSRES`	manually eliminated, (two occurrences of *row a* pattern) (Section 2.3.2.3)	*not applicable*

Step 4 result: regular expressions generated *(Row a example only)* (Section 2.3.4)	`(^|[\s\(\[\'"/,\-])(?P<wt_res>(RES_PATTERN))(?P<pos>[1-9][0-9]*)(?P<mut_res>(RES_PATTERN))(?=([\.,\s)\]\'":;\-?!/]|$))`

2.3.1.2 Raw pattern generation

In a given sentence, if the amino acid pattern matched at least twice, and the position pattern matched at least once, a *raw pattern* was generated by selecting the shortest span of text that contained all of the amino acid residue and integer mentions. Amino acid mentions were replaced with `RES` and integer mentions were replaced with `POS`. Example input texts and the resulting raw patterns are presented for both true positive (Table 2.3a-e,h) and false positive (Table 2.3f,g) matches.

We applied this rule to a local installation of MEDLINE in its entirety (in October of 2006). 89,108,055 sentences from 16,333,215 MEDLINE-record title and abstract fields were provided as input for creating raw patterns. 262,157 unique raw patterns were generated from 295,618 sentences complying with the bootstrap rule. The most commonly occurring patterns included `RESPOSRES`, `RESPOS-->RES`, and `RESPOS to RES` (Table 2.3a, 2.3b, and 2.3c, respectively). 258,067 of these raw patterns occurred only once in MEDLINE. The remaining 4090 patterns occurred at least twice. Since raw patterns were generated from the shortest span of text containing *all* of the `RES` and `POS` entities, each sentence could generate only a single raw pattern. The complete set of patterns generated by this process is available on the MutationFinder project webpage.

As mentioned earlier, this step is intended to achieve high recall and is very prone to false postives. 99.7% of these raw patterns were *not* used for generating regular expressions because the vast majority represented false positives.

2.3.2 Step 2: Raw pattern refinement

Many irrelevant patterns are obtained by applying the bootstrap rule to Medline. Through a partially automated process of refinement, the number of raw patterns was reduced from 262,157 to 634. This required two hours of work by an expert.

2.3.2.1 Automatic raw pattern refinement

First, to eliminate raw patterns likely to be unimportant, all raw patterns which occurred only once in Medline were filtered out. This reduced the number of raw patterns by 98.44%, from 262,157 to 4090.

2.3.2.2 Manual raw pattern refinement: false positives

Second, the remaining 4090 patterns were manually reviewed to determine their relevance. If it was intuitively obvious that a pattern represented a mutation mention, it was retained; otherwise, the pattern was deleted. (Questionable patterns tended to have a low number of occurrences, and since an objective of this project was to minimize the efforts of humans in pattern development, it was decided that they should be deleted without investing time to investigate their validity.) Following this process, all false positive (i.e., non-mutation) patterns were deleted.

2.3.2.3 Manual raw pattern refinement: non-unique patterns

Finally, raw patterns which would not provide novel mutation patterns were manually deleted. For example, the raw pattern `RESPOSRES and RESPOSRES` (Table 2.3h) would not provide a novel mutation pattern, because it contained two mentions of the simpler pattern `RESPOSRES` (Table 2.3a). These manual refinement steps further reduced the number of raw patterns from 4090 to 634, and took a total of two hours.

2.3.3 Step 3: Mutation pattern generation

Since raw patterns could contain more than one mutation mention (e.g., Table 2.3d,e), and since *Step 1* did not assign wild-type and mutant identities to each residue mention, the next step was to tag the elements to be extracted as the wild-type residue, mutant residue, and sequence position in each raw pattern. In this step, each remaining raw pattern was manually reviewed and edited to assign semantic tags indicating whether a given term was

the wild-type residue, mutant residue or sequence position entity. The resulting patterns are referred to as *mutation patterns*, since they can be used to represent point mutation mentions unambiguously. Each mutation pattern will match, at most, a single mutation.

For raw patterns containing only a single mutation mention (e.g., Table 2.3a-c), `RES` and `POS` entries were replaced with `WRES`, `MRES`, and the `POS` entry with `SPOS`. The decision of which `RES` term to be converted to `WRES` versus `MRES` was straight-forward to someone familiar with biomedical literature (i.e., a domain expert). In 83.13% of the mutation patterns, the first residue mention was assigned `WRES`.

100 raw patterns contained more than one mutation mention (e.g., Table 2.3d,e). Semantic tag assignment was slightly more complicated for these. The raw pattern was duplicated to appear once for each mutation mention, and each raw pattern was edited to assign identities. This duplication of patterns containing more than one mutation mention resulted in an addition of 125 patterns, from 634 raw patterns to 759 mutation patterns. In Table 2.3d and 2.3e, the raw patterns are duplicated to yield three and two mutation patterns, respectively, corresponding to the number of mutation mentions in each.

In duplicated patterns (e.g., Table 2.3d,e), extraneous text remained which would result in the over-specialization of a pattern. These were edited so that only the shortest piece of text containing the `WRES`, `MRES`, and `SPOS` entities was retained, and leading or trailing text was removed. (For example, in the first mutation pattern in Table 2.3e, the trailing `and POS` was removed when the pattern was duplicated.)

Automating this procedure is possible, but would require a large development corpus to achieve acceptable recall. Given a collection of raw patterns, mutation patterns with greater than two occurrences of `RES` or greater than one occurrence of `POS` were deleted. In the remaining raw patterns, the first `RES` mention was replaced with `WRES`, the second with `MRES`, and the position with `SPOS`; this resulted in an automatically generated collection of mutation patterns. Beginning with a simple mutation finder that would match

only *wNm-formatted* mutation mentions using single-letter amino acid abbreviations, each mutation pattern was tested to determine if it increased the F-measure of the system on the development data. If so, that pattern was retained. Otherwise it was deleted.

Due to the infrequent occurrence of many mutation patterns in MEDLINE, the patterns actually occurring in the MutationFinder corpus is limited. Very few patterns end up being retained by this method, and the resulting system is over-fit to the development data. This approach was not incorporated into our final system, but illustrates a way to automate mutation pattern generation. Manually generating mutation patterns, which took approximately 2.5 hours, was far more time-efficient than developing a larger corpus to support the process.

2.3.4 Step 4: Automatic regular expression generation

Translation from mutation patterns to regular expressions is a computationally trivial task. Each of the entity types in a raw pattern is replaced with a regular expression to match the text strings which might appear in that slot. Residues are matched in their three-letter abbreviation or full name, and sequence positions are matched as positive integers between 1 and 9,999. The entities to be extracted from the text (`WRES`, `MRES`, and `SPOS`) are matched using symbolic grouping expressions. In addition to the regular expressions contained in the collection of mutation patterns, an additional pattern was always added to match *wNm-formatted* mutation mentions using single-letter abbreviations. This pattern was case sensitive, and is the only pattern that matches single-letter-abbreviated residue mentions.

A minor processing step was applied at this stage to generalize regular expressions. Occurrences of either `a` or `an` were replaced with the regular expression `(a|an)`, so the texts *'Ala42 to a glycine'* and *'Ala42 to an isoleucine'* would both be matched by the same regular expression.

2.3.5 Step 5: System optimization

After the set of regular expressions are generated and refined, it is possible to optimize the rule set and individual rules by testing their performance on annotated and unannotated corpora. For MutationFinder, two types of system optimization were performed: optimization of precision and optimization of recall. Our experience suggests that optimizing precision is essential, but that optimizing recall is optional. Optimization of precision was performed via a post-hoc error analysis and took very little time; optimization of recall required the use of our human-annotated development data. While the error analysis itself did not take long, approximately 18 person-hours were devoted to corpus annotation. (Until this point, no annotated data has been required for system development.) Because optimization of system recall is not required (as we will show), we do not include the time for corpus development in our estimate of time for system development.

2.3.5.1 Optimizing precision with post-hoc error analysis on unannotated data

We used an unannotated corpus of GeneRIFs [75] to optimize MutationFinder's precision. Before precision-optimization, we applied MutationFinder to the complete collection of GeneRIFs (through June, 2006). We randomly selected 100 GeneRIFs which MutationFinder identified as containing mutations. We then manually scored MutationFinder's Document Retrieval precision on these GeneRIFs. False positives were categorized, and adjustments were made to avoid the most significant error types. Three major adjustments to MutationFinder (of the six mentioned in Section A.2.2) were made as a result of this error analysis.

Prior to optimization, certain gene/protein names were commonly mistaken for mutation mentions. For example, *H4A* and *E2F* were commonly mistaken as representing mutations. MutationFinder's regular expression for matching *wNm-format* mutation men-

tions with *single-letter abbreviations only* was modified to require that N be greater than 9. This lead to a 30% increase in precision on the same collection of GeneRIFs (Table 2.4b). Next, to further avoid confusion with gene names, we required that the same regular expression be case sensitive, only matching uppercase letters. This led to an 11% increase in precision (Table 2.4c). Third, this error analysis led us to implement a post-processing rule to ignore 'mutations' where the wild-type and mutant entities were identical. For example, mention of a homozygous genotype, *C1298C*, was initially extracted as a point mutation. This rule led to a 3% increase in precision (Table 2.4d).

This post-hoc error analysis required only a minimal time commitment from a human domain expert. Scoring the precision of the 100 random GeneRIFs took approximately 10 minutes, and the resulting error analysis led to a large increase in system precision. Prior to implementation of these rules, the precision achieved on GeneRIFs was 58% (Table 2.4a); after optimization, when using the same data set, precision was 97% (Table 2.4e), and no false negatives were incurred. (Note that the precision improvements displayed in Table 2.4b-d are for these changes in isolation. The optimizations are not mutually exclusive. For example, a false positive resulting from the text *'crucial for C1q binding to ligands'*[8] is avoided by both the N > 9 and case-sensitivity rules.)

Table 2.4: Post-hoc precision estimates on a sample of 100 GeneRIFs.
Pre-optimized MutationFinder refers to an implementation of MutationFinder which does not incorporate these three precision-optimizing rules. These data show the relative contribution of each individual rule.

	Description	Precision
a	Pre-optimized MutationFinder	0.58
b	Pre-optimized MutationFinder + N > 9 rule	0.88
c	Pre-optimized MutationFinder + case-sensitivity rule	0.69
d	Pre-optimized MutationFinder + wild-type ≠ mutant rule	0.61
e	MutationFinder	0.97

[8]Source Entrez Gene ID: 714

A danger in a precision-oriented optimization step such as this is the over-fitting of rules to the test collection. Care should be taken only to incorporate rules that make intuitive sense and are supported by the analysis. For example, rather than requiring that sequence positions always be greater than 9, we tailored this rule specifically to what we observed and what made intuitive sense: many genes names appear similar to single-letter abbreviated *wNm-formatted* mutation mentions, but we do not expect to see gene names that conform to many other mutation patterns. We tailored this rule based on domain knowledge and the results of this error analysis. Incidentally, the remaining false positives were all cell line names that were extracted as mutation mentions. The problem of mistaking gene and cell line names for mutations is revisited in Section 2.6.1.

Since the MutationFinder corpus was available for optimization, these rules were additionally tested on that data set. Comparison of the pre-optimized and post-optimized MutationFinder on our development data set showed an approximate 11% precision increase with only a 1% decrease in recall for extracting Normalized Mutations.

2.3.5.2 Optimizing recall — optional

Recall optimization of the complete rule set generally took one of two forms. First, the overall utility of general rules was tested (e.g., replacing occurrences of `a` or `an` with `(a|an)` in regular expressions), by testing the performance of MutationFinder on the development data with the rule alternately incorporated and not incorporated. Second, error analyses performed by looking at false negatives incurred by the system were used to inform future decisions about system design.

The methods described in the six-step approach are biased toward high recall. By applying the bootstrap rule to a very large corpus of relevant documents, important patterns are compiled without the need for much human interaction. Recall optimization on the MutationFinder corpus did little to improve the performance of the system on the same data.

Table 2.5 compares the different approaches that were applied. Table 2.5a presents the performance of the best collection of regular expressions on the development data. Implementation details are now discussed in the context of their effect on performance.

Table 2.5: Extracted Mention performances of recall-optimization rules.
True positives (TP), false positives (FP), false negatives (FN), precision (P), recall (R), and F-measure (F) are presented for the recall optimization rules. These metrics are introduced in Section 2.1.1.

	Description	**TP**	**FP**	**FN**	**P**	**R**	**F**
a	top-performing system	456	8	94	0.983	0.829	0.899
b	one-letter abbreviations included	474	265	76	0.641	0.862	0.736
c	no linguistic processing of a,an	456	8	94	0.983	0.829	0.899

2.3.5.3 Utility of single-letter abbreviations

As described in Section 2.3.4, when automatically generating regular expressions amino acid residues were matched by their three-letter or full name abbreviations, but not by their single-letter abbreviations. As illustrated by comparing lines *a* and *b* in Table 2.5, matching on single-letter abbreviations is associated with an Extracted Mentions recall increase of 3.3 percentage points and precision decrease of 34.2 percentage points. While this decrease in precision could likely have been reduced by allowing one-letter abbreviations only in certain patterns (rather than globally), the recall increase presented is the maximum that could be achieved. Because a major emphasis of this study is to minimize the amount of time invested in manual pattern generation and refinement, it was determined that hand-selecting patterns would take too long to be worth a maximum of 3.3 percentage point increase. It is important also to note that the Normalized Mutation recall increase was only 0.6 percentage points (data not shown), suggesting that in many practical applications of MutationFinder the recall difference would be minimal.

2.3.5.4 Utility of linguistic processing

As noted in Section 2.3.4, some regular expressions were generalized by substituting the words *a* and *an* with the regular expression `(a|an)`. This processing step resulted in no change in performance on our development corpus. (See Table 2.5a,c) Since we observed no precision degradation, but expect a recall improvement on a larger corpus, we opted to incorporate this processing step into our system.

2.3.6 Step 6: System testing

After optimization, MutationFinder was applied to the previously unseen test corpus. Performance in terms of our three performance metrics is presented in Table 2.7. As noted earlier, system testing on unseen data is not required for rule development and the time for corpus development is therefore not included in the estimate of time for system development.

MutationFinder was evaluated on the blind test subset of the gold-standard corpus, which consists of 910 mutation mentions in 508 documents annotated by Annotators 1 and 2. MutationFinder achieved a precision of 0.984 and a recall of 0.819 on Extracted Mentions on this test collection. Complete test results are presented in Section 2.5.1 and Table 2.7.

To test the hypothesis that automatically generated patterns can achieve equivalent or improved performance when compared to manually generated patterns, MutationFinder was compared with a collection of manually generated rules. These manually generated rules were constructed by Annotator 3 prior to the automatically generated rules, in an initial attempt to build a high-performance mutation recognition system. The test collection is blind with respect both to the automatically-generated and manually-generated rules. The manually generated system is available via the MutationFinder project webpage as `MutationFinder-0_2-beta`.

2.4 Gold standard corpus

The gold standard data set consists of 1515 annotated complete point mutation mentions in the title and abstract fields of 813 MEDLINE records (the gold standard corpus). To compile a corpus likely to contain many mutation mentions, documents were randomly selected from the collection of articles cited as primary citations for mutant Protein Data Bank[76] (PDB) entries.

Three mutation types were annotated: point mutations, insertions, and deletions[9]. The structure of each mutation annotation was based on a simple mutation event ontology that was developed for this purpose. The vast majority of mutations annotated were point mutations; this corpus proved to contain too few insertion and deletion mentions to be useful for testing insertion/deletion extraction systems.

Mutation annotations could be *complete* or *partial*. For example, the text *'mutation at alanine-64'* only mentions a wild-type residue and a sequence position. If the associated mutant residue was not present in the same document (i.e., title and abstract), a partial point mutation annotation would be created, which contained a blank `Mutant Element` field. If a mention contained the information necessary to fill the `Sequence Position`, `Wild-type Element`, and `Mutant Element` fields (for a `Substitution Event`), the resulting annotation was complete. Comprehensive information on the annotation process and the mutation ontology is available on the MutationFinder project webpage.

2.4.1 Preprocessing

To reduce the time needed to annotate the corpus, all of the documents (with the exception of 25, which will be discussed shortly) were preprocessed to automatically annotate obvious mutation mentions. Mutations described in the *wNm format* were automatically

[9]Although insertions and deletions were annotated, MutationFinder does not attempt to extract these mutation types. These may be targeted in a future version.

annotated. All automatic annotations were manually inspected for correctness during the annotation process.

2.4.2 Annotation

Knowtator [77], a text annotation tool, was used for corpus annotation. Annotators marked up mentions of mutations that were not identified in the preprocessing step and validated mutation mentions that were labeled during the preprocessing step. Preprocessing errors (e.g., the text *'the E2F protein'* erroneously annotated as a point mutation) were repaired or deleted, as necessary.

The corpus was divided into five subcollections for the annotation process:

1. 25 abstracts, preprocessed, annotated by all annotators;
2. 25 abstracts, not preprocessed, annotated by all annotators;
3. 254 abstracts, preprocessed, annotated by Annotator 1;
4. 254 abstracts, preprocessed, annotated by Annotator 2;
5. 255 abstracts, preprocessed, annotated by Annotator 3.

Subcollections 1 and 2 were used to calculate interannotator agreement (IAA); by preprocessing only subcollection 1 the effect, if any, that preprocessing had on the annotation process could be determined. The 50 documents in subcollections 1 and 2 were randomly selected from the full corpus and added (randomly dispersed) to each annotator's annotation set (subcollections 3, 4 and 5). Some bias may have been introduced by not preprocessing subcollection 2 (because these documents were sometimes readily identifiable).

2.4.3 Annotation guidelines

To guide the annotation process and promote continuity between annotators, a draft set of annotation guidelines was constructed prior to the annotation process. Meetings among

the annotators were periodically conducted whereby difficult cases were discussed and the annotation guidelines adjusted.

2.4.4 Annotation quality assurance

After the completion of the annotation process, each of the annotators independently performed a review and consistency check of their entire annotation set based on the finalized set of annotation guidelines.

2.4.5 Interannotator agreement

Interannotator agreement (IAA) was calculated in a pairwise manner using the same metrics that were used to evaluate MutationFinder. Each annotator's data was alternately treated as the gold standard, and the average F-measure (harmonic mean of precision and recall) was calculated for Extracted Mentions, Normalized Mutations, and Document Retrieval. IAA was calculated over all complete point mutations, and mean pairwise IAA was 95.0% for Extracted Mentions. Complete IAA data is presented in Section 2.5.2 and Table 2.9.

2.4.6 Division of gold standard into development and test data

Annotator 3 (Caporaso) had the role of both corpus annotator and primary developer of MutationFinder. To avoid over-fitting MutationFinder to the gold standard corpus, the corpus was divided into development and test subsets.

The development subset of the gold standard was composed of documents from subcollections 1, 2, and 5 (i.e., the 305 documents that were annotated by Annotator 3). The test subset was composed of documents from subcollections 3 and 4 (i.e., the 508 documents that were not annotated by Annotator 3). For a detailed breakdown of the gold-standard corpus and the division between development and test data, see Table 2.6.

The development corpus was used for informing decisions about patterns and rules

which should be included in the system. The test set served as a completely blind test for MutationFinder. Evaluation was only performed using the test data following system development and optimization.

Table 2.6: Corpus contents.
Contents of the development (DEV) and test (TEST) sets derived from the generated gold-standard corpus. For the three mutation types that were annotated, the number of complete annotations is presented, followed by the number of partial annotations in parenthesis. The vast majority of annotations were point mutations.

	DEV	TEST	TOTAL
Documents	305	508	813
Documents containing mutation mentions	111	212	323
Documents not containing mutation mentions	194	296	490
Point mutation annotations (partial)	605 (56)	910 (150)	1515 (206)
Insertion annotations (partial)	0 (0)	0 (3)	0 (3)
Deletion annotations (partial)	4 (0)	10 (5)	14 (5)

2.5 Results

The two central goals of this work were to: first, test the hypothesis that automatic methods can be used to learn patterns of sufficient quantity and quality to match or outperform a manually-built rule set; and second, to develop a high-quality corpora of mutation mentions with good inter-annotator agreement to serve as a gold standard for mutation extraction systems. I have met both of these goals, and present supporting data herein.

2.5.1 Manually versus automatically constructed rules

My initial attempt at building a high-performance concept recognition system involved manually compiling rules, informed during my annotation of the development corpus. This annotation process required approximately 18 hours to complete, and was an important step in development of the manually compiled rule set. The automatic approach required only 5.5 person-hours for rule development, and achieved higher F-measure that the manually

constructed rule set when tested on data that were blind with respect to both rule sets (Table 2.7). I conclude that automatically generated rule sets can achieve performance that is at least equivalent to manually generated rule sets. In the case of MutationFinder, the automatically constructed rule set took less than 1/3 of the time required for compiling the manually constructed rule set.

Table 2.7: Comparison of manually and automatically generated rule sets.
Counts of true positives (TP), false positives (FP), and false negatives (FN) are presented, along with precision (P), recall (R), and F-measure (F) for the three different performance metrics. These data can be recreated by applying versions `0_2-beta` (for manually generated rules) and `0_3-beta` (for automatically generated rules) of MutationFinder to the blind test data, and scoring with the provided performance script. All code and data is available at *http://mutationfinder.sourceforge.net*.

Manually generated rules	TP	FP	FN	P	R	F
Extracted Mentions	686	4	221	0.994	0.756	0.859
Normalized Mutations	352	2	124	0.994	0.740	0.848
Document Retrieval	146	0	36	1.000	0.802	0.890
Automatically generated rules	TP	FP	FN	P	R	F
Extracted Mentions	743	12	164	0.984	0.819	**0.894**
Normalized Mutations	384	10	92	0.975	0.807	**0.883**
Document Retrieval	162	1	20	0.994	0.890	**0.939**

MutationFinder achieves higher Normalized Mutation F-measure than reported for the previous mutation extraction systems (88% MutationFinder versus a maximum of 86% for MutationGraB, see Tables 2.7 and 2.1). While scores in Tables 2.7 and 2.1 are not directly comparable because different test corpora were used, in an additional test of MutationFinder's automatically constructed rule set, MutationFinder was applied to the full text corpus recently published with Mutation GraB [63] (Table 2.8)[10]. For this analysis their three development and three test corpora were combined into a single test corpus. The Normalized Mutations metric discussed in this Chapter is equivalent to MutationGraB's *'Cited mutation'* metric. Since Lee *et al.* only present performance data on this corpus for

[10]This analysis was performed by my collaborator on this project, David Randolph.

the more ambitious task of extracting associated pairs of mutations and genes/proteins, a direct comparison with their system cannot be performed at this time. The nearly identical performance of MutationFinder on the Mutation GraB corpus and the MutationFinder corpus (Table 2.7) further suggests that MutationFinder's performance will scale well to unseen data.

The automated pattern generation approach presented here yielded 759 patterns. The frequency of these patterns in the literature roughly matched a power-law (Zipfian) distribution, and the resulting regular expressions in our `regex.txt` file (available on MutationFinder project webpage) are listed in order from most commonly matching to least commonly matching. While most of the very high frequency regular expressions were included in the manually generated pattern set, it is compilation of the lower-frequency patterns that is truly valuable. These patterns *could* be compiled manually, but the time required would be prohibitive.

Table 2.8: Performance of MutationFinder on the MutationGraB corpus.
Counts of true positives (TP), false positives (FP), and false negatives (FN) are presented, along with precision (P), recall (R), and F-measure (F) for the three different performance metrics. This corpus did not annotate mutation mentions, so Extracted Mentions can not be calculated. All code and data, including a translation of the corpus that is compatible with our performance script, is available as part of the MutationFinder software package.

Automatically generated rules	TP	FP	FN	P	R	F
Normalized Mutations	3572	177	837	0.952	0.808	0.874
Document Retrieval	503	1	26	0.998	0.951	0.974

2.5.2 Interannotator agreement

1515 complete point mutation mentions were annotated in the 813 documents. Mean pairwise interannotator agreement (IAA), when comparing extracted mentions calculated on the fifty documents annotated by all three annotators, was 95%. Similar IAA results were obtained when calculating over the preprocessed vs. non-preprocessed data; preprocessing

does not appear to have an effect on IAA. We expect that the observed differences are not statistically significant. (See Table 2.9) The resultant gold standard corpus is available as part of the MutationFinder software package.

Table 2.9: Mean pairwise interannotator agreement.
Mean pairwise F-measure is presented, using three evaluation metrics, on the 25 documents in subcollection 1 (PREPROCESSED), the 25 documents in subcollection 2 (NOT PREPROCESSED) and the 50 documents in the union of subcollections 1 and 2 (ALL IAA).

Task	PREPROCESSED	NOT PREPROCESSED	ALL IAA
Extracted Mentions	0.949	0.951	0.950
Normalized Mutations	0.916	0.967	0.941
Document Retrieval	0.926	1.000	0.961

2.6 Conclusions

2.6.1 Precision limitations

MutationFinder achieves very high precision on all of the data sets on which it was tested. There were, however, several categories of residual errors. In some cases, MutationFinder extracts other entities, such as genes, proteins, or cell lines, whose names look similar to mutation mentions. Additionally, MutationFinder occasionally incurs false positives due to promiscuous pattern matching. For example, in the text *'the transfer of a proton from the catalytic cysteine to a His 207-Asp 205 diad via a system of ordered water molecules,'*[11] `H207D` is extracted as a point mutation because MutationFinder's pattern `WRES SPOS-MRES` is matched.

To quantify the problem of extracting gene or cell line names as mutations, lists of gene names and cell line names were generated and provided as input to MutationFinder[12]. A list

[11]Source PMID: 10841779

[12]This work was performed by my collaborator on the MutationFinder project, Bill Baumgartner.

containing 2,706,089 unique gene names was constructed by parsing out the symbol, official symbol, full name, and synonyms for each gene record in the Entrez Gene `gene_info`[13] file. A collection of 8102 unique cell line names was compiled from three online sources [78–80]. These lists were supplied as input to MutationFinder, and the number of gene and cell line names which MutationFinder erroneously identified as mutation mentions were counted. Only 679 (0.025%) of the gene names and 30 (0.370%) of the cell line names were identified as mutations by MutationFinder, leading to the conclusion that these false positives will be rare in practice.

The `H207D` example presented above represents a false positive mutation mention extracted from our development corpus. Promiscuous patterns do not appear to be overly problematic for MutationFinder at this stage, in part due to the post-hoc error analysis presented in Section 2.3.5.1. Since post-hoc error analyses of precision are not dependent on the availability of annotated data, in most cases problematic patterns should be easily identifiable and could be removed or modified, as was done with MutationFinder. An explicit fault model [81] is helpful for the sometimes surprisingly difficult task of error analysis. Cohen *et al.* [82] describes the use of a fault model in designing test suites for a biomedical named entity recognition system, and Johnson *et al.* [83] describes the design of a fault model for post hoc error analysis of ontology linking systems. A strength of our approach is that it is very easy to review, modify, and delete patterns. (We retained the pattern that caused the `H207D` error, but modified other patterns as discussed earlier.)

2.6.2 Recall limitations

MutationFinder does not achieve the full recall that it is capable of (with high precision maintained). An error analysis was performed by reviewing all false negatives incurred by MutationFinder on the development corpus at the document-retrieval level. Errors fell into four categories, and in some circumstances a single error fell into more than one category.

[13] ftp://ftp.ncbi.nih.gov/gene/DATA/gene_info.gz

First, in five cases, mutations were missed due to the presence of *intervening text* in the patterns. For example, the text *His-554 of IIAMtl was mutated to glutamine*[14] describes the mutation `H554Q`. The first mutation pattern presented in Table 2.3d comes close to matching this mention, except that there is intervening text in the input: *of IIAMtl*. Second, in four cases, mentions were missed because patterns did not comply with the bootstrap rule (e.g., a mutation mention was split across sentence boundaries). Third, two errors arose due to the presence of compound complex coordinations[15] being used to describe several mutations at once. For example, if the patterns in Table 2.3e were applied to the text *Ala-42 was mutated to Gly, Phe, Leu, and Trp*, MutationFinder would only extract two of the four mutations (i.e., `A42G` and `A42F`). Finally, two more errors arose due to the exclusion of single-letter abbreviations when matching mutation mentions in all patterns except `WRESSPOSMRES`.

The first three of these limitations could be avoided by modifying MutationFinder's rule set, while the fourth represents an underlying difficulty in concept recognition. The problem of intervening text can be addressed by applying the method presented by Huang and colleagues [71] (by attempting to align patterns with text), or by syntactically analyzing the text and allowing optional extensions to the patterns. In the example provided above, specifying that a prepositional phrase may optionally be present in the pattern could fix the issue. While syntactic processing may be necessary to address these few remaining problems, the complexities introduced by its application (perhaps even increasing the complexity of the pattern language beyond regular expressions) can at best address a relatively small number of errors. Limitations arising from the bootstrap rule also could be addressed. In the error analysis, two of these four errors arose because a mutation mention was split across two sentences. By rebuilding raw patterns, but allowing for occurrences of residues and sequence positions in neighboring sentences, patterns could be obtained that would match these mentions. Next, by adding an additional syntactic processing step to Step 4, compound

[14]Source PMID: 16443929

[15]Phrases where a noun phase is meant to be associated with each element in a list.

complex coordinations could be generalized to allow a variable number of intermediate residue mentions. The mentions missed due to single-letter abbreviated residue mentions, however, represent a more inherent problem: the trade-off between precision and recall that must be made when developing a concept recognizer. As discussed in Section 2.3.5.3, a large precision hit is incurred by matching single-letter-abbreviated residue mentions, while only a small increase in recall is achieved. Without investing a large amount of time to investigate when single-letter-abbreviations should be accepted, mentions of this type would always be missed by MutationFinder to maintain precision.

If higher recall is required than is initially achieved when applying the rule-learning methodology to other tasks, error analyses could be performed to identify false negatives, as was done here. Addressing those that cause the most significant problem first should improve recall at the expense of additional person hours for system development. Recall optimization however, will often require some sort of annotated test data. If resources are unavailable for corpus development, Craven and Kumlien [84] recommend approaches for rapidly developing *weakly annotated data*. Based on the error analysis presented here, most false negatives appear avoidable through additional syntactic and other processing in *Step 4*. In many applications recall optimization should not be necessary. This is discussed further in Section 2.6.4.

2.6.3 Generating patterns for other concept recognition tasks

To apply the methodology presented here to develop patterns for extracting different types of events or relationships, raw patterns would need to be compiled with an alternate bootstrap rule. For example, I briefly examine the problem of extracting descriptions of *protein transport events*[85] from biomedical literature. Imagine that we would like to design a system to extract a `TRANSPORTED-PROTEIN`, a `SOURCE` *and/or* a `DESTINATION` from unseen text. The bootstrap rule could specify that if a protein and a cellular location are mentioned in a single sentence, a protein transport event may have been described. Named

entity recognizers could be applied to identify sentences complying with this rule. Both a protein-name recognizer and a cellular-location recognizer would be required to match at least once. Raw patterns could then be generated from complying sentences and manually converted into *transport patterns*. Example texts and patterns are presented in Table 2.10. This example differs from MutationFinder to illustrate how the system could be modified to handle extraction of incomplete events (see Table 2.10b,c). Recognizing mutations in text is relatively simple compared to recognizing protein transport events, and applying this approach to more complex concept recognition problems may require more work or result in lower performance.

Table 2.10: Hypothetical patterns for a *protein transport event* extraction system.
These examples were manually obtained from GeneRIFs describing protein transport events.

	Input text (Entrez Gene ID)	**Pattern**
a	***Bcl-xL*** *from the* ***cytosol*** *to the* ***outer mitochondrial membrane*** (598)	`TRANSPORTED-PROTEIN` from the `SOURCE` to the `DESTINATION`
b	***MDMX*** *is transported to the cell* ***nucleus*** *upon DNA damage* (4194)	`TRANSPORTED-PROTEIN` is transported to the cell `DESTINATION`
c	***Stau2(59)*** *is exported from the* ***nucleus*** *by two distinct pathways* (171500)	`TRANSPORTED-PROTEIN` is exported from the `SOURCE`

2.6.4 Optionality of recall optimization

MutationFinder's precision and recall were optimized to construct a high-performance mutation recognition system. The findings presented in this chapter are consistent with the hypothesis that the construction of rule sets for mutation recognition is largely automatable. I conclude that this approach, being recall-oriented, does not require recall optimization, although it can be helpful. Precision optimization, on the other hand, is necessary. This is convenient: precision optimization can be performed inexpensively via post-hoc analyses,

while recall optimization typically requires more costly human-annotated data.

MutationFinder achieves high recall prior to optimization (Table 2.5a,c). While an error analysis has suggested four causes of false negatives, for many practical applications expending additional effort to optimize recall may not be worthwhile, especially if full-text will be provided to the concept recognition system rather than abstracts alone. The evaluations of MutationFinder involved collections of abstracts or GeneRIFs as corpora. In a follow-up study (presented in Section 3.2.3.2), when the performance of MutationFinder is compared on abstracts versus full-text, recall was greatly improved by the presence of the full text. When a mutation is the subject of an article, it is usually mentioned more than once. With each additional mention of a specific mutation, the chances of it being matched by a pattern increases. As full text continues to become more available, concept recognitions systems should be applied to full text rather than abstracts alone[16]. Additionally, for most practical concept recognition applications, extracting at least one instance of each targeted event is sufficient (i.e., the Normalized Mutations task is more important than the Extracted Mentions task).

Taken together, the already high recall achieved by this approach, the increasing availability of full text, and the generally greater significance of extracting normalized information as opposed to individual mentions, supports the claim that recall-optimization (with human-annotated data) of systems developed by our approach is not required.

[16]It should be noted that precision may be adversely effected by the presence of more text, simply because there are more places to go wrong. In the test of MutationFinder on the Mutation GraB corpus, there was a slight decrease in precision when MutationFinder was tested on full text, compared with the precision on abstracts in our test collection. In the abstracts versus full-text comparison presented in Section 3.2.3.2 there was no decline in precision, but that corpus was small in comparison to the MutationGraB corpus, so may be a less accurate evaluation.

2.6.5 Associating mutation mentions with genes or proteins

MutationFinder extracts mentions of point mutations from literature with high precision and recall. An even more useful system would associate mutations with genes or proteins. Work is currently being done to address this, and the pattern learning approach presented here should scale well to this problem. By using MutationFinder as an entity-recognizer, and employing a high-performance gene/protein name recognizer, pattern can be learned for extracting these associations from literature. In addition to providing a useful concept recognition system, this future work will provide insight on automatically learning patterns to extract associated entity mentions from text.

2.6.6 Applications of MutationFinder

Several groups have used MutationFinder as a component to help achieve broader text mining goals [86–89]. Additionally, in several collaborative projects, MutationFinder has been applied to compile mutations which have been described in the literature for specific protein families. In the next chapter, I describe an application of MutationFinder to annotate mutant proteins in the PDB. The study described therein allows for an additional evaluation of MutationFinder which illustrates some practical limitations of both automated and manual mutation annotation, and discusses how MutationFinder might be applied to improve the quality of database annotation.

CHAPTER III

EVALUATION OF TEXT MINING TOOLS ON REALISTIC TASKS

3.1 Background

In Chapter II I introduced MutationFinder and an approach to rapidly develope similar concept recognition systems. When MutationFinder was evaluated on blind human-annotated test data, it performed very well (see Figures 2.7 and 2.8). The high performance of MutationFinder and similar tools suggests that the possibility of applying these systems to automate biological database construction or annotation may be becoming practical.

As was done in Chapter II, biomedical text mining systems are generally evaluated intrinsically—for example, against a gold standard data set with named entities that are tagged by human annotators, judging the system on its ability to replicate the human annotations. Systems are less commonly evaluated extrinsically—i.e., by measuring their contribution to the performance of some task. Intrinsic evaluations of text mining tools are critical to accurately assessing their basic functionality, but they do not necessarily tell us how well a system will perform in practical applications. In this Chapter I present an extrinsic evaluation of MutationFinder, and two additional approaches to annotate mutations in a database: the first is a sequence-alignment-based approach, and the second is human (author) annotation.

Hunter and Cohen [1] list four text mining systems that are being or have been used to assist in the population of biological databases (LSAT [90], MuteXt [28], Textpresso [33], and PreBIND [91]). Of these four, data on the actual contribution of the tool to the database curation effort is available for only one: the PreBIND system is reported to have reduced the time necessary to perform a representative task by 70%, yielding a 176-person-day time

[17]The work presented in this chapter was originally presented in: *Intrinsic evaluation of text mining tools may not predict performance on realistic tasks* (Pacific Symposium on Biocomputing 2008, V13 Pg. 640-651).

savings. More recently, Karamanis *et al.* [92] recorded granular time records for a "paper-by-paper curation" task over three iterations in the design of a curator assistance tool, and noted that curation times decreased as user feedback was incorporated into the design of the tool. In the information retrieval (IR) domain, Hersh *et al.* [93] assessed the ability of an IR tool (Ovid) to assist medical and nurse practitioner students in finding answers to clinical questions, and found that performance of the system in intrinsic evaluation did not predict the ability of the system to help users identify answers. Some tasks in the recent BioCreative shared tasks (particularly the GO code assignment task in BioCreative 2004, PPI task in BioCreative 2006, and the GN tasks in both years), and to a lesser extent, of the TREC Genomics track in some years, can be thought of as attempts at extrinsic evaluations of text mining technologies. Camon *et al.* [94] gives an analysis of the shortcomings of the text mining systems that participated in the BioCreative 2004 GO code assignment task. We are not aware of any work that directly assesses the ability of an automated technique to recreate a large, manually curated data set, although the importance of such evaluations has been noted [24].

As reviewed in Section 2.1.2, there has recently been much interest in the problem of automatically identifying point mutations in text [28, 29, 39, 40, 63–65, 95]. Briefly, comprehensive and accurate databases of mutations that have been observed or engineered in specific biological sequences are often extremely valuable to researchers interested in those sequences, but because the requisite information is generally dispersed throughout the primary literature, manually compiling these databases requires many expert hours.

The availability of MutationFinder allows us to ask subsequent questions. First, how effective are manual biological database annotation techniques in terms of accuracy and coverage; and second, does the performance of an automated annotation technique in intrinsic evaluation predict the performance of that system in an extrinsic evaluation? The first question addresses the issue of whether replacement or augmentation of manual database

annotation methods with automatic methods is worth exploring, while the second addresses whether the most commonly performed evaluations of automatic techniques translate into information regarding their applicability to real-world tasks.

To address these questions, we compare and evaluate three approaches for annotating mutations in Protein Data Bank (PDB) [15] entries[18]: manual annotation, which is how mutations are currently annotated in the PDB; and two automatic approaches—text-mining-based annotation using MutationFinder, and alignment-based annotation—which are being explored as possibilities to replace or augment manual annotation. (The PDB is the central repository for 3D protein and nucleic acid structure data, and one of the most highly accessed biomedical databases.) In the following section I present methods to address these questions and the results of my analyses. Problems are identified with all of the annotation approaches, automatic and manual, and I conclude with ideas for how to best move forward with database annotation to produce the best data at the lowest cost.

3.2 Methods and results

In this section the methods and results of three experiments are presented: first, an evaluation of the accuracy and comprehensiveness of the manual mutation annotations in the Protein Data Bank; next, an extrinsic evaluation of two automated techniques by comparing their results with the manually deposited mutation data in the PDB; and finally, a comparison of MutationFinder's performance when run over abstracts and full text to address the hypothesis, mentioned briefly in Section 2.6.4, that MutationFinder's low recall in extrinsic evaluation is a result of the lack of information in article abstracts. Unless otherwise noted, all evaluations use a snapshot of the PDB containing the 38,664 PDB entries released through 31 December 2006. All data files used in these analyses are available

[18]Entries in the PDB are composed of atomic cartesian coordinates defining the molecular structure, and metadata, including primary sequence(s) of the molecule(s) in the structure, mutations, primary citation, structure determination method, etc.

via the MutationFinder project webpage.

3.2.1 Evaluation of manual annotations

When a structural biologist submits a structure to the PDB, they are asked to provide a list of any mutations in the structure. Compiling this information over all PDB entries yields a collection of manually annotated mutations associated with PDB entries, and this mapping between PDB entries and mutations forms the basis of this analyses. The accuracy of these annotations are evaluated by comparing mutation field data with sequence data associated with the same entries. The completeness of this data is evaluated by looking for PDB entries which appear to describe mutant structures but do not contain data in their mutation fields.

3.2.1.1 Manual mutation annotations

Manual mutation annotations were compiled from the *mutation field* associated with PDB entries. The mutation field is a free-text field in a web-based form that is filled in by researchers during the structure deposition process. The depositor is expected to provide a list of mutations present in the structure (e.g., 'Ala42Gly, Leu66Thr'[19]), but the information provided is not always this descriptive. For example, many mutation fields contain indecipherable information, or simply the word *yes*. In cases where the depositor does not provide any information in the mutation field (as is often the case), differences identified by comparison with an aligned sequence are suggested to the author by a PDB annotator. The author can accept or decline these suggestions.

[19]This is a common format for describing point mutations, which indicates that alanine at position 42 in the sequence was mutated to glycine, and leucine at position 66 was mutated to threonine.

3.2.1.2 Normalization of manual mutation annotations

Because the mutation field takes free text input, automated analysis requires normalization of the data. This was done by applying MutationFinder to each non-empty mutation field. Point mutations identified by MutationFinder in a mutation field were normalized. To evaluate this normalization procedure, a non-author biologist manually reviewed a random subset (n=400) of non-empty mutation fields and the normalized mutations output by MutationFinder. Precision of the normalization procedure was 100.0%. Recall was 88.9%. This high performance is not surprising, since the task was relatively simple. It suggests that normalizing mutation fields with MutationFinder is acceptable. 10,504 point mutations in 5971 PDB records were compiled by this approach. This data set is referred to as the *manually deposited mutation annotations*.

3.2.1.3 Accuracy of manually deposited mutation annotations

To assess the accuracy of the manually deposited mutation annotations, each mutation was validated against the sequence data associated with same PDB entry. (This process is similar to that employed by the MuteXt [28] system.) If a mutation could not be validated against the sequence data, that entry was considered to be inaccurately annotated and was reported to PDB. (Note that this discrepancy could indicate an error in the sequence, an error in the mutation annotation, or a mismatch in sequence numbering.)

Validation of mutations against sequence data was performed as follows. Sequences were compiled for all PDB entries. For a given entry, the annotated sequence position was checked for the presence of putative mutant residue. For example, PDB entry 3CGT is annotated with the mutation `E257A`. The sequence associated with 3CGT was scanned to determine if alanine, the mutant residue, was present at position 257. In this case it was, so the annotation was retained. If alanine were not present at position 257, the annotation would have been labelled as invalid. In cases where PDB entries contain multiple sequences

(e.g., a protein composed of several polypeptide chains), each sequence was checked for the presence of the mutant residue.

3.2.1.4 Coverage of manually deposited mutation annotations

To assess the coverage of the manually annotated data, an attempt was made to identify PDB records of mutant structures that did not contain data in their mutation field. To identify records of mutant structures, PDB entry titles were searched for any of several keywords that suggest mutations (case insensitive search query: `muta* OR substitut* OR varia* OR polymorphi*`). MutationFinder was also applied to search titles for mentions of point mutations. If a title contained a keyword or mutation mention and the entry's mutation field was blank, the entry was labelled as insufficiently annotated. An informal review of the results suggested that this approach was valid.

3.2.1.5 Results of manual annotation evaluation

40.6% (4260/10504) of mutations mentioned in mutation fields were not present at the specified position in the sequence(s) associated with the same PDB entry. These inconsistencies were present in 2344 PDB entries, indicating that 39.3% of the 5971 PDB entries with MutationFinder-normalizable mutation fields may be inaccurately annotated. As mentioned, these inaccurate annotations could be due to errors in the mutation annotation or the sequence, or mismatches between the position numbers used in the mutation and the sequence. In the majority of cases, the errors likely arise from mismatches in numbering, as there is generally some confusion in how mutations should be numbered (i.e., based on the sequence in the structure or based on the UniProt reference sequence). PDB entries now contain mappings between the structure and UniProt sequences, and in a future analysis these mappings could be used to determine if any of these apparent errors are instead inconvenient discrepancies which could be avoided automatically.

Additionally, 21.7% (1243/5729) of the PDB entries that contained a mutation keyword

or mention in the title were found to contain an empty mutation field. These entries appear to be underannotated. (As a further indication of the scope of the "underannotation problem," note that 12.9% (1024/7953) of the non-empty mutation fields simply contain the word *yes.*) Again, this is likely to be an overestimate of the true number of underannotated PDB entries (due to promiscuity of the search query), but even if this is overestimated by a factor of 10, underannotation is still a problem.

These results suggest that the manually deposited mutation data is far from perfect, and that not just the quantity, but the *quality* of manual database annotation stands to be improved. In the next section, automated techniques for mutation annotation in the PDB are explored to determine if they may provide a means to replace or augment manual annotation. These automated techniques are evaluated against the manually deposited mutation annotations, although it has just been shown that it is far from perfect. Performance of the automated techniques is therefore underestimated.

3.2.2 Automated mutation annotation evaluated extrinsically

In this section two automated mutation annotation techniques are evaluated by assessing their ability to reproduce the manually deposited mutation annotations in the PDB. The first automated method, text mining for mutations using MutationFinder, has been shown to perform very well on blind test data (i.e., in intrinsic evaluation). The second approach, detecting differences in pre-aligned sequences, is not inherently error-prone and therefore does not require intrinsic evaluation. We might expect that the near-perfect and perfect abilities of these systems (respectively) to perform the basic function of identifying mutations would suggest that they are capable of compiling mutation databases automatically. Assessing their ability to recreate the manually deposited mutation annotations allows for evaluation of this expectation.

3.2.2.1 Text-mining-based mutation annotation: MutationFinder

Two sets of MutationFinder mutation annotations were generated—with and without the sequence validation step described in Section 3.2.1.3. The unvalidated data should have higher recall, but more false positives. To compile the unvalidated MutationFinder annotation set, MutationFinder was applied to primary citation abstracts associated with PDB records. For each record in the PDB, the abstract of a primary citation (when both a primary citation and an abstract were available) was retrieved, and MutationFinder was applied to extract normalized point mutations. 9625 normalized point mutations were associated with 4189 PDB entries by this method, forming the *unvalidated MutationFinder mutation annotations*. To compile the *validated MutationFinder mutation annotations*, sequence validation was applied to the unvalidated MutationFinder mutation annotations. This reduced the results to 2602 normalized mutations in 2061 PDB entries.

3.2.2.2 Alignment-based mutation annotation

Validated and unvalidated data sets were also compiled using an alignment-based approach. Sequences associated with PDB entries were aligned with UniProt sequences using `bl2seq`. Differences between aligned positions were considered point mutations, and were associated with the corresponding entries. The alignment approach yielded 23,085 normalized point mutations in 9807 entries (the *unvalidated alignment mutation annotations*[20]). Sequence validation[21] reduced this data set to 14,284 normalized mutations in 6653 entries (the *validated alignment mutation annotations*).

[20]This data set was compiled by my collaborator on this project, Nita Deshpande.

[21]Sequence validation was somewhat redundant in this case, but was included for completeness. Surprisingly, it was not particularly effective here. The positions assigned to mutations in this approach were taken from the aligned UniProt sequence when sequence start positions did not align perfectly, or when the alignment contained gaps. This resulted in different position numbering between the manually- and alignment-produced annotations, and reduced performance with respect to the manual annotations.

3.2.2.3 Extrinsic evaluation of automated annotation data

To assess the abilities of the MutationFinder- and alignment-based annotation techniques to recreate the manual annotations, mutation annotations generated by each approach were compared with the manually deposited mutation annotations in terms of precision, recall, and F-measure using the `performance.py` script (available as part of the MutationFinder package). Two metrics were scored: *mutant entry identification*, which requires that at least one mutation be identified for each mutant PDB entry, and *normalized mutations*, which requires that each manually deposited mutation annotation associated with a PDB entry be identified by the system. *Mutant entry identification* measures a system's ability to identify structures as mutant or non-mutant, and is the corollary of the *document retrieval* metric presented in Section 2.2, while *normalized mutations* measures a system's ability to annotate the structure with specific mutations.

Normalized mutations were judged against the manually deposited mutation annotations, constructed as described in Section 3.2.1.1. This set contained 10,504 mutations in 5971 PDB records from a total of 38,664 records. As noted earlier, many non-empty mutation fields do not contain mutations (e.g., when they contain only the word *yes*). However, in the vast majority of cases, a non-empty mutation field indicates the presence of mutations in a structure. I therefore constructed a different data set for scoring mutant entry identification. This *manually curated mutant entry* data set was constructed from all PDB entries which contained non-empty mutation fields, and contained 7953 entries (out of 38,664 entries in the PDB snapshot).

3.2.2.4 Extrinsic evaluation results

The utility of each automated techniques (and combinations of both) was evaluated for identifying mutant PDB entries (*mutant entry identification*, Table 3.1a) and annotating mutations associated with PDB entries (*normalized mutations*, Table 3.1b).

Table 3.1: Performance of six automated methods for identifying mutant PDB entries.

Six automated methods for identifying mutant PDB entries are assessed against (a) manually curated mutant entry data, and (b) manually deposited mutation annotations. True positives (TP), false positives (FP), false negatives (FN), precision (P), recall (R), and F-meaure (F) are presented. Sequence validation (methods 2 and 4) is described in Section 3.2.1.3.

	(a) Mutant Entry Id.	TP	FP	FN	P	R	F
1	MutationFinder	2690	1499	5263	0.642	0.338	0.443
2	MutationFinder + validation	1665	396	6288	**0.808**	0.209	0.333
3	Alignment	6079	3728	1874	0.620	**0.764**	0.685
4	Alignment + validation	4104	2549	3849	0.617	0.516	0.562
5	2 *and* 3	1403	275	6550	**0.836**	0.176	0.291
6	2 *or* 3	6258	3816	1695	0.621	**0.787**	0.694

	(b) Normalized Mutations	TP	FP	FN	P	R	F
1	MutationFinder	2575	7050	7929	0.268	0.245	0.256
2	MutationFinder + validation	1803	799	8701	**0.693**	0.172	0.275
3	Alignment	7681	15404	2823	0.333	**0.731**	0.457
4	Alignment + validation	5059	9225	5455	0.354	0.482	0.408
5	2 *and* 3	1584	532	8920	**0.749**	0.151	0.251
6	2 *or* 3	7900	15671	2604	0.335	**0.752**	0.464

On both metrics, the highest precision results from the intersection of the validated MutationFinder mutation annotations (method 2) and the unvalidated alignment mutation annotations (method 3) data, while the highest recall results from the union of these. Generally, method 2 achieves high precision, and method 3 achieves high recall. None of these approaches achieves a respectable F-measure, although as was pointed out in Section 3.2.1.5, these performance values are likely to be underestimates due to noise in the manually deposited mutation annotations.

3.2.3 MutationFinder applied to abstracts versus full-text

Table 3.1 shows that MutationFinder (with and without validation) achieves very low recall with respect to the manually deposited mutation annotations. I next evaluated the hypothesis that this was a result of mutations not being mentioned in article abstracts, but rather only in the article bodies. A PDB snapshot containing the 44,477 PDB records released through 15 May 2007 was used for this analysis.

3.2.3.1 Compiling and processing full-text articles

PubMed Central was downloaded through 15 May 2007. XML tags and metadata were stripped. All articles were searched for occurrences of a string matching the format of a PDB ID. (IDs are four characters long: a number, a letter, and two letters or numbers, e.g. 3CGT.) If such a string was found, it was compared to a list of valid PDB IDs; if the string matched a valid PDB ID, the article was retained. This returned 837 articles. From this set, articles that were primary citations for PDB structures were selected, resulting in a set of 70 PDB entries (with 13 manually annotated mutations) for which full text was available[22].

3.2.3.2 Comparing abstracts versus full text

MutationFinder with sequence validation (as described in Section 3.2.1.3) was applied to the abstracts and full-text articles, yielding two mutation data sets. The results were compared against the manually annotated mutation data, allowing for direct assessment of the contribution of the article bodies to MutationFinder's performance.

3.2.3.3 Abstract versus full text results

A 10-fold increase in recall was observed when the article body was provided to MutationFinder in addition to the abstract, with no associated degradation of precision (Table

[22]The full text articles were compiled by my collaborator on this project, Lynn Fink.

Table 3.2: MutationFinder applied to abstracts versus full text.

MutationFinder with sequence validation was applied to abstracts and full text (abstract + article body) for 70 PDB entries. Results are compared with manually annotated data. True positives (TP), false positives (FP), false negatives (FN), precision (P), recall (R), and F-measure (F) are presented, describing each approach's ability to replicate manually curated data.

Metric	Input	TP	FP	FN	P	R	F
Normalized Mutations	Abstracts	1	0	12	1.000	0.077	0.143
	Full text	10	0	3	1.000	0.769	0.870
Mutant Entry Id.	Abstracts	1	0	9	1.000	0.100	0.182
	Full text	7	0	3	1.000	0.700	0.824

3.2). While 70 PDB entries with 13 mutations is a small data set, these data strongly suggest that access to full text is critical for automated mutation annotation by text mining tools. When sequence validation was not applied, normalized mutation and mutant entry identification recall were perfect, but precision was 11.7% and 38.5%, respectively.

3.3 Conclusions

These experiments address two questions. First, how effective are manual biological database annotation techniques in terms of accuracy and coverage; and second, does the performance of an automated annotation technique in intrinsic evaluation predict the performance of that system in an extrinsic evaluation? I now present my conclusions regarding these questions, and discuss their implications for database curation.

3.3.1 Reliability of mutation annotation approaches

The manual *and* automatic approaches to annotating mutations appear to yield significant Type I (precision-related) and Type II (recall-related) errors when analyzed on the PDB as a whole. This suggests that these methods may be insufficient to generate the required quality and quantity of annotation that is necessary to handle the barrage of data

in the biomedical sciences.

Manual annotation of PDB entries is error-prone, as illustrated by the sequence-validation of these data described in Section 3.2.1.5, and does not guarantee complete annotation. (It should be noted that many of the results that are classified as errors in the manually annotated data are likely to be due to sequence numbering discrepancies. Mappings between PDB sequences and UniProt sequences in the PDB can be used to identify these, and in a future analysis these mappings could be used to reevaluate the manually annotated data.)

The automated mutation annotation approaches also appear to have limitations. MutationFinder (with validation against sequence data) performs well, but full text is probably required for any text mining approach to achieve sufficient recall. Conversely, the alignment-based approach is comprehensive, but overzealous. The manual and automatic methods do frequently validate and complement one another (data not shown)—in parallel, they may provide a means for improving the quality, while reducing the cost (in person-hours), of database annotation.

3.3.2 MutationFinder: Intrinsic versus extrinsic evaluations

In an intrinsic evaluation against blind gold standard data, MutationFinder achieved 97.5% precision and 80.7% recall on normalized mutations extraction, and 99.4% precision and 89.0% recall on document retrieval [39, 40]. In the extrinsic evaluation against manually deposited mutation annotations in the PDB, the exact same system achieved 26.8% precision and 24.5% recall for normalized mutation extraction, and 64.2% precision and 33.8% recall for mutant entry identification (the equivalent of document retrieval in this work). While these are likely to be underestimates of the true utility (Section 3.2.1.5), the large difference in performance cannot be explained completely by the imperfect extrinsic evaluation. The discrepancy appears to be chiefly explained by two factors: introduction of a systematic source of false positives, and missing data. These issues illustrate that accurately and

comprehensively pulling desired information from text is just the beginning of deploying a text mining system as a database curation tool.

False positives were systematically introduced when a single article was the primary citation for several PDB entries, and MutationFinder associated all mutations mentioned in the article with all the citing entries. For example, PDB entries 1AE3, 1AE2, and 1GKH all share the same primary citation (PMID: 9098886). This abstract mentions five mutations, all of which MutationFinder associates with each of the three PDB entries. Each of the structures contains only one of the five mutations, so four false positives are incurred for each entry. (The other two mutations referred to are in other structures.) Sequence validation eliminated all of these false positives while retaining all of the true positives, and overall improved normalized mutation precision by 42.5 percentage points with an associated degradation in recall of 7.4 percentage points.

False negatives were most common when the targeted information was not present in the primary citation abstracts. The abstract versus full text analysis showed that processing the full text with MutationFinder plus sequence validation resulted in a nearly 70 percentage point increase in recall, with no precision degradation. These data result from an analysis of only a small subset of the PDB, but they clearly illustrate the importance of full text for high-recall mutation mining.

While it is an essential step in building a text mining system, evaluating a system's performance on gold standard data (intrinsic evaluation) is not necessarily indicative of its performance as a database curation tool (extrinsic evaluation). Identifying and gaining access to the most relevant literature, and identifying and responding to sources of systematic error, are central to duplicating the performance observed on a (well-chosen) gold standard data set in practical applications.

3.3.3 Alignment-based mutation annotation: Extrinsic evaluation

Compiling differences in aligned sequences is not inherently error prone, unlike text mining—beyond software testing to avoid programming errors, no intrinsic evaluation is necessary. However, this method does not perform perfectly for annotating mutations in the PDB, but rather achieves high recall with low precision.

Error analysis suggests that the primary cause of both false positives and false negatives obtained by alignment-based mutation annotation with respect to the manually deposited mutation annotations is differences in sequence position numbering between the PDB sequence and the UniProt sequence. In PDB entry 1ZTJ, for example, the authors annotated mutations S452A, K455A, T493A, and C500S, while sequence comparison identified S75A, K78A, T115A, and C123S. The (almost identical) relative sequence positions and wild-type and mutant residues suggest that these are the same mutations, but the sequence position offset results in four false positives and four false negatives. Utilizing the mappings between PDB sequence positions and UniProt sequence positions in the PDB should help to alleviate these discrepancies in position numbering. This will be explored in future work, and is expected to significantly reduce these types of errors.

False positives additionally occur as a result of slight differences in the sequence of the solved structure and the closest sequence in UniProt. Differences in the sequences are not *necessarily* mutations induced for analysis, and are therefore not annotated as such. For example, sequence comparison identified six mutations in the PDB entry 1QHO, and the primary citation authors acknowledge several of these as sequence 'discrepancies.' False negatives can also occur when a sequence cannot be aligned to a UniProt sequence, and the alignment-based method cannot be applied, or alternatively if inaccurate information was provided by the depositor. For example, PDB entry 1MWT is annotated with the Y23M mutation, but valine is present at position 23 in the associated sequences. In this case the classification as false negative is an artifact of a problematic manual annotation, rather than

a statement about the performance of the annotation technique.

CHAPTER IV

DETECTING COEVOLUTION WITHOUT PHYLOGENETIC TREES

4.1 Background

In Section 1.3.2 I introduced the idea of coevolution in biological sequences, and noted several applications of algorithms that identify coevolving pairs of positions in multiple sequence alignments. In this Chapter I change focus from biomedical text mining to comparing and evaluating algorithms designed to identify pairs of coevolving positions in multiple sequence alignments of proteins.

Coevolutionary analyses have frequently been performed on one or a few protein families. However, just as The Adaptive Evolution Database [96] allows proteome-wide studies of evolutionary rates, proteome-wide studies of coevolution could also be performed if sufficiently fast and well-characterized methods for detecting coevolution were available.

As biological sequences are the product of an evolutionary process, it intuitively makes sense that the accuracy of analyses of the historical processes affecting them will be improved by explicit representation of those historical processes. Incorporation of phylogenetic information has benefited diverse classes of bioinformatics algorithms, including multiple sequence alignment [42], comparison of microbial communities [97], and functional annotation of genes [98]. Accordingly, incorporation of phylogenetic knowledge to control for patterns in biological sequence data that arise from ancestry is regarded as essential for coevolutionary analyses and best achieved by directly incorporating the phylogeny in the metric [54, 99, 100].

Many coevolution algorithms ('tree-aware' methods) have explicitly attempted to control for phylogeny (e.g., [51–53, 56, 99, 101, 102]), while others ('tree-ignorant' methods)

[23]The work presented in this chapter was originally presented in: *Detecting coevolution without phylogenetic trees? Tree-ignorant metrics of coevolution perform as well as tree-aware metrics.* (BMC Evolutionary Biology 2008, 8(1):327).

have implicitly assumed a star phylogeny (e.g., [57, 59, 103, 104]). Drawbacks have been identified for both approaches. Likelihood based tree-aware methods have the disadvantage of being sensitive to model mis-specification, a property common to all likelihood methods, and generally have a much longer compute time than tree-ignorant methods. Tree-ignorant methods are thought to have decreased specificity due to confounding of correlations arising from selective pressure with correlations arising from shared ancestry represented by the phylogeny [56, 105, 106]. Past evaluations of the effect of tree topology on the performance of coevolution algorithms have used simulated data, and have confirmed that non-star tree topologies can cause false positives [103, 105, 107, 108].

Clearly, controlling for shared ancestry is essential but approaches that do so without explicitly representing the phylogenetic tree are possible. In this chapter I test the hypothesis that a tree-ignorant statistic can be informative if it is compared to a distribution of the same statistic with the same embedded ancestry but variable in coevolution. In this case, the shared ancestry origin of correlated evolution dominates the background distribution. A greater magnitude of correlated evolution than this background is thus evidence of coevolution.

An additional consideration for estimating coevolution is that encoding protein alignments with reduced-state amino acid alphabets reduces computational complexity, and may also increase statistical power [52, 54, 103, 109]. In a reduced-state alphabet, the twenty amino acids are collapsed to a smaller number of states. For example, a three-state 'charge' alphabet can be achieved by treating His, Lys, and Arg as the 'positively charged' state; Asp and Glu as the 'negatively charged' state; and, all other residues as the 'uncharged' state. The recoding chosen for a group of sequences constitutes an explicit hypothesis concerning the primary biochemical property subjected to coevolutionary pressures by natural selection. A coevolution algorithm applied to sequences with fewer states should have more power to identify pairs of positions which coevolve as a result of the physicochemical

property being modeled (e.g., charge) because variability within each state is hidden. Information is lost in recoding to a reduced-state alphabet however, so the power for detecting coevolution arising from other properties of amino acid residues (e.g., side-chain volume) decreases. The sensitivity of coevolution algorithm performance to the alphabet choice is unclear.

Evaluations of coevolution algorithms on simulated data elucidate the strengths and limitations of the algorithms, but are forced to rely on simplifying assumptions about the biological systems being modeled. Evaluations on biological data are therefore important for understanding how an algorithm will perform under more realistic circumstances. Biologically relevant evaluations are difficult however, because we have little knowledge of when sequence positions truly coevolve, and therefore do not have a good idea of what the true positives are.

Different approaches have been employed to define coevolutionary positive-controls. Individual cases of coevolution are directly supported by observation of compensated pathogenic deviations (CPDs), or variants known to cause disease in one species but which are present as wild-type in another species [110]. CPDs have been reported for both RNA and protein coding genes and does exhibit strong statistical evidence for coevolution [109]. The suitability of this class of variation for examining the properties of coevolution, however, is low for both practical and biological reasons: the number of cases for which there is sufficient data from related species is low; and, the identification of these variants as pathological suggests the selection coefficients operating on them is very strong and thus may not be representative of the strength of selection responsible for most coevolution.

An alternative approach has been to focus on structural influences likely to be subjected to natural selection. Past evaluations on (non-simulated) RNA alignments have treated base pairs as positive controls and all other pairs as negative controls [48, 49, 102, 106]. These have been useful for comparing algorithms on RNA, but it is not clear that performance on

RNA alignments translates well to performance on protein alignments because interactions between residues in proteins are generally more complex. Protein gold standards have been designed to evaluate a method's ability to identify residue contacts in tertiary structure by defining residue pairs within a certain C_β distance in a representative crystal structure as positive controls, and all other pairs as negative controls [52, 56, 102–104, 111–114]. The set of residue pairs within a small C_β distance in tertiary structure in a representative crystal structure is recognized as a coarse criteria because it is not clear that close physical proximity is an essential precondition for coevolution [114], and because a single crystal structure may not accurately describe the tertiary contacts in all sequences in the alignment.

To complement residue-contact-based comparisons, I developed a novel secondary-structure-based method for comparing coevolution methods where the known periodic stabilizing interactions between stacked residues in protein alpha helices are taken as positive controls. Double-mutant studies of protein alpha helices have shown that stacked residues in alpha helices interact to stabilize the helix [115–118]. Statistical analyses support these results by showing that the interactions are present in diverse families of alpha helices [52, 56, 119–121]. Stabilization is thought to result from ionic interaction, aromatic-aromatic interaction, or hydrogen bonding between stacked side chains. Although there has been discussion on the validity of these studies [122], biophysical and statistical analyses continue to support the case for stabilizing interactions. These interactions occur between the stacked positions in the alpha helix, or the positions separated by three residues in primary structure $(i, i+4)$ (where i refers to the sequence position), and to a lesser extent between positions separated by two residues $(i, i+3)$, corresponding to the 3.6 residue per turn periodicity of the alpha helix. Since interactions between stacked residues appear important for alpha helix stability, positions should coevolve to conserve these interactions. Methods for detecting coevolution in proteins should therefore identify stacked residues in alpha helices, as illustrated in [52, 108, 120, 123] providing a positive control for coevolution

detection algorithms. The purpose of this study is therefore to exploit the known regular structure of the helix as a gold standard for detecting coevolution: methods for detecting coevolution should, at minimum, be able to recapture these regularities.

In this chapter I report an assessment of the hypothesis that appropriately transformed tree-ignorant metrics have similar statistical power to tree-aware approaches by performing a systematic comparison of nine coevolution algorithms. Five of the algorithms — Mutual Information (MI), Normalized Mutual Information (NMI) [103], Resampled Mutual Information (RMI) [61][24], Statistical Coupling Analysis (SCA) [59], and Corrected Mutual Information (MIp) [104] — use multiple sequence alignments but no phylogenetic trees. The other four methods — LnLCorr [52, 54], Ancestral States (AS) [51, 99], the Generalized Continuous-Time Markov Process Coevolutionary Algorithm (GCTMPCA) [50, 53], and CoMap [102, 106] — use multiple sequence alignments and phylogenetic trees. The algorithms were compared by application to real (i.e., non-simulated) protein sequence alignments.

In a secondary study, our alignments are recoded in 52 different reduced-state amino acid alphabets to evaluate the utility of amino acid alphabets which model different chemical properties, and to test the hypothesis that alphabets with fewer states are generally better for detecting coevolution.

4.2 Methods

Five tree-ignorant methods and four tree-aware methods for detecting coevolving positions were compared on four multiple sequence alignments using the full amino acid alphabet and, when applicable, 52 reduced amino acid alphabets. Two of the alignments, tetrapod myoglobin (42 sequences, 153 positions) and chordate myosin rod (114 sequences, 1064 positions), represent mostly alpha helical protein sequences and thus the positive controls.

[24]RMI was developed by my collaborators on this project, Brett Easton and Gavin Huttley, and was initially presented in the published version of this study.

The other two alignments are matched negative controls generated by shuffling the order of positions in each observed alignment to remove all structural information.

The statistical significance of coevolution metrics between alignment columns separated by a specified distance was determined using the t-test transformation (Figure 4.1). Applying a coevolution algorithm to an alignment results in a 'coevolution matrix,' where each position in the lower triangle contains a pairwise coevolutionary score. Each coevolution matrix, constructed based on a combination of method, alignment, and alphabet, was evaluated to determine to what degree the combination allowed detection of the periodicity of the alpha helix. The distribution of scores from $(i, i+n)$ positions in a coevolution matrix were compared with the distribution of all other scores in the same matrix using a two-sample t-test [124]. Significant p-values at $n = 3$ or $n = 4$ were taken as evidence of a method's ability to detect coevolution, as these are the stacked positions in alpha helices which are expected to coevolve to retain alpha helix stability. Although the coevolutionary scores are not directly comparable between the methods, the t-test transformation standardizes the metrics, allowing evaluation of the relative performance of each method. I note here that although the distance matrix structure of the result matrices violates the independence clause of the t-test, the robustness of results were validated using a non-parametric matrix permutation test (described in Methods).

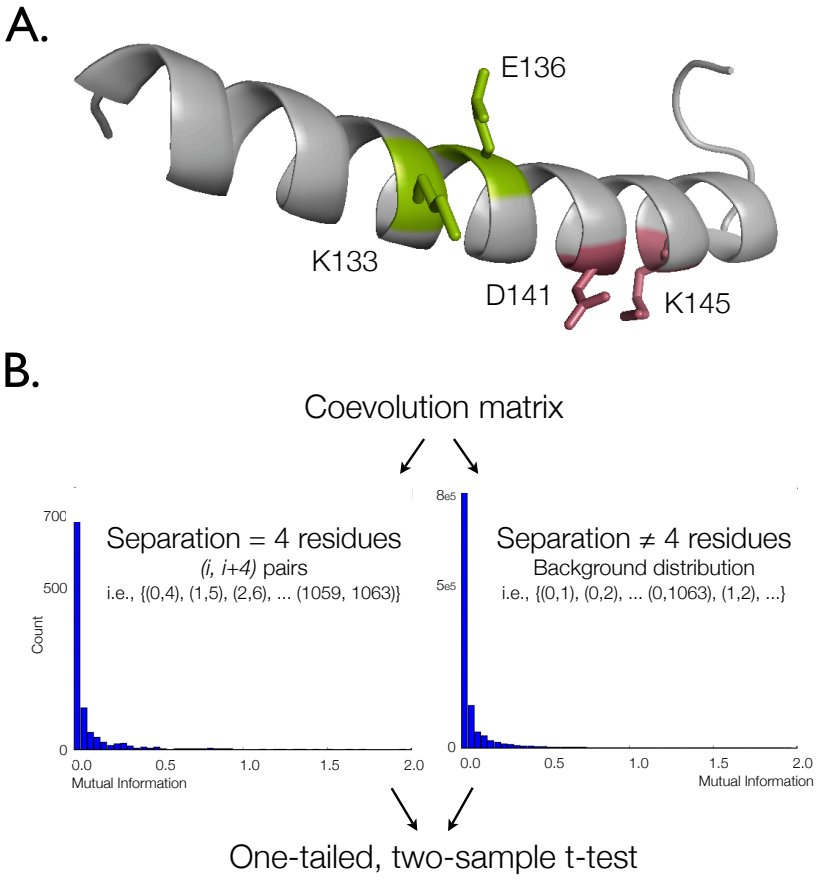
4.2.1 Coevolution detection methods: Tree-ignorant

4.2.1.1 Mutual information

Mutual information (MI) was calculated over all pairs of positions (columns) in the alignment, as described in [103]. MI is a measure of the degree to which knowing the value of one discrete random variable (in this case, the identity of the amino acid residue at a specific position in a protein) informs you of the value of another discrete random variable (the identity of the residue at another position). Pairs of positions with high MI scores are

Figure 4.1: Diagram of alpha-helix-based evaluation.

(A) Stacked positions in protein alpha helices have been observed to coevolve to maintain ionic interactions which stabilize the helix. Stacked positions are the $(i, i+3)$ and $(i, i+4)$ position pairs, corresponding to the 3.6 residue per turn periodicity of the alpha helix. Two such pairs are highlighted here (PDB: 1MBD). (B) The distribution of coevolution scores from $(i, i+n)$ position pairs are compared against a background distribution generated from all other position pairs in the same alignment using a one-tailed, two-sample t-test. The scale of the y-axes differ due to the large difference in the number of position pairs in each distribution. These t-test p-values were compared against p-values generated with other statistical tests to confirm the applicability of the t-test to these data (see Methods).



those that have undergone correlated substitution events. The MI calculation is discussed in detail in Section 5.1.1.

4.2.1.2 Normalized mutual information

Normalized mutual information (NMI) was calculated by dividing the MI score for each pair of alignment positions by the joint entropy of that pair of positions. This normalization, as described in [103], attempts to reduce the minimizing effect of higher sequence conservation on mutual information. For example, if two pairs of columns have perfectly correlated substitutions, but one pair is more highly conserved than the other, the more highly conserved pair will have a lower MI. Normalizing by joint entropy combats this: the pairs will have equal NMI. This method was reimplemented based on the description in [103]. As NMI is the primary method used in other chapters, a detailed presentation of the computation and features of this metric are discussed in Section 5.1.1.

4.2.1.3 Resampled mutual information

Resampled mutual information (RMI) is similar to the Jackknife technique [125] but avoids the complication arising from deletion of observations. The same resampling approach as reported previously for a tree-dependent probabilistic approach was applied [109] but here it is used to generate a null distribution of MI for data with arbitrary alphabet sizes. Note that when the branch lengths on a phylogeny are infinitely long, for a sample of n sequences the statistic from the approach of [109] is equivalent to n times MI (Easton and Huttley, unpublished). The resampling approach rescales MI by generating permutations of the data for the pair of aligned columns. The modified data sets are identical to the observed data aside for a specific residue whose observed state is replaced by one of the alternate states present in other sequences at that position. As a result, the modified data set has near identical shared ancestry. The frequency with which such permuted data sets result in a MI less than that from the observed data is taken as the probability of observing

a permutation with less dependence. Thus, RMI explicitly adjusts for shared ancestry and computes probabilities of coevolution between residue pairs.

4.2.1.4 Statistical coupling analysis

Statistical coupling analysis (SCA) [59] measures the change in distribution of residues at one position associated with a change in the distribution of residues at another position. If a correlated change in the distribution of residues exists between a pair of positions, that pair is said to be statistically coupled and potentially coevolving. Statistical coupling of a pair of positions i and j is calculated by selecting a subset of the MSA (the subalignment) where the residue at position j is fixed. The difference in the distribution of amino acid residues occurring at position i is calculated between the full MSA and the subalignment as the difference in their Euclidean norms. If the difference between the two distributions is large, the positions are said to be statistically coupled suggesting possible coevolution of those positions. SCA was reimplemented based on the description of the algorithm in [59]. This implementation was verified against the Ranganathan group's Matlab implementation and their published results[25].

In addition to the input alignment, SCA requires that the percentage of sequences containing a fixed residue at a position of interest be specified by the user (the cutoff parameter). To study the effect of the cutoff parameter on SCA's performance, I evaluated SCA using six cutoff values: 0.4, 0.5, 0.6, 0.7, 0.8, and 0.9.

4.2.1.5 Corrected mutual information

Corrected mutual information (MIp) is a MI-based metric which attempts to control for 'background MI', or MI arising from random variation or shared ancestry. As described in [104], MIp is calculated first by computing MI for a pair of positions. (See Section

[25]The original work on the SCA reimplementation was performed by my collaborator on this project, Sandra Smit.

5.1.1 for a detailed description of the MI calculation.) The MIp score is then calculated by subtracting the product of the mean MIs for the two positions divided by the overall mean MI score from the MI score for the pair of positions. MIp data were calculated with a Perl implementation of the algorithm provided by the authors.

4.2.2 Coevolution detection methods: Tree-aware

4.2.2.1 Ancestral states

An ancestral-state-reconstruction-based method (AS) for detecting coevolving positions was implemented based on the method described in [51, 99]. In this method, ancestral states are inferred for each internal node of the provided phylogenetic tree using maximum likelihood with a substitution model calculated from the alignment and tree. For each pair of positions in the alignment, all pairs of organisms were evaluated to score position pairs on the number of times both underwent a substitution since their last common ancestor (LCA). Scores were calculated as a weighted count of correlated substitutions, by summing the inverse branch lengths between organisms when both residues had changed since the LCA. If neither or only one residue changed since the LCA, nothing was added to the score. This weighting has the effect of favoring correlated changes that happen closer to each other in evolutionary time, since these are less likely to be the results of random substitutions. (Many alternative scoring methods are possible, and a comparison of the various approaches is deferred to a future study.)

Coevolving pairs are expected to score higher than non-coevolving pairs. [99] illustrated that the method for inferring ancestral sequences (parsimony or likelihood, in their analysis), and the method for inferring the phylogenetic tree, have little effect on the ability of this method to detect coevolution.

4.2.2.2 LnLCorr

LnLCorr is available in two versions from the authors. The first (LnLCorr99) [52], is available as a C binary. The second (LnLCorr07) [54], is available as an open source MPI/C++ package. Different results were obtained with the different implementations, and data on both is presented. (Since the methods are very similar, we treat them as a single tree-aware method.) LnLCorr uses a likelihood ratio (LR) to compare the probability of the data under independent and dependent models of evolution. In this implementation, a larger LR between a pair of positions suggests coevolution. (When I refer to LnLCorr, I am discussing properties of the algorithm common to the LnLCorr99 and LnLCorr07 implementations.)

LnLCorr differs from the other methods presented in that it incorporates its own amino acid alphabet reduction step based on a residue metric provided by the user. Each implementation is packaged with a default metric which are referred to here as DEF99 and DEF07. At each sequence position, the mean value of the metric is determined, and residues are categorized as above or below the mean. We evaluated LnLCorr using the default metric in addition to the five Atchley metrics (see Section 4.2.4.1), but point out that LnLCorr always recodes to a two-state alphabet. These alphabets, while based on the same residue metrics as the alphabets provided to our other methods, result in different encodings of the alignments for LnLCorr than for the other algorithms.

4.2.2.3 Generalized continuous-time Markov process coevolutionary algorithm

The generalized continuous-time Markov process coevolutionary algorithm (GCTM-PCA) [50, 53] employs maximum likelihood to determine if pairs of substitution events are more likely under a dependent or independent model of evolution. These data were calculated using the C++ implementation provided by the authors as open-source software.

GCTMPCA requires a single parameter *epsilon* (ϵ), the penalty incurred for a single residue change (as opposed to a correlated change between two residues), be provided by the user. Based on empirical evidence the authors define 0.7 as the optimal value of ϵ. Unless otherwise noted this default value is used, and was validated as optimal from a range of ϵ values. The instantaneous rate matrix for single substitution events must also be provided by the user. The authors use a rate matrix derived from the Dayhoff model [8], and that is also adopted here as the default. In studies using reduced amino acid alphabets, this matrix was modified to represent substitution rates in the reduced alphabet. To reduce the substitution matrix, the counts and frequencies from the original Dayhoff data were collapsed in accordance with the reduced amino acid alphabet and used to recalculate the instantaneous rate matrix.

4.2.2.4 CoMap

The CoMap algorithm is similar to AS in that it relies on reconstruction of the ancestral states of all positions in the alignment. However, instead of simply counting the number of co-occurring substitutions, CoMap builds 'substitution vectors' for each position, where each element in the vector represents a change in a corresponding branch of the phylogenetic tree. Coevolving positions are identified as those with correlated substitution vectors.

Two variants of CoMap were used in this study. The algorithm presented in [106] builds binary substitution vectors indicating whether a substitution occurred on each branch. In [102] an updated algorithm is presented which incorporates 'weighted substitution vectors', which score changes based on the difference in a pre-specified physicochemical property. Data computed with the binary substitution vector is presented as using an unreduced alphabet (`ORIG`) since there is no adjustment made for the physicochemical properties of the residues, and data computed using the weighted vectors are presented as reduced-state alphabets since they are designed to represent physicochemical properties of the residues. Three of CoMap's built-in weighting schemes are evaluated (`GRANTHAM.POLARITY`,

GRANTHAM.VOLUME, and CHARGE), in addition to weighting schemes based on the five Atchley factors (see Section 4.2.4.1). CoMap data were calculated with CoMap-1.3.0, provided by its authors.

4.2.3 Alignments and trees

4.2.3.1 Myoglobin and randomized myoglobin

The first two alignments analyzed were a 153-position myoglobin alignment containing sequences from 42 tetrapods, and a randomized version of this alignment which was used as a negative control. Myoglobin is a mostly alpha helical protein, and therefore serves as a good test case for detecting alpha helix periodicity. This alignment and the corresponding phylogenetic tree were published with [52], where the details of its construction can be found. This alignment contains one gap in one sequence.

The randomized myoglobin alignment was generated by reordering the columns, thereby removing all structural information. Because the periodicity of the alpha helix should no longer be detectable, this alignment served as a negative control. The randomized alignment used the same tree as the myoglobin alignment.

For the purpose of alphabet recoding, it was necessary to replace ambiguous characters in the alignments with non-ambiguous characters. The five B characters (which represent either asparagine or aspartic acid) and two Z characters (which represent either glutamine or glutamic acid) in the myoglobin alignment were replaced with D and E, respectively. D and E were chosen over N and Q based on residue frequency in the full alignment. No X or J characters (which represent any amino acid, and leucine or isoleucine, respectively) were present in the myoglobin alignment.

4.2.3.2 Myosin rod domain and randomized myosin rod domain

The myosin heavy chain alignment was originally constructed for the MyoMapr database [62] with ClustalW [42], and subsequently adjusted using PyCogent [126]. Sequences that

introduced gaps in other sequences were deleted, as were sequences that contained gaps that were not shared by other sequences at the 99.5% gap identity level (i.e. pairs of sequences left in the alignment were 99.5% identical in gap pattern over the aligned region). A subalignment consisting of only the rod domain was used as the second alpha helical data set.

The rod domain of myosin is a continuous alpha helix which forms a coiled-coil homodimer. This alignment contains 1064 positions, is derived from 114 chordate sequences, and contains no gaps or ambiguous characters. The positions in this alignment were also shuffled to serve as a negative control. A phylogenetic tree was constructed by neighbor-joining [127]. Since the alignment represents a continuous alpha helix, it is ideal for detecting alpha helix periodicity.

4.2.4 Amino acid alphabets

4.2.4.1 Atchley-factor alphabets

Atchley et al. (2005) calculated values for the twenty canonical amino acid residues by reducing 54 amino acid attributes to five condensed metrics using factor analysis. We used these five metrics (A1 - A5) to define reduced-state amino acid alphabets of varied sizes. To define each 'Atchley-factor alphabet,' we ordered the amino acid residues based on their values for each metric, and then grouped neighboring residues into n roughly equal sized groups, where n ranged from 2 through 10. Each of the five Atchley factors was used to define nine Atchley-factor alphabets, resulting in forty-five reduced alphabets. Each alphabets is referred to based on the factor it is derived from and the alphabet size. For example, `A1_4` refers to the four-state alphabet derived from Atchley-factor 1 (A1).

The five factors are described in [128] as follows: A1, related to residue polarity; A2, related to propensity for different secondary structures; A3, a molecular size/volume factor; A4, related to amino acid composition in proteins and number of codons; and A5, a

electrostatic charge factor.

4.2.4.2 Canonical reduced alphabets

The full twenty-state amino acid alphabet was broken into seven additional reduced-state alphabets based on four commonly recognized features of amino acids: hydropathy index (a three-state alphabet), charge with histidine treated as a charged residue (two- and three-state alphabets), charge with histidine treated as an uncharged residue (two- and three-state alphabets), polarity (a four-state alphabet), and size (a two-state alphabet). These reduced alphabets were generated by splitting the full alphabet based on natural breaks in the properties of the amino acids, and I therefore refer to these as 'rationally-designed' alphabets (as opposed to the heuristically defined Atchley-factor alphabets). All amino acid alphabet definitions are presented in Table 4.4.

4.2.4.3 Correlations between alphabet size and performance

Kendall's *tau* (τ) rank correlation test [129] was applied to compute correlation coefficients (τ) and p-values for the relationship between alphabet size and method performance. For each Atchley-factor alphabet, the performance at each alphabet size (2 states through 10 states, and the full twenty-state alphabet) were ranked, and the ranks compared against the rank alphabet size to determine if there was a correlation between alphabet size and performance. Performance ranking for a method/alphabet combination was performed based on the -log(p-value) for identifying $(i, i+4)$ pairs as different from the background signal. The $(i, i+4)$ pairs were chosen over the $(i, i+3)$ pairs to define significance because these are generally where the strongest signals were observed for either method, and the biochemical data more strongly support these interactions. A positive τ indicates that an increase in the number of states is correlated with an increase in performance.

4.2.5 Detecting periodicity of the alpha helix

To detect coevolutionary signal resulting from the periodicity of the alpha helix, coevolution scores were compiled for pairs of positions differing by n in position number, or $(i, i+n)$ pairs. For each value of n, the distribution of scores for the $(i, i+n)$ pairs was compared to the distribution of all other positions pairs in the alignment (the background distribution) using a one-tailed (H_a: $\mu_{other} < \mu_n$), two-sample t-test. For example, to look for coevolutionary signal between positions separated by three residues $(i, i+4)$, the distribution of coevolution scores at position pairs {(1,5), (2,6), (3,7), (4,8), ...} was compared with the distribution of all other positions pair coevolution scores {(1,2), (1,3), (1,4), (1,6), ..., (2,3), (2,4), (2,5), (2,7), ...}. When sample sizes are large, the two-sample t-test is robust to deviations of the background distribution from normality and differences in the population sample sizes, making it suitable for this application. An important feature of this test is that methods with higher false positive rates do not achieve more significant p-values. This is because the higher false positive rate tends to increase the coevolution scores of all pairs, thus elevating the background distribution and resulting in a less significant p-value.

The multiple comparisons problem is significant, but is unavoidable in full-molecule coevolutionary analyses. P-values are presented from all tests in relation to several baselines, including the Bonferroni-adjusted alpha to control for multiple comparisons [130]. The Bonferroni adjustment circumvents the multiple comparison problem, but is often considered too stringent, greatly decreasing the power of the statistical test.

The distance matrix structure of the result matrices violates the independence clause of the t-test. To determine if this affected the conclusions, p-values were computed empirically using a non-parametric matrix permutation test [131] for several result matrices. (This method is very computationally intensive, so was not applied to all results.) Results suggest that the p-values obtained from the t-tests are sometimes exaggerated, but the ranking of

p-values is consistent. The coevolution score distributions are not normal (see Figure 4.1), which is also potentially problematic for the t-test. To address this, the Wilcoxon rank-sum test [132] (implemented in the R statistical package) was applied to confirm the t-test p-values for a selection of result matrices and negative controls. The Wilcoxon p-values also suggest that the t-test p-values are sometimes exaggerated, but rank similarly to the t-test p-values. These results support the use of t-test for comparing the performance of algorithms.

4.2.6 Identifying individual coevolving pairs

To identify individual coevolving pairs, coevolution scores corresponding to each pair of positions were compared to all coevolution scores in the same matrix using one-observation t-tests to generate a p-value matrix indicating the statistical significance of each score. Significance thresholds (α) were varied from 1.0×10^{-2} and 1.0×10^{-7} in steps of one order of magnitude, and statistically significant scores corresponding to $(i, i+3)$ and $(i, i+4)$ pairs were counted as true positives (TP). Statistically significant scores corresponding to all other position pairs were counted as false positives (FP). Scores corresponding to $(i, i+3)$ and $(i, i+4)$ pairs which were not statistically significant were counted as false negatives (FN). This allowed for calculation of precision (P) as $TP/(TP+FP)$, or statistically significant scores arising from the periodicity of the alpha helix, divided by the total number of statistically significant scores (total hits); and calculation of recall (R) as $TP/(TP+FN)$, or statistically significant scores arising from the periodicity of the alpha helix, divided by the total number of $(i, i+3)$ and $(i, i+4)$ pairs. It must be noted that pairs are expected to coevolve for effects other than the stability of the alpha helix, and the false positive count is therefore contaminated with many scores which should be counted as true positives. Similarly, not all stacked positions in the alpha helix are expected to coevolve, and the false negative count is therefore contaminated with many true negatives. Since the exact set of coevolving positions is not known, it is not possible to adjust these counts

accordingly. This issue is common to all methods however, so while the true precisions and recalls are likely significantly higher than those presented here, the relative values should be meaningful comparisons of the methods. While for simplicity these scores are referred to as precision and recall, more accurately *precision* can be described as the proportion of statistically significant coevolution scores associated with pairs of stacked positions in the alpha helix, and *recall* can be described as the proportion of stacked positions observed to coevolve. These statistics are summarized by presenting F-measures, the harmonic mean of precision and recall, calculated as $(2 \times P \times R)/(P + R)$.

Precision, recall, F-measure, and total hits data are summarized with area under the curve (AUC) scores for each method, alphabet combination. Six variates of α were tested, and AUC was calculated for each metric (Figure 4.4). Each step in α contributed an equal length to the area under that segment of the curve ($\log_{10} \alpha^2 - \log_{10} \alpha^1$, opposed to $\alpha^2 - \alpha^1$). If no statistically significant scores were returned for one value of α (i.e., total hits = 0), precision could not be calculated because the denominator was equal to zero. For these values of α, precision was set to zero to facilitate the calculation of AUC. Similarly, when precision and recall were both equal to zero, F-measure could not be calculated and was therefore set to zero. This is convenient because, for example, if a method, alphabet combination achieved no statistically significant hits for any value of α, precision AUC was equal to zero. All precision, recall, F-measure, total hits, and AUC data are provided as supplementary material to Caporaso *et al.* [61][26].

4.2.7 χ^2 comparisons of tree-aware and tree-ignorant methods

The tree-aware and tree-ignorant categories were compared using χ^2 goodness-of-fit tests (Table 4.1). A 2x2 contingency table was compiled by counting the runs (algorithm/alphabet combinations) in each category which identify a set of pairs as statistically significant with $\alpha = 0.01$. The performance was evaluated for identifying $(i, i+3)$ pairs and

[26]This data set is too large to include in this dissertation.

$(i, i+4)$ pairs in myosin and myoglobin. In each case, counts were tallied using the best and worst performing SCA cutoff values. (SCA cutoff values obtaining the most and least significant median p-values at $(i, i+3)$ and $(i, i+4)$ in each alignment were selected as the best and worst SCA sets, respectively. For example, the $(i, i+3)$ myoglobin tree-ignorant counts in row 1 of Table 4.1A, and the associated negative control, use cutoff=0.90 because those values achieved the highest median p-value for that specific data point.) Because SCA performance varies widely with the cutoff value (see Figure 4.6), and optimization is not always practical, it is useful to see how the methods compare when the best and worst cutoff values are used. Tree-aware counts are always tallied using GCTMPCA runs with the recommended and empirically validated optimal value of $\epsilon = 0.70$.

Comparisons of the tree-aware methods (Table 4.2) and the tree-ignorant methods (Table 4.3) were performed similarly. For both categories, 4x2 contingency tables were compiled by counting the significant and insignificant scores at $(i, i+3)$ and $(i, i+4)$, with $\alpha = 0.01$. LnLCorr07 and GCTMPCA were not included in the comparison on myosin because the computation time was prohibitive. Counts for GCTMPCA on myoglobin were compiled with $\epsilon = 0.70$. All tree-ignorant methods were compared on myoglobin and myosin, and counts were compiled using the empirically determined optimal SCA cutoff values for each data point.

4.3 Results

4.3.1 Tree-aware versus tree-ignorant methods

The tree-ignorant methods are more capable of detecting coevolutionary signal at $(i, i+3)$ and $(i, i+4)$ in myoglobin than the tree-aware methods (Table 4.1A, row 1, p=1.50×10^{-3} and row 3, p=4.97×10^{-13}, respectively). There is no statistically significant difference in performance between the tree-aware methods on myoglobin or the myoglobin negative control (Table 4.2). Between the tree-ignorant methods, there is a statistically significant

performance difference at (i,$i+3$) in favor of MIp, NMI, and SCA (Table 4.3A, row 1); and at (i,$i+4$) in favor of NMI and SCA (Table 4.3A, row 2). MI incurred more Type 1 error than the other methods (Table 4.3B, row 1, p=2.71×10^{-3}).

Table 4.1: Comparison of tree-aware methods and tree-ignorant methods.
χ^2 goodness-of-fit tests comparing all applicable tree-aware and all tree-ignorant methods for detecting alpha helix periodicity at separations of (i,$i+3$) and (i,$i+4$), using all reduced-state amino acid alphabets, where applicable. Some of the methods (LnLCorr07 and GCTMPCA) were not run on myosin (see text), causing different numbers of runs on the myosin versus the myoglobin data sets. Additionally, the reduced-state alphabets were not applicable to all tree-aware methods, causing the sum of the p-value counts to differ for the tree-aware versus tree-ignorant methods. Bolded rows highlight statistically significant χ^2 results, and bolded counts in these rows highlight which class of methods achieved the higher ratio of significant to insignificant scores. Counts are calculated including both the empirically determined best and worst SCA cutoffs to illustrate the effect of this free parameter on method performance, and with the empirically validated optimal $\epsilon = 0.70$ value for GCTMPCA. $\alpha = 0.01$.

(A) Tetrapod Myoglobin						
	Tree-aware		Tree-ignorant			
	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	χ^2	$p-value$
(i,i+3) w best SCA (0.9)	3	125	**32**	**233**	**10.08**	$\mathbf{1.50 \times 10^{-3}}$
(i,i+3) w worst SCA (0.5)	3	125	23	242	5.61	1.79×10^{-2}
(i,i+4) w best SCA (0.8)	14	114	**128**	**137**	**52.22**	$\mathbf{4.97 \times 10^{-13}}$
(i,i+4) w worst SCA (0.4)	14	114	**100**	**165**	**30.10**	$\mathbf{4.10 \times 10^{-8}}$
(B) Randomized Tetrapod Myoglobin						
	Tree-aware		Tree-ignorant			
	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	χ^2	$p-value$
(i,i+3) w best SCA (0.9)	1	127	4	261	0.36	5.46×10^{-1}
(i,i+3) w worst SCA (0.5)	1	127	9	256	2.38	1.23×10^{-1}
(i,i+4) w best SCA (0.8)	0	128	1	264	0.48	4.87×10^{-1}
(i,i+4) w worst SCA (0.4)	0	128	1	264	0.48	4.87×10^{-1}
(C) Chordate Myosin						
	Tree-aware		Tree-ignorant			
	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	χ^2	$p-value$
(i,i+3) w best SCA (0.4)	54	14	221	44	0.60	4.40×10^{-1}
(i,i+3) w worst SCA (0.9)	54	14	198	67	0.65	4.21×10^{-1}
(i,i+4) w best SCA (0.6)	58	10	209	56	1.41	2.36×10^{-1}
(i,i+4) w worst SCA (0.9)	**58**	**10**	177	88	**8.92**	$\mathbf{2.82 \times 10^{-3}}$
(D) Randomized Chordate Myosin						
	Tree-aware		Tree-ignorant			
	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	χ^2	$p-value$
(i,i+3) w best SCA (0.4)	7	61	56	209	4.14	4.18×10^{-2}
(i,i+3) w worst SCA (0.9)	7	61	55	210	3.91	4.81×10^{-2}
(i,i+4) w best SCA (0.6)	0	68	0	265	n/a	n/a
(i,i+4) w worst SCA (0.9)	0	68	0	265	n/a	n/a

In myosin, there is no significant difference in the performance of the tree-aware and

Table 4.2: Tree-aware methods compared on myoglobin and myosin.

χ^2 goodness-of-fit tests comparing the performance of tree-aware methods for detecting alpha helix periodicity at (i,$i+3$) and (i,$i+4$). Counts are calculated for GCTMPCA with the empirically validated optimal $\epsilon = 0.70$. In most cases, there is no statistically significant difference in performance between the tree-aware methods with the single exception being $(i, i+4)$ in myosin, where AS and CoMap out-perform LnLCorr99. Data is not presented on the myosin alignment for tree-aware methods which proved too computationally intensive to be practical. Counts in Table 4.2 can be directly compared with counts in Table 4.3. $\alpha = 0.01$.

(A) Tetrapod Myoglobin												
	GCTMPCA ($\epsilon = 0.70$)		LnLCorr99		LnLCorr07		Ancestral States		CoMap			
	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	χ^2	$p - value$
(i,i+3)	0	53	1	5	0	7	2	51	0	9	7.51	1.11×10^{-1}
(i,i+4)	10	43	0	6	0	7	4	49	0	9	6.75	1.50×10^{-1}
(B) Randomized Tetrapod Myoglobin												
	GCTMPCA ($\epsilon = 0.70$)		LnLCorr99		LnLCorr07		Ancestral States		CoMap			
	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	χ^2	$p - value$
(i,i+3)	1	52	0	6	0	7	0	53	0	9	1.43	8.40×10^{-1}
(i,i+4)	0	53	0	6	0	7	0	53	0	9	n/a	n/a
(C) Chordate Myosin												
	GCTMPCA ($\epsilon = 0.70$)		LnLCorr99		LnLCorr07		Ancestral States		CoMap			
	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	χ^2	$p - value$
(i,i+3)	-	-	5	1	-	-	41	12	8	1	0.69	7.09×10^{-1}
(i,i+4)	-	-	2	4	-	-	**48**	**5**	8	1	**14.18**	$\mathbf{8.33 \times 10^{-4}}$
(D) Randomized Chordate Myosin												
	GCTMPCA ($\epsilon = 0.70$)		LnLCorr99		LnLCorr07		Ancestral States		CoMap			
	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	χ^2	$p - value$
(i,i+3)	-	-	0	6	-	-	7	46	0	9	2.21	3.31×10^{-1}
(i,i+4)	-	-	0	6	-	-	0	53	0	9	n/a	n/a

Table 4.3: Tree-ignorant methods compared on myoglobin and myosin.

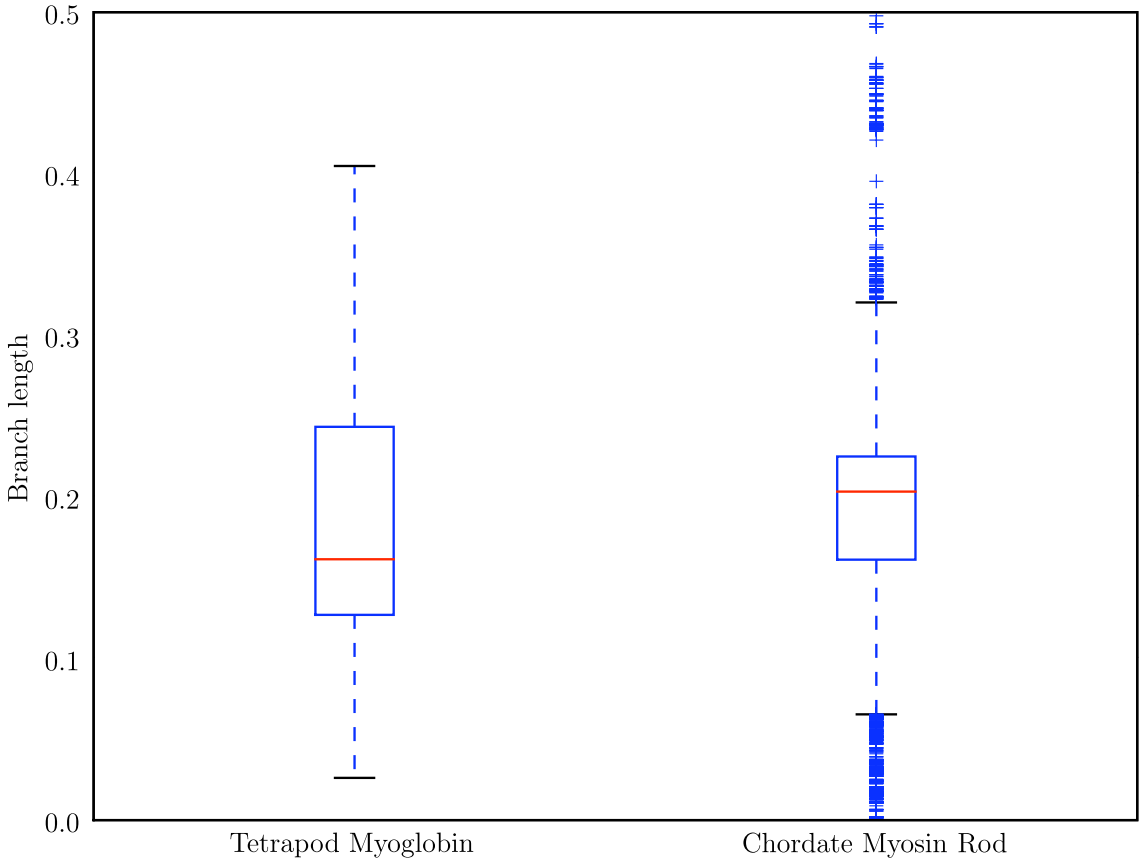
χ^2 goodness-of-fit tests comparing the performance of tree-ignorant methods for detecting alpha helix periodicity at (i,i+3) and (i,$i+4$). Bolded rows highlight statistically significant χ^2 results, indicating a difference between the methods. Bold counts in these rows highlight which method achieved the highest ratio of significant to insignificant scores. Counts are calculated using the empirically determined optimal SCA cutoff for each positive control data point. SCA cutoff is 0.9 for A-B row 1, 0.8 for A-B row 2, 0.4 for C-D row 1, and 0.6 for C-D row 2. $\alpha = 0.01$.

(A) Tetrapod Myoglobin												
	MI		NMI		RMI		SCA		MIP			
	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	χ^2	$p - value$
(i,i+3)	2	51	9	44	0	53	10	43	**11**	**42**	**17.98**	$\mathbf{1.24 \times 10^{-3}}$
(i,i+4)	32	21	**41**	**12**	1	52	36	17	18	35	**79.28**	$\mathbf{2.48 \times 10^{-16}}$
(B) Randomized Tetrapod Myoglobin												
	MI		NMI		RMI		SCA		MIP			
	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	χ^2	$p - value$
(i,i+3)	**4**	**49**	0	53	0	53	0	53	0	53	**16.25**	$\mathbf{2.71 \times 10^{-3}}$
(i,i+4)	0	53	0	53	0	53	0	53	1	52	4.02	4.04×10^{-1}
(C) Chordate Myosin												
	MI		NMI		RMI		SCA		MIP			
	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	χ^2	$p - value$
(i,i+3)	**51**	**2**	46	7	38	15	40	13	46	7	**14.83**	$\mathbf{5.08 \times 10^{-3}}$
(i,i+4)	**52**	**1**	44	9	26	27	44	9	43	10	**41.30**	$\mathbf{2.33 \times 10^{-8}}$
(D) Randomized Chordate Myosin												
	MI		NMI		RMI		SCA		MIP			
	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	$p \leq \alpha$	$p > \alpha$	χ^2	$p - value$
(i,i+3)	5	48	18	35	4	49	2	51	**27**	**26**	**53.30**	$\mathbf{7.38 \times 10^{-11}}$
(i,i+4)	0	53	0	53	0	53	0	53	0	53	n/a	n/a

tree-ignorant methods at $(i, i+3)$ or $(i, i+4)$ after SCA cutoff optimization, but before SCA optimization the tree-aware algorithms achieve better performance at $(i, i+4)$ (Table 4.1C). The absence of a significant difference could reflect either real properties of the methods, or arise from reduced statistical power of this particular comparison. There is more variability in the branch lengths in the myosin data set (Figure 4.2), suggesting the possibility that as the relationships between sequences becomes more variable, accounting for those relationships becomes more important. However, the statistical power of the method comparisons are not identical between the myoglobin and myosin cases. Due to the computational intensity of GCTMPCA and LnLCorr07, these methods were not practical to run on the myosin alignment and tree. (Single runs of GCTMPCA and LnLCorr07 on the myosin rod were stopped after running for greater than 78 processor hours and 383 processor hours, respectively.) As a result, the myosin case has a reduced number of observations compared with that for myoglobin which will reduce the statistical power to identify differences between the method classes. Comparing LnLCorr99, CoMap, and AS on myosin shows that there is no statistically significant difference in these methods at $(i, i+3)$ (Table 4.2C, row 1) but that AS and CoMap outperform LnLCorr99 at $(i, i+4)$ (Table 4.2C, row 2, p=8.33×10^{-4}). Of the tree-ignorant methods, there was a statistically significant difference in the performance of the individual methods (Table 4.3C), and MI achieved the highest ratio of significant to insignificant scores, and fared well in terms of Type 1 error (Table 4.3D). On the myosin negative control, MIp achieved significantly more false positives than the other methods (Table 4.3D, row 1, p=7.38×10^{-11}). The ability of MI to out-perform all of the other methods, including the tree-aware methods, is extremely surprising.

Figure 4.3 shows the relative performance of all methods and parameter settings (alphabet choice, SCA cutoff and GCTMPCA ϵ). These data illustrate the results of the χ^2 tests (Figure 4.3A,C), and the negative controls confirm the validity of the evaluations by

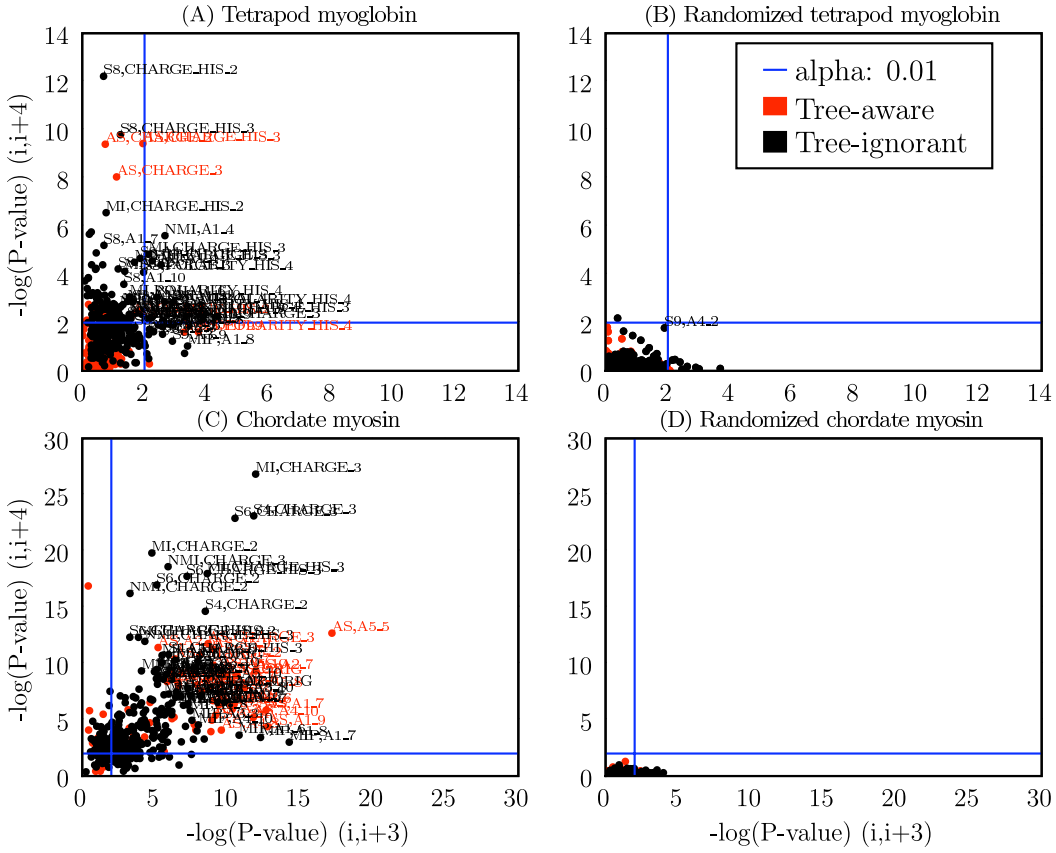
Figure 4.2: Branch lengths associated with the myoglobin and myosin alignments. Red line indicates the median value, and the top and bottom of the box indicate the upper and lower quartile values, respectively. Whiskers represent the largest and smallest values within 1.5 x IQR (inter-quartile range), and pluses represent outliers, or points outside of 1.5 x IQR. The distribution of branch lengths in the myosin tree is clearly wider and more variable than the distribution of branch lengths in the myoglobin tree.



consistently showing no coevolution where it is not expected (Figure 4.3B,D).

The tree-aware and tree-ignorant methods were additionally evaluated on their ability to identify individual coevolving pairs by comparing area under the curve (AUC) scores for precision and recall (see Methods for definitions) for each method, alphabet combination

Figure 4.3: Method, alphabet combination performance for detecting helix stacking. Each combination of method and alphabet is represented as a single point, with coordinates defined by the -log(p-value) at $(i, i+3)$ (x-axis) and $(i, i+4)$ (y-axis) for each alignment. SCA cutoffs included for (A-B) are 0.9 and 0.8, and for (C-D) are 0.4 and 0.6. These represent the best cutoff values at $(i, i+3)$ and $(i, i+4)$, respectively, for each positive control data set. Label key: `method,alphabet`, so `S8,CHARGE_HIS_2` refers to SCA with cutoff=0.80, and the `CHARGE_HIS_2` alphabet. Method abbreviations: AS: Ancestral States; L07: LnLCorr07; L99: LnLCorr99; MI: Mutual Information; NMI: Normalized Mutual Information; RMI: Resampled Mutual Information; Sn: Statistical Coupling Analysis, cutoff $= n/10$; MIP, Corrected Mutual Information; Gn: Generalized Continuous-Time Markov Process Coevolutionary Algorithm, $\epsilon = n/10$; CM: CoMap. Alphabet definitions in Table 4.4.

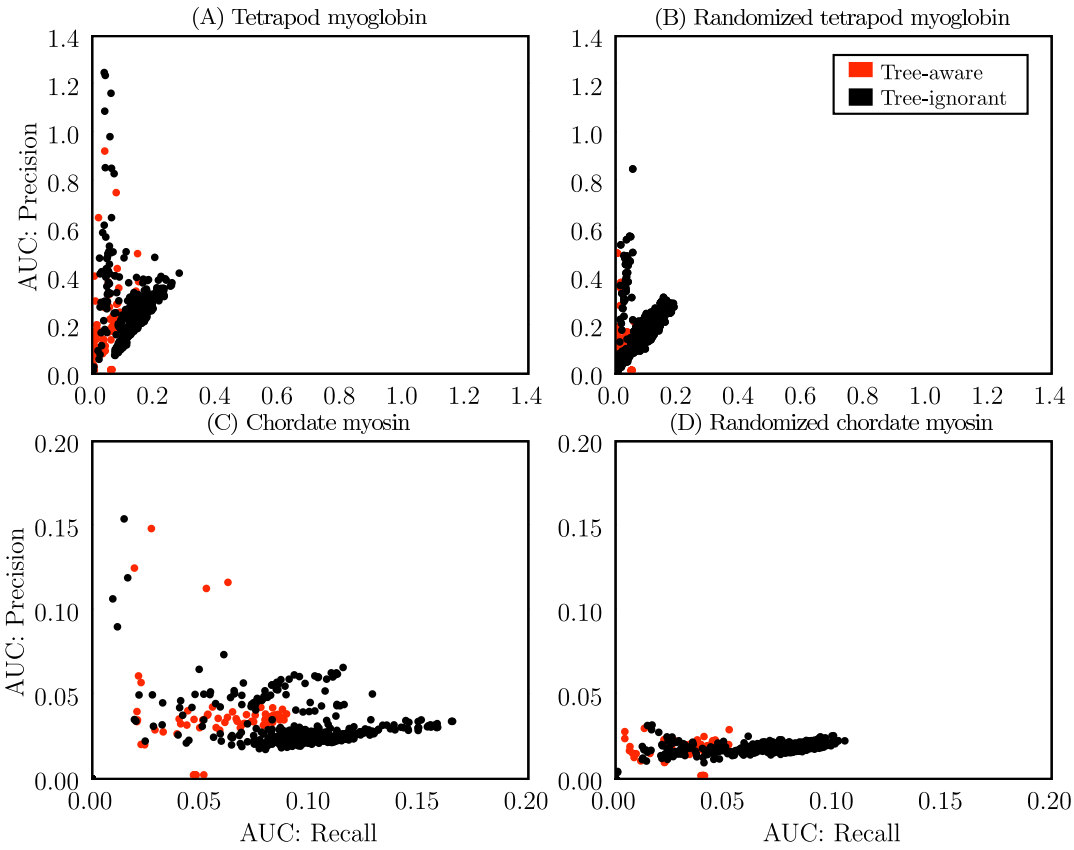


(Figure 4.4). On the myoglobin data set, the tree-ignorant methods achieved higher AUC scores for precision and recall than the tree-aware methods. On the myosin alignment, the tree-ignorant methods achieved higher AUC scores for recall than the tree-aware methods, but the highest precision scores appear similar between the two classes of methods. On the myoglobin negative control, the tree-ignorant methods achieved higher AUC scores for precision and recall, suggesting more false positives for the tree-ignorant methods. On the myosin negative control, precision appears similar between the two classes, while the tree-ignorant methods appear to have achieved higher recall, indicating more false positives. Because the negative controls are a shuffled version of the original alignments, the total hits count is the same for the positive and negative control coevolution matrices for a given method, alphabet combination (with the exception of the LnLCorr methods, where the coevolution scores differ slightly between positive and negative control alignments). Due to the difficulty of defining positives and negatives (discussed in Methods), the precision, recall, and F-measure values for the negative controls are less meaningful than for the positive control, but are provided for completeness. Precision and recall results are summarized via F-measure in Figure 4.5, which presents the F-measure AUCs achieved by each method with each alphabet. The median F-measure AUCs appear higher for the tree-ignorant methods in both positive and negative controls. Precision, recall, F-measure, total hits, and AUC data for all method, alphabet combinations are provided as supplementary material to Caporaso *et al.* [61].

4.3.2 Alphabet reduction

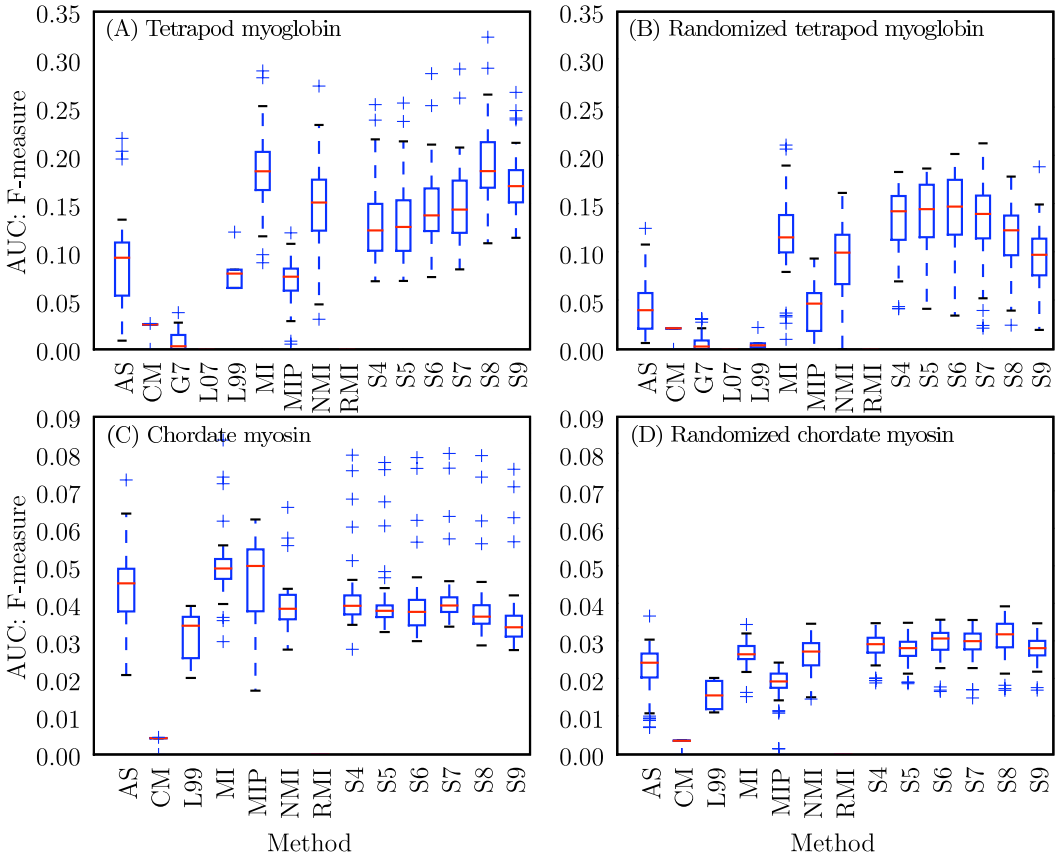
To evaluate the effect of the size and type of reduced-state alphabets on algorithm performance, each alignment was recoded with 52 reduced-state alphabets. Coevolution algorithms were applied to each recoded alignment in addition to the original (full-alphabet) alignments. The 45 'Atchley-factor' alphabets are based on five metrics (A1-A5) presented in [128], and are used primarily to evaluate the effect of alphabet size. The 7 'rationally-

Figure 4.4: Method, alphabet combination performance for detecting coevolving positions. Each combination of method and alphabet is represented as a single point, with coordinates defined by area under the curve (AUC) values for recall (y-axis) and precision (x-axis) for the (A) tetrapod myoglobin alignment, (B) randomized tetrapod myoglobin alignment, (C) chordate myosin rod alignment, and (D) randomized chordate myosin rod alignment. The computation of AUC values is described in Methods.



designed' alphabets were developed based on more canonical amino acid metrics, and are used primarily to evaluate the effect of alphabet type. Alphabet names describe the alphabet type (i.e., the property being modeled) followed by alphabet size (i.e., the number of states). For example, `A1_4` refers to the A1-based alphabet with four states, and `CHARGE_3` refers to the charge-based alphabet with three states. Alphabet definitions are presented in Table

Figure 4.5: F-measure distributions for each method for detecting coevolving positions. Modified box plots illustrate the distribution of F-measures obtained with a given method over all alphabets. Values are F-measure area under the curve (AUC) scores for each alignment. The computation of AUC values is described in Methods. Red lines indicate the median AUC, and the top and bottom of the boxes indicate the upper and lower quartile values, respectively. Whiskers represent the largest and smallest AUC values within 1.5 x IQR (inter-quartile range), and pluses represent outliers, or points outside of 1.5 x IQR. Methods with more condensed distributions are those that appear more robust to alphabet choice. AS: Ancestral States; L07: LnLCorr07; L99: LnLCorr99; MI: Mutual Information; NMI: Normalized Mutual Information; RMI: Resampled Mutual Information; Sn: Statistical Coupling Analysis, cutoff = $n/10$; MIP, Corrected Mutual Information; Gn: Generalized Continuous-Time Markov Process Coevolutionary Algorithm, $\epsilon = n/10$; CM: CoMap.



4.4.

The data illustrate markedly different results with different alphabet definitions, both in terms of number of states and amino acid property modeled.

4.3.2.1 The effect of alphabet size

In the myoglobin alignment (42 sequences by 153 positions), there is no clear relationship between performance and number of alphabet states (Table 4.5A). The three of thirty-five correlations that are significant after correction for multiple comparisons are positive correlations, suggesting increased performance with more states. In the negative control (Table 4.5B) however, the ratio of significant to insignificant correlations is similar. These data therefore do not strongly support a relationship between alphabet size and method performance.

In the myosin rod alignment (114 sequences by 1064 positions) a positive correlation frequently exists between alphabet size and ability to detect alpha helix periodicity (Table 4.6A), particularly for MI, NMI, and MIp. This suggests that more alphabet states improve performance for these methods on the myosin rod. Seven of the thirty correlation coefficients are significant after correction for multiple comparisons, compared with one of thirty in the negative control (Table 4.6B).

4.3.2.2 The effect of alphabet type

The 'rationally-designed' alphabets generally out-perform the Atchley-factor alphabets, although the A1-based and A5-based alphabets were frequently among the top performing alphabets (Table 4.7). The `CHARGE_*` alphabets very commonly yield the most significant p-values at $(i, i+4)$ in both alignments. As A1 and A5 are charge and polarity factors, these were expected to perform well. In myoglobin, of the methods that identified $(i, i+4)$ pairs with statistical significance at $\alpha = 0.01$, seventeen of the thirty-five top-performing alphabets were the rationally designed alphabets. Eleven of the eighteen top-performing

Table 4.4: Reduced-state alphabet definitions.

The 52 reduced-state amino acid alphabets. Each state is defined as a group of characters followed by a semicolon, so for example, 'KRDEH' and 'ACFGILMNPQSTVWY' are reduced to the charged and uncharged states, respectively, in the `CHARGE_HIS_2` alphabet.

(A) Rationally defined alphabets			
Alphabet Identifier	States		
`CHARGE_2`	KRDE;ACFGHILMNPQSTVWY		
`CHARGE_HIS_2`	KRDEH;ACFGILMNPPQSTVWY		
`CHARGE_3`	KR;DE;ACFGHILMNPQSTVWY		
`CHARGE_HIS_3`	KRH;DE;ACFGILMNPQSTVWY		
`SIZE_2`	GAVLISPTCND;MFYWQKHRE		
`POLARITY_HIS_4`	DE;RHK;AILMFPWV;GSTCYNQ		
`HYDROPATHY_3`	RKDENQH;YWSTG;PAMCFLVI		
(B) Heuristically defined 'Atchley-factor' alphabets			
Alphabet Identifier	States	Alphabet Identifier	States
`A1_2`	CVILFMWAGS;TPYHQNDERK	`A1_3`	CVILFMW;AGSTPY;HQNDERK
`A1_4`	CVILF;MWAGS;TPYHQ;NDERK	`A1_5`	CVIL;FMWA;GSTP;YHQN;DERK
`A1_6`	CVI;LFMW;AGS;TPY;HQND;ERK	`A1_7`	CVI;LFM;WAG;ST;PYH;QND;ERK
`A1_8`	CVI;LF;MWA;GS;TPY;HQ;NDE;RK	`A1_9`	CV;IL;FMW;AG;ST;PY;HQN;DE;RK
`A1_10`	CV;IL;FM;WA;GS;TP;YH;QN;DE;RK		
`A2_2`	MEALFKIHVQ;RWDTCNYSGP	`A2_3`	MEALFKI;HVQRWD;TCNYSGP
`A2_4`	MEALF;KIHVQ;RWDTC;NYSGP	`A2_5`	MEAL;FKIH;VQRW;DTCN;YSGP
`A2_6`	MEA;LFKI;HVQ;RWD;TCNY;SGP	`A2_7`	MEA;LFK;IHV;QR;WDT;CNY;SGP
`A2_8`	MEA;LF;KIH;VQ;RWD;TC;NYS;GP	`A2_9`	ME;AL;FKI;HV;QR;WD;TCN;YS;GP
`A2_10`	ME;AL;FK;IH;VQ;RW;DT;CN;YS;GP		
`A3_2`	SDQHPLCAVK;WNGERFITMY	`A3_3`	SDQHPLC;AVKWNG;ERFITMY
`A3_4`	SDQHP;LCAVK;WNGER;FITMY	`A3_5`	SDQH;PLCA;VKWN;GERF;ITMY
`A3_6`	SDQ;HPLC;AVK;WNG;ERFI;TMY	`A3_7`	SDQ;HPL;CAV;KW;NGE;RFI;TMY
`A3_8`	SDQ;HP;LCA;VK;WNG;ER;FIT;MY	`A3_9`	SD;QH;PLC;AV;KW;NG;ERF;IT;MY
`A3_10`	SD;QH;PL;CA;VK;WN;GE;RF;IT;MY		
`A4_2`	WHCMYQFKDN;EIPRSTGVLA	`A4_3`	WHCMYQF;KDNEIP;RSTGVLA
`A4_4`	WHCMY;QFKDN;EIPRS;TGVLA	`A4_5`	WHCM;YQFK;DNEI;PRST;GVLA
`A4_6`	WHC;MYQF;KDN;EIP;RSTG;VLA	`A4_7`	WHC;MYQ;FKD;NE;IPR;STG;VLA
`A4_8`	WHC;MY;QFK;DN;EIP;RS;TGV;LA	`A4_9`	WH;CM;YQF;KD;NE;IP;RST;GV;LA
`A4_10`	WH;CM;YQ;FK;DN;EI;PR;ST;GV;LA		
`A5_2`	DSQPVLECWA;HFINMTYKGR	`A5_3`	DSQPVLE;CWAHFI;NMTYKGR
`A5_4`	DSQPV;LECWA;HFINM;TYKGR	`A5_5`	DSQP;VLEC;WAHF;INMT;YKGR
`A5_6`	DSQ;PVLE;CWA;HFI;NMTY;KGR	`A5_7`	DSQ;PVL;ECW;AH;FIN;MTY;KGR
`A5_8`	DSQ;PV;LEC;WA;HFI;NM;TYK;GR	`A5_9`	DS;QP;VLE;CW;AH;FI;NMT;YK;GR
`A5_10`	DS;QP;VL;EC;WA;HF;IN;MT;YK;GR		

Table 4.5: Alphabet size and performance in myoglobin alignments.

For each heuristically defined alphabets, Kendall correlation coefficients (τ) and corresponding p-values quantitate the correlation between performance rank and alphabet size. Bolded values highlight p-values ≤ 0.05, and starred values indicated p-values significant after adjustment for multiple comparisons. (Adjusted α A-B: 0.0014)

(A) Tetrapod Myoglobin										
Method	A1		A2		A3		A4		A5	
	τ	p-value	τ	p-value	τ	p-value	τ	p-value	τ	p-value
MI	-0.20	4.8×10^{-1}	0.42	1.1×10^{-1}	0.16	6.0×10^{-1}	-0.33	2.2×10^{-1}	0.24	3.8×10^{-1}
NMI	-0.29	2.9×10^{-1}	0.24	3.8×10^{-1}	0.24	3.8×10^{-1}	0.16	6.0×10^{-1}	0.56	$\mathbf{2.9 \times 10^{-2}}$
RMI	0.11	7.3×10^{-1}	0.33	2.2×10^{-1}	-0.51	$\mathbf{4.7 \times 10^{-2}}$	-0.02	1.0	0.64	$\mathbf{9.1 \times 10^{-3}}$
SCA8	-0.24	3.8×10^{-1}	0.78	$\mathbf{9.5 \times 10^{-4}}$*	0.56	$\mathbf{2.9 \times 10^{-2}}$	-0.02	1.0	0.87	$\mathbf{1.2 \times 10^{-4}}$*
MIp	0.16	6.0×10^{-1}	0.47	7.3×10^{-2}	0.29	2.9×10^{-1}	-0.11	7.3×10^{-1}	0.78	$\mathbf{9.5 \times 10^{-4}}$*
AS	-0.02	1.0	0.64	$\mathbf{9.1 \times 10^{-3}}$	0.29	2.9×10^{-1}	-0.47	7.3×10^{-2}	0.07	8.6×10^{-1}
G7	0.16	6.0×10^{-1}	0.64	$\mathbf{9.1 \times 10^{-3}}$	0.33	2.2×10^{-1}	0.42	1.1×10^{-1}	0.47	7.3×10^{-2}
(B) Randomized Tetrapod Myoglobin										
Method	A1		A2		A3		A4		A5	
	τ	p-value	τ	p-value	τ	p-value	τ	p-value	τ	p-value
MI	-0.38	1.6×10^{-1}	-0.29	2.9×10^{-1}	-0.07	8.6×10^{-1}	0.16	6.0×10^{-1}	0.60	$\mathbf{1.7 \times 10^{-2}}$
NMI	-0.29	2.9×10^{-1}	-0.16	6.0×10^{-1}	-0.47	7.3×10^{-2}	-0.07	8.6×10^{-1}	0.16	6.0×10^{-1}
RMI	-0.38	1.6×10^{-1}	-0.29	2.9×10^{-1}	-0.11	7.3×10^{-1}	-0.16	6.0×10^{-1}	0.56	$\mathbf{2.9 \times 10^{-2}}$
SCA8	-0.73	$\mathbf{2.2 \times 10^{-3}}$	-0.69	$\mathbf{4.7 \times 10^{-3}}$	-0.60	$\mathbf{1.7 \times 10^{-2}}$	-0.64	$\mathbf{9.1 \times 10^{-3}}$	0.07	8.6×10^{-1}
MIp	-0.38	1.6×10^{-1}	-0.69	$\mathbf{4.7 \times 10^{-3}}$	-0.42	1.1×10^{-1}	-0.11	7.3×10^{-1}	0.29	2.9×10^{-1}
AS	-0.56	$\mathbf{2.9 \times 10^{-2}}$	-0.42	1.1×10^{-1}	-0.38	1.6×10^{-1}	-0.38	1.6×10^{-1}	-0.24	3.8×10^{-1}
G7	0.60	$\mathbf{1.7 \times 10^{-2}}$	0.29	2.9×10^{-1}	-0.29	2.9×10^{-1}	0.56	$\mathbf{2.9 \times 10^{-2}}$	0.20	4.8×10^{-1}

Table 4.6: Alphabet size and performance in myosin alignments.

For each heuristically defined alphabets, Kendall correlation coefficients (τ) and corresponding p-values quantitate the correlation between performance rank and alphabet size. Bolded values highlight p-values ≤ 0.05, and starred values indicated p-values significant after adjustment for multiple comparisons. (Adjusted α A-B: 0.0017)

(A) Chordate Myosin										
Method	A1		A2		A3		A4		A5	
	τ	p-value	τ	p-value	τ	p-value	τ	p-value	τ	p-value
MI	0.78	$\mathbf{9.5 \times 10^{-4}*}$	0.78	$\mathbf{9.5 \times 10^{-4}*}$	0.64	$\mathbf{9.1 \times 10^{-3}}$	0.60	$\mathbf{1.7 \times 10^{-2}}$	0.82	$\mathbf{3.6 \times 10^{-4}*}$
NMI	0.69	$\mathbf{4.7 \times 10^{-3}}$	0.78	$\mathbf{9.5 \times 10^{-4}*}$	0.56	$\mathbf{2.9 \times 10^{-2}}$	0.29	2.9×10^{-1}	0.82	$\mathbf{3.6 \times 10^{-4}*}$
RMI	0.29	2.9×10^{-1}	0.29	2.9×10^{-1}	-0.07	8.6×10^{-1}	-0.51	$\mathbf{4.7 \times 10^{-2}}$	0.60	$\mathbf{1.7 \times 10^{-2}}$
SCA6	0.56	$\mathbf{2.9 \times 10^{-2}}$	0.20	4.8×10^{-1}	0.16	6.0×10^{-1}	-0.07	8.6×10^{-1}	0.16	6.0×10^{-1}
MIp	0.78	$\mathbf{9.5 \times 10^{-4}*}$	0.78	$\mathbf{9.5 \times 10^{-4}*}$	0.69	$\mathbf{4.7 \times 10^{-3}}$	0.60	$\mathbf{1.7 \times 10^{-2}}$	0.73	$\mathbf{2.2 \times 10^{-3}}$
AS	-0.16	6.0×10^{-1}	0.29	2.9×10^{-1}	0.73	$\mathbf{2.2 \times 10^{-3}}$	0.42	1.1×10^{-1}	0.29	2.9×10^{-1}
(B) Randomized Chordate Myosin										
Method	A1		A2		A3		A4		A5	
	τ	p-value	τ	p-value	τ	p-value	τ	p-value	τ	p-value
MI	-0.51	$\mathbf{4.7 \times 10^{-2}}$	-0.33	2.2×10^{-1}	-0.07	8.6×10^{-1}	0.16	6.0×10^{-1}	-0.16	6.0×10^{-1}
NMI	-0.20	4.8×10^{-1}	-0.78	$\mathbf{9.5 \times 10^{-4}*}$	-0.29	2.9×10^{-1}	0.51	$\mathbf{4.7 \times 10^{-2}}$	0.20	4.8×10^{-1}
RMI	-0.33	2.2×10^{-1}	-0.60	$\mathbf{1.7 \times 10^{-2}}$	-0.24	3.8×10^{-1}	0.16	6.0×10^{-1}	-0.11	7.3×10^{-1}
SCA6	-0.16	6.0×10^{-1}	-0.56	$\mathbf{2.9 \times 10^{-2}}$	-0.60	$\mathbf{1.7 \times 10^{-2}}$	0.20	4.8×10^{-1}	-0.24	3.8×10^{-1}
MIp	-0.29	2.9×10^{-1}	-0.47	7.3×10^{-2}	-0.64	$\mathbf{9.1 \times 10^{-3}}$	0.38	1.6×10^{-1}	-0.51	$\mathbf{4.7 \times 10^{-2}}$
AS	0.20	4.8×10^{-1}	0.20	4.8×10^{-1}	-0.11	7.3×10^{-1}	0.02	1.0	-0.11	7.3×10^{-1}

Atchley-factor alphabets were A1- or A5-based. In myosin, fifteen of the thirty top performing alphabets were rationally designed, and five of the fourteen top-performing Atchley-factor alphabets were A1- or A5-based. One of the top-performing alphabets for MIp was the unreduced alphabet (`ORIG`). It is not clear why the A3- and A4-based alphabets performed better in myosin, although these mostly did well with RMI. RMI exhibits very little variance in p-value based on choice of alphabet (Figure 4.6), which may make p-value-based ranking less meaningful. Because LnLCorr99 and CoMap use alphabets differently from the other algorithms (see Methods), the corresponding counts of Atchley-factor and 'rationally-designed' alphabets were not included in these computations. Top-performing alphabets for these methods are however presented.

The four-state alphabet based on Atchley Factor 4 (`A1_4`) shows up frequently in the top alphabets, and on inspection appears to mirror the decisions that might be made if manually defining an alphabet based on this metric (Bin 1: CVILF, Bin 2: MWAGS, Bin 3: TPYHQ, Bin 4: NDERK). These bins mimic natural break-points in this metric, with the exception of Q being in Bin 3. The generally high performance of coevolution methods working on alignments encoded in `A1_4` suggests that it may compliment the use of the more canonical reduced-state alphabets, and that other Atchley-factor alphabets defined based on natural breaks in the data may also yield good results.

4.3.3 GCTMPCA and SCA parameters

SCA and GCTMPCA both require a single parameter (cutoff and ϵ, respectively) from the user. Experimentation with the values of these parameters was incorporated into the analyses, and was found to have a strong effect on method performance. The authors of GCTMPCA suggest 0.70 as an optimal setting for ϵ [50, 53]. This was empirically validated here by comparing performance for detecting signal at $(i, i+3)$ and $(i, i+4)$ in myoglobin using the full amino acid alphabet and two reduced-state alphabets (data not shown).

The optimal value for the SCA cutoff parameter was empirically found to be variable

Table 4.7: Top five amino acid alphabets for each method.

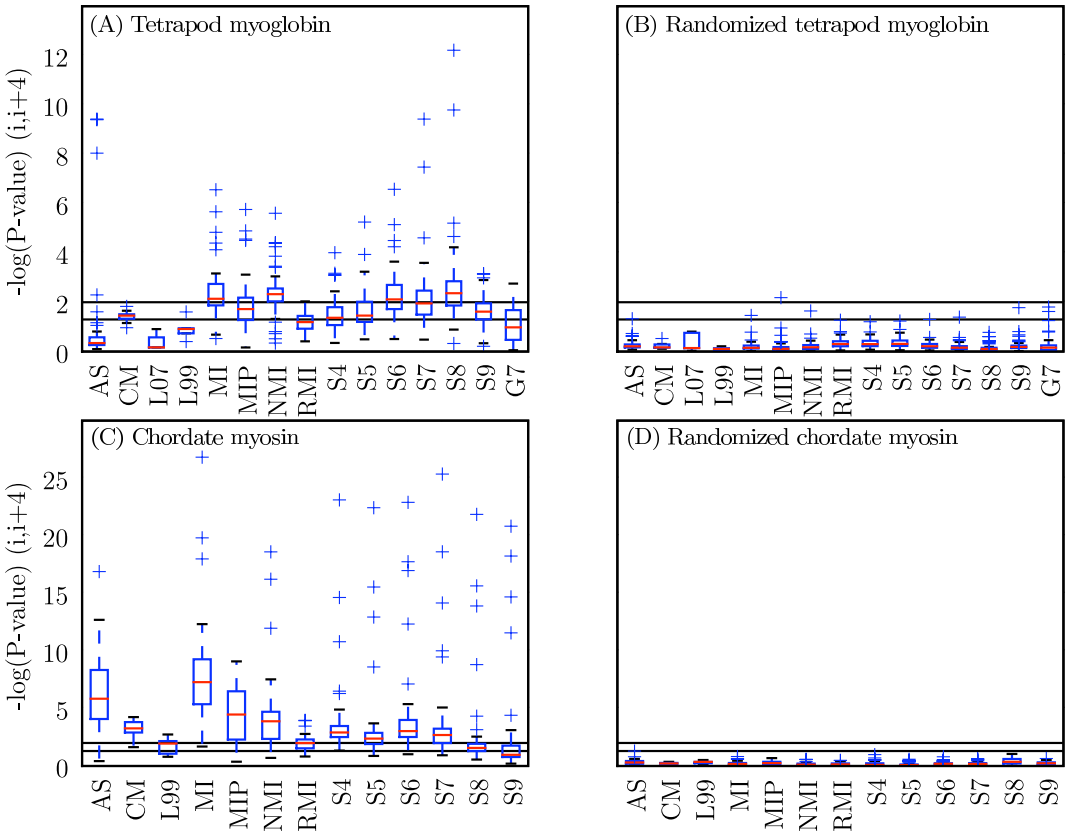
For each method and alignment combination, the top five ranking alphabets are presented. Rankings are determined based on the p-value at $(i, i+4)$ in the given alignment. Methods that did not achieve any significant p-values (at $\alpha = 0.01$) for the given alignment are not included as those rankings are not meaningful.

(A) Tetrapod Myoglobin							
Rank	MI	NMI	RMI	SCA8	MIp	GCTMPCA7	AS
1	CHARGE_HIS_2	A1_4	A3_2	CHARGE_HIS_2	CHARGE_2	A4_7	CHARGE_HIS_3
2	CHARGE_2	A1_3	A3_4	CHARGE_HIS_3	CHARGE_HIS_2	A1_10	CHARGE_2
3	CHARGE_HIS_3	A1_7	A4_4	A1_7	CHARGE_3	A4_10	CHARGE_3
4	A1_4	CHARGE_HIS_2	A5_10	A1_4	CHARGE_HIS_3	A5_5	CHARGE_HIS_2
5	CHARGE_3	CHARGE_2	A4_7	A1_2	A4_4	A5_10	POLARITY_HIS_4

(B) Chordate Myosin								
Rank	MI	NMI	RMI	SCA6	MIp	LnLCorr99	AS	CoMap
1	CHARGE_3	CHARGE_3	A4_3	CHARGE_3	A5_9	DEF99	A1_2	ORIG
2	CHARGE_2	CHARGE_2	A4_4	CHARGE_HIS_3	A2_9	A4	A5_5	A4
3	CHARGE_HIS_3	CHARGE_HIS_3	A3_3	CHARGE_2	A5_10	A5	CHARGE_3	VOLUME
4	CHARGE_HIS_2	CHARGE_HIS_2	A3_4	CHARGE_HIS_2	ORIG	A2	A2_5	A2
5	A3_9	A4_4	CHARGE_2	SIZE_2	A3_9	A1	A5_9	A1

Figure 4.6: Distribution of scores obtained with each method for all alphabets.

Modified box plots illustrate the distribution of scores obtained with a given method over all alphabets. Values are -log(p-value) at $(i, i+4)$ for each alignment. Black lines indicate statistical significance threshold of $\alpha = 0.05$ (bottom line) and $\alpha = 0.01$ (top line). Red lines indicate the median value, and the top and bottom of the boxes indicate the upper and lower quartile values, respectively. Whiskers represent the largest and smallest values within 1.5 x IQR (inter-quartile range), and pluses represent outliers, or points outside of 1.5 x IQR. Methods with more condensed distributions are those that appear more robust to alphabet choice. AS: Ancestral States; L07: LnLCorr07; L99: LnLCorr99; MI: Mutual Information; NMI: Normalized Mutual Information; RMI: Resampled Mutual Information; Sn: Statistical Coupling Analysis, cutoff = $n/10$; MIP, Corrected Mutual Information; Gn: Generalized Continuous-Time Markov Process Coevolutionary Algorithm, $\epsilon = n/10$; CM: CoMap.



(see distributions of performance by cutoff in Figure 4.6A,C) and did not match the values obtained as recommended in [59]. When applied to the alignments without alphabet reduction, the recommended steps identify 0.947 as the optimal value for myosin, and 1.0 as the optimal value for myoglobin (meaning there are not enough sequences for the analysis). The empirically-determined optimal cutoff for detecting coevolutionary signal at $(i, i+4)$ in myosin was 0.6, and in myoglobin was 0.8, when no alphabet reduction was applied. These results suggest that the steps presented in [59] will not always identify the optimal cutoff, and experimentation with the cutoff value should therefore be incorporated into applications of SCA.

4.4 Discussion

4.4.1 Tree-aware versus tree-ignorant techniques

Tree-ignorant coevolution algorithms matched or out-performed the tree-aware coevolution algorithms for identifying coevolving positions in alpha helices. With the exception of CoMap, the tree-aware methods also generally required vastly more compute time than the tree-ignorant methods, despite the slowest of these algorithms (GCTMPCA and LnL-Corr) being implemented in C while the other algorithms are predominantly implemented in Python (C implementations are frequently orders of magnitude faster than equivalent implementations in Python). The C++ implementation of CoMap was the fastest of the methods. In some cases, standard Mutual Information out-performed all other algorithms (Table 4.3C). These observations suggest that when transformed in a manner that adjusts for shared ancestry, as demonstrated here, tree-ignorant methods can be more reliable than tree-aware methods. Although including additional sequences with short branch lengths would likely result in a convergence in performance of the two method classes, the computational performance advantage of tree-ignorant methods establishes them as the metric of choice for comprehensive surveys of molecular coevolution.

Figure 4.6 illustrates that with the correct parameter choices (i.e., amino acid alphabet and free parameters to algorithms) nearly all of the methods can identify the $(i, i+4)$ stacked residues in the two alpha helical proteins. In most applications however, the 'correct' amino acid alphabet will not be known prior to the analysis, and it will not be practical to optimize the free parameters and alphabet. For example, if trying to infer protein-protein interactions on a genome-wide scale based on coevolutionary relationships, true positives would not be known. Even if a subset of known interactions could be used for optimization, the compute time would be prohibitive. Methods that are fast and robust to parameter choice are preferable. In myoglobin, MI and NMI stand out in this respect: the distribution of scores achieved by these two metrics is skewed toward statistical significance compared with the other methods (Figure 4.6A). The same appears true in myosin, with RMI additionally appearing extremely robust to alphabet choice but achieving a lower median p-value compared to MI and NMI (Figure 4.6C). AS, MIp, and CoMap generally perform well on myosin, although some alphabet choices with AS and MIp can lead to very poor performance. The variance introduced by the cutoff parameter choice with SCA is large in both alignments, and GCTMPCA and LnLCorr do not appear robust to alphabet choice and achieve median p-values below the $\alpha = 0.05$ significance threshold, suggesting that these methods would be less useful in cases where the choice of alphabet cannot be optimized or confidently specified *a priori*.

The ability of tree-ignorant methods, and MI in particular, to out-perform tree-aware methods is a seemingly surprising result given previous demonstrations of high false positives for this method class. These results demonstrate that the t-test transformation provides sufficient control for the shared ancestry cause of associations. Because every column (position) in an alignment has the same underlying relationship among the rows (sequences), every estimated pairwise coevolution score will have this influence in common. Pairs with a true coevolutionary history will have both that shared ancestry and the additional influence

of coevolution, and so should stand out from the non-coevolving pairs.

4.4.2 Alphabet size

Larger alphabets frequently improve performance of coevolution algorithms (MI, NMI, and MIp in particular) on the myosin data set, but there is no obvious relationship between alphabet size and coevolution algorithm performance on the myoglobin data. The differences observed on myoglobin and myosin may be explainable by differences in the quantity of input data. The myosin rod data set has more sequences than the myoglobin data set, and therefore more states to observe when calculating a pairwise coevolution score between two columns. Grouping residues via reduced alphabets is expected to improve statistical performance by providing a clearer picture of the distribution of residue types. The additional observations (sequences) present in the myosin alignment appear sufficient to describe the residue distribution without recoding, and the information loss associated with recoding in fewer states may therefore result in decreased statistical performance. The lack of a significant positive or negative correlation between alphabet size and statistical performance on myoglobin suggests that (unlike on myosin) higher-state alphabets are not better, and that a generic benefit associated with fewer-state alphabets may only arise on alignments with less sequences than the myoglobin data.

An alternative explanation for the positive correlation between alphabet size and performance in myosin is that the Atchley-factor alphabets are not useful categorizations of the data. If true, as the number of states increases (and the alphabets become more similar to the full amino acid alphabet) they should perform better. The strongest correlations however are achieved with the A1 and A5 alphabets, which are the Atchley-factor alphabets that are expected to perform the best (because they model charge and polarity, the features known to be important to interactions in alpha helices). Some of the best performances overall were achieved using these alphabets (see Table 4.7), so this explanation appears unlikely.

Much previous work, including [52] which used the same myoglobin alignment and tree, has focused on the assumption that smaller alphabets are generally better. If true, a negative correlation should exist between alphabet size and method performance. This is not observed, suggesting that in practice smaller alphabets do not necessarily improve statistical performance.

4.4.3 Alphabet type

If probing for coevolution resulting from a specific type of physicochemical interaction, a reduced alphabet which models that physicochemical property in fewer states should increase power. There is serious risk, however, of losing power by using the incorrect reduced alphabet. For example, the p-value for identifying signal at $(i, i+4)$ with MI increased (i.e., became less significant) from 1.5×10^{-27} with the `CHARGE_3` alphabet, to 1.0×10^{-3} with the `SIZE_2` alphabet. For all methods analyzed on myosin with reduced alphabets — AS, MI, NMI, RMI, MIp, and SCA — the `SIZE_2` alphabet always resulted in less significant p-values than the `CHARGE_3` alphabet. This illustrates that recoding with the correct alphabet (`CHARGE_3` here) will highlight a coevolutionary relationship based on that property, but that recoding with an incorrect alphabet (`SIZE_2` here) will obscure the coevolutionary signal. Non-sensical alphabet recoding should destroy the coevolutionary signal. If the interaction type (e.g., charge, size) is not known ahead of time, using the unreduced alphabet can provide useful results and should be safer than making an uninformed choice of reduced alphabet.

The importance of choosing the 'correct' reduced alphabet is illustrated by variable performance of a single algorithm even when the alphabet choices have the same number of states. For detecting coevolutionary signal between stacked residues in alpha helices, the reduced alphabets which model charge/polarity perform the best. These include the four `CHARGE_*` alphabets (`CHARGE_2`, `CHARGE_3`, `CHARGE_HIS_2`, `CHARGE_HIS_3`) and the alphabets based on Atchley factors 1 and 5. These would not be the best alphabets if the interactions

were based on a different physicochemical property.

The success of charge-based alphabets for identifying stacked positions in alpha helices confirms the previous biochemical and statistical observations, and suggests that alphabet recoding can be applied to probe interaction types. For example, if a coevolutionary interaction is apparent when working with a size-based alphabet, but disappears when working with a charge-based alphabet, the interaction is likely more related to size than charge. This is observed when looking at $(i, i+1)$ pairs in the myosin rod using MI with the `SIZE_2` and `CHARGE_3` alphabets (data not shown). Coevolution on the basis of side-chain volume between $(i, i+1)$ pairs, which form the set of residues which are presumably closest to one-another in the folded protein, seems likely. Classification of interaction types based on these characteristics is often likely to be overly simplistic, but probing interactions this way may serve as a starting point for more in-depth analysis. Care should be taken however to avoid spurious conclusions arising from multiple comparisons when repeatedly applying a coevolution algorithm to the same alignment recoded with multiple alphabets.

4.4.4 Utility of alpha-helical proteins for comparing coevolution algorithms

This chapter presents a comparison of nine coevolution detection methods on two different molecules which are entirely (myosin rod) or almost entirely (myoglobin) alpha helical. The two alignments are different in terms of number of sequences, diversity of sequences, and sequence length, yet in both cases the algorithms are able to detect (to varying degrees) the periodicity of the alpha helix. The coevolution of stacked residues in protein alpha helices, identifiable computationally and supported by double-mutant studies, makes alignments of alpha helices useful for comparing techniques for detecting coevolution. Unfortunately, this approach cannot be generalized to beta sheets, because the length of the beta strands is highly variable and there is thus no consistent periodic signal expected to be consistent across different proteins.

The CHARGE_HIS_2 alphabet, where residues are recoded to charged or uncharged with histidine counted as charged, was consistently among the best alphabets for detecting coevolution at $(i, i+3)$ and $(i, i+4)$ in myoglobin. Figure 4.7 reports p-values for ten algorithms for detecting coevolution at $(i, i+n)$, for n of 1 through 20, where alignments were recoded in the CHARGE_HIS_2 alphabet. (In cases where this specific alphabet recoding was not applicable – LnLCorr and CoMap – results represent the DEF99 and ORIG data sets, respectively.) These graphs are a useful visualization of the relative performance of different algorithms or parameters because positive and negative controls are built-in. Since myoglobin is composed of non-contiguous alpha helices, a signal should be visible at $n = 3$ $(i, i+3)$ and $n = 4$ $(i, i+4)$, and not at other values of n. We see that most of the methods identify at least one of these with statistical significance, even after Bonferroni correction for multiple comparisons. AS appears to yield a false positive at $n = 19$.

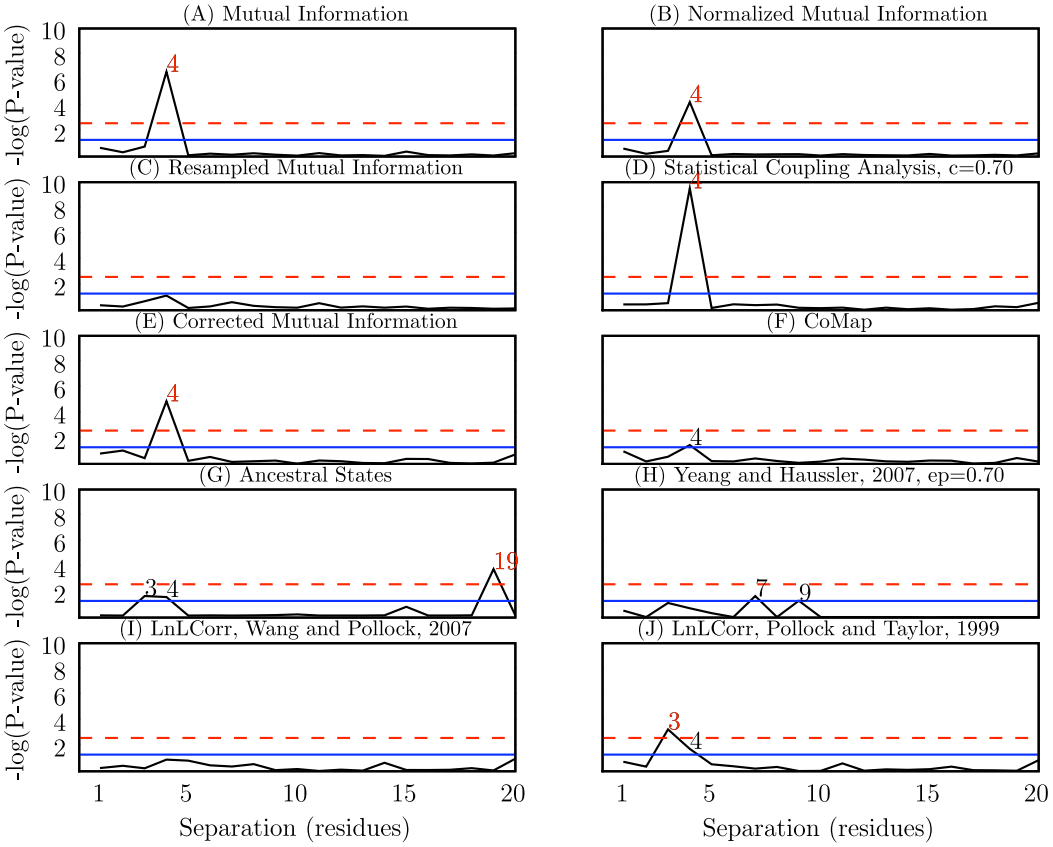
Figure 4.8 reports p-values for Mutual Information at separations of 1 through 50 ($(i, i+1)$ through $(i, i+50)$) in the myosin rod with no alphabet recoding. Since the myosin rod is a contiguous alpha helix and the input alignment contains more data than the myoglobin alignment (in terms of sequence length and number of sequences), some weaker interactions become apparent. While $(i, i+3)$ and $(i, i+4)$ still generate the most significant p-values, other pairs nearby in sequence also show significant p-values. Notably, multiples of seven between seven and thirty-five obtain p-values suggestive of statistical significance, likely resulting from longer-distance stacking interactions in the myosin rod. Alpha helices allow for a comparison of the power of each method not only to detect the strongly coevolving pair sets, but possibly sets of pairs undergoing weaker coevolution as well.

4.4.5 Long-distance signal in Myosin rod

Figure 4.8 illustrates that coevolutionary signal is periodically detectable at multiples of seven to distances of thirty-five residues in the myosin rod. Two possible explanations for this signal are: (i) that direct interactions cause coevolution between the residue pairs, or (ii)

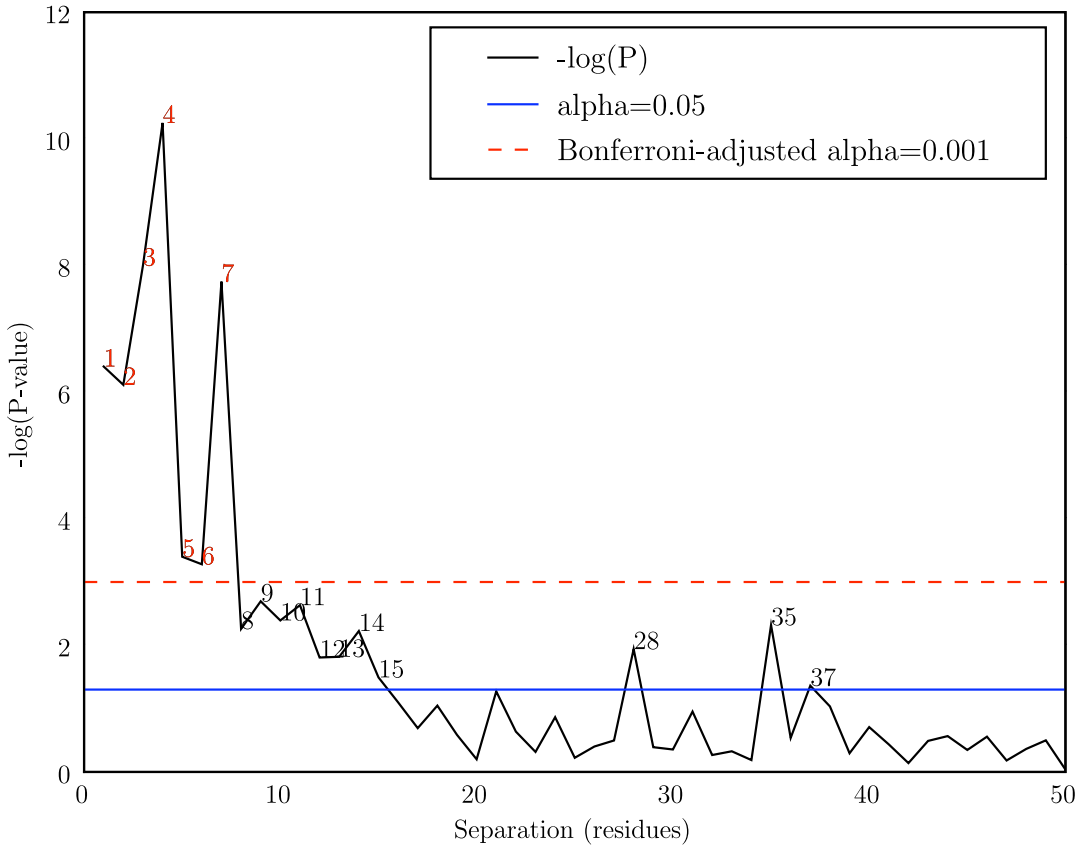
Figure 4.7: Coevolutionary signal in myoglobin at $(i, i+n)$ pairs.

-log(p-values) are presented for each separation of n residues $(i, i+n)$ for n ranging from 1 to 20. Biochemical studies of alpha helices suggest that a statistically significant signal should be detectable at $(i, i+3)$ and $(i, i+4)$. Black digits indicate values of n significant at $\alpha = 0.05$, and red digits indicate values of n significant after Bonferroni adjustment for multiple comparisons, $\alpha = 0.0025$. Each graph (A-J) represents performance with a different algorithm. Where applicable, alignments were recoded with the `CHARGE_HIS_2` alphabet, which consistently yielded among the best results. When recoding with `CHARGE_HIS_2` was not applicable, LnLCorr and CoMap, the `DEF99` and `ORIG` data sets (respectively) are presented.



that indirect interactions cause coevolution between the residue pairs (e.g., residues 1 and 7 directly interact and coevolve, as do residues 7 and 14, and an indirect correlation results in

Figure 4.8: Mutual Information coevolutionary signal in the myosin rod.
-log(p-values) are presented for each separation of n residues $(i, i+n)$ for n ranging from 1 to 50. Biochemical studies of alpha helices suggest that a statistically significant signal should be detectable at $(i, i+3)$ and $(i, i+4)$. Black digits indicate values of n significant at $\alpha = 0.05$, and red digits indicate values of n significant after Bonferroni adjustment for multiple comparisons, $\alpha = 0.001$. No alphabet recoding was applied before analysis with Mutual Information. In addition to $n = 3$ and $n = 4$, multiples of seven (7,14,21,28,35) have suggestive p-values through $n = 35$.



a weaker signal between residues 1 and 14). The data support the latter explanation. The long distance interactions detected in myosin should not be expected in myoglobin, because unlike myosin, the alpha helices in myoglobin are not contiguous.

The period of the alpha helix is 3.6. Direct interactions due to stacking of more distant

pairs should therefore occur at multiples of 7.2 (rounded to 7, 14, 22, 29, 36, ...). Instead, we see distant interactions at multiples of 7.0 (separations of 7, 14, 21, 28, and 35 residues). This suggests that the direct interactions occur locally: we see a signal at 35, not at 36, because it is the result of a series of direct local interactions (effectively) 7 positions apart.

The myosin rod is composed of 28 residue homologous units, each composed of four heptad repeats. These repeat regions could be an alternative explanation for the signal at multiples of seven, but this seems less likely. Following evolution of the myosin rod, it is not clear what would drive coevolution of these units, and because conservation reduces MI for a pair of positions, the common ancestry followed by conservation would have the opposite effect.

The process of coevolution is not well understood and there has been very little conclusive evidence about the forces driving coevolution in proteins. In the case of alpha helices, it appears that distant coevolutionary relationships are the result of indirect correlations rather than residue stacking.

CHAPTER V

OPTIMIZING JOINT-ENTROPY-NORMALIZED MUTUAL INFORMATION

5.1 Background

While a single method does not emerge as the overall best choice from Chapter IV, joint-entropy-normalized mutual information (NMI) has become my method-of-choice for applied coevolutionary analyses. *I stress that the results obtained from the alpha-helix-based method comparison do not suggest or imply that accounting for phylogeny is unimportant in coevolutionary analyses.* Rather, while tree-aware methods such as LnLCorr and GCTM-PCA are surely the most elegant choices available for modeling and studying the process of coevolution in biological sequences, for applications which attempt to make predictions based on coevolution metrics, the metrics that explicitly incorporate the phylogenetic tree typically appear to be less sensitive than those that do not. As an important corollary, comparing scores to a distribution of scores with the same underlying phylogeny can avoid false positives arising from phylogenetic effects, as illustrated by our negative controls.

For the downstream applications in Chapter VI and Appendix II, I have chosen NMI as my primary metric for identifying covarying positions. Because of the influence of phylogeny, NMI results are treated skeptically, and in light of the fact that covariation does not imply coevolution. Results are reviewed in the context of phylogeny and structural data (when available) to make predictions about when a pattern of covariation between a pair of positions may have been caused by coevolution of those positions.

In the current chapter, the information gained from the alpha-helix-based comparison is assembled into practical steps that I have used to apply coevolutionary analyses to protein sequence alignments. Guidelines are presented for choosing a suitable set of sequences, and post-processing coevolution matrices to reduce signal arising from phylogeny and random

sequence variation. The steps presented here form the basis of the analyses in Chapter VI. I begin by introducing in more detail the joint-entropy-normalized mutual information metric.

5.1.1 Joint-entropy-normalized mutual information

The Shannon entropy (H) of a position a in a multiple sequence alignment is a measure of its variability. For a set of discrete states $X = \{x_1, x_2, \ldots x_n\}$, Shannon entropy is computed as:

$$H_a = -\sum_{i=1}^{n} p(x_i) \cdot \log_2 p(x_i). \tag{5.1}$$

In the case of protein sequence alignments, the states are the amino acid residues, and the probability for observing each state ($p(x_i)$) is computed as the frequency of that state at position a in the alignment. In practice, the base of the logarithm is not important as long as it is consistent; conventionally, base 2 is used making *bits* the units of H. If one of the states is not observed at position a, as is nearly always the case in protein sequence alignments, it is taken that $0 \log_2 0 = 0$. The entropy at a position decreases with conservation, so a perfectly conserved position has $H = 0$.

The Shannon entropy for a pair of positions a and b, or the joint entropy, is computed similarly except that the set of states is now all possible pairs of states: $XY = \{x_1y_1, x_1y_2, \ldots x_my_n\}$. The joint entropy calculation is:

$$H_{ab} = -\sum_{i=1}^{m}\sum_{j=1}^{n} p(x_iy_j) \cdot \log_2 p(x_iy_j). \tag{5.2}$$

In the context of a multiple sequence alignment, the Mutual Information for a pair of positions a and b (MI_{ab}) is a measure of the degree to which knowing the identity of the residue at position a informs you of the residue at position b (or vice versa: $MI_{ab} = MI_{ba}$)[27].

[27]More generally, MI is a measure of the degree to which knowing the value of one discrete random variable informs you of the value of another discrete random variable.

MI_{ab} is calculated as the sum of the Shannon entropies (H_a and H_b) at each position minus the joint entropy of the positions (H_{ab}).

$$MI_{ab} = H_a + H_b - H_{ab} \tag{5.3}$$

Joint-entropy normalized Mutation Information for positions a and b (NMI_{ab}) is simply:

$$NMI_{ab} = \frac{MI_{ab}}{H_{ab}}. \tag{5.4}$$

In addition to consistently showing up among the top-performing methods, there are a number of convenient features associated with NMI.

5.1.2 Convenient features of NMI

5.1.2.1 NMI removes the effect of evolutionary rate heterogeneity among sites

Because mutual information is normalized by the joint entropy of the pair of sites, rate heterogeneity among sites is controlled for, and therefore does not effect the covariation statistic. It has been noted that rate heterogeneity is an inherent problem with many covariation algorithms, including MI and SCA. [112]

5.1.2.2 Minimum and maximum NMI scores are clearly defined

As with standard mutual information (MI), the minimum NMI value is 0.0. Since the maximum mutual information score for a pair of positions a and b is obtained when the residue patterns of the two positions are identical, and are therefore both identical to the pattern of the combined positions,

$$H_a = H_b = H_{ab} \tag{5.5}$$

If we call this quantity H_0, the NMI calculation follows as:

$$MI_{ab} = H_a + H_b - H_{ab} = H_0 + H_0 - H_0 = H_0 \tag{5.6}$$

$$NMI_{ab} = \frac{MI_{ab}}{H_{ab}} = \frac{H_0}{H_0} = 1.0 \tag{5.7}$$

NMI therefore has a maximum value of 1.0, and is interpreted as the proportion of the maximum possible MI at a pair of positions which is observed. The clear upper and lower bounds on NMI make it a convenient statistic to work with.

5.1.2.3 NMI does not require an evolutionary model

Because NMI does not require an evolutionary model (as the tree-aware methods do) it is not possible to misspecify the evolutionary model. Additionally, gap characters do not pose a problem for the analysis, as they can be treated simply as any other alignment character. It is not clear how to properly handle gap characters in coevolutionary analyses, but the flexibility afforded by not needing to model gaps is convenient. Gap handling is further discussed below.

5.1.2.4 NMI is fast

NMI is relatively fast to compute for all pairs of positions in an alignment. When run on the myoglobin and myosin data sets, it was consistently among the fastest methods.

5.2 Alignment pre-processing with DivergentSet

DivergentSet [133] is a web-based tool which, given a multiple sequence alignment, can be used to generate a subalignment containing no pair of sequences with greater than n% sequence identity, for a user-specified value of n. The alpha-helix-based study presented in Chapter IV was designed to compare methods on alignments containing both closely and distantly related sequences, but in practice it is useful to see if selecting a divergent subset

of sequences can improve the performance of covariation detection. In a follow-up to that study presented here, I compared the ability of NMI to detect the periodicity of the alpha helix with the chordate myosin rod alignment and the corresponding randomized alignment, to observe the difference in performance resulting from the application of DivergentSet. The myoglobin alignment was not included in this study because it contained too few sequences to be divided into subalignments while still retaining a sufficient number of sequences.

The myosin alignment was passed through DivergentSet with percent-sequence-identity thresholds of 0.95 (the ds95 alignment) and 0.99 (the ds99 alignment). The ds95 and ds99 alignments contained 28 and 56 sequences, respectively, compared with the 114 sequences in the full alignment. The randomized myosin alignment was pruned to contain only the sequences in the ds95 and ds99 alignments, to create two randomized alignments for a negative control. Performance of NMI with its top-performing alphabets (see Table 4.7) was compared on these alignments using the t-test and AUC alpha-helix-based comparisons described in Sections 4.2.5 and 4.2.6, respectively.

5.3 Post-processing filters

In this section several coevolution result post-processing filters are introduced and compared using the alpha-helix-based evaluations. The filters discussed are gap percent filtering (introduced here), multiple interdependency filtering [107], and non-parsimony informative filtering [134]. A key of abbreviations, including brief descriptions of the filters, is presented in Table 5.1.

5.3.1 Gap percent filtering

Alignment positions containing gap characters have frequently been ignored in coevolutionary analyses (e.g., [106, 135]), in part because it is not clear what it would mean for a substitution at one position to be compensated by a deletion at another positions (or vice versa). However, excluding a position from an analysis because a small percentage of

Table 5.1: Filter abbreviations.
Abbreviations and brief definitions of the post-processing filters described in Section 5.3.

Abbreviation	Name	Alignment positions excluded
None	No post-processing	None
10p_gap	Gap percent filter	Positions containing greater than 10% gap characters.
mid	Multiple interdependency filter	Positions which significantly covary with more than one other position. Presented in [107].
md$x$mc$y$ (e.g., md2mc2 and md2mc10p)	Non-parsimony-informative filter	Positions containing less than x different characters (md: minimum differences) which appear in at least y sequences (mc: minimum counts). Most common variants are md2mc2 (md and mc = 2) and md2mc10p (md = 2 and mc = 10% of the total number of sequences).
md$x$mc$y$_strict	Strict non-parsimony informative filter	Positions containing less that x different characters, and where ANY characters present appear in less than y sequences. md2mc2_strict is the filter presented in [134].

the sequences contain a gap at that position can greatly reduce the number of positions included. To address this limitation, the analyses presented here treat gap characters the same as all other alignment characters in the coevolutionary analysis[28]. Positions composed of 10% or more gap character are subsequently excluded from a coevolutionary analysis as a post-processing step. This compromise allows for inclusion of more data in the coevolution analysis, while effectively ignoring regions of the alignment which do not appear to align well or where indels have occurred in many sequences. This filter is referred to as 10p_gap. This strategy has been applied in previous studies, such as Fodor and Aldrich [112].

5.3.2 Multiple interdependency filtering

Multiple interdependency filtering of coevolution results (abbreviated mid) has been suggested to reduce the covariation signal arising from phylogenetic effects rather than functional interactions [107]. After a coevolutionary analysis, this filter is applied to exclude all positions which significantly covary with more than one other position in the alignment, reasoning that positions which covary with many other positions are likely to represent

[28]Note that this increases the size of the unreduced protein alphabet, for example, from 20 to 21 characters: the twenty canonical amino acid residues plus the - character indicating a gap.

covariation arising from the pattern of the phylogeny. For example, if two clusters of closely related sequences are connected by a long branch, many substitutions are likely on the long branch simply because of the amount of divergence occurring along that branch. Positions which incur substitutions on that branch but are generally conserved otherwise will covary with one another, and this covariation would be far more likely to be a result of the structure of the tree than to be of functional significance. Ignoring these positions is suggested to be a way to control for the phylogeny without explicitly incorporating the phylogeny in a covariation statistic.

5.3.3 Non-parsimony-informative filtering

Non-parsimony-informative filtering is introduced as a method which can be applied to remove coevolutionary signal arising from stochastic sequence variation [134]. This filter requires that to be included in a coevolutionary analysis, a position a must contain at least two different characters (`minimum_differences` $= 2$), and each character which appears at position a must appear in at least two sequences (`minimum_counts` $= 2$). This is intended to ignore positions which either exhibit no variation (which NMI already treats appropriately), and positions which incur substitutions at a very high rate and are therefore unlikely to be under selective pressure, which is a prerequisite for coevolution. Filtering non-parsimony-informative sites from coevolutionary analyses was shown to significantly reduce the false positive rate when using coevolution to identify groups of functionally relevant positions in GroEL[134].

A subtle point related to parsimony informative filtering is whether the filtering should be applied to alignments before or after recoding them in reduced-state amino acid alphabets (when applicable) for the covariation analysis. In practice, there was little difference in the performance when both variations were tested in alpha-helix-based comparisons, although slightly better performance was frequently achieved when filtering was applied to already-recoded alignments (data not shown). Because most variants on these filters in-

volved changes to `minimum_counts` (leaving `minimum_differences` as its default value), it makes intuitive sense that when working in a recoded alphabet we should be interested in the number of times each state appears, rather than the number of times each specific amino acid appears. All work presented here that incorporates both non-parsimony-informative filtering and reduced-state amino acid alphabets have the alignments recoded before filtering is applied.

5.3.4 Filter evaluation methods

The non-parsimony-informative and multiple interdependency filters were compared with both the t-test-based and AUC precision/recall alpha-helix-based evaluations previously used to compare algorithms. In these comparisons, only the unreduced protein alphabet and the top five reduced-state alphabets for NMI on the myosin rod *or* myoglobin alignment are included (see Table 4.7). Data is presented to compare the filters (and combinations of filters) to one another and the unfiltered results. Details on the t-test-based evaluations are discussed in Section 4.2.5, and details of the AUC precision/recall evaluations are discussed in Section 4.2.6. Neither of these alignments contain any gaps, so the percent gap filter is not evaluated here.

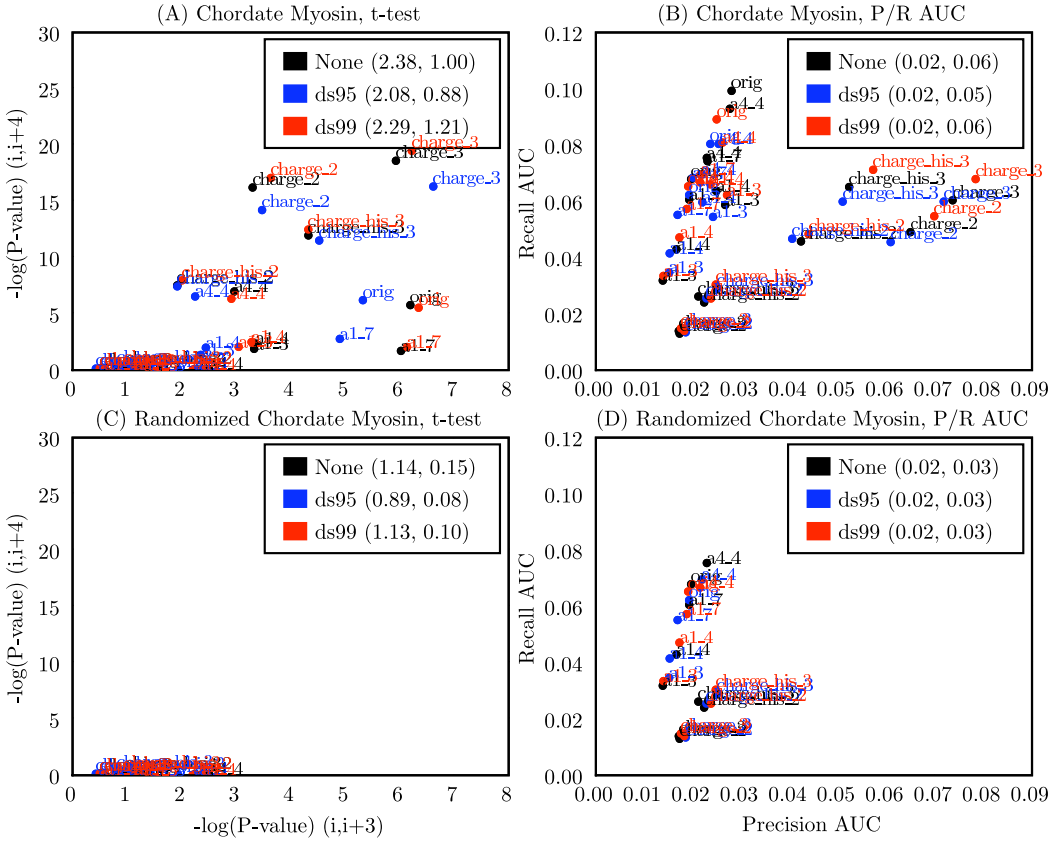
5.4 DivergentSet and filter evaluation results

5.4.1 DivergentSet

Very little difference is observed when comparing the full myosin rod alignment with the DivergentSet (ds95 and ds99) variants (see Figure 5.1). Normalization of MI with joint entropy accounts for the amount of sequence variation at a specific position, and thereby appears to control for the clusters of closely related sequences that are pruned by DivergentSet. In a future study, these results will be compared with the results obtained with standard MI, as this would be a way to directly test this hypothesis.

Figure 5.1: DivergentSet comparison results.

NMI results for the myosin and randomized myosin alignments are presented and compared to results following DivergentSet application to the alignments in t-test and AUC-based analyses (presented in Sections 4.2.5 and 4.2.6, respectively).

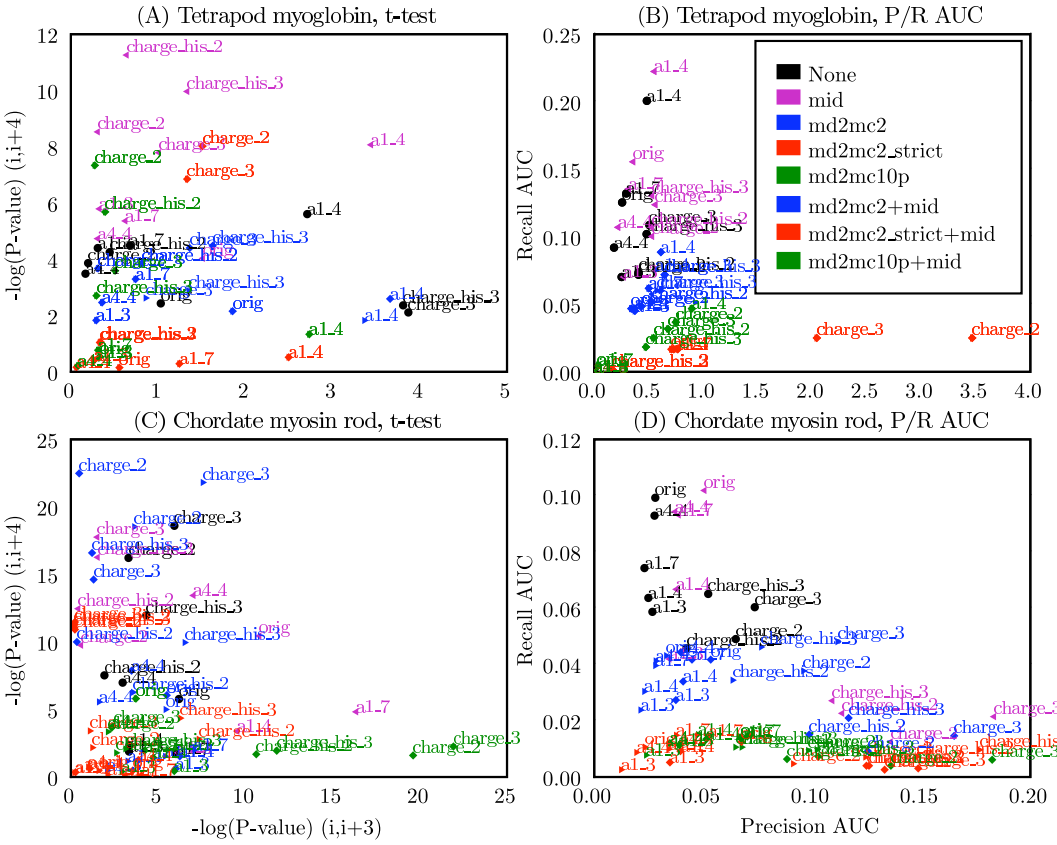


5.4.2 Multiple interdependency filtering

Multiple interdependency filtering is effective for increasing the median AUC precision of NMI in alpha helix-based comparisons (Figure 5.2), and in several cases, it is responsible for the highest AUC precision values achieved. Multiple interdependency filtering is also associated with large drops in recall. This is expected as covariation arising from phylogeny and covariation arising to conserve function are not mutually exclusive.

Figure 5.2: Filter comparison results (positive control).

Raw NMI results are presented and compared to results following post-processing filters for t-test and AUC-based analyses (presented in Sections 4.2.5 and 4.2.6, respectively). mid results are left-pointing triangles, non-parsimony-informative results are right-pointing triangles, and results combining mid and non-parsimony results are diamonds.



In many cases, positions with multiple significant covarying pairs are likely to be interesting. For example, Gloor *et al.* [136] presents evidence that positions which are observed to covary with multiple other positions are more likely to be found at protein-protein interaction interfaces or protein active sites than those that coevolve with only one other position. The Ranganathan group has provided data suggesting that positions which co-

vary with more than one other position may be involved in energetic pathways that are important for allosteric interactions in proteins [58, 59]. Because there are hypotheses regarding the functional relevance of positions which covary with multiple other positions, and this filter rules those positions out by definition, I consider this filter to be useful only when very high precision is required at the expense of recall, or when the structure of the phylogeny is very likely to cause a problem. In the latter case, however, it is probably more appropriate to work with a coevolution metric that incorporates phylogeny.

5.4.3 Non-parsimony informative filtering

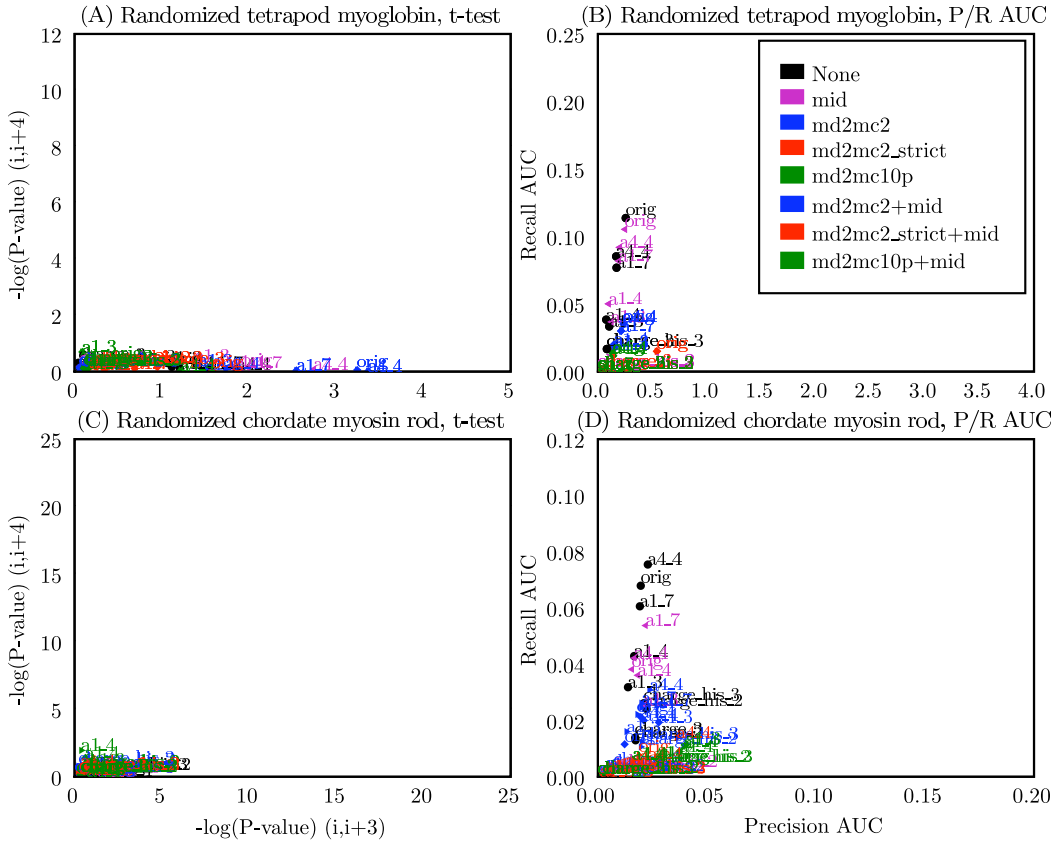
In alpha-helix-based evaluations of this filter (and variations on it), precision was often greatly increased but with a significant decrease in recall (Figure 5.2). The best balance of precision and recall was frequently achieved with the md2mc2 variant of this filter. In subsequent chapters, I frequently apply the md2mc2 filter, as well as the md2mc2_strict and md2mc10p variants of this filter.

5.5 Summary

In this Chapter, I have reviewed the joint-entropy-normalized mutual information coevolution metric (NMI), and several processing steps for multiple sequence alignments and coevolutionary analysis results. In Chapter VI and Appendix II, I apply NMI and the non-parsimony-informative filters to make inferences about protein-protein interactions.

Figure 5.3: Filter comparison results (negative control).

Raw NMI results are presented and compared to results following post-processing filters in the randomized alignment variants for t-test and AUC-based analyses (presented in Sections 4.2.5 and 4.2.6, respectively). mid results are left-pointing triangles, non-parsimony-informative results are right-pointing triangles, and results combining mid and non-parsimony results are diamonds.



CHAPTER VI

SEQUENCE COVARIATION IN THE TYPE VI SECRETION SYSTEM

6.1 Background

Modeling molecular complexes is an early step in designing drugs and is important in the detailed understanding of how biological processes emerge from chemical interactions. Assembling a model using biochemical techniques is a slow and often tedious process, and there is much interest in algorithms to speed up (or eventually replace) this process. Just as macroscopic structures such as the bills of hummingbirds and the corolla tubes of flowering plants have coevolved to form highly specific functional interactions, it is expected that proteins involved in molecular complexes coevolve to conserve critical interactions. In this chapter, sequence covariation is identified to infer sequence coevolution, and thereby physical interactions between the component proteins of the Type VI Secretion System (T6SS), a molecular syringe implicated in the pathogenicity of diverse Gram-negative bacteria.

The two goals of this project are: (1) to identify the conserved pairwise protein-protein interactions in functional Type VI Secretion System complexes; and (2) to test the hypothesis that sequence covariation can suggest pairs of proteins that are likely to be involved in pairwise interactions, thereby reducing the cost (in dollars and person-hours) required to identify pairwise interactions in molecular complexes. This project requires both computational and biochemical experiments. Here I present the results of the computational aspects of this project, and preliminary data resulting from the biochemical studies which are currently being performed by collaborators in the Marcelo Sousa Laboratory at the University of Colorado at Boulder. The experimental results will allow us to develop a detailed model of the T6SS, fulfilling the first goal, and to perform a statistical (permutation-based) test of our hypothesis, fulfilling the second goal.

6.1.1 The Type VI Secretion System

The Type VI Secretion System, or T6SS, is a recent discovery in the functionally-related Gram-negative bacterial secretion system family. Evidence for the T6SS began to surface in around 1996, and the system wasn't named T6SS until 2006. Bingle *et al.* [137] provide a review of the T6SS, including the history of its discovery. These secretion systems are literal molecular syringes, which span the two membranes of the bacterial cell, intercellular space, and the membrane of a host cell, to transfer effector proteins from the bacterial cell into the host cell. The identities of the effector proteins and their specific roles in the host cells are still the subject of much research, although some evidence has associated *V. cholerae* T6SS effectors with actin cross-linking in mammalian microphages, and *A. hydrophilia* T6SS effectors with increased rates of apoptosis in host cells [138].

The T6SS is implicated in the pathogenicity of many medically relevant species including *Vibrio cholerae*, *Pseudomonas aeruginosa*, and *Samonella enterica* [139]. A recent computational analysis suggests the presence of the T6SS genes in 42 pathogenic proteobacteria species [140], and another study found T6SS clusters in over a quarter of the sequenced bacterial genomes at the time [137]. Different species typically contain about 12 to 20 genes in a T6SS cluster [138, 140], and many species contain several clusters of T6SS genes in their genomes (see [137] supplementary material). For example, in this study I have identified four clusters of T6SS proteins in *Yersinia pseudotuberculosis* (see Section 6.2.1), and this is not at all uncommon. The evolutionary relationship between these sequences, described previously [137] and confirmed in my analysis, suggest that the gene clusters have frequently undergone horizontal transfer rather than duplication events within a single species.

At this stage, very little is known about the physical interactions between the proteins suspected to be involved in the T6SS complex, although detailed information is beginning to surface (e.g., [141, 142]). A goal of this study is to augment the current knowledge of the protein-protein interactions important in functional T6SS complexes.

6.1.2 Sequence coevolution and protein-protein interactions

Inferring protein-protein interactions via coevolving positions in multiple sequence alignments has the potential to greatly reduce the cost of identifying interactions and modeling molecular complexes. Biochemical data has suggested coevolution of interacting proteins, for example [143], which analyzes the coevolution of lutropin and follitropin receptors with their respective ligands, luteinizing hormone and follicle-stimulating hormone. The authors showed that chimeric receptors or hormones could be made to reverse the receptor specificity to the other binding partner (e.g., lutropin receptor could be made to bind the follicle-stimulating hormone), showing that is is possible for a receptor (or hormone) to change specificity with relatively few amino acid substitutions. Based on this analysis, the authors propose that the two homologous receptors coevolved with their ligands from an ancestral receptor/ligand pair. This study biochemically illustrates the phenomena of intermolecular coevolution to tune binding specificity.

If coevolution between binding partners occurs, that coevolutionary signal should be present in multiple sequence alignments. Identifying that signal should allow for generating hypotheses about potential interactions. An attempt to achieve this was presented in [144], where the authors evaluated the ability of a correlation-coefficient-based coevolution metric to discriminate between physically-interacting and non-physically-interacting pairs of proteins, defined by several test data sets. The authors show that their method (termed *in silico* two-hybrid) could generally identify the physically interacting pairs, although precision appeared to be low.

In a more recent study, Yeang and Haussler [53] applied their coevolution algorithm (GCTMPCA, included in the study presented in Chapter IV) to detect coevolution between pairs of domains in Pfam. They found that coevolving positions were over-represented in functionally related proteins, at protein-protein interfaces, and in functionally important regions of the proteins.

Related approaches have involved looking for correlations in evolutionary distances between proteins in a sequence alignment (e.g., [60, 145–147], although a recent article has questioned this methodology [148]); and several other approaches including phylogenetic profiling, which were reviewed and compared in [149]. These methods focus on detecting functionally related groups of proteins, while the coevolutionary analysis presented here attempts to identify physical interactions. Sequence co-occurrence (although not true phylogenetic profiling) has been applied in this study in addition to sequence covariation/coevolution (see Table 6.1).

The analysis presented here differs from the previous work in that predictions are being made about proteins where very little is known *a priori* about physical interactions, and where predictions will be subsequently tested biochemically. The previous studies have instead tested against a gold-standard derived for that purpose. This difference is similar to the intrinsic versus extrinsic evaluation dichotomy discussed in Section III, and will allow for a very convincing statistical test of the hypothesis that sequence covariation can be used to infer physical interactions. This will be discussed further in Section 6.2.5.

6.2 Methods

Multiple sequence alignments of suspected T6SS proteins were analyzed for covariation in order to identify which proteins may be involved in physical interactions with one another. The T6SS complex appears to have been frequently horizontally transferred [137], which leads to significant challenges in compiling sets of orthologous sequences and particularly in pairing sequences which should be matched in covariation analyses (the sequence pairing problem). To avoid these issues as much as possible, close manual review of all sequences was incorporated into the sequence compilation and pairing process. Details of the sequence compilation procedure and the subsequent co-occurrence and covariation analyses are presented here.

6.2.1 Sequence compilation and alignment

Protein sequences for 21 *Salmonella enterica* serovar typhimurium (stm) genes were obtained (via their accession numbers from the KEGG database [150]. The genes and their corresponding accession numbers are: SciA (STM0266); SciB (STM0267); SciC (STM0268); SciD (STM0269); SciE (STM0270); SciF (STM0271); SciG (STM0272); SciH (STM0273); SciI (STM0274); SciK (STM0276); SciM (STM0279); SciN (STM0280); SciO (STM0281); SciP (STM0282); SciS (STM0285); SciT (STM0286); SciU (STM0287); SciV (STM0288); VrgS (STM0289); SciW (STM0290); and STM0291 (unnamed in stm as of this writing).

These 21 genes (the seed sequences) colocate in an approximately 31kb region of the genome of *Salmonella enterica* serovar typhimurium. Genome-colocation-based clusters[29] of seed sequence homologs were identified in proteobacterial genomes using the KEGG SSDB Conserved Gene Cluster search tool. The seed sequences homologs found in these clusters, determined based on KEGG best-best annotation, were compiled on a per genome basis. In genomes with more than one cluster of homologous genes, the homologous sequences were compiled on a genome and genome-location basis. The outcome of this manual process was a collection of sequences for suspected T6SS proteins for 71 gene clusters from 51 proteobacterial genomes. The KEGG species codes (and number of T6SS gene clusters when greater than one) were: aav; acb; aci; aeh; aha; asa; atu; azo (2); bam (2); bbr; bch; bcn; bja; bma (4); bmv; bpa; csa; cvi; dar; eca; eci (2); ecs; hch; hha; hhe; jan; maq; mlo; mxa; pae (3); pca; pde; pfo (2); pin; plu (4); ppr; rba; reh (2); rle; rme; rsp; sde; sfr; stm; stt; sus; vch; vvu; xcv (2); ypa (4); yps (4).

To identify sequences which may have been missed by this process, MUSCLE alignments [43] were constructed for each set of homologous protein sequences. These alignments were used to construct hidden Markov model (HMM) profiles for each protein using HMMER

[29]The term 'gene cluster' has been used to refer to different categorizations of genes. I use the term here to refer to genes found within close physical proximity in a single genome, and not, for example, a cluster of orthologous genes.

[151], which were used in global and local profile searches of KEGG. The search results were manually reviewed for each protein sequence and sequences which were not previously identified, but that clustered with genes already identified, were added to that gene cluster's sequence collection. In cases where more than one homolog was identified in a single gene cluster, only the most significant hit was included. Finally the sequences identified for each protein in the previous steps were aligned using MUSCLE with default parameters.

The sequences generated by this process form the basis of the co-occurrence and covariation analyses. The sequence compilation results are summarized in Table 6.1. The number of sequences for each protein are presented on the diagonal, and the number of times each pair of sequences cooccur in a gene cluster are presented in the lower triangle of the matrix.

6.2.2 The sequence pairing problem

A significant difficulty with this study was associating sequences suspected to form a functional T6SS with one another. Typically this could be done by associating sequences cooccurring in a genome, but because the T6SS genes appear to have frequently undergone horizontal gene transfer events, it is dangerous to assume that any SciH gene (for example) in a genome would be capable of a functional interaction with any SciI gene from the same genome. Because some genomes in this analysis contained as many as four T6SS gene clusters, pairing sequences for the downstream analyses was a difficult process. Sequences were only considered to be involved in the same T6SS apparatus if they were observed to cluster in the genome, and this was determined by reviewing the loci manually based on accession numbers.

6.2.3 Covariation analysis

For each of the seventy-six pairs of proteins that cooccurred in 30 or more gene clusters (referred to as the cooccurring proteins, see the italicized and bolded values in Table 6.1), joint-entropy-normalized mutual information was applied to score the degree of covaria-

Table 6.1: Cooccurrence of suspected T6SS proteins.

Values on the diagonal are the total number of gene clusters containing a homolog for the corresponding protein, and off-diagonal values are the number of gene clusters in which a pair of proteins was observed to cooccur. Pairs of proteins which cooccurred in forty of more gene clusters are bolded, and pairs of proteins which occur in less than forty at least thirty gene clusters are italicized. These pairs of proteins were included in the covariation analyses, with greater confidence associated with pairs achieving co-occurrence values of forty or greater. This table is symmetric, and values on the diagonal indicate the total number of gene clusters in which the protein was identified.

	SciA	SciB	SciC	SciD	SciE	SciF	SciG	SciH	SciI	SciK	SciM	SciN	SciO	SciP	SciS	SciT	SciU	SciV	SciW	291	VrgS
SciA	38																				
SciB	*37*	68																			
SciC	*38*	**68**	70																		
SciD	*34*	**43**	**44**	44																	
SciE	16	22	23	22	23																
SciF	7	6	7	7	7	7															
SciG	*35*	**60**	**60**	**40**	21	6	61														
SciH	*37*	**64**	**66**	**43**	23	7	**57**	67													
SciI	*37*	**65**	**67**	**43**	22	6	**58**	**66**	68												
SciK	*35*	**45**	**47**	**42**	23	7	**41**	**46**	**46**	47											
SciM	13	12	13	11	8	6	11	13	12	13	13										
SciN	28	**50**	**52**	*30*	17	7	**49**	**49**	**50**	*32*	10	53									
SciO	*38*	**67**	**69**	**43**	23	7	**61**	**66**	**67**	**46**	13	**53**	70								
SciP	*38*	**67**	**69**	**44**	23	7	**61**	**66**	**67**	**47**	13	**53**	**69**	70							
SciS	28	**52**	**53**	*36*	20	5	**48**	**51**	**52**	*39*	8	**44**	**52**	**53**	53						
SciT	10	15	15	13	10	2	14	15	15	14	2	12	15	15	15	15					
SciU	4	4	4	4	4	2	4	4	3	4	2	4	4	4	3	3	4				
SciV	4	4	4	4	4	2	4	4	3	4	2	4	4	4	3	3	4	4			
SciW	2	5	5	2	2	2	5	4	3	2	2	5	5	5	4	1	2	2	5		
0291	1	4	4	1	1	1	4	3	2	1	1	4	4	4	3	0	1	1	4	4	
VrgS	*32*	**56**	**57**	*39*	20	7	**51**	**55**	**56**	**40**	13	**44**	**57**	**57**	**44**	12	3	3	5	4	58

tion between all pairs of intermolecular positions. Three reduced-state alphabets, a1_4m, charge_2, and charge_3, were applied, in addition to the full (unreduced) alphabet. In all cases, the gap character was treated as an ordinary alphabet character. This resulted in a covariation score between 0.0 and 1.0 for all intermolecular positions pairs between all pairs of cooccurring proteins. Pairs of proteins that cooccurred in fewer than 30 gene clusters were not included in this analysis, as these were considered to contain too few sequences for the analysis. The threshold of 30 was selected to be more conservative than studies performed by other groups, which achieved high false positive rates: a threshold of 11 was used in [144], and a threshold of 20 was used in [53]. Additional sequences are expected to reduce false positives by providing protection against spurious correlations because if two positions are evolving independently, as more observations (sequences) are included it becomes increasingly unlikely to find a correlation.

All post-processing steps presented in Section 5.3 were applied to the results of the covariation analysis to evaluate the differences and similarities between the resulting protein-protein interaction predictions. In all results, alignment positions are excluded if they contain more than 10% gap characters (10p_gap, see Section 5.3.1 for a detailed description of this gap filtering process). This step serves to exclude poorly aligned sections of the alignment, as these generally contain many gaps, and positions where indels have frequently occurred, as it is not clear how 'coevolution' involving these positions should be interpreted. Additionally, variations and combinations of the other two post-processing steps, parsimony informative and multiple interdependency filtering, were applied. The set of protein-protein interaction predictions which appeared best, based on being the near-exact intersection of the predictions generated with all other filtering strategies, was obtained using the 10p parsimony informative filter (md2mc10p, Section 5.4.3). This filter excludes alignment positions which do not contain at least two different characters which are each present in at least 10% of the sequences. This imposes a minimum degree of variation that must

be observed at a position before searching for covariation, and was shown to be useful for reducing false positives in NMI-based covariation analyses (Section 5.4.3).

6.2.4 Defining significantly covarying positions to predict protein interactions

Distributions of NMI scores in covariation analyses are skewed very heavily toward low scores, and in some cases have one or more extreme outliers which achieve very high scores (see Table 6.2). In this study a threshold-based test was applied, where an NMI score greater than or equal to 0.95, equating to an extreme outlier, was taken as an indicator of sequence covariation. A pair of proteins was predicted to interact if one or more pairs of positions achieved NMI greater than or equal to 0.95. This is a very high threshold value, and is intended to identify pairs of proteins which exhibit the strongest covariation signal.

6.2.5 Statistical analysis of protein-protein interaction predictions

A strength of this study is that the protein-protein interaction predictions will be tested biochemically following the computational analysis. As of this writing, those tests are in progress, and the results will be used to drive a statistical test of the hypothesis that intermolecular sequence covariation can be used to infer protein-protein interactions.

A permutation-based test will be applied to statistically test the hypothesis that sequence covariation can provide information about protein-protein interactions. In this test, I will use information about which pairs appear to physically interact and which appear not to physically interact based on biochemical analysis. By generating random sets of predictions about interacting and non-interacting pairs of proteins (PPI prediction sets), it is possible to count the number of random PPI prediction sets that more correct (e.g., based on the number of correct predictions), than the covariation-based PPI prediction set. If less than 5% of the random PPI prediction sets are better than the covariation PPI prediction set, that would support the hypothesis that covariation can be used to identify

Table 6.2: Distribution of NMI scores.
Distributions of NMI covariation scores are heavily skewed toward low scores. In some cases there are outliers which achieve very high scores. This pattern is evident when binning NMI scores and looking at the count of scores in each bin (Bin count), the total number of scores greater than or equal to the lower bound of a bin (count $\geq$ NMI), and the total percent of scores greater than or equal to the lower bound of a bin (percent $\geq$ NMI). Presented here are distributions of intermolecular NMI scores between the SciG and SciK alignments, which are predicted to interact; and SciA and SciK alignments, which are not predicted to interact. In both cases, alignments are recoded with the charge_2 alphabet, and the 10p_gap and md2mc10p filters were applied.

	SciG and SciK: predicted to interact			SciD and SciP: not predicted to interact		
NMI	Bin count	count $\geq$ NMI	percent $\geq$ NMI	Bin count	count $\geq$ NMI	percent $\geq$ NMI
0.00	23854	28982	1.0e+00	9170	11049	1.0e+00
0.05	4012	5128	1.8e-01	1508	1879	1.7e-01
0.10	760	1116	3.9e-02	295	371	3.4e-02
0.15	221	356	1.2e-02	51	76	6.9e-03
0.20	77	135	4.7e-03	15	25	2.3e-03
0.25	12	58	2.0e-03	6	10	9.1e-04
0.30	6	46	1.6e-03	1	4	3.6e-04
0.35	6	40	1.4e-03	1	3	2.7e-04
0.40	9	34	1.2e-03	1	2	1.8e-04
0.45	6	25	8.6e-04	1	1	9.1e-05
0.50	0	19	6.6e-04	0	0	0.0e+00
0.55	0	19	6.6e-04	0	0	0.0e+00
0.60	0	19	6.6e-04	0	0	0.0e+00
0.65	2	19	6.6e-04	0	0	0.0e+00
0.70	0	17	5.9e-04	0	0	0.0e+00
0.75	0	17	5.9e-04	0	0	0.0e+00
0.80	5	17	5.9e-04	0	0	0.0e+00
0.85	0	12	4.1e-04	0	0	0.0e+00
0.90	0	12	4.1e-04	0	0	0.0e+00
0.95	12	12	4.1e-04	0	0	0.0e+00

protein-protein interactions with a p-value less than 0.05. Two limitations of this test are that it requires biochemical information on all pairs of proteins, and it assumes that a negative result in a biochemical test suggests that the corresponding pair of proteins do not physically interact. The design of this test may need to be revised if these limitations prove problematic.

6.3 Results

6.3.1 Cooccurence of putative T6SS proteins in bacterial gene clusters

Of the twenty-one *Salmonella enterica* serovar typhimurium proteins, I identified thirteen which frequently cooccur in bacterial gene clusters. Cooccurrence has been shown to be evidence of the functional relationship between bacterial genes (e.g., [100, 149]) and supports the idea that some or all of the seed sequences are involved in a common molecular function. The number of gene clusters in which each pair of proteins were observed to cooccur are presented Table 6.1. Our covariation analysis focused on the thirteen cooccurring proteins, and therefore tested seventy-six pairs of cooccurring proteins for sequence covariation.

6.3.2 Covariation

Two sets of protein-protein interaction predictions were generated. The first set (referred to as md2mc10p_mid) was the result of applying the 10p parsimony informative (md2mc10p) and multiple interdependency (mid) filters, and contained eleven predictions. The second set (referred to as md2mc10p) was the result of applying only the md2mc10p filter, and contained sixteen predictions. While all filters discussed in Section 5.3 were applied in the covariation analysis, yielding seven different (but overlapping) sets of protein-protein interaction predictions, I have focused on the md2mc10p and md2mc10p_mid result sets as they form the near-perfect intersection of seven result sets, and are therefore expected to be the most conservative predictions.

While the multiple interdependency filter has been shown to improve precision, the assumption on which this filter is based is questionable (see Section 5.4.2). I therefore treat the sixteen md2mc10p predictions as the best result set, and these predictions are being tested biochemically as of this writing. These predictions, along with the specific covarying positions pairs, are presented in Table 6.3.

Of these sixteen md2mc10p predictions, two have been biochemically confirmed (SciH-SciI; SciG-SciI[30]); an additional five appear to be correct based on current models of the T6SS (SciG-SciK; SciK-SciN; SciK-SciS; SciG-SciS; SciK-VrgS); eight involved predicted cytoplasmic proteins that have not been previously characterized (SciD-SciO; SciI-VrgS; SciD-SciS; SciD-VrgS; SciD-SciI; SciA-SciK; SciO-VrgS; SciG-VrgS); and one prediction is inconsistent with the current model (SciD-SciN)[31]. Figure 6.1a presents the T6SS model created based on the current literature, and Figure 6.1b presents proposed extensions on this model incorporating the predictions presented herein.

6.4 Discussion

Based on previous studies and the computational analyses presented here, Figure 6.1b proposes a model of the T6SS. The majority of the predictions made appear to either be correct (2/16), likely (5/16), or possible (8/16) based on biochemical studies in progress and the current model (see Figure 6.1a). All sixteen predictions are currently being validated biochemically. These results are a promising start, and while they have suggested information on the organization of the T6SS complex the true power of this study lies in the biochemical evaluation of these predictions, which will further elucidate the structure of the T6SS complex. This biochemical confirmation will additionally allow for a statistical test of the central hypothesis of this chapter, that sequence covariation can be used to infer protein-protein interactions.

To the best of my knowledge, this is the first application of sequence covariation to predict protein-protein interactions where the predictions are being validated biochemically following the computational analysis. As discussed in Chapter III, this type of evaluation

[30]Confirmed by a different group and presented in [142], along with data confirming the SciH-SciI.

[31]The 'current models' referred to here are those presented in [137, 138, 152], and the categorizations of these predictions were provided by my collaborator on this project, Amy Dear.

Figure 6.1: Models of the Type VI Secretion System.

(a) A model of the T6SS based on a review of the T6SS literature. (b) A proposed model of the T6SS based on sequence covariation, biochemistry, and previous models. This figure was created by my collaborator on this project, Amy Dear.

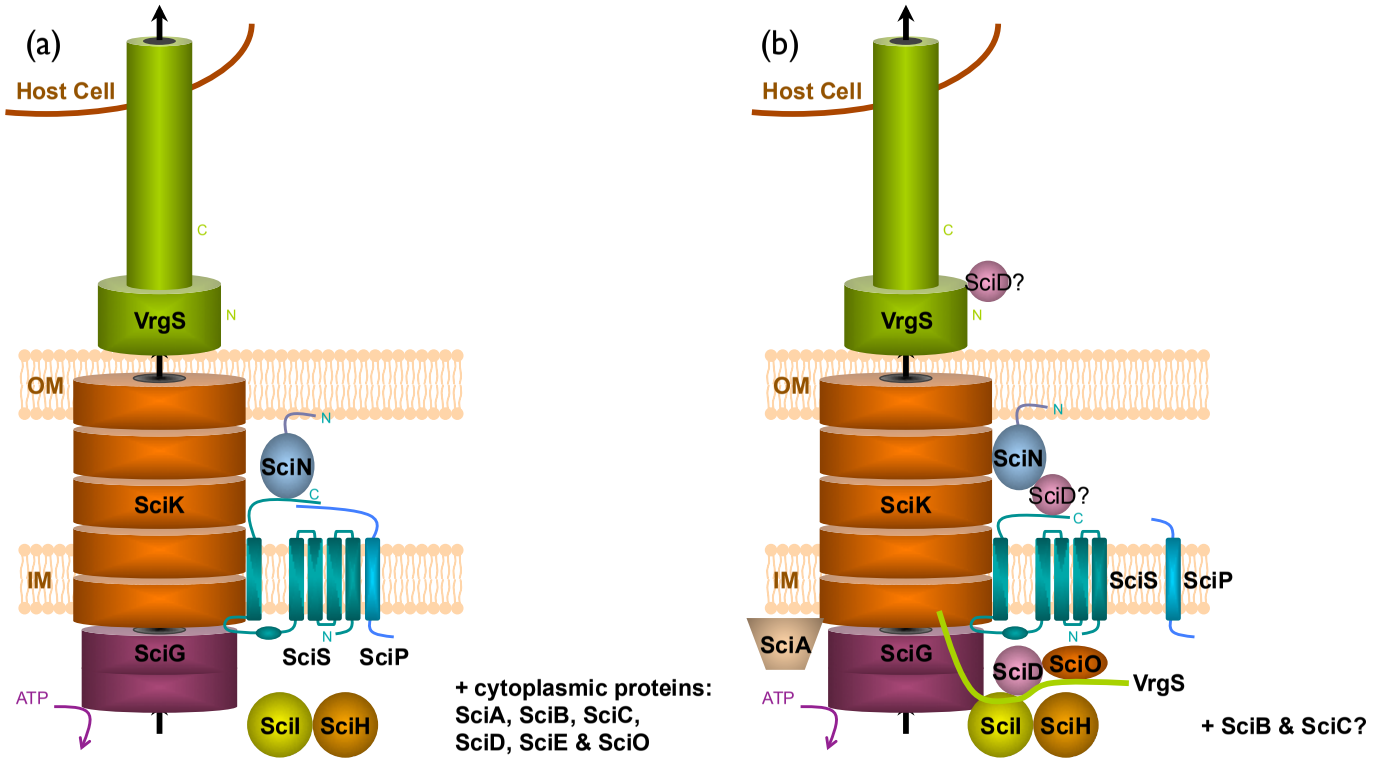


Table 6.3: Predicted T6SS protein-protein interactions.

Predicted physical interactions of T6SS proteins based on sequence co-occurrence and co-variation. Position pairs are observed to covary with NMI ≥ 0.95 in at least one of four included amino acid alphabets. The first position number in each pair corresponds to the first protein in the pair, and the second position number in each pair to the second protein in the pair. These position numbers are based on the alignments used in this analysis. Covarying position pairs are separated by semi-colons. The eleven bolded predictions were returned from each alternative filtering process. The starred prediction was returned only in the md2mc10p prediction set.

Protein pair	Covarying position pairs
SciG, SciK	323,41; 651,36; 651,72; 653,36; 653,72; 655,36; 655,72; 660,36; 660,72; 698,36; 698,72; 713,36; 713,72; 775,38
SciO, VrgS	27,527; 27,572
SciD, SciI	163,296
SciG, SciS	394,410; 405,817; 641,817
SciD, SciO	149,350
SciG, VrgS	709,495; 709,526; 801,157; 801,203
SciD, SciS	101,617; 101,825; 163,739; 163,823
SciI, VrgS*	361,527; 361,572; 422,527; 422,572; 469,527; 469,572
SciK, VrgS	88,261
SciG, SciI	232,412; 232,413; 232,426; 232,437; 340,436; 425,413; 425,426; 425,437; 584,300; 584,306; 584,384; 651,241; 658,300; 658,306; 658,384; 698,241; 701,413; 701,426; 701,437; 709,342; 713,300; 713,306; 713,384
SciH, SciI	131,306; 139,130
SciK, SciS	54,295; 63,252; 63,535; 130,222
SciK, SciN	103,60; 130,58
SciD, VrgS	101,261
SciD, SciN	149,133
SciA, SciK	145,66

(an extrinsic evaluation) tests the contribution of a technique to assist in a realistic task, rather than testing its ability to recreate a possibly problematic gold-standard data set (an intrinsic evaluation).

CHAPTER VII

CONCLUDING REMARKS AND FUTURE DIRECTIONS

I have addressed two areas of computational biology in this dissertation: biomedical language processing, specifically identifying and extracting mentions of protein mutations from free text; and protein sequence analysis, specifically identifying protein positions which coevolve. While very different problems, unifying aspects of this work include protein point mutations and substitutions (Chapters II, IV, VI, and Appendix II), evaluations of methods in bioinformatics and computational biology (Chapters II, III, IV, VI, and Appendix I), and development and publication of bioinformatics software (II, IV, and Appendix I). In the following sections I review the most significant conclusions drawn from the studies presented herein, and I conclude with my future research plans.

7.1 Automatic pattern development for concept recognition systems

An unprecedented quantity of biomedical text is available via MEDLINE, and full-text journal publications are becoming increasingly available to the public. Biomedical language processing offers hope for managing this ever-increasing literature base, but it has been noted that 'Knowledge-based [natural language processing] systems will be practical for real-world applications only when their domain-dependent dictionaries can be constructed automatically.' [66] In this dissertation, I have presented one mechanism, along with a case study, illustrating how MEDLINE itself can be used to generate these 'domain-dependent dictionaries' and thereby make data currently available only in biomedical literature more accessible to researchers.

In Chapter II I presented the automated pattern generation methodology employed to develop MutationFinder, my high-performance system for extracting point mutation mentions from biomedical literature; the publicly available mutation corpus, distributed with MutationFinder, which should prove valuable for enabling direct comparisons between

mutation recognition systems; and several measures of MutationFinder's performance. The automatic pattern generation methodology is extensible to other concept recognition goals, and provides a means for achieving high precision and recall while minimizing the investment of time when developing rule-based concept recognition systems.

7.2 The MutationFinder project

The goal of the MutationFinder project was to develop a useful and usable tool for extracting point mutation mentions from text. The best evidence that this goal has been achieved is the adoption of MutationFinder by several groups [86–89] since its publication in 2007. Further illustrating the interest in MutationFinder are recent citations of the project in secondary literature: *Cyberinfrastructure Technology Watch* Quarterly newsletter [153] and *Genome Technology* [154].

I recognize that a system for identifying the gene or protein incurring a mutation, in conjunction with the mutation data, would be even more valuable. Additionally, the ability to distinguish between mutations in genes and proteins would be useful. (For example, *'C10T'* could refer to either a cysteine to threonine mutation in a protein, or a cytosine to thymine mutation in DNA.) Accurately identifying gene and protein names in text is an open area of research [26], and classifying extracted names as referring to genes or proteins is a task which human experts only do with around 80% agreement [155]. I view mutation extraction, gene/protein named-entity recognition, and gene/protein mutation disambiguation as separate language processing problems, which may collectively be solved by combining independent systems. MutationFinder provides reliable extraction of mutation data, and its modular nature and open-source availability in multiple languages facilitate its incorporation into more complex systems.

Combining the output of MutationFinder with the output of an independent gene/protein name extraction system could provide a basis for improving the performance of gene/protein

recognizers, and this is the subject of a Master's project currently in progress in the Hunter lab (David Randolph, personal communication). The ability to distinguish gene and protein names could in turn provide the requisite information to disambiguate mutation types, and assist in associating mutations with their source gene or protein.

7.3 Manual and automated database annotation procedures

In Chapter III, I evaluated manual curation and two automated methods for annotating protein mutations in the PDB, and showed that all three were unreliable. Genomic data and their reliable annotation are essential to progress in biomedical sciences. It has been shown empirically that manual annotation cannot keep up with the rate of biological data generation [27]; furthermore, I have shown that even if manual annotation could keep pace with data generation, it is still error-prone.

A reasonable database annotation approach to pursue is incorporating automated techniques into manual annotation processes. For example, when a scientist deposits a new PDB structure, their primary citation and sequences can be scanned for mutations. The depositor could be presented with suggestions: *In your abstract, you mention an A42G mutation—is this mutation present in your structure?*[32] Additionally, these tools can be applied as quality control steps. Before a mutation annotation is accepted, it could be validated against sequence data. Responses to such prompts could be recorded and used to generate new gold standards that could be used to improve existing or future tools for automating annotation procedures. 'Smart' annotation deposition systems could be the key to improved quality of data in the present and improved automated techniques in the future.

Biomedical text mining tools have historically focused on abstracts, largely because

[32]Note that this could also be applied retroactively when structures are deposited before the corresponding article is published. In this case, authors could be e-mailed with any suspected erroneous or missing annotations, and prompted to correct them.

it has been difficult to access full-text. Full-text is typically more tightly guarded and distributed via journal websites. Abstracts, on the other hand, are easily accessible via MEDLINE. In Chapter III I clearly illustrated that to comprehensively extract mutation data from literature, it is essential to have access to full text: abstracts alone do not contain all of the relevant information (see Table 3.2).

7.4 Coevolution comparison

In Chapter IV, I changed focus to evaluating coevolution algorithms, or algorithms designed to identify coevolving pairs of positions in multiple sequence alignments. I hypothesized that by comparing coevolution scores to a background distribution generated from the same alignment, methods which did not incorporate phylogeny (tree-ignorant) could match or out-perform methods that did incorporate phylogeny (tree-aware). The analyses presented in Chapter IV support that hypothesis: I demonstrated that transformed tree-ignorant methods detect coevolution with equivalent or better power than tree-aware methods when applied to detect the periodicity of protein alpha helices.

I again stress that *the results do not imply that accounting for phylogeny is unimportant in coevolutionary analyses*, but rather that the coevolution metrics that explicitly incorporate the phylogenetic tree typically appear to be less sensitive than those that do not. While explicitly incorporating phylogeny is necessary to study the process of coevolution, metrics that do so may be too conservative in applications which attempt to make predictions based on hypothesized coevolving positions. I address this further, with an example, in Section 7.4.2.

A useful next step will involve confirming the results of Chapter IV with other evaluations, perhaps based on identifying positions proximal in tertiary structure or individual pairs of positions which have been biochemically shown to coevolve, for example in compensatory mutation studies. The robust statistical properties of transformed tree-ignorant

metrics coupled with their generally many orders of magnitude faster computational speed opens up new prospects for detection of coevolving residues within and between proteins. The application of coevolution algorithms on a massive scale should provide new insights into the structures, functions, and interactions of a wide range of protein families.

Separating positions which truly coevolve from those which covary for other reasons such as phylogeny or random correlations is an extremely difficult challenge and still a very open area of research. Two problems that are not dealt with by the current coevolution algorithms, and which I consider to be two of the more difficult aspects of detecting coevolution, are detecting coevolution that does not occur in a pairwise manner, and identifying patterns of coevolution which mirror patterns arising phylogeny.

7.4.1 Positions involved in interactions may not be conserved

Although the lack of appropriate algorithms makes it difficult to support this claim, I suspect that in many cases it is not possible to identify coevolving positions because the interactions between amino acid residues are not necessarily pairwise, or because the positions involved in a pairwise interaction may change. For example, imagine a substitution at a position in the hydrophobic core of a protein which reduces the net hydrophobicity of the core and thereby has a slightly deleterious effect on the function of the protein. That substitution may effect the substitution rate at several other positions in the core of the protein, and could be compensated by a mutation at any of those positions. If compensating mutations occur at different positions in different sequences, the coevolution would not be detectable by algorithms designed to identify pairwise coevolution.

In Figure 7.1, I present a possible example of the loss of a pairwise interaction being compensated by an interaction between a nearby pair of positions in the myosin rod alignment used in Chapter IV. In sequence gi:6708502 it appears as though the loss of an ionic interaction between positions 244 and 248 may be compensated by an ionic interaction between positions 252 and 256. This type of compensatory relationship is not possible to

detect with the current coevolutionary algorithms[33]. It is possible that Statistical Coupling Analysis (or other residue-distribution-based methods) are the best equipped of the current coevolution algorithms to handle this issue since they are theoretically capable of identifying a relaxation of selective pressure on a position associated with a change at another position.

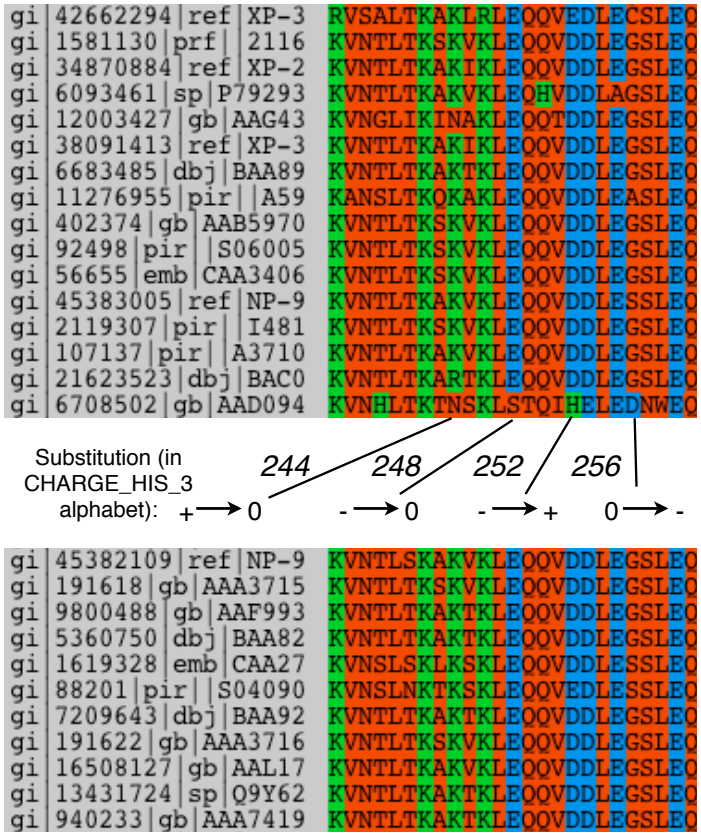
7.4.2 Phylogenetic and coevolutionary covariation patterns appear similar

If a pair of positions are truly coevolving, but the pattern of covariation mirrors the phylogeny, tree-aware methods would treat the data as weaker evidence for coevolution between the pair of positions. Covariation patterns should however be expected to frequently resemble patterns arising from the phylogeny. Coevolving positions are suspected to be subject to moderate or strong selective pressures, and many mutations – especially those that would be deleterious in the absence of a second, compensating mutation – will not be tolerated. This may explain the difficulty of identifying coevolving position pairs in proteins using tree-aware methods: covariation patterns of truly coevolving positions may frequently mirror covariation patterns arising from phylogeny. I suspect that this, in part, explains the difference in performance of the tree-aware and tree-ignorant methods in Chapter IV.

Kondrashov *et al.* [156] present an example of an apparent compensated pathogenic deviation in beta hemoglobin. The authors proposed that van der Waals interactions between V20 and G69 of human beta hemoglobin are structurally significant, and cite evidence that a V20E mutation is pathogenic in humans. In *Equus caballus* (horse) however, E20 is wild-type, and appears to be compensated by H69, replacing the van der Waals interaction with a hydrogen bonding or ionic interaction between E20 and H69. The authors cite the PDB structures of human and horse beta hemoglobin (4HHB and 2DHB, respectively) as evidence for the interaction between these two positions.

[33]The position numbers are based on the myosin_rod_chordate alignment included as supplementary material to [61]. Without a biochemical analysis, we cannot assume that this is truly a compensatory relationship, but must instead treat this as a suggestive pattern.

Figure 7.1: Compensation may involve different sequence positions.
An ionic interaction between positions 244 and 248 may be compensated by an ionic interaction between positions 252 and 256 in this subalignment of the myosin rod. An analysis of this alignment in the context of the phylogeny (to infer ancestral states) suggested that these substitutions could all be achieved via single nucleotide substitutions. Several pathways of four single substitution events are possible which would retain an ionic interaction between at least two $(i, i+4)$ positions in all intermediate sequences. Position colors correspond to the `CHARGE_HIS_3` alphabet, where positively charged residues are colored green, negatively charged residues blue, and uncharged residues red. The alignment schematic was generated using Seaview.



To determine if this putative compensatory mutation, G69H, would be identifiable via covariation algorithms, I compiled and aligned beta hemoglobin sequences. In agreement with the sequence alignment presented in [156], the E20/H69 pair appears monophyletic,

apparently present only in the *Perissodactyla*[34] family (Figure 7.2). (In addition to *Equus caballus*, E20/H69 is found in the Indian rhino, *Rhinoceros unicornis*, and the South American tapir, *Tapirus terrestris*, which are not included in Figure 7.2, but would be found on the same branch.) Because tree-aware methods are designed to treat monophyletic covariation as weaker evidence of coevolution, tree-ignorant methods should generally be more capable of detecting interactions of this type.

Interestingly, one additional sequence in the tree presented in Figure 7.2 deviates from the pattern of uncharged residues at position 69. In sequence gi:122681 (derived from the Egyptian fruit bat, *Rousettus aegyptiacus*), a negatively charged residue is present at position 69 (D69). D69 is noted as associated with β^0 thalassemia in dbSNP and HbVar[35], suggesting that this may be a compensated pathogenic deviation in *Rousettus aegyptiacus*. In the *Rousettus aegyptiacus* sequence, position 19 contains a histidine (H19), which is rare at this position. If D69 is a CPD compensated by H19, which appears possible based on a review of related structures, it would support the idea presented in Section 7.4.1 and Figure 7.1 that the compensating position may frequently change, thereby making it difficult to detect with pairwise coevolution detection algorithms.

7.5 Intermolecular coevolution and the Type VI Secretion System

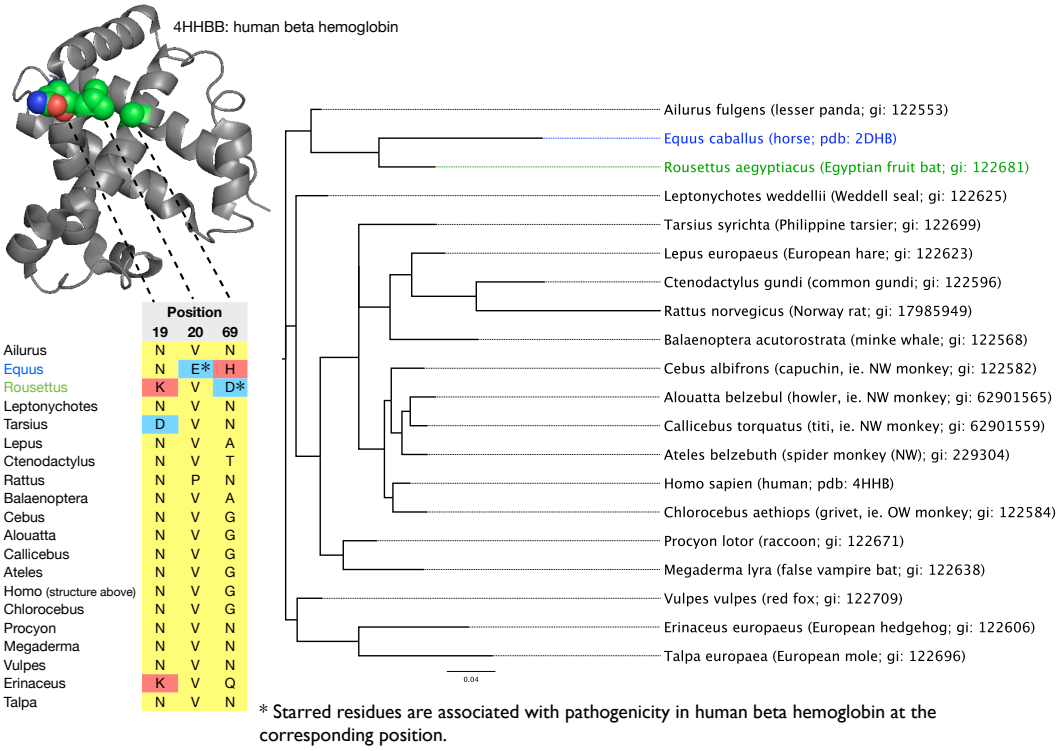
In Chapter VI I applied the techniques developed in Chapters IV and V in an attempt to identify intermolecular coevolution between component proteins of the Type VI Secretion System. I successfully identified covarying positions between ten putative T6SS proteins, and used this information to make sixteen predictions of physically interacting pairs. These

[34] *Perissodactyla* are grazing mammals, characterized morphologically by having an odd number of toes.

[35] See dbSNP mutation rs34718174 and HbVar ID 376. Note there is a position numbering difference so sequence position 69 here and in HbVar refers to position 70 in the sequence associated with rs34718174. HbVar is the Database of Human Hemoglobin Variants and Thalassemias, and as of this writing can be found at: http://globin.bx.psu.edu/hbvar/menu.html.

Figure 7.2: Compensation pattern may mirror phylogeny.

The compensatory relationship identified in [156] presented in the context of alignment, phylogeny, and structure. The blue branch (2DHB) is the only one containing the E20/H69 interaction suggested by [156]. All other sequences contain uncharged residues at these positions, with the exception of the green branch (gi:122681), which contains a negatively charged residue at position 69. The PDB sequences for 4HHB and 2DHB match their UniProt reference sequences (P68871 and P02062, respectively) perfectly, and I have therefore retained the PDB sequence identifiers for consistency with [156].



predictions shed some light on the organization of the T6SS complex, but the even greater value of this study will result from the biochemical tests of these predictions that are currently in progress. In collaboration with the Sousa biochemistry lab in Boulder, Colorado, my sixteen protein-protein interaction predictions are being tested *in vitro*. The biochemical results will provide hard evidence for the structure of the T6SS complex, and will be used to further evaluate my approach for detecting sequence coevolution in this study, as well as other algorithms presented in Chapter VI.

7.6 The Sequence Pairing Problem

Possibly the biggest limitation in intermolecular coevolution studies right now is what I refer to as the sequence pairing problem, or the problem of associating sequences from different proteins suspected to interact with one another. For example, if searching for coevolution between protein A and protein B, I must generate multiple sequence alignments for each protein, and then correctly associate the protein A and protein B sequences which are suspected to interact with one another. Frequently, these would be the pairs of sequences which are found in the same genome. The two issues that make the sequence pairing problem difficult are limited access to the relevant information in databases, and sequence paralogy.

The first issue is related to data management: in many cases, if you BLAST against a sequence database to compile sequences, the source species for each sequence is not directly accessible from the BLAST results. KEGG avoids this issue by including species codes in sequence identifiers, and I therefore frequently use KEGG. The bigger problem however results from paralogy. If there are, say, two copies each of protein A and protein B in a single genome, you can not assume that either protein A would form a functional interaction with either protein B, so you must know how to associate the sequences with one another correctly. For intermolecular coevolution studies to scale, sequence pairing must be automatic.

In my T6SS study, I handled the sequence pairing process in a tedious, partly manual process, where I only associated sequences with one another when they were found to co-locate in a genome. While effective, this technique is not scalable, and represents a sequence annotation issue that will have to be addressed. As noted in previous studies which attempted to identify intermolecular coevolution [53, 144], the sequence pairing problem is a significant hurdle which has yet to be solved.

7.7 The role of software engineering in computational biology

Developing, testing, and publishing software is the computational biologist's primary means for conducting reproducible experiments. I strongly believe that documenting and versioning software is a key component of record keeping for the computational biologist, and that the quality and comprehensiveness of software tests is an indicator of the quality of the science. I have briefly presented two software projects that I have contributed to in this dissertation: MutationFinder and PyCogent. In both cases, close attention has been paid to software testing to ensure its quality, and to software documentation both at the end-user level via tutorials and examples, and at the developer-level via structured and detailed comments.

7.8 Future directions

Two questions have arisen out of my studies of phylogeny and coevolving protein positions that I plan to address in the near future. The first question, briefly discussed in Section 7.4.1, is how frequently do positions change their interaction partners within proteins? Figures 7.1 and 7.2 suggest that this may occur, and if confirmed widely, this data could explain why pairwise coevolution algorithms do not identify more coevolving positions. The second question, briefly presented in Section 7.4.2, is should truly coevolving positions be expected to exhibit covariation patterns that mirror the phylogeny? I presented a single example where this appears to be the case in Figure 7.2, but an analysis of many

pairs of positions known to coevolve, such as base pairing positions in RNA, in the context of phylogeny would allow me to test if this is more generally the case. Understanding how positions coevolve in this way should allow for the improvement of coevolution algorithms by suggesting ways to control for patterns arising from phylogeny in a less-stringent fashion. Additionally, I would like to continue my evaluation of coevolution algorithms in an attempt to see if the results of my alpha-helix-based comparison are consistent with results on other biologically realistic and simulated test sets. My PyCogent coevolution module makes further tests of this type relatively straight-forward.

The quantity of all flavors of biological data is exploding, including biomedical literature, fully sequenced genomes, gene expression data, individual gene and protein sequences, and molecular structures. As specific needs arise, I am interested in expanding my automatic pattern generation approach to other information extraction tasks, and incorporating the resulting tools into database annotation tasks. I believe that a combination of manual and automated database annotation techniques can be applied to improve to coverage and accuracy of current biomedical databases, while simultaneously generating gold-standard data for future automated annotation approaches. An on-going collaboration resulting from my text mining work involves interfacing MutationFinder with the Protein Databank. This will serve as a proof-of-concept for combining manual and automated database annotation techniques, and should result in improved quality of mutation data in the PDB.

Several on-going collaborations have resulted from applications of my sequence coevolution work, including predicting tertiary interactions in chemotaxis proteins and making informed mutagenesis decisions in protein engineering studies. The results of these studies will provide new biological insight, and will allow for additional extrinsic evaluations of my approach to identifying coevolving protein positions.

In addition to my biological research goals, I am devoted to developing high quality, highly reusable, open source software based on established software engineering principles. I

will continue to develop and support my two open-source software projects, MutationFinder and PyCogent, and I expect that my future research will result in new open source software projects.

The research presented in this thesis has created new questions for me, and provides a pathway into my post-doctoral research. As the toolset of the biologist continues to expand and change, computational tools and statistical methods are consistently becoming more important. The ability to efficiently work with large data sets, and to meaningfully interpret patterns emerging from data are critical. As a computational biologist, my role is to test biological hypotheses using the tools of computer science and mathematics. I look forward to continuing to collaborate with both computer scientists, mathematicians, and traditional wet-lab biologists to address the ever-expanding and far-reaching questions of biology.

REFERENCES

[1] Hunter L, Cohen KB: **Biomedical Language Processing: What's Beyond PubMed?** *Mol Cell* 2006, **21**(5):589–594.

[2] Marcotte E, Date S: **Exploiting big biology: integrating large-scale biological data for function inference.** *Brief Bioinform* 2001, **2**(4):363–374.

[3] Caporaso JG, Yarus M, Knight R: **Error minimization and coding triplet/binding site associations are independent features of the canonical genetic code.** *J Mol Evol* 2005, **61**(5):597–607.

[4] Yarus M, Caporaso GJ, Knight R: **Origins of the genetic code: the escaped triplet theory.** *Annu Rev Biochem* 2005, **74**:179–198.

[5] Nelson DL, Cox MM: *Lehninger: Principles of Biochemistry*, Worth Publishers. 3rd edition 2000 chap. 5, :141–145.

[6] Nelson DL, Cox MM: *Lehninger: Principles of Biochemistry*, Worth Publishers. 3rd edition 2000 chap. 5, :146–150.

[7] Steen H, Mann M: **The ABC's (and XYZ's) of peptide sequencing.** *Nat Rev Mol Cell Biol* 2004, **5**(9):699–711.

[8] Dayhoff MO, Schwartz RM, Orcutt BC: *Atlas of Protein Sequence and Structure*, National Biomedical Research Foundation, *Volume 5* 1978 :345–352.

[9] Berg JM, Tymoczko JL, Stryer L: *Biochemistry*, W. H. Freeman and Company 2002 chap. 2, :21–28.

[10] Alberts B, Johnson A, Lewis J, Raff M, Roberts K, Walter P: *Molecular Biology of the Cell*, Garland Science 2002 chap. 7, :453–465.

[11] Felsenstein J: *Inferring Phylogenies*. Sinauer Associates, Inc. 2004.

[12] Durbin R, Eddy S, Krogh A, Mictchison G: *Biological sequence analysis*. Cambridge University Press 2006.

[13] Sterelny K, Griffiths PE: *Sex and Death: An Introduction to the Philosopy of Biology*. The University of Chicago Press 1999.

[14] Nelson DL, Cox MM: *Lehninger: Principles of Biochemistry*, Worth Publishers. 3rd edition 2000 chap. 6, :159–202.

[15] Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, Shindyalov IN, Bourne PE: **The Protein Data Bank.** *Nucleic Acids Res* 2000, **28**:235–242.

[16] Deshpande N, Addess KJ, Bluhm WF, Merino-Ott JC, Townsend-Merino W, Zhang Q, Knezevich C, Xie L, Chen L, Feng Z, Green RK, Flippen-Anderson JL, Westbrook J, Berman HM, Bourne PE: **The RCSB Protein Data Bank: a redesigned query system and relational database based on the mmCIF schema.** *Nucleic Acids Res* 2005, **33**(Database issue).

[17] Shu X, Leiderman P, Gepshtein R, Smith NR, Kallio K, Huppert D, Remington SJ: **An alternative excited-state proton transfer pathway in green fluorescent protein variant S205V.** *Protein Sci* 2007, **16**(12):2703–2710.

[18] Stenson PD, Ball EV, Mort M, Phillips AD, Shiel JA, Thomas NST, Abeysinghe S, Krawczak M, Cooper DN: **Human Gene Mutation Database (HGMD): 2003 update.** *Hum Mutat* 2003, **21**(6):577–581.

[19] McKusick VA: **Mendelian Inheritance in Man and its online version, OMIM.** *Am J Hum Genet* 2007, **80**(4):588–604.

[20] Dantzer J, Moad C, Heiland R, Mooney S: **MutDB services: interactive structural analysis of mutation data.** *Nucleic Acids Res* 2005, **33**(Web Server issue):W311–W314.

[21] Horn F, Bettler E, Oliveira L, Campagne F, Cohen FE, Vriend G: **GPCRDB information system for G protein-coupled receptors.** *Nucleic Acids Res* 2003, **31**:294–297.

[22] Horn F, Vriend G, Cohen FE: **Collecting and harvesting biological data: the GPCRDB and NucleaRDB information systems.** *Nucleic Acids Res* 2001, **29**:346–349.

[23] Rhee SY, Gonzales MJ, Kantor R, Betts BJ, Ravela J, Shafer RW: **Human immunodeficiency virus reverse transcriptase and protease sequence database.** *Nucleic Acids Res* 2003, **31**:298–303.

[24] Cohen AM, Hersh W: **A survey of current work in biomedical text mining.** *Briefings in Bioinformatics* 2005, **6**:57–71.

[25] Rebholz-Schuhmann D, Kirsch H, Couto F: **Facts from text—is text mining ready to deliver?** *PLoS Biol* 2005, **3**(2).

[26] Yeh A, Morgan A, Colosimo M, Hirschman L: **BioCreAtIvE task 1A: gene mention finding evaluation.** *BMC Bioinformatics* 2005, **6 Suppl 1**.

[27] Baumgartner WA, Cohen KB, Fox LM, Acquaah-Mensah G, Hunter L: **Manual curation is not sufficient for annotation of genomic databases.** *Bioinformatics* 2007, **23**(13):i41–i48.

[28] Horn F, Lau AL, Cohen FE: **Automated extraction of mutation data from the literature: application of MuteXt to G protein-coupled receptors and nuclear hormone receptors.** *Bioinformatics* 2004, **20**(4):557–568.

[29] Rebholz-Schuhmann D, Marcel S, Albert S, Tolle R, Casari G, Kirsch H: **Automatic extraction of mutations from Medline and cross-validation with OMIM**. *Nucl. Acids Res.* 2004, **32**:135–142.

[30] Caporaso JG, Baumgartner WA, Cohen KB, Johnson HL, Paquette J, Hunter L: **Concept recognition and the TREC Genomics tasks.** In *The Fourteenth Text REtrieval Conference (TREC 2005) Proceedings* 2005.

[31] Caporaso JG, Baumgartner WA, Kim H, Lu Z, Johnson HL, Medvedeva O, Lindemann A, Fox L, White EK, Cohen KB, Hunter L: **Concept Recognition, Information Retrieval, and Machine Learning in Genomics Question-Answering**. In *The Fifteenth Text REtrieval Conference (TREC 2006) Proceedings* 2006.

[32] Baumgartner WA, Lu Z, Johnson HL, Caporaso JG, Paquette J, Lindemann A, White EK, Medvedeva O, Cohen KB, Hunter L: **Concept recognition for extracting protein interaction relations from biomedical text.** *Genome Biol* 2008, **9 Suppl 2**:S9.

[33] Müller HM, Kenny EE, Sternberg PW: **Textpresso: an ontology-based information retrieval and extraction system for biological literature.** *PLoS Biol* 2004, **2**(11):e309.

[34] Hakenberg J, Royer L, Plake C, Strobelt H, Schroeder M: **Me and My Friends: Gene Mention Normalization with Background Knowledge**. In *Proceedings of the Second BioCreative Challenge Evaluation Workshop* 2007.

[35] Leaman R, Gonzalez G: **BANNER: an executable survey of advances in biomedical named entity recognition.** *Pac Symp Biocomput* 2008, :652–663.

[36] Hunter L, Lu Z, Firby J, Baumgartner WA, Johnson HL, Ogren PV, Cohen KB: **OpenDMAP: an open source, ontology-driven concept analysis engine, with applications to capturing knowledge regarding protein transport, protein interactions and cell-type-specific gene expression.** *BMC Bioinformatics* 2008, **9**:78.

[37] Consortium U: **The Universal Protein Resource (UniProt) 2009.** *Nucleic Acids Res* 2009, **37**(Database issue):D169–D174.

[38] Jackson P, Moulinier I: *Natural language processing for online applications: text retrieval, extraction, and categorization.* John Benjamins Publishing Company. 2002.

[39] Caporaso JG, Baumgartner WA, Randolph DA, Cohen KB, Hunter L: **MutationFinder: A high-performance system for extracting point mutation mentions from text.** *Bioinformatics* 2007, **23**(14):1862–1865.

[40] Caporaso JG, Baumgartner WA, Randolph DA, Cohen KB, Hunter L: **Rapid pattern development for concept recognition systems: Application to point mutations.** *J Bioinform Comput Biol* 2007, **5**(6):1233–1259.

[41] Caporaso JG, Deshpande N, Fink JL, Bourne PE, Cohen KB, Hunter L: **Intrinsic evaluation of text mining tools may not predict performance on realistic tasks.** *Pac Symp Biocomput* 2008, :640–651.

[42] Thompson JD, Higgins DG, Gibson TJ: **CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice.** *Nucleic Acids Res* 1994, **22**(22):4673–4680.

[43] Edgar RC: **MUSCLE: a multiple sequence alignment method with reduced time and space complexity.** *BMC Bioinformatics* 2004, **5**.

[44] Katoh K, Kuma K, Toh H, Miyata T: **MAFFT version 5: improvement in accuracy of multiple sequence alignment.** *Nucleic Acids Res* 2005, **33**(2):511–518.

[45] Thompson JD, Koehl P, Ripp R, Poch O: **BAliBASE 3.0: latest developments of the multiple sequence alignment benchmark.** *Proteins* 2005, **61**:127–136.

[46] Wallace IM, Blackshields G, Higgins DG: **Multiple sequence alignments.** *Curr Opin Struct Biol* 2005, **15**(3):261–266.

[47] Essoussi N, Boujenfa K, Limam M: **A comparison of MSA tools.** *Bioinformation* 2008, **2**(10):452–455.

[48] Freyhult E, Moulton V, Gardner PP: **Predicting RNA structure using mutual information.** *Appl Bioinformatics* 2005, **4**:53–59.

[49] Lindgreen S, Gardner PP, Krogh A: **Measuring covariation in RNA alignments: physical realism improves information measures.** *Bioinformatics* 2006, **22**(24):2988–2995.

[50] Yeang CH, Darot JFJ, Noller HF, Haussler D: **Detecting the coevolution of biosequences–an example of RNA interaction prediction.** *Mol Biol Evol* 2007, **24**(9):2119–2131.

[51] Shindyalov IN, Kolchanov NA, Sander C: **Can three-dimensional contacts in protein structures be predicted by analysis of correlated mutations?** *Protein Engineering* 1994, **7**(3):349–358.

[52] Pollock DD, Taylor WR, Goldman N: **Coevolving protein residues: maximum likelihood identification and relationship to structure**. *J Mol Biol* 1999, **287**:187–198.

[53] Yeang CH, Haussler D: **Detecting coevolution in and among protein domains.** *PLoS Comput Biol* 2007, **3**(11):e211.

[54] Wang ZO, Pollock DD: **Coevolutionary patterns in cytochrome c oxidase subunit I depend on structural and functional context.** *J Mol Evol* 2007, **65**(5):485–495.

[55] Pazos F, Citterich HM, Ausiello G, Valencia A: **Correlated mutations contain information about protein-protein interaction**. *J Mol Biol* 1997, **271**(4):511–523.

[56] Wollenberg KR, Atchley WR: **Separation of phylogenetic and functional associations in biological sequences by using the parametric bootstrap**. *Proc Natl Acad Sci U S A* 2000, **97**(7):3288–3291.

[57] Saraf MC, Moore GL, Maranas CD: **Using multiple sequence correlation analysis to characterize functionally important protein regions**. *Protein Eng* 2003, **16**(6):397–406.

[58] Lockless SW, Ranganathan R: **Evolutionarily Conserved Pathways of Energetic Connectivity in Protein Families**. *Science* 1999, **286**(5438):295–299.

[59] Suel GM, Lockless SW, Wall MA, Ranganathan R: **Evolutionarily conserved networks of residues mediate allosteric communication in proteins.** *Nat Struct Biol* 2003, **10**:59–69.

[60] Waddell PJ, Kishino H, Ota R: **Phylogenetic methodology for detecting protein interactions.** *Mol Biol Evol* 2007, **24**(3):650–659.

[61] Caporaso JG, Smit S, Easton BC, Hunter L, Huttley GA, Knight R: **Detecting coevolution without phylogenetic trees? Tree-ignorant metrics of coevolution perform as well as tree-aware metrics.** *BMC Evol Biol* 2008, **8**:327.

[62] Buvoli M, Hamady M, Leinwand LA, Knight R: **Bioinformatics assessment of beta-myosin mutations reveals myosin's high sensitivity to mutations.** *Trends Cardiovasc Med* 2008, **18**(4):141–149.

[63] Lee LC, Horn F, Cohen FE: **Automatic Extraction of Protein Point Mutations Using a Graph Bigram Association.** *PLoS Comput Biol* 2007, **3**(2):e16.

[64] Bonis J, Furlong LI, Sanz F: **OSIRIS: a tool for retrieving literature about sequence variants**. *Bioinformatics* 2006, **22**(20):2567–2569.

[65] Baker CJO, Witte R: **Mutation Mining – A Prospector's Tale**. *Journal of Information Systems Frontiers* 2006, **8**:47–57.

[66] Riloff E: **Automatically Constructing a Dictionary for Information Extraction Tasks**. In *Proceedings of the Eleventh National Conference on Artifical Intelligence*, MIT Press 1993:811–816.

[67] Riloff E: **Automatically Generating Extraction Patterns from Untagged Text**. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, MIT Press 1996:1044–1049.

[68] Hovy EH, Hermjakob U, Lin CY: **The Use of External Knowledge in Factoid QA**. In *Text REtrieval Conference* 2001.

[69] Ravichandran D, Hovy E: **Learning Surface Text Patterns for a Question Answering System**. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, ACL 2002:41–47.

[70] Tanabe L: **Text Mining the Biomedical Literature for Genetic Knowledge**. *PhD thesis*, George Mason University 2003.

[71] Huang M, Zhu X, Hao Y, Payan DG, Qu K, Li M: **Discovering patterns to extract protein-protein interactions from full texts.** *Bioinformatics* 2004, **20**(18):3604–3612.

[72] Hao Y, Zhu X, Huang M, Li M: **Discovering patterns to extract protein-protein interactions from the literature: part II.** *Bioinformatics* 2005, **21**(15):3294–3300.

[73] Salus PH: *A Quarter Century of UNIX (Addison-Wesley UNIX and Open Systems Series)*. Addison-Wesley Professional 1994.

[74] *http://www.alias-i.com/lingpipe/*.

[75] Mitchell JA, Aronson AR, Mork JG, Folk LC, Humphrey SM, Ward JM: **Gene indexing: characterization and analysis of NLM's GeneRIFs**. In *Proceedings of the AMIA 2003* 2003.

[76] Deshpande N, Addess KJ, Bluhm WF, Merino-Ott JC, Townsend-Merino W, Zhang Q, Knezevich C, Xie L, Chen L, Feng Z, Green RK, Flippen-Anderson JL, Westbrook J, Berman HM, Bourne PE: **The RCSB Protein Data Bank: a redesigned query system and relational database based on the mmCIF schema.** *Nucleic Acids Res* 2005, **33**(Database issue).

[77] Ogren PV: **Knowtator: a plug-in for creating training and evaluation data sets for Biomedical Natural Language systems.** In *Proceedings of the 9th International Protégé Conference* 2006:73–76.

[78] *http://www.iclc.it/Listanuova.html*.

[79] *http://www.biotech.ist.unige.it/cldb/cname-1c.html*.

[80] *http://www.cabri.org/HyperCat/cells/CellLines.html*.

[81] Binder RV: *Testing Object-Oriented Systems: Models, Patterns, and Tools (The Addison-Wesley Object Technology Series)*. Addison-Wesley Professional 1999.

[82] Cohen KB, Tanabe L, Kinoshita S, Hunter L: **A resource for constructing customized test suites for molecular biology entity identification systems.** In *BioLINK 2004: Linking biological literature, ontologies, and databases: tools for users.*, Association for Computational Linguistics 2004:1–8.

[83] Johnson HL, Cohen KB, Hunter L: **A Fault Model for Ontology Mapping, Alignment, and Linking Systems**. In *Pacific Symposium on Biocomputing* 2007.

[84] Craven M, Kumlien J: **Constructing biological knowledge bases by extracting information from text sources.** *ISMB 1999* 1999.

[85] Lu Z: **Text Mining on GeneRIFs**. *PhD thesis*, University of Colorado Health Sciences Center 2007.

[86] Schuman J, Bergler S: **Interactive Retrieval UsingWeights**. In *The Sixteenth Text REtrieval Conference (TREC 2007) Proceedings* 2007.

[87] Tari L, Tu P, Lumpkin B, Leaman R, Gonzalez G, Baral C: **Passage relevancy through semantic similarity**. In *The Sixteenth Text REtrieval Conference (TREC 2007) Proceedings* 2007.

[88] Wong W, Martinez D, Cavedon L: **Extraction of Named Entities from Tables in Gene Mutation Literature**. In *Australasian Document Computing Symposium Proceedings* 2008.

[89] Stokes N, Li Y, Cavedon L, Huang E, Rong J, Zobel J: **Entity-Based Relevance Feedback for Genomic List Answer Retrieval**. In *The Sixteenth Text REtrieval Conference (TREC 2007) Proceedings* 2007.

[90] Shah PK, Bork P: **LSAT: learning about alternative transcripts in MEDLINE.** *Bioinformatics* 2006, **22**(7):857–865.

[91] Donaldson I, Martin J, de Bruijn B, Wolting C, Lay V, Tuekam B, Zhang S, Baskin B, Bader GD, Michalickova K, Pawson T, Hogue CWV: **PreBIND and Textomy–mining the biomedical literature for protein-protein interactions using a support vector machine.** *BMC Bioinformatics* 2003, **4**:11.

[92] Karamanis N, Lewin I, Seal R, Drysdale R, Briscoe E: **Integrating natural language processing with FlyBase curation.** *Pac Symp Biocomput* 2007, :245–256.

[93] Hersh WR, Crabtree MK, Hickam DH, Sacherek L, Friedman CP, Tidmarsh P, Mosbaek C, Kraemer D: **Factors associated with success in searching MEDLINE and applying evidence to answer clinical questions.** *J Am Med Inform Assoc* 2002, **9**(3):283–293.

[94] Camon EB, Barrell DG, Dimmer EC, Lee V, Magrane M, Maslen J, Binns D, Apweiler R: **An evaluation of GO annotation retrieval for BioCreAtIvE and GOA.** *BMC Bioinformatics* 2005, **6 Suppl 1**:S17, [http://dx.doi.org/10.1186/1471-2105-6-S1-S17].

[95] Witte R, Kappler T, Baker CJO: **Enhanced semantic access to the protein engineering literature using ontologies populated by text mining.** *Int J Bioinform Res Appl* 2007, **3**(3):389–413.

[96] Roth C, Betts MJ, Steffansson P, Saelensminde G, Liberles DA: **The Adaptive Evolution Database (TAED): a phylogeny based tool for comparative genomics.** *Nucleic Acids Res* 2005, **33**(Database issue):D495–D497.

[97] Lozupone C, Knight R: **UniFrac: a new phylogenetic method for comparing microbial communities.** *Appl Environ Microbiol* 2005, **71**(12):8228–8235.

[98] Zhou Y, Wang R, Li L, Xia X, Sun Z: **Inferring functional linkages between proteins from evolutionary scenarios.** *J Mol Biol* 2006, **359**(4):1150–1159.

[99] Tuffery P, Darlu P: **Exploring a phylogenetic approach for the detection of correlated substitutions in proteins**. *Mol Biol Evol* 2000, **17**(11):1753–1759.

[100] Barker D, Pagel M: **Predicting functional gene links from phylogenetic-statistical analyses of whole genomes.** *PLoS Comput Biol* 2005, **1**:e3.

[101] Dimmic MW, Hubisz MJ, Bustamante CD, Nielsen R: **Detecting coevolving amino acid sites using Bayesian mutational mapping.** *Bioinformatics* 2005, **21 Suppl 1**:i126–i135.

[102] Dutheil J, Galtier N: **Detecting groups of coevolving positions in a molecule: a clustering approach.** *BMC Evol Biol* 2007, **7**:242.

[103] Martin LC, Gloor GB, Dunn SD, Wahl LM: **Using information theory to search for co-evolving residues in proteins.** *Bioinformatics* 2005, **21**(22):4116–4124.

[104] Dunn SD, Wahl LM, Gloor GB: **Mutual information without the influence of phylogeny or entropy dramatically improves residue contact prediction.** *Bioinformatics* 2008, **24**(3):333–340.

[105] Pollock DD, Taylor WR: **Effectiveness of correlation analysis in identifying protein residues undergoing correlated evolution.** *Protein Eng* 1997, **10**(6):647–657.

[106] Dutheil J, Pupko T, Jean-Marie A, Galtier N: **A model-based approach for detecting coevolving positions in a molecule.** *Mol Biol Evol* 2005, **22**(9):1919–1928.

[107] Tillier ER, Lui TW: **Using multiple interdependency to separate functional from phylogenetic correlations in protein alignments**. *Bioinformatics* 2003, **19**(6):750–755.

[108] Easton BC: **Novel techniques for detecting correlated evolution**. *PhD thesis*, Australian National University 2006.

[109] Easton B, Maxwell P, Isaev A, Huttley G: **A probabilistic method to identify compensatory substitutions for pathogenic mutations.** In *Proceedings of the 5th Asia-Pacific Bioinformatics Conference, volume 5 of Advances in Bioinformatics and Computational Biology*, Imperial College Press 2007:195–205.

[110] Kern AD, Kondrashov FA: **Mechanisms and convergence of compensatory evolution in mammalian mitochondrial tRNAs.** *Nat Genet* 2004, **36**(11):1207–12.

[111] Fodor AA, Aldrich RW: **Influence of conservation on calculations of amino acid covariance in multiple sequence alignments**. *Proteins* 2004, **56**(2):211–221.

[112] Fodor AA, Aldrich RW: **On evolutionary conservation of thermodynamic coupling in proteins**. *J Biol Chem* 2004, **279**(18):19046–19050.

[113] Wang ZO, Pollock DD: **Context dependence and coevolution among amino acid residues in proteins.** *Methods Enzymol* 2005, **395**:779–790.

[114] Horner DS, Pirovano W, Pesole G: **Correlated substitution analysis and the prediction of amino acid structural contacts.** *Brief Bioinform* 2008, **9**:46–56.

[115] Marqusee S, Baldwin RL: **Helix stabilization by Glu-...Lys+ salt bridges in short peptides of de novo design.** *Proc Natl Acad Sci U S A* 1987, **84**(24):8898–8902.

[116] Serrano L, Bycroft M, Fersht AR: **Aromatic-aromatic interactions and protein stability. Investigation by double-mutant cycles.** *J Mol Biol* 1991, **218**(2):465–475.

[117] Huyghues-Despointes BM, Scholtz JM, Baldwin RL: **Helical peptides with three pairs of Asp-Arg and Glu-Arg residues in different orientations and spacings.** *Protein Sci* 1993, **2**:80–85.

[118] Blaber M, Baase WA, Gassner N, Matthews BW: **Alanine scanning mutagenesis of the alpha-helix 115-123 of phage T4 lysozyme: effects on structure, stability and the binding of solvent.** *J Mol Biol* 1995, **246**(2):317–330.

[119] Sundaralingam M, Drendel W, Greaser M: **Stabilization of the long central helix of troponin C by intrahelical salt bridges between charged amino acid side chains.** *Proc Natl Acad Sci U S A* 1985, **82**(23):7944–7947.

[120] Klingler TM, Brutlag DL: **Discovering structural correlations in alpha-helices.** *Protein Sci* 1994, **3**(10):1847–1857.

[121] Meier M, Burkhard P: **Statistical analysis of intrahelical ionic interactions in alpha-helices and coiled coils.** *J Struct Biol* 2006, **155**(2):116–129.

[122] Fernandez-Recio J, Sancho J: **Intrahelical side chain interactions in alpha-helices: poor correlation between energetics and frequency.** *FEBS Lett* 1998, **429**:99–103.

[123] Fuchs A, Martin-Galiano AJ, Kalman M, Fleishman S, Ben-Tal N, Frishman D: **Co-evolving residues in membrane proteins.** *Bioinformatics* 2007, **23**(24):3312–3319.

[124] Sokal RR, Rohlf FJ: *Biometry*, W.H. Freeman and Company. 3rd edition :813–819.

[125] Sokal RR, Rohlf FJ: *Biometry*, W.H. Freeman and Company. 3rd edition :820–824.

[126] Knight R, Maxwell P, Birmingham A, Carnes J, Caporaso JG, Easton BC, Eaton M, Hamady M, Lindsay H, Liu Z, Lozupone C, McDonald D, Robeson M, Sammut R, Smit S, Wakefield MJ, Widmann J, Wikman S, Wilson S, Ying H, Huttley GA: **PyCogent: a toolkit for making sense from sequence.** *Genome Biol* 2007, **8**(8):R171.

[127] Saitou N, Nei M: **The neighbor-joining method: a new method for reconstructing phylogenetic trees.** *Mol Biol Evol* 1987, **4**(4):406–425.

[128] Atchley WR, Zhao J, Fernandes AD, Druke T: **Solving the protein sequence metric problem.** *Proc Natl Acad Sci U S A* 2005, **102**(18):6395–6400.

[129] Sokal RR, Rohlf FJ: *Biometry*, W.H. Freeman and Company. 3rd edition :593–601.

[130] Sokal RR, Rohlf FJ: *Biometry*, W.H. Freeman and Company. 3rd edition :240.

[131] Sokal RR, Rohlf FJ: *Biometry*, W.H. Freeman and Company. 3rd edition :813–819.

[132] Sokal RR, Rohlf FJ: *Biometry*, W.H. Freeman and Company. 3rd edition :423–439.

[133] Widmann J, Hamady M, Knight R: **DivergentSet, a tool for picking non-redundant sequences from large sequence collections.** *Mol Cell Proteomics* 2006, **5**(8):1520–1532.

[134] Codoner FM, O'Dea S, Fares MA: **Reducing the false positive rate in the non-parametric analysis of molecular coevolution.** *BMC Evol Biol* 2008, **8**:106.

[135] Fares MA, Travers SA: **A Novel Method for Detecting Intramolecular Coevolution: Adding a Further Dimension to Selective Constraints Analyses**. *Genetics* 2006, **173**:9–23.

[136] Gloor GB, Martin LC, Wahl LM, Dunn SD: **Mutual information in protein multiple sequence alignments reveals two classes of coevolving positions.** *Biochemistry* 2005, **44**(19):7156–7165.

[137] Bingle LE, Bailey CM, Pallen MJ: **Type VI secretion: a beginner's guide.** *Curr Opin Microbiol* 2008, **11**:3–8.

[138] Cascales E: **The type VI secretion toolkit.** *EMBO Rep* 2008, **9**(8):735–741.

[139] Yahr TL: **A critical new pathway for toxin secretion?** *N Engl J Med* 2006, **355**(11):1171–1172.

[140] Shrivastava S, Mande SS: **Identification and functional characterization of gene components of Type VI Secretion system in bacterial genomes.** *PLoS ONE* 2008, **3**(8):e2955.

[141] Aschtgen MS, Bernard CS, Bentzmann SD, Lloubs R, Cascales E: **SciN is an outer membrane lipoprotein required for type VI secretion in enteroaggregative Escherichia coli.** *J Bacteriol* 2008, **190**(22):7523–7531.

[142] Bnemann G, Pietrosiuk A, Diemand A, Zentgraf H, Mogk A: **Remodelling of VipA/VipB tubules by ClpV-mediated threading is crucial for type VI protein secretion.** *EMBO J* 2009.

[143] Moyle WR, Campbell RK, Myers RV, Bernard MP, Han Y, Wang X: **Co-evolution of ligand-receptor pairs.** *Nature* 1994, **368**(6468):251–255.

[144] Pazos F, Valencia A: **In silico two-hybrid system for the selection of physically interacting protein pairs.** *Proteins* 2002, **47**(2):219–227.

[145] Goh CS, Bogan AA, Joachimiak M, Walther D, Cohen FE: **Co-evolution of proteins with their interaction partners.** *J Mol Biol* 2000, **299**(2):283–293.

[146] Pazos F, Valencia A: **Similarity of phylogenetic trees as indicator of protein-protein interaction.** *Protein Eng* 2001, **14**(9):609–614.

[147] Kann MG, Shoemaker BA, Panchenko AR, Przytycka TM: **Correlated evolution of interacting proteins: looking behind the mirrortree.** *J Mol Biol* 2009, **385**:91–98.

[148] Hakes L, Lovell SC, Oliver SG, Robertson DL: **Specificity in protein interactions and its relationship with sequence diversity and coevolution.** *Proc Natl Acad Sci U S A* 2007, **104**(19):7999–8004.

[149] Karimpour-Fard A, Leach SM, Gill RT, Hunter LE: **Predicting protein linkages in bacteria: which method is best depends on task.** *BMC Bioinformatics* 2008, **9**:397.

[150] Kanehisa M, Goto S: **KEGG: Kyoto Encyclopedia of Genes and Genomes.** *Nucleic Acids Res* 2000, **28**:27–30.

[151] Eddy SR: **Profile hidden Markov models.** *Bioinformatics* 1998, **14**(9):755–763.

[152] Filloux A, Hachani A, Bleves S: **The bacterial type VI secretion machine: yet another player for protein transport across membranes.** *Microbiology* 2008, **154**(Pt 6):1570–1583.

[153] Fink JL, Bourne PE: **Reinventing Scholarly Communication for the Electronic Age**. In *The Coming Revolution in Scholarly Communications & Cyberinfrastructure, Volume 3*, CTWatch Quarterly 2007.

[154] Dublin M: **Mining for Text Treasure**. In *Genome Technology Newsletter*, Genome Technology 2008.

[155] Hatzivassiloglou V, Duboué PA, Rzhetsky A: **Disambiguating proteins, genes, and RNA in text: a machine learning approach.** *Bioinformatics* 2001, **17 Suppl 1**:S97–106.

[156] Kondrashov AS, Sunyaev S, Kondrashov FA: **Dobzhansky-Muller incompatibilities in protein evolution.** *Proc Natl Acad Sci U S A* 2002, **99**(23):14878–14883.

[157] Tardiff J: **Sarcomeric Proteins and Familial Hypertrophic Cardiomyopathy: Linking Mutations in Structural Proteins to Complex Cardiovascular Phenotypes**. *Heart Failure Reviews* 2005, **10**(3):237–248.

[158] Berg JM, Tymoczko JL, Stryer L: *Biochemistry, Fifth Edition : International Version.* W. H. Freeman 2002.

APPENDIX A

MUTATIONFINDER

A.1 Background

Chapters II and III of this dissertation focus on MutationFinder, an open-source text mining software package of which I am the primary developer. While those chapters focused on building and evaluating MutationFinder, the purpose of this appendix is to describe the MutationFinder software.

As noted earlier, comprehensive and accurate data on mutations that have been studied in specific genes or proteins is often required by researchers, but is time-consuming and tedious to compile manually. Traditional keyword-based information retrieval is often inadequate for retrieving mutation literature because mutation mentions are often abbreviated in non-standard ways. Automatically recognizing mentions of point mutations in text is therefore important for basic information retrieval, in addition to providing the framework for more complex processing of biomedical literature, such as automating the construction of mutation databases.

The seminal papers on extracting mentions of point mutations from text [28, 29] provide the initial bases on which to develop mutation recognition systems. These techniques have proven useful in applications such as OSIRIS [64], an IR engine for compiling literature documenting mutations in specific proteins, and Mutation Miner [65], which annotates protein structures with mutation data extracted from biomedical literature. However, prior to the release of MutationFinder, no concept recognition system for extracting descriptions

[36]The work presented in this chapter was originally presented in: *MutationFinder: A high-performance system for extracting point mutation mentions from text* (Bioinformatics, 2007 23(14):1862-1865)

of point mutations from free text had been made publicly available, and a corpus for directly comparing different approaches was lacking.

The novel contributions of this work are (i) improved precision and recall over existing approaches for identifying mutation mentions in text, (ii) an open-source and publicly available system for extracting mutation mentions from text, (iii) a large, high-quality gold standard data set for judging and comparing the performance of mutation extraction systems, and (iv) a script for judging the performance of mutation mention extraction systems.

A.2 Methods

A.2.1 Baseline mutation extraction system

A partial reimplementation of MuteXt served as a baseline for performance comparisons. The rules described in [28] for recognizing point mutation mentions were implemented, although extracted mutations were not validated against sequence data; the follow-up study (see Table 3.1) suggests that this validation step would likely have led to higher precision but lower recall in the baseline system.

A.2.2 MutationFinder

MutationFinder builds on the baseline system, and similarly uses regular expressions to identify mutations. MutationFinder's regular expressions are documented with examples in the source code. The baseline system is modified in six ways to improve precision and recall:

1. wNm[37] format mentions with one-letter abbreviations must have N > 9;

2. wNm format mentions with one-letter abbreviations must appear in upper-case letters;

[37]In this notation, w and m refer to the wild-type and mutant residue/base identities, respectively, and N refers to the sequence position. For example, *A42G* and *Ala42Gly* are both *wNm-formatted* mutation mentions.

3. Wild-type and mutant residue/base identities must not be the same;

4. MutationFinder specifies patterns incorporating non-alphanumeric characters, whereas the baseline system removes non-alphanumerics;

5. MutationFinder identifies mutations described in natural language (as opposed to completely abbreviated formats)[38] with specific patterns, whereas the baseline system uses a heuristic to match these mentions;

6. MutationFinder splits text on sentences and applies its regular expressions to each sentence, whereas the baseline system splits both on words and sentences and applies different regular expressions to each.

These modifications yield the performance data presented in this document; 1 through 5 result in improvements to precision, and 4 through 6 result in improvements to recall. Modifications to future versions of MutationFinder will be described via updates to the software documentation.

A.2.3 Gold standard data

Two independent gold standard data sets were developed along with MutationFinder: one for developing and one for evaluating the performance of mutation extraction systems. The data sets consisted of abstracts, which were randomly selected from the set of primary citations in the PDB for mutant protein structures. Annotation was performed with Knowtator [77], using an ontology that was developed for describing point mutations. To build the development data set, the primary system developer (Caporaso) annotated 605 point mutation mentions in 305 abstracts. The annotation process is described in detail in Section 2.4.

The development set informed the manual construction of MutationFinder's regular expressions. Development and tuning of these regular expressions for mutation extraction was

[38]For example, *alanine 42 was mutated to glycine*, as opposed to *A42G*.

performed by testing variations of MutationFinder, (e.g., by comparing case-sensitive and case-insensitive regular expressions when matching amino acid residue names), on this mutation corpus. These performance comparisons additionally motivated the six modifications presented earlier.

To test MutationFinder, the system was applied to an independent (completely blind) test data set. This data set consisted of 910 mutation mentions from 508 article abstracts, annotated by two of the authors of the original study who were not involved in system development (Baumgartner and Randolph).[39] To calculate interannotator agreement, fifty of the abstracts annotated by Caporaso were randomly selected and dispersed throughout these annotators' corpora. Mean pairwise interannotator agreement, calculated on the fifty overlapping abstracts, was 94%. Details on the interannotator agreement study are provided in Section 2.4.5. In total, 1515 point mutation mentions were annotated in 813 abstracts, and this complete mutation corpus is publicly available via the MutationFinder project webpage.

A.2.4 Performance metrics

As in Chapter II, we use Precision, Recall, and F-measure on three separate tasks for evaluating mutation recognition systems: *Extracted Mentions*, *Normalized Mutations*, and *Document Retrieval*. These metrics are introduced in Section 2.2 but briefly, the Extracted Mentions task requires that all mentions of all mutations be extracted from input text, while the Normalized Mutations task requires that at least one mention of each mutation be extracted from input text, and the Document Retrieval task requires that at least one mention of any mutation be extracted from input text.

[39]These authors were involved with porting MutationFinder from Python to Java and Perl, but no changes were made to the system during this process.

A.3 System details: The MutationFinder project

Three functionally identical implementations of MutationFinder, in Python, Perl, and Java, along with source code, extensive unit tests, the annotated development and test data, and a scoring script for mutation extraction systems, are available via the MutationFinder project webpage[40]. MutationFinder can be used either as a stand-alone application, or imported into other Python, Perl, or Java applications. The annotation set and scoring script are provided with the hope that these resources will foster continued work on this problem and provide a means for comparing approaches between groups.

The simplest way to use MutationFinder is via the provided scripts. These take as input a single file comprising a document collection, where each line in the input file defines an independent "document" in the following format:

`identifier` <*tab*> `text`

MutationFinder's test data utilizes PubMed identifiers and article abstracts, but any combination of identifiers and text sources can be used. For example, if users wish to process abstracts at the sentence level, they can split their abstracts on sentences and assign a unique identifier to each. As output, the system prints one line for each text source in the following tab-delimited format:

`identifier` <*tab*> `mutation1` <*tab*> `mutation2` ...

Mutations are listed in *wNm* format with one-letter residue/base abbreviations. Start and end byte-offsets of the matched text span can optionally be provided for each mention as well.

[40]http://mutationfinder.sourceforge.net

Table A.1: MutationFinder compared to the baseline system.
MutationFinder and a baseline system are compared in terms of precision (P), recall (R), and F-measure (F). MutationFinder achieves higher performance on all metrics than the baseline system. For direct comparison, the numbers of true positives (TP), false positives (FP), and false negatives (FN) are also provided. EM, NM, and DR refer to Extracted Mentions, Normalized Mutations, and Document Retrieval, respectively. (These data are reproducible with MutationFinder-0_3-beta.)

	Baseline			MutationFinder			Baseline			MutationFinder		
	TP	FP	FN	TP	FP	FN	P	R	F	P	R	F
EM	584	37	323	743	12	164	0.940	0.644	0.764	0.984	0.819	0.894
NM	326	29	150	384	10	92	0.918	0.685	0.785	0.975	0.807	0.883
DR	140	8	42	162	1	20	0.946	0.769	0.849	0.994	0.890	0.939

A.4 System performance

Mutation Finder was compared with the baseline system on our independent test collection. It achieved nearly perfect precision and a marked increase in recall over the baseline system on this blind test set. (See Table A.1.)

To estimate the error of MutationFinder and the baseline system, the independent test set was subsampled into 100 random data sets, ranging in size from 50% of the test set (204 entries) to 85% of the test set (431 entries). (This range was selected to allow for performance variability which might arise from having test sets of different sizes, but constrained to avoid variance from being over- or under-estimated.) Both systems exhibited low variance on these samples (Table A.2), and mean performances were nearly identical to the values presented in Table A.1. MutationFinder's low variance suggests that its high performance should generalize well to other mutation-related literature.

A.5 Summary

The MutationFinder software project, available at http://mutationfinder.sourceforge.net, contains MutationFinder, an open-source, rule-based system for extracting point mutation mentions from literature with high precision and recall; a high-quality, human-annotated

Table A.2: Mean ± standard deviation performance of systems.

Mean ± standard deviation for precision (P), recall (R), and F-measure (F) of MutationFinder and the baseline system calculated on 100 randomly selected subsets of the blind test data. Subsets randomly range in size from 50-85% of the original test set. EM, NM, and DR refer to Extracted Mentions, Normalized Mutations, and Document Retrieval, respectively. (These sample variance calculations may underestimate true variance.)

	Baseline			MutationFinder		
	P	R	F	P	R	F
EM	0.938 ± 0.012	0.642 ± 0.030	0.762 ± 0.023	0.984 ± 0.004	0.817 ± 0.022	0.893 ± 0.013
NM	0.917 ± 0.013	0.683 ± 0.032	0.783 ± 0.023	0.974 ± 0.007	0.803 ± 0.029	0.880 ± 0.018
DR	0.946 ± 0.013	0.771 ± 0.021	0.849 ± 0.014	0.993 ± 0.007	0.890 ± 0.016	0.939 ± 0.009

mutation corpus; and a scoring script for mutation extraction systems. Three functionally identical implementations of MutationFinder are available in Python, Perl, and Java.

APPENDIX B

NMI DOES NOT SUGGEST COMPENSATORY MUTATIONS IN MYOSIN

B.1 Background

Familial Hypertrophic Cardiomyopathy (FHC) is a relatively common autosomal dominant myocardium disease, clinically defined by an enlarged left ventricle and most known as the cause of sudden cardiac death in people under 40. Mutations in sarcomeric proteins are implicated in FHC, with the majority of known mutations (43% of 270) being single-residue substitutions in cardiac β myosin heavy chain (βMyHC) [157].

Sarcomeres are multi-protein complexes which form the basis of myofibrils, bundles of filaments which connect opposite ends of the interiors of muscle cells. Contraction in sarcomeres, achieved by ATP-dependent sliding of myosin across actin filaments, causes contraction of muscle cells, and thereby of muscle tissue. In myocytes (heart cells) this contraction is responsible for the beating of the heart, and in the left ventricle, ultimately for pumping oxygenated blood away from the heart. Sarcomeres are composed of thick filaments, thin filaments, and elastic filaments, cumulatively composed from nine sarcomeric proteins. Thick filaments are composed of the hexameric myosin, thin filaments of actin, and elastic filaments of titin. (The other sarcomeric proteins have structural and regulatory roles.) Two βMyHC polypeptides form the coiled-coil "rod" domain of myosin, and each interacts with a Regulatory Myosin Light Chain (MLC-2), Essential Myosin Light Chain (MLC-1) heterodimer to form the complete hexameric myosin protein [158].

More than 100 mutations in βMyHC, occurring in its two functional domains, have been linked to FHC. βMyHC mutations implicated in FHC range in clinical severity from mild, and barely detectable, to extremely severe, involving high rates of sudden cardiac

death in people under 40. Recently, functional assays have been performed to elucidate the molecular nature of βMyHC misfunction arising from mutations. One mutation, R403Q, which is implicated in very severe cardiac dysfunction, was observed to result in a lower than normal myosin/actin sliding velocity and with an increased rate of actin-activated ATPase activity. This suggests that the R403Q mutation may result in a sarcomeres requiring more energy and producing less contraction, an observation which intuitively seems linkable to an enlarged heart.

FHC has an incidence rate of 1 in 500, and causes an array of symptoms ranging from in severity from shortness-of-breath to sudden cardiac death. FHC, and therefore βMyHC, are of great clinical interest. A puzzling aspect of βMyHC substitutions is that several which are associated with severe effects in humans can appear as wild-type in other species. Understanding how this is tolerated will provide insight into the mechanistics of the sarcomere and to the molecular nature of FHC, and is the subject of this Appendix.

MyoMapr is a database designed to facilitate the study of mutations in the human βMyHC [62]. Based on a review of MyoMapr, thirteen mutations in the head domain were identified as pathogenic in humans but were found to be wild-type in other species. In this Appendix, I apply coevolutionary analyses to test the hypothesis that these mutations represent compensated pathogenic deviations[41] (CPDs), and that the *compensating* mutation can be identified as other positions in the protein that covary with the CPD position.

So far, the covariation analysis has not resulted in predictions of potentially compensating positions, and the negative results of this analysis are documented here. Of course the failure to detecting potentially compensating positions does not imply that they do not exist. I conclude this chapter with a discussion of possible explanations of why compensating positions may have eluded detection thus far (Section B.4).

[41]Compensated pathogenic deviations, defined in Section 1.5, are mutations that are deleterious in one species, but where the mutant residue was found as wild-type in another species. The term *compensating pathogenic deviation* was introduced in [156].

B.2 Methods

Four alignments, derived from data generated for the MyoMapr database, were used in this analysis. The alignments are all expected to yield very similar results, but simply reflect different multiple sequence alignment strategies. Each alignment was subjected to a coevolutionary analysis, and positions which achieved high scores with each of the putative CPD position was analyzed in the context of the structure and phylogeny to see if the cumulative data support a possible compensatory mutation relationship between the positions. The steps in this process are now detailed.

B.2.1 Alignments

Two myosin sequence alignments associated with the MyoMapr database were the subject of these studies: myosin_995 and myosin_98. The positions corresponding to the myosin head domain only were extracted from these alignments as this study focused exclusively on the head domain, and additional variations on the alignments were constructed by realigning the sequences with MUSCLE.

B.2.1.1 myosin_995_head

The myosin_995 alignment was originally constructed for the MyoMapr database with ClustalW, and subsequently adjusted using PyCogent. Sequences that introduced gaps in other sequences were deleted, as were sequences that contained gaps that were not shared by other sequences at the 99.5% gap identity level (i.e. pairs of sequences left in the alignment were 99.5% identical in gap pattern over the aligned region). The first 885 positions of the myosin_995 alignment were extracted to create the myosin_995_head subalignment, an alignment corresponding to the head domain only of the myosin heavy chain. This alignment contained 114 sequences of length 855.

B.2.1.2 myosin_995_head_muscle

All gaps were removed from the myosin_995_head alignment, and the sequences were realigned with MUSCLE using default parameters. This alignment contained 114 sequences of length 855.

B.2.1.3 myosin_98_head

The myosin_98 alignment was originally constructed for the MyoMapr database with ClustalW, and subsequently adjusted using PyCogent. Sequences that introduced gaps in other sequences were deleted, as were sequences that contained gaps that were not shared by other sequences at the 98% gap identity level (i.e. pairs of sequences left in the alignment were 98% identical in gap pattern over the aligned region). The first 1600 positions of the myosin_98 alignment were extracted to create the myosin_98_head subalignment, an alignment corresponding to the head domain only of the myosin heavy chain. This alignment contained 257 sequences of length 1600.

B.2.1.4 myosin_98_head_muscle

All gaps were removed from the myosin_98_head alignment, and the sequences were manually reviewed. Several sequences that did not appear relevant to the head domain were deleted. (These were sequences that were on the order of 90% or greater gaps in the head-only alignment, or were sequences that were much longer than the others.) The remaining sequences were realigned with MUSCLE, using default parameters. This alignment contained 225 sequences of length 1619.

B.2.2 Coevolutionary analysis

Each alignment was evaluated with joint-entropy-normalized mutual information (NMI), using the full amino acid alphabet and six reduced-state alphabets (charge_2, charge_3, a1_4m, polarity_his_4, hydropathy_3, and size_2). For each of the suspected CPD positions,

all other positions in the alignment were evaluated for significant covariation by selecting positions which achieved a NMI of greater than or equal to a pre-specified threshold. Thresholds of 0.75, 0.85, and 0.95 were evaluated. Any positions that were found to significantly covary with the CPD position were then evaluated in the context of a neighbor-joining phylogeny constructed from the alignment, and a structure of the myosin head domain (structure reference: 2MYS).

B.2.3 Coevolution result post-processing

Two filtering steps were incorporated to reduce false positive hits. These were implemented as post-processing steps applied to the coevolution result matrices. In the first, positions which did not contain the wild-type and mutant residues in at least two sequences each were ignored. This step is similar to the md2mc2 filter introduced in Section 5.3.3, except it required that the two residues of interest, the wild-type and the mutant (rather than any two residues), are present in at least two sequences. This ensures that there is a reasonable amount of data about the mutation being studied. Next, positions which contained greater than 10% gap characters were excluded from the analysis (i.e., the 10p_gap filter described in Section 5.3.1). This rules out sections of the alignment with many gap characters, which are frequently poorly aligned segments or positions which would be unlikely compensators. These two filters were always applied.

Additional experimentation was performed using the md2mc10p filter (see Table 5.1) as this appeared useful in the T6SS analysis (Chapter VI). The md2mc10p filter requires that a position contain at least two different amino acids which appear in 10% of the sequences each. Each alignment was evaluated with and without incorporation of this filter. The results were identical in both cases however, so the resulting data is not included in this report.

B.3 Results

Covariation with the CPD positions was infrequent. Potentially compensating positions were identified for only two of the thirteen mutations tested: two hits were identified for the S148I mutation, and one hit was identified for the P731L mutation. These hits had relatively low NMI scores as well – for both mutations, no hits were achieved with NMI greater than or equal to 0.85.

Figure B.1 shows the results in the context of the 2MYS PDB structure for two hits on the S148I mutation: the potentially compensating positions, 446 and 230. In all sequences containing I148, methionine is present at position 446, and tyrosine at position 230. As illustrated in Figure B.1, these positions are distant in the folded protein (Cα distances: 38.63$\mathring{A}$ and 40.19$\mathring{A}$, respectively), suggesting that this is likely a false positive hit. When the 148/446 residue pair is reviewed in the context of the phylogeny (Figure B.2), all I148 sequences are found to exist in a single branch of the tree, suggesting the possibility that the high NMI is the result of a phylogenetic effect[42]. Taking the structural and phylogenetic data together, the correlation between these positions seems more likely to have arisen from common ancestry than from a compensatory mutation relationship between the two positions.

The myosin_98_head alignment achieved a hit for the P731L mutation. On analyzing the data however, it appears that this is a false positive resulting from poor alignment quality at this region in the protein. Supporting that claim, when the myosin_98_head sequences are realigned with MUSCLE, which generally out-performs Clustal on benchmark data, the P731L hit disappears.

In the myosin_98_head_muscle alignment, there is a single hit for the S147I mutation which is different from the hit achieved with the myosin_995_head alignments. While the

[42]The correlated pattern does not negate the possibility that this is a compensatory relationship however. See Section 7.4.2 for a discussion of this point.

Figure B.1: Gln149 and covarying positions on myosin head structure 2MYS.

When the NMI threshold is set to 0.75, two positions covary with the CPD position. Residues belonging to a common functional domain are highlighted in a common color. Gln149 corresponds to the CPD position, and positions 232 and 448 are the covarying positions. Note that these position numbers differ slightly from those presented in the text, as a result of mapping position numbers between sequences with different gap patterns. The numbers appearing in yellow below the covarying positions are the Cα distances, in Angstroms, of the covarying position from the CPD position. When viewed in the context of the structure, these appears to be false positives as the residues are quite distant in the folded protein, and not in the same functional domain. The coloring of regions of the protein by domain was performed by my collaborator on this project, Massimo Buvoli.

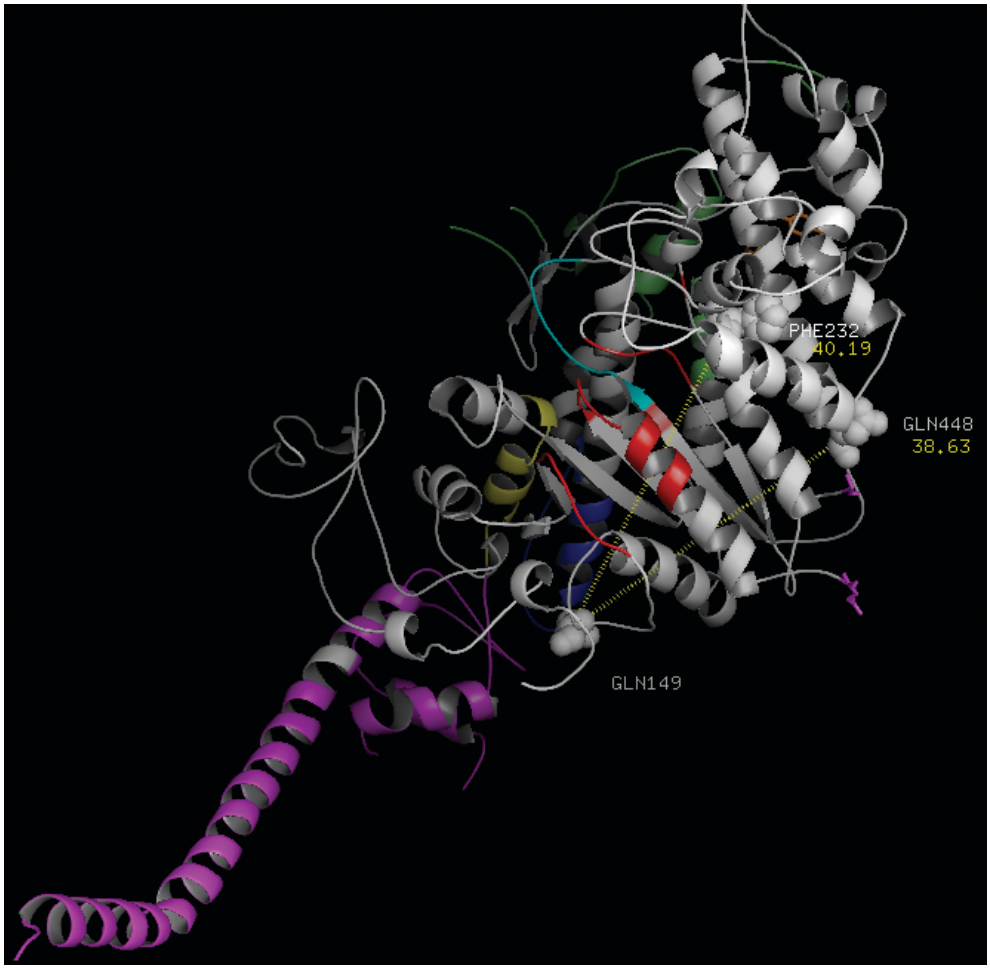
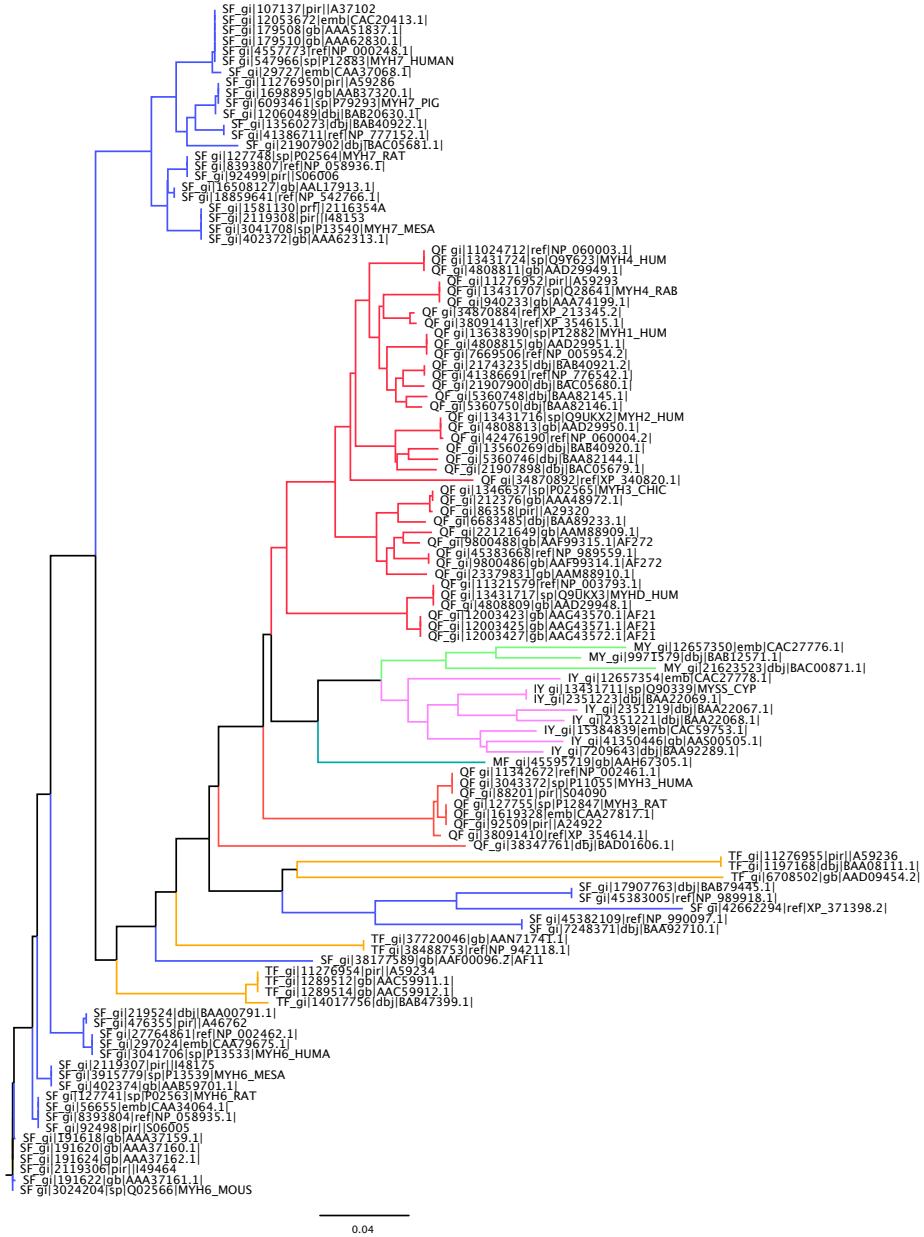


Figure B.2: Positions 148 and 230, with phylogenetic tree.

The residue identities at positions 148 and 230 are presented in the context of the phylogeny. The first and second characters on each tip label are the residue identities at positions 148 and 230, respectively. Each monochromatic clade indicates cluster of sequences with identical residues at positions 148 and 230. The tree was constructed by neighbor-joining using the myosin_995_head alignment.



compensated and compensating mutation (position 175) are closer in tertiary structure in this case (Cα distance: 20.04 $\mathring{A}$), the residue pattern at these positions is inconsistent with a compensatory mutation explanation because the same residue (isoleucine) is present at position 175 in sequences containing S147 or I147. The compensatory mutation hypothesis states that a different residue must appear at the compensating position in sequences which contain the mutant and wild-type compensated residue. This false positive hit appears to be a result of the low NMI threshold (0.75) when it was achieved; higher thresholds (0.85, 0.95) do not return this result.

B.4 Discussion

As mentioned above, the failure to detect compensating positions in this analysis does not imply that they do not exist. There are many possible reasons why this analysis did not yield compensating positions, and I discuss several of these possibilities here.

The first explanation that I address for our inability to identify compensating mutations is that the compensators may not detectable with the current coevolution detection algorithms. One scenario where this may be the case, and which I discuss further in Section 7.4.1, is that the position which compensates the mutation may not conserved. To illustrated this issue, imagine two clades, A and B, and three positions, x, y, and z. If x and y are involved in a salt bridge interaction in clade A, but x instead forms a salt bridge with z in clade B, y and z will be constrained by different evolutionary pressures in clade A then in clade B. In this example, the position with which x interacts and coevolves changes in different lineages. Because the current incarnation of coevolutionary algorithms focus on pairwise interactions, they can not address that situation. It is possible that this type of scenario prevents us from identifying the compensating position (or positions, in this case) in the myosin head.

Another explanation is that, for at least some of the mutations, we may be looking in the

wrong place by searching only for coevolving positions in the myosin head. If positions in a different protein (e.g., actin for mutations in the actin-binding domain) are compensating for the mutations, they will obviously not be detectable via an intramolecular coevolutionary analysis. This possibility has not yet been explored, primarily because of the time that would be required to address the sequence pairing problem (see Section 7.6).

Finally, it is possible that our hypothesis is simply wrong, and that the thirteen mutations are not CPDs. This would be a difficult or impossible alternative to test, as it would require accepting (rather than failing to reject) the null hypothesis. In future work I may try to address the first two of these possibilities in a further attempt to explain the apparent paradox of amino acid residues having different phenotypic consequences in different species. Only after more extensive analysis would I begin to lean more toward the conclusion that the mutations are not CPDs.