

On the identification and investigation of homologous gene families, with particular emphasis on the accuracy of multidomain families

Jacob M. Joseph

August 2012

CMU-CB-12-103

Publisher:

Lane Center for Computational Biology
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Committee:

Dannie Durand (advisor)
Takis Benos
Tanya Berger-Wolf
Russell Schwartz
Mona Singh

*This document is submitted in partial fulfillment
of the requirements for the degree of Doctor of Philosophy.*

©2012 Jacob M. Joseph

This work has been funded by NIH T32 training grant T32 EB09403 as part of the HHMI-NIBIB Interfaces Initiative, NSF grant DBI-0641313, NIH grant 1 K22 HG 02451-01, a Pittsburgh Supercomputing Center grant, and a David and Lucille Packard Foundation fellowship. These organizations have had no role in the design or execution of this work.

Keywords: genomics, gene family, homology, gene duplication, multidomain, network rewiring, neighborhood correlation, homology network, domain mutual information, gene family classification

Abstract

This dissertation addresses the identification and characterization of homologous gene families in large-scale, genomic data. Particular emphasis is paid to multidomain gene families, as multidomain sequences represent at least half of the sequence universe, but present an especially challenging case for family classification. Often, these sequences are excluded from analyses because they tend to interfere with classification performed with existing methods. This thesis develops the theoretical context for family classification of datasets that contain multidomain sequences, and demonstrates the implementation necessary for performing classification on large data sets.

Five primary results are presented in this work. First, a definition of homology that encompasses the evolutionary scenarios that result in multidomain families is formulated. Second, the techniques and implementation of family classification are presented. The methodology developed takes protein sequence data as input, and, by explicitly considering the evolutionary signal of gene duplication inherent in a sequence similarity network, derives a network that is an accurate estimate of homology. Third, the structure of this network is examined, and compared to the theoretical construct of a network of homology. Fourth, an approach for predicting families from this network is developed. Importantly, a statistical framework is presented for evaluation of the result using a limited set of curated families. Finally, the interplay between domains and the clustering result is examined using an information-theoretic approach.

Contents

Contents	v
1 Introduction	1
1.1 Areas addressed	4
1.2 Evolution of gene families	5
1.3 Locus model of multidomain homology	6
1.4 Classification of gene families	7
1.5 Relationship of domains and families	8
1.6 Summary of results	9
2 Background and preliminaries	13
2.1 Study of multidomain gene families	13
2.1.1 Descriptive studies	13
2.1.2 Modeling	14
2.2 Domain CONSISTENCY and Promiscuity	15
2.3 The homology network	16
2.4 Family classification	16
2.4.1 Use cases	18
2.4.2 Challenges of multidomain families	18
2.4.3 Clustering	20
2.5 Evaluation	21
2.5.1 Curated family benchmark	22
3 Network rewiring	25
3.1 Limitations of sequence similarity	26
3.2 Network rewiring	30
3.2.1 Empirical classification performance	32
3.3 BLAST sequence similarity	36
4 Optimization for large scale	43
4.1 Architecture	45
4.2 Key data structures	49

4.2.1	Networks of sequences	50
4.2.2	Hierarchical tree storage	55
4.3	Implementation of Neighborhood Correlation	58
4.3.1	Symmetric sequence similarity	59
4.3.2	Calculation order	61
5	Analysis of network properties	65
5.1	Network measures	67
5.1.1	Measure definitions	68
5.2	Interpretation of measures	70
5.3	Simulation	73
5.4	Analysis of Yeast networks	77
5.5	Analysis of human and mouse networks	81
6	Clustering and its evaluation	85
6.1	Evaluation metrics	87
6.2	Clustering methodology	89
6.2.1	Agglomerative, hierarchical clustering	90
6.3	Results	91
6.3.1	Sequence similarity	92
6.3.2	Neighborhood Correlation	99
6.3.3	Influence of additional data	105
7	The relationship between domains and clusters	109
7.1	Introduction	109
7.2	Mutual Information Formulation	110
7.3	Working Example of Framework	112
7.4	Areas of focus	116
7.4.1	Structure of predicted families	116
7.4.2	Characteristics of domains	117
7.4.3	Domain co-occurrence	119
7.5	Data	120
7.6	Clustering entropy	120
7.7	Relationship of domains and clustering	122
7.8	Mutual information of domains versus entropy	125
7.9	Characterization of clusters by domain content	128
7.9.1	Principal component analysis	129
7.9.2	Projection of clusters	131
8	Conclusions and future directions	141
A	Infrastructure	143
A.1	Software	143
A.1.1	Existing tools and packages	143
A.1.2	Developed software	143
A.2	SQL Schema	144

B Data	153
C Family score distributions	155
D Domains and clusters – supplementary data	161
D.1 Examination across lineages	161
D.2 Ordering of clusters by PCA component	174
D.3 Stability in single genomes	176
List of Figures	189
List of Tables	195
List of Code	197
Bibliography	199

Introduction

This dissertation focuses upon the identification of multidomain gene families, and examination of the genomic processes by which they arise. A gene family comprises the set of all sequences that evolved by vertical descent from a single common ancestor. Gene duplication followed by modification of one or both copies is the primary means by which new genes arise. Duplicated genes, or homologs, lead to a number of evolutionary scenarios by which new functionality can arise while still retaining the function of the ancestral gene. The novel function is most often related to the original function of the gene, and as a consequence, families of genes tend to exhibit similar properties of function. Gene families, and means of their identification, are of great interest for characterization of the processes by which genomes evolve, and provide insight into the functions of individual genes as a result of their memberships in gene families.

The structure of many proteins may be decomposed into domains, sequence fragments that fold to unique structures independent of the rest of the protein. A gene that encodes a protein that contains at least two domains is referred to as a multidomain gene, and a multidomain gene family includes at least one multidomain gene. Domains typically correspond to functional units, and multidomain proteins may exploit the combined function of all domains. Through domain insertion, the inclusion of another domain into a new context, such function need not be independently evolved.

Multidomain genes comprise a large fraction of the total number of known genes, and are widely recognized to be correlated with increases in organismal complexity, such as multicellularity. They comprise approximately 37% of the genes in multicellular organisms, and 39% in metazoans [142]. Multidomain proteins are central to signal transduction [17, 118], apoptosis [9], tissue repair, and the vertebrate immune system [142]. They often comprise scaffolds that facilitate the interaction of the functional modules of signaling pathways [62]. Interestingly, multidomain proteins are highly relevant to matters of human health; more than half of the members of the multidomain Kinase family, the largest protein family, have known roles in one or more cancer processes [64].

Despite the high prevalence and intriguing evolutionary relevance of multidomain genes, the study of multidomain gene families has been hindered in two primary ways. First, they do not fit the model implicit to existing family classification methods and lead to inaccurate results. Second, there has been ambiguity as to how gene families of multidomain sequences may be defined, and debate as to whether they can be thought of as families at all. Both have led to a common practice

of omitting multidomain genes from efforts to classify families.

The underlying difficulty is that the sequence fragments that encode domains may be mobile: a domain may arise in a sequence because it occurred in the ancestor of the sequence, but may also arise through insertion from some other gene. Such a history of a gene does not fit the original definition of homology proposed by Fitch, where all portions of a sequence must have the same history [51]. This dissertation demonstrates an extension of Fitch's definition, and allows reasoning about multidomain gene families. Additionally, and according to this new definition, this work presents a means of family classification that is robust to the complexities of multidomain gene families.

Classification of families is an important prerequisite for a number of biological questions. Post-genomic research relies heavily upon inference of the evolutionary relationships between genes. The use of model organisms frequently presupposes a mapping of genes to corresponding orthologs in other organisms, and an understanding of more complex relationships to other genes. Homologous gene families are the basis of phylogenomic inference [24] and evolution-based methods for function prediction and annotation transfer [71, 60, 61, 152]. Knowledge of family structure facilitates the study of the processes that drive family evolution (e.g., [43]). Whole genome sequencing efforts have inspired the construction of large-scale gene family databases with the goal of characterizing the full complement of homologous families over a broad range of genomes [72, 150, 39, 137]. These and other genome-scale applications require methods for accurate, automated, and high-throughput family classification.

A substantial goal of this dissertation is to support those research efforts, while developing a robust system architecture that facilitates rapid investigation in new directions. A major emphasis throughout this work has been to design and implement a framework that allows comprehensive, flexible exploration of source and intermediate data, and clear means of delineating the evidence used to reach conclusions in the output.

The approach undertaken in this dissertation is data-driven: amino acid sequences are the primary input. Comparison of sequences within a genome, as well as among several genomes, is used to derive signal to discover the underlying relationships among the proteins, and, in turn, infer the organization of homologous gene families. The input data are heterogeneous in source, evolutionary lineage, and degree of annotation, and are very large. Such data must be well organized to facilitate computation. Further, the scale of the data can present great computational requirements. This dissertation develops the methods and demonstrates efficient implementation for performing accurate family classification using practical resources, on large-scale genome data.

Much has been learned about the evolution of gene families from studies of specific families. These studies comprise many examples of gene family evolution and illustrate a set of fundamental genomic processes that drive their evolution. Yet, these are a small sampling of the protein space. Reliable, general means of family classification can facilitate discovery of characteristic properties of families and their evolution.

There are many unknowns about domains, and the relationship between domains and families. To the extent that this has been studied, much of the focus has been upon the problems this may cause for classifying families. The evolution of many families appears to be associated with specific genomic processes, acting at a variety of levels. Abstractly, events upon genes include

duplication, the unifying characteristic of a family; wholesale gene loss; and mutation of single codons. A hallmark of multidomain sequences is the transfer, or *shuffling*, of protein domains between sequences. Identification of families allows for consideration of these processes within the context in which they occur. For example, domain shuffling events modify individual sequences, and the propensity for particular events appears to be conserved within families. These propensities may only be evident, and will certainly be more clearly observed, when the relationships between sequences are known.

This work presents the framework to study the interplay of domains and families. This is a substantial, novel advancement over studies of bulk domain statistics within the context of individual sequences. Studies of families have constructed detailed histories of individual families (e.g., [41, 66, 125, 140, 147]). Identification of these families is not automated, and, importantly, these studies provide little ability to compare different families within a genome, or establish an understanding of a “typical” family. Further, while domain events within the family may be characterized, their scope is generally not such that they provide information about the contained domains that also occur outside of the context of that family. In an attempt to understand domain behavior, other studies consider statistics such as domain order, position, or co-occurrence within genes (e.g., [8, 27, 36, 144, 154]). These illustrate the distribution of domains over individual sequences, but cannot resolve their relationship with the evolution of families.

The quantity of protein sequence data available is increasing exponentially [120]. This, alone, motivates the development of methods that can cope with large datasets using practical computational resources. Yet, it is important to clarify *why* one should consider large datasets, and identify the realm of biological questions that one seeks to investigate. This is not merely of academic interest; the rate of growth of sequence data is at least as fast as increases in computational capacity and data storage. The use of more data increases the conserved phylogenetic signal. This allows identification of weak signal, and provides the level of discrimination necessary to identify disparate signals, such as between lineages that exhibit distinct evolutionary histories. In all cases, it lends greater confidence to the results obtained, distinguishing the hypotheses made from those merely supported by background noise.¹

The scope of the data required to support conclusions about gene families must be distinguished from that of, say, use of the entire universe of available protein sequences. As in all disciplines, a balance must be sought between the complexity and the quantity of data employed. More targeted selection of genomes can enable the use of more sophisticated techniques that would be impractical in extremely large datasets. Here, balance is sought to accommodate investigation within and between evolutionary lineages. In practice, this means constraining the set of genomes used to include as few genomes as may be required to address the granularity within a given lineage, while sufficient to cover the breadth of lineages relevant to the study. Make no mistake: even a carefully selected set of data remain sufficiently large to demand sophisticated approaches to data management and careful optimization of techniques employed.

¹Many types of *signal* and *noise* are addressed in this dissertation. Beyond the discussion in this Introduction, these are developed in a more detailed manner in Chapter 2: Background and preliminaries (p.13).

1.1 Areas addressed

This work may be decomposed into, and is presented as, five major emphases:

First, I frame a definition of gene homology that encompasses multidomain sequences, developed in early collaborative work [134].

Second, I present the techniques I have developed for performing family classification, including the framing of homology as a transitive network of genes. This mathematical foundation is used to reason about how closely homology may be estimated by typical measures, such as sequence similarity. Further, such a framework motivates an approach to exploit the structure inherent in an imperfect network. This can yield substantially better measures than what may be derived from pairwise measurements alone. The developed *network rescoring* approach, Neighborhood Correlation, uses the local structure of a weighted network of sequence similarity to correct for missed or incorrect inferences between all members of a network. A final step in the family classification pipeline developed in this work involves clustering of the resulting weighted network to establish a partitioning into discrete families.

Third, I describe the techniques and methodology that facilitate this research. The scale of the data sets under consideration requires careful attention to storage and computational demands. Most of the results presented in this dissertation are derived from a dataset of 48 genomes of disparate evolutionary lineage, comprising approximately 600k sequences [104]. The approach and methods here have been developed with an eye toward substantially larger datasets. To meet these needs, I have developed a robust, flexible infrastructure for exploratory research of evolutionary families using proteome sequences. The value of such an architecture is multi-fold:

Exploratory research is undertaken within the frame of general research directions, though is best accomplished when the data may be used to ask new questions and guide the specific methodology undertaken. The general framework is comprised of a central data store that interacts with discrete tools, each designed to accomplish a single task in a pipeline of related tools. This separation facilitates incremental, iterative development of the research. A central store of source, intermediate, and output data reduces duplication of the data, and greatly aids in cross-checking of methods, and, at least as importantly, consistency of the data itself. In this work, a wide range of input, intermediate, and derived data must be handled, including sequences, sequence meta-data, domain annotations, known families, networks of sequences, and derived clusters.

The scale of the data used here is not amenable to *ad hoc* approaches to computation, as when individual, “quick and dirty” programs are used to each investigate a question using myriad data sources. This is particularly relevant in that the goal of my research has not been solely to develop potentially useful methods, but also to employ them to increase our understanding of genome evolution. Organization and planning of infrastructure requirements become inevitable when one cannot simply re-compute an entire pipeline throughout development.

Many of the solutions developed here generalize to other areas of research, where one may not require a complete infrastructure built for sequence data and investigation of genome evolution. A modular framework allows separation of components for use within other infrastructures, possibly very different from that employed here.

Fourth, I discuss validation of family classification prediction. Evaluation is accomplished from two

primary perspectives: Extrinsic measures are used to evaluate accuracy of the classification using known families that we have drawn from primary sources. Intrinsic measures, such as measures of the connectivity of the sequence network, or the composition of clusters that are derived from such a network, are used in the absence of ground-truth data. Since the composition of very few families is known, intrinsic measures are especially valuable for demonstrating the consistency of family classification predictions. Further, the measures developed provide insight into the structure of gene families.

Finally, having proposed evolutionary families, it is possible to examine the biological properties of the genome in the context of those families. The interplay between domains and families is explored through an information-theoretic approach.

1.2 Evolution of gene families

The classical model of new gene formation is duplication of an existing gene, followed by mutation. Redundancies due to duplication tend to reduce the evolutionary pressures that conserve the function and composition of the gene. Four general scenarios of the fate of the duplicates are recognized [51, 74, 111, 156]. First, the redundant function may be neutral in pressure or deleterious, with the result that one gene becomes non-functional. This is the most likely fate, and is evidenced by the existence of large numbers of pseudogenes. Second, the duplication may facilitate greater expression of a gene product that is required in large quantities. Both copies will be retained. In this case, gene conversion or purifying selection can help preserve their identical composition. With a third scenario, subfunctionalization, the duplicates of a multifunctional gene may exhibit complementary loss of the duplicate functionality. Finally, one of the duplicates may acquire new function. This is referred to as neofunctionalization.

Homologous genes are those that have descended from a common ancestral gene (i.e., by vertical descent) [52]. An evolutionary gene family is the closure of all genes that are homologous. That is, each member of a gene family is related to all other members of that family through a common ancestor. By the same measure, any pair of genes that do not share a common ancestor are not homologous, and cannot be members of the same family. These properties connote that families do not overlap.

Duplication may arise at a continuum of scales, from small fragments to complete genes, chromosomes, and entire genomes. The processes of evolution by which point mutations occur are well studied [63]. The processes of multidomain protein evolution are not well understood in comparison with gene duplication.

The sharing and insertion of domains facilitates rapid evolution by reuse of components. Domains inserted into existing genes are likely to be expressed and conserved if they lend auxiliary function to the proteins in which they are found. Domains may evolve by vertical descent, being copied only via gene duplication, or may be widely dispersed among sequence contexts. Canonical examples of the latter include binding domains that may be integrated into several distinct proteins common only in binding mechanism.

Domain insertion can be mediated by a number of biological mechanisms. A body of work has looked at domain architectures observed in genomes to make inferences regarding domain processes. These studies abstract away from specific biological mechanisms and seek to describe the

relation of domain shuffling operations [54]. Abstract domain shuffling operations include insertion, duplication, and deletion. These act in concert with gene fusion and fission. Studies of genomic sequence DNA have led to many conclusions about the processes of sequence evolution that drive such operations.

The biological mechanisms responsible for rearrangement of sequences have been most extensively studied in the context of new gene formation, where the mechanisms of atypical splicing, unequal recombination, and retrotransposition are thought to dominate [11, 97]. For example, analyses of genome context illustrate that fusion of neighboring genes may frequently be attributed to faulty stop codons. Remote insertions are more likely at exon boundaries, via atypical splicing [11], mobile elements, exon shuffling, and non-allelic homologous recombination [97]. Gene and domain duplication by retrotransposition are evidenced by genes found in duplicate, but absent of introns.

1.3 Locus model of multidomain homology

The traditional definition of homologous families does not apply directly to multidomain proteins. Homology is not a divisible property, though multidomain proteins regularly contain sequence fragments of disparate lineage. Walter Fitch has stated that “We must recognize that not all parts of a gene have the same history and thus, in such cases, that the gene is not the unit to which the terms orthology, paralogy, *etcetera* apply” [52]. An inability to work at the level of genes limits our ability to apply homology-based methods to multidomain families. The goal of this dissertation is to employ the inferences about genes made possible by examining their evolutionary relationships, and devise methods to identify families of these genes. Gene families are a particularly suitable basis to examine the evolution of genomes. Domains are structural units, but genes are the units that are ultimately expressed. Accordingly, it is necessary to extend this definition.

In preliminary, collaborative work [134], we proposed an extension to the definition of homology that extends to multidomain sequences. The evolutionary unit, or gene, may be recognized as the locus of that gene on a chromosome. Homologous genes are related through vertical descent, by duplication and modification. In the context of a multidomain sequence, the locus may similarly be duplicated, resulting in two homologous genes. This paralogous relationship remains unchanged should one of the duplicates subsequently acquire mutations *or* acquire additional fragments of genetic material from some other source. Here, the history of the gene locus, not the origin of an inserted sequence fragment, defines the relationship between duplicates.

In this dissertation, I adopt the following definition of homology: Two sequences are homologous if the loci that encode them are descendants of the same ancestral locus. This definition expands the applicability of the gene family framework to a large and important class of proteins. It facilitates family classification on datasets comprised of complete genomes, without removal of sequences that may not fit the classification method.

A hypothetical history of such a multidomain family is illustrated in Figure 1.1. Here, we can retrace the history of the loci that encode the currently observable genes w , x , y , and z . Gene loci are represented as lines, and shaded polygons depict unique domains. Genes x , y , and z comprise a family, having arisen from a common ancestor, a_0 , through gene duplication events. Their domain composition differs; the ancestor of genes x and y acquired a domain through domain insertion, whereas z diverged prior to this insertion event. This domain does not affect the relationship of

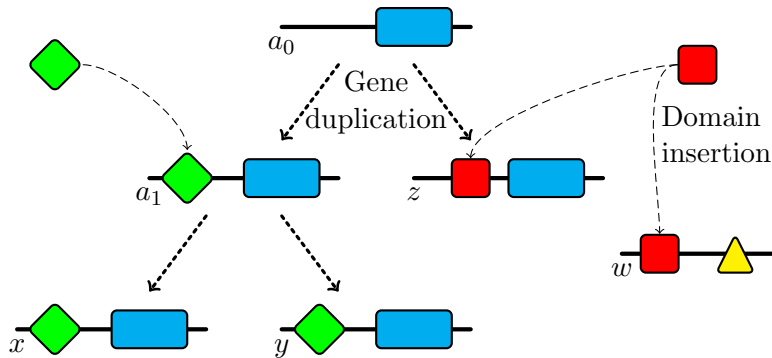


Figure 1.1: The evolutionary history of a hypothetical multidomain family, showing both gene duplications and domain insertions. Gene loci are depicted as lines, overlaid by polygons that represent domains. Genes x , y , and z share a common ancestor (a_0), but do not have identical domain composition; a domain (diamond, in green) was inserted into the ancestral gene a_1 after the divergence of gene z . Gene w shares a homologous domain with these genes, though there is no gene that is ancestral to both w and a member of the family.

x , y , and z to their common ancestor, a_0 . Genes z and w do contain a common domain, acquired by insertion. This common domain does not confer homology; gene w shares no common ancestor with z , or the other genes here.

The use of genes as the unit of evolution neither discounts the evolutionary history of single domain-sequences that evolve purely by vertical descent, nor the evolutionary history of sequence fragments acquired by insertion. Rather, it allows comprehensive study of the history of domains *and* of the history of the families in which they occur. The definition of homology adopted here applies to gene families that evolve through domain insertion into an existing gene. This is a plausible assumption for the evolution of many families because existing genes already have in place the promoter and regulatory components required for expression. Insertions into genes that already have regulatory signals are more likely to be expressed, and retained, than those into other locations. However, this also suggests that exceptions are possible. In what appear to be rare instances, a gene architecture may be assembled *de novo*; for example, by recruitment of a promoter to an existing sequence. Under the model used here, such a sequence would be considered the progenitor of a new family.

1.4 Classification of gene families

Grouping homologous genes into families facilitates (1) reasoning about the evolutionary history of those families, especially as compared between diverse lineages, and (2) investigation of the molecular or abstract processes that drive the evolution of families, in general.

Having formulated a formal definition of homologous families that includes domain shuffling, a further step is to devise means of identifying family membership using real data. The goal of gene family classification is to partition a set of unlabeled sequences into homologous families; i.e., sets of sequences derived from a common ancestral gene by speciation, gene duplication, and, in some species, horizontal gene transfer.²

²Note that this is distinct from orthology, in which only sequences related through speciation are considered.

Classification of multidomain gene families requires distinguishing genes related through vertical descent from those related only through domain insertion [134]. For example, in Figure 1.1, sequence-based approaches will tend to assign w to the homologous family, $\{x, y, z\}$, because genes w and z share a homologous domain. This assignment is incorrect; there is no ancestral genome that contains an *entire* gene that is ancestral to both w and z .

In the context of Figure 1.1, the task of family classification is to identify genes x , y , and z as a family, while excluding gene w . Regions of high similarity, such as due to a recent domain insertion, among unrelated sequences tend to confound traditional sequence-based approaches to classification. This is the primary effect that may lead to inappropriate inference of gene w as a member of the family illustrated.

At the same time, some families are highly divergent, resulting in weak sequence similarity among members. This can result from drift of the sequence, as well as domain events such as insertion or shuffling of the primary order of existing domains. Sequence divergence will decrease the similarity between homologous pairs, to a degree related to the amount of time since divergence, and, inversely, the extent to which the sequences are constrained by other evolutionary pressures. This can result in weak sequence similarity between homologous sequences, an effect known as *remote homology*.

1.5 Relationship of domains and families

The mechanisms by which multidomain families evolve are varied, and complex. In particular, the mechanisms by which they evolve are very different from single-domain families in ways that violate the assumptions of existing classification methods. A primary complication is that some domains are mobile within a genome, with the result that sequence fragments are transferred between gene loci that do not share ancestry.

There is reason to believe that some domains are unique to, or characterize, specific families. Others occur in many distinct families, and in concert with a variety of other domains. Some domains are mobile within a genome, with the result that large sequence fragments – and the corresponding function – are transferred between gene loci that do not share common ancestry. Others may be mobile within a specific sequence context. Others still may not be mobile at all, arising only through gene duplication.

I define the CONSISTENCY of a domain to be the propensity of that domain to evolve exclusively in genes that share common ancestry; i.e., within a single evolutionary family. A domain that appears in multiple families is INCONSISTENT, though this is not a binary property. For example, a domain that occurs exclusively in a single family, save one other sequence, is rather similar in behavior to a domain that does occur in only one family — and these patterns are quite distinct from a domain that occurs, say, in one sequence each of many families. Note that this concept refers to a biological property of a domain, *not* a count of the number of instances we may observe. (One estimate of this property may be such a count, but the biological property, and observed instances of it, should not be confounded.)

Domains, as single structural units, tend to confer a specific function to the protein in which they occur. Insertion of domains is a means by which genes may acquire functional modules. One may consider this interplay from the perspective of a domain, and characterize the functional properties of the domain, as well as its propensity to be exchanged. This approach does not harness

information from the sequence context associated with an instance of a domain. Relevant properties of a domain, including how likely it is to be inserted elsewhere, may depend upon sequence context.

An opposing, and equally extreme, perspective is to consider only the properties and functions of genes. This would not delineate the function or evolution of the domains as separate from the containing gene. In doing so, this disregards the value of aggregating information about a domain from many sequence contexts, and other genes of well-characterized function.

Neither domains nor genes need be considered in isolation. A balanced perspective that considers both the relevance of genes as units of evolution, and domains as elements that may be inserted, is most appropriate. Automated identification of gene families facilitates a means for understanding protein domains within the context of genes, and the families to which they belong.

While CONSISTENCY is a conceptual property of a domain, it will be useful to estimate it numerically. The approach taken in this work is to first identify families, and then evaluate the relationship of domains to those families. It should be noted that study of the interplay between families and domains requires that each be identified. The ability to accomplish either is dependent upon the other, but such a degree of interdependence also means that a better understanding of one may avail progress toward understanding the other. An effective estimate of CONSISTENCY could make it much easier to distinguish families. Examination of the relationship correlation between gene families and domain CONSISTENCY is a major focus of this work, and is specifically addressed in Chapter 7: The relationship between domains and clusters (p.109).

At the same time, INCONSISTENCY presents a major obstacle to family identification. Modular exchange violates many assumptions of methods for family classification that are based on sequence similarity. Domains that occur in unrelated sequences are not evidence of homology, regardless of how strongly similar the two instances of that domain may be. Sequence similarity does not distinguish between CONSISTENT and INCONSISTENT domains in comparing sequences.

1.6 Summary of results

The primary emphasis for family classification in this dissertation has been to develop methods that perform well on datasets that include multidomain sequences.

Early, collaborative work focused on developing a scoring method, Neighborhood Correlation [134], for pairs of multidomain proteins, where a high score indicates a prediction of homology. This method used the local structure of the sequence similarity network, as calculated by BLAST, as input.

In Song *et al.* [134], we demonstrated, using a curated set of known homologous sequences, that Neighborhood Correlation achieves high accuracy on the pairwise classification problem. Development of a curated set of homologous genes that included multidomain sequences was a significant contribution of this work. In addition to its utility for evaluating our own methods, this allowed us to elucidate the behavior of sequence similarity on such heterogeneous data. The success of Neighborhood Correlation demonstrated the efficacy of exploiting the local structure of the sequence similarity network to estimate homology.

In my dissertation research, I exploit this idea to obtain a novel and more powerful approach to classifying protein families through a focus on rewiring a sequence similarity network to better

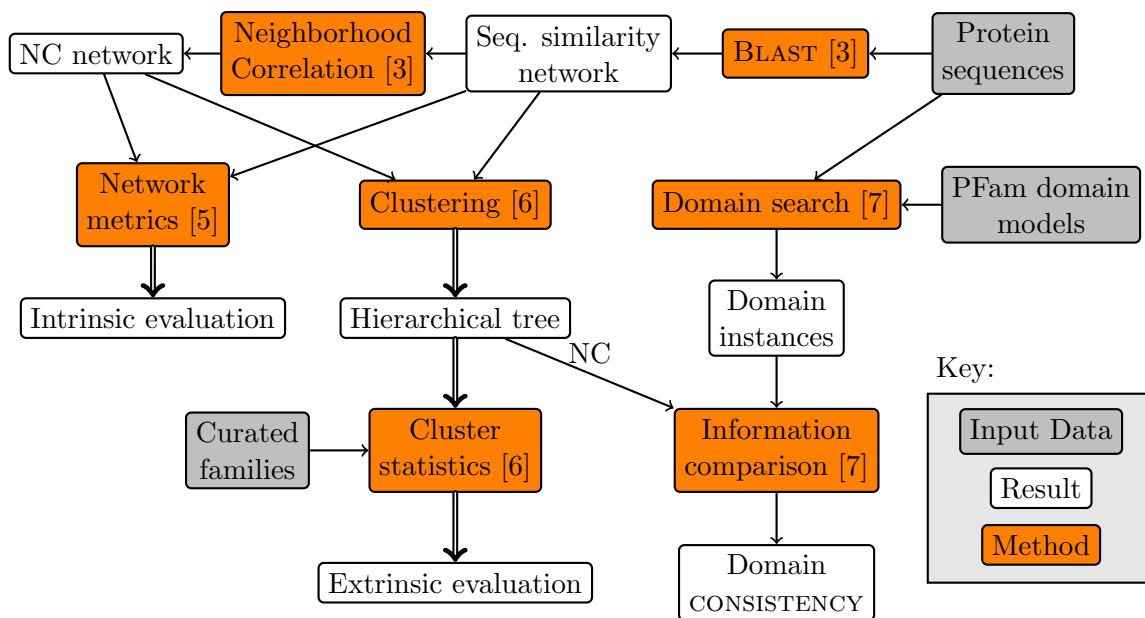


Figure 1.2: Work-flow of data and methods used in this thesis. Arrows represent data flow. Double lines represent parallel data paths; e.g., clustering is performed on either a sequence similarity network or a Neighborhood Correlation network, and the resulting tree is specific to that data input. Brackets indicate the chapter number in which a method is discussed.

approximate homology. This is followed by clustering nodes in this network to establish sets that represent predicted homologous gene families. Figure 1.2 depicts the sources of data used in this dissertation, the flow of intermediate derived data into individual methods, and the result of those methods. My approach to network rewiring is detailed in Chapter 3: Network rewiring (p.25). I applied this approach to a number of genome-scale data sets, described in Appendix B: Data (p.153).

Application of these ideas to very large data sets poses challenges to computation and storage. Most of the results presented in this dissertation are derived from a dataset of 48 genomes of disparate evolutionary lineage, comprising approximately 600k sequences. To accommodate this challenge, the approach and methods here have been developed with an eye toward substantially larger datasets. I developed a robust, flexible infrastructure for exploratory research of evolutionary families using proteome sequences. Briefly, this consists of a central SQL-based data store, a library of code for fundamental functionality, and discrete tools for specific methods. The design choices made and resulting architecture are discussed in Chapter 4: Optimization for large scale (p.43).

The underlying rationale of the rewiring approach used in this work is that the history of evolution is reflected in the structure of the sequence similarity network. My work is based on the hypothesis that this network structure reflects the events that occurred during evolution, and that it may be used to reconstruct the network of homology. It is therefore particularly important to understand the structural properties of networks based on real data. The results of these investigations are presented in Chapter 5: Analysis of network properties (p.65).

Reconstruction of a network of homology from sequence similarity is a substantial step toward family prediction, both in the theoretical constructs that may be employed, and toward a practical means of predicting homology. If the goal is to predict complete families, a means of partitioning this network into discrete families is required. Hierarchical clustering is employed to infer this partitioning from residual noise in the estimated network of homology. The techniques considered, and methods evaluation of the clustering result using curated data are discussed in 6: Clustering and its evaluation (p.85).

I use the results of the family classification pipeline developed to investigate the relationship between domain content and the underlying organization of protein families. In Chapter 7: The relationship between domains and clusters (p.109), I introduce an information-theoretic approach to examine these relationships. This allows for identification and quantitative comparison of the properties of domains and sequences which best characterize family structure.

Background and preliminaries

This chapter provides context for the challenge of family classification. The current understanding of the relevance of multidomain gene families is reviewed first. These works frame the theoretical concepts developed used to define the problem of family classification, and provide the empirical backing necessary to demonstrate their relevance to biology. After the introduction of the theoretical concept of a homology network, this chapter reviews existing approaches for family classification, and focuses upon the specific challenges that multidomain families pose for the accurate identification of families. The final section of this chapter describes issues related to effective evaluation of family predictions, and introduces the curated benchmark of 20 families used throughout this work.

2.1 Study of multidomain gene families

The advent of genomic sequence data has facilitated large-scale, empirical studies to characterize the structure of genomes and elucidate the processes by which they evolve. As the amount of genome data has increased, it has become possible to pose broad questions about the underlying biological processes, and examine the differences that exist between lineages. Two primary perspectives have guided these studies. The first is a descriptive approach, whereby a set of characteristics about genes and domains are enumerated and measured using the available data. The second perspective aims to construct a generative model that can explain the data.

2.1.1 Descriptive studies

Complete genome sequences have facilitated the study of novel and complex domain architectures in the evolution of new genes. Descriptive studies using this data have led to a characterization of the processes by which new genes arise, and enumeration of the molecular mechanisms that are responsible for abstract domain shuffling events. New genes are formed through duplication of DNA, which is understood to occur through retro transposition and tandem duplication events of single loci, segmental duplications of whole or partial chromosomes, as well as duplication of complete genomes [97, 111, 138]. Highlighting the relevance of large-scale duplications, it has been

hypothesized that whole-genome duplication presents an opportunity for entire signaling cascades to evolve *en masse* [37].

The formation of novel and varied domain architectures has been posited to have a driving role in corresponding increases in regulatory and organismal complexity [11]. The structure of sequence fragments that encode domains is correlated to a large degree with exon structure. As a consequence, the processes of exon shuffling are thought to be a primary driver for the formation of novel domain architectures [115]. This is particularly true in metazoa, where the rate of exon shuffling is observed to have greatly accelerated, and is believed to have been key to the formation of structural proteins involved in multicellularity [115]. A variety of molecular mechanisms are involved in exon shuffling, including atypical splicing events, and incorporation of existing exons into new contexts by mobile elements [11, 46, 54, 82, 107, 145]. Additionally, gene fission and fusion events can be responsible for genes with novel architectures [21, 47, 48, 91].

The inferred history of biological mechanisms can depend upon context. Whereas some domain instances may unambiguously correspond to genes derived through vertical descent, other instances may be better explained through complex rearrangement. This may, for example, result in insertion of a domain into different families. Further, the history of specific events responsible for a gene's evolution may not be fully defined by the resulting genome structure, and many genomic mechanisms may be indistinguishable. For example, insertion of a domain by read-through error or insertion at the C-terminus might be both modeled simply as a domain insertion.

More generally, the organization of contemporary genomes has been measured, and the variation of gene composition between lineages has been evaluated [98]. Increases in the complexity of genome structure and regulatory networks has been attributed to the processes of modular construction that result in multidomain genes [9, 142]. The evolution of individual domain families¹ has been studied extensively [27, 32, 62, 92].

Other studies have focused upon study of the functional consequences of new genes and domain architectures. Genomic datasets have enabled direct study of the evolutionary fates of these new genes with respect to regulation and function [55, 99]. Such studies of function have revealed examples of how evolutionary scenarios can differ among distinct lineages [138].

2.1.2 Modeling

The second perspective taken by empirical studies has been to examine the structure of genomes by fitting the data to parameters of a generalized model. These models tend to be based upon network theory, in which a graph theoretical framework is used to make conclusions about the processes that act upon the genome. In these, a network is typically constructed of nodes that represent either genes or domains, joined by edges that represent similarity, co-occurrence, or functional interaction. This approach is then typically defined by a fit of the data to a model of network growth. Such models reflect the scenario of ongoing evolution as growth of the network [18, 50, 110, 139]. The emergent behavior of such a network may then be ascribed to fundamental, natural processes inherent to genome evolution [90]. A fit of the data to the results of a generative model can suggest a mapping between the emergent properties of the model, and the underlying biological processes.

¹Domain families, which are comprised of all instances of a particular domain, should not be confused with gene families.

Network models have been used to describe general structural features of genomes [12, 21, 50, 105, 129, 154]. In response to the general explanatory power of some models of network growth, much of the literature concerns the fitting of a given generative model based upon the degree distribution observed in networks of sequence similarity and domain interaction networks. The most frequent discussion concerns the fitting of such data to a power-law distribution of node degree. Such distributions are inherent to many natural and physical processes [33], so it is plausible to imagine that these distributions exist in genomic data. Such fits lead to direct inference of the mode of duplication; i.e., through creation of a scale-free network by preferential attachment of new nodes to existing nodes with a tendency directly related to the degree of those nodes. This has led to a search for power-law distributions in genomic data. Finding such distributions suggests an elegant analog to the processes of gene duplication and the formation of novel domain architectures. The preferential attachment model of domain co-occurrence has been used to propose fundamental models of modular gene formation [92, 153]. Similarly, birth-death models of gene formation have been used to explain observed power-law distributions of the degree of related genes [78, 80, 79, 86, 87]. Birth-death models are conceptual match for the processes of gene duplication and loss.

Most modeling approaches are based upon fitting the observed genomic data to a theoretical model, and then using that model to explain the dynamics of the genome. However, if biological conclusions are to be reached, caution is necessary to consider the plausibility of the model with respect to underlying biological processes, even when the data fit a model with high statistical significance [136]. That is, many models may fit the data, but lack explanatory power. These inferences are critically dependent upon the accuracy of the fit to the data distribution. In the case of the universality of power law distributions, there is reason to question whether the fit to a power law is warranted, and whether such a fit is useful when used to explain the biological processes [33, 93, 136].

With respect to networks of domain co-occurrence, further evidence has suggested that a preferential attachment model does not accurately describe homologous gene families [121, 122].

2.2 Domain consistency and Promiscuity

The concept of CONSISTENCY was introduced in the preceding chapter is closely related to a property sometimes referred to as Promiscuity [101]. That term was coined in reference to domains that occur in several proteins, but are not evidence of a common biological function between those proteins. The goal of that initial study was to identify protein-protein interactions from sequence, and domains that could be identified as leading to incorrect inference of interaction were labeled as promiscuous, and excluded. The term has since been more broadly used, and definitions in the literature vary widely.

The most common usage is with respect to the frequency of co-occurrence of a domain with other domains in a sequence. This has been used to infer a propensity of a domain to be found in multidomain sequences [14, 142]. The underlying goal of many of these studies has been to better describe domain occurrence in genomic datasets, and, in turn, better understand the processes at work in the genomes. Toward such understanding, some have worked to extend statistics of domain promiscuity to account for the background frequency of domain occurrence, including the domain versatility index [148]. These measures consider the contexts in which a domain occurs, but do not

consider the dynamics of how the genes were formed. Others have explicitly related promiscuity to the mobility of a domain in the genome [15], and the modes of formation of individual domain families [27, 28]. Such studies tend to follow the initial usage of promiscuity, and frame it as a value that may be calculated from the data at hand, rather than an independent biological property of domains.

While domain promiscuity has been considered in the context of different lineages, and found to vary [14, 35], these formulations do not posit, or necessarily have a relationship to homology of the containing proteins. In this dissertation, CONSISTENCY is defined as a theoretical concept, and refers to the propensity of a domain to occur exclusively in genes that belong to a single homologous family.

2.3 The homology network

Many approaches to gene family classification represent the sequence universe as a network of sequences. Different networks use edges that correspond to a variety of measurable properties between each pair of sequences, such as the degree of sequence similarity, protein-protein interaction, or functional similarity. It is important to recognize that many such measures are not a prerequisite of homology between a pair of sequences. For example, while functional similarity tends to follow from homology, functional similarity is not *a priori* evidence of homology [52]. Homology is defined by common ancestry. Accordingly, some measures, such as sequence similarity, may be expected to comprise direct proxies of homology, in the absence of noise.

It is useful to frame this problem in the theoretical context, before proceeding toward a network defined by some measure. I define the homology network, $G_H = (V, E_H)$, where V is the set of all sequences and $(x, y) \in E_H$, if and only if x and y are homologous. Because gene families are defined through vertical descent, each sequence is homologous to all other sequences in its family, and only to those sequences. Otherwise stated, homology is a transitive property. If x and y share common ancestry, and y and w share common ancestry, then x and w must also share common ancestry. G_H is a disjoint union of cliques, in which each clique corresponds to exactly one gene family.

In practice, G_H is unknown, and homology is typically estimated using sequence comparison. This yields a sequence similarity network, $G_S = (V, E_S)$, where $(u, v) \in E_S$ if the sequences u and v have a significant degree of similarity. This network is weighted, by the measure of sequence similarity. Since sequence similarity is not a perfect predictor of homology, G_S will not, in general, be a transitive graph. Remote homology will result in missing edges, while spurious similarity, convergent evolution, and shared INCONSISTENT domains will introduce false edges. As a result, families no longer correspond to cliques or, in many cases, even to isolated connected components.

2.4 Family classification

The goal of family classification is to identify sets of sequences that share common ancestry. Conceptualizing this problem in the context of the homology network, G_H , provides a theoretical basis common to many approaches. Namely, given a network of sequence similarity, G_S , the goal of family classification reduces to that of reconstructing the edges in G_H , by restoring the edges that

are missing within the subgraphs that represent families, and removing spurious edges that may exist between unrelated sequences.

The defining evolutionary process of homology is that of speciation and gene duplication. In the context of the sequence similarity network, this may be represented as the creation of a new node, with an edge of strong similarity to the corresponding duplicate. Additionally, and importantly, all edges incident upon that corresponding node will be replicated to the new node. Were duplication the only relevant process in family evolution, dense regions of the sequence similarity network could readily be expected to directly correspond to families.

Sequences may also have similarity to other sequences that are not related. Such similarity is not a reflection of family evolution. Some similarity may be spurious; i.e., the consequence of chance in a large dataset. A short sub-string of one sequence is likely to be observed in another sequence when the size of the dataset is sufficiently large. Similarity between unrelated sequences may also arise due to convergent evolution.

Two strategies have been exploited to improve family classification accuracy. First, the prediction of homology could be improved over that of sequence similarity, reducing the number of errors that result in false or missing edges in G_S . Second, a clustering methodology could be applied to the network, to derive families from the structure of the network, without also including sequences that are connected by false edges in G_S . Efforts to improve family classification fall into both of these categories.

Many efforts have focused upon accurate strategies of pairwise homology prediction. Sequence similarity to a given sequence is typically measured by performing a BLAST [4] search, which is a heuristic to quickly identify regions of similarity between that sequence and any other similar sequence in a database. This yields a measure of similarity based upon the length of the common region and a model of the frequency of amino-acid substitutions [2, 3], as well as a statistical significance of this similarity with respect to other sequences in the database. With continued development, BLAST has incorporated additional statistical measures that more accurately account for the composition of sequences [7, 155]. Many studies have worked to improve the accuracy of sequence similarity search heuristics by improving the initial seed used for searching the database [22, 26, 157]. Others have sought to develop novel pairwise metrics that better correspond to homology [100].

BLAST is reliant upon the possibility of achieving an initial degree of similarity between a specific query sequence and another in the database. PSI-BLAST [5, 128] extends this methodology to consider queries based upon a profile constructed from one or more sequences. This approach facilitates iterative searches that identify homologs that might not be found directly using a particular query sequence. As a consequence, it greatly aids in the identification of remote homologs. Improvements in remote homology detection have helped to identify members of distant protein families, but, by design, they do not benefit the accuracy of homology detection that is not remote [134]. Additionally, the iterative, interactive nature of these techniques is not ideally suited toward large-scale or automated studies.

Both edges that are missing and those that exist, but falsely suggest homology in the sequence similarity network, G_S , may be thought of as the result of noise in the network. First, however, it is useful to understand that the errors in G_S are the result of biology, and generally are not an effect of measurement noise. The recovery of community structure amid noise is a typical task

of clustering [56]. That is, if a network has an inherent structure, such as that of the homology network, G_H , a clustering algorithm could be applied to resolve this structure, even if false or missing edges also exist. Clustering is a natural approach for organizing unstructured data, and a myriad of methods have been proposed for sequence data [20, 58, 76, 81, 83, 85, 88, 89, 100, 106, 117, 127, 124, 131, 149, 151]. The goal in many studies to group sequences, without an emphasis upon their evolutionary relationship. Others have pursued clustering for the application of identifying homologous sequences, families, and sub-families [1, 25, 49, 69, 94, 102, 109, 113].

The approaches of improved homology detection and clustering are interdependent, and may be combined. Many clustering algorithms seek specific structural features in graphs, based on the assumption that G_S still retains clique-like structures corresponding to families, despite noise. Conversely, better pairwise homology prediction methods will yield a network that better approximates homology and is more amenable to clustering algorithms. Reducing the amount of error in the estimated network will lead to a graph that better approximates the homology network. Clustering can then be applied to establish a partitioning that resolves the families.

2.4.1 Use cases

Two general use cases for family classification may be recognized. First, the construction of large-scale databases of gene families requires automated means of predicting the relationships between all genes in one, or, typically, a great many genomes. This demands methods that are robust to differences in the evolutionary processes of distinct families, which vary widely, as well as differences among disparate species lineages. Targeted, interactive homology detection methods represent a second approach. Here, the typical use case is to begin with a sequence, and iteratively search for other members of the same gene family. Widely used tools that fit this model include BLAST, which approximates an exhaustive search of pairwise similarities to the initial sequence, and PSI-BLAST, which iteratively performs searches using a model of sequences found in previous rounds.

Interactive tools exploit knowledge from the user and complement it with sequence data. They constrain the problem greatly by performing a local search from a given sequence of interest. However, note that this is not a general solution to discerning the organization of the entire space of all sequences. Local searches are not designed to provide insight into the statistics of or relationships between families not under immediate investigation. While this methodology has yielded much insight, and many useful tools, it demonstrates a clear compromise where small portions of data may be considered in detail, at the cost of robustness and consistency on large, heterogeneous data sets.

General and targeted approaches remain separate largely because homology detection methods are not perfect. If they were, a reasonable, perhaps preferable, approach would be to compute the automated analysis, with interactive searches being reduced to mere look-up of the precomputed result. The focus in this dissertation is upon automated, large scale classification and analysis.

2.4.2 Challenges of multidomain families

The evolutionary histories of multidomain sequences tend to violate key assumptions implicit in the use of sequence similarity for homology prediction. Sequence similarity is a function of the length and identity of the shared region between two sequences. Its use in family classification is

premised on the property that shared sequence is evidence of common ancestry. Additionally, it relies upon the property that the entirety of two homologous sequences are of the same origin and retain evidence of their relationship. Insertion of a domain into a sequence clearly violates each of these criteria, and, in practice, disrupts the result of methods that use sequence similarity to predict homology. Insertion of a domain from a non-homologous sequence has the effect of increasing the similarity between non-homologous sequences, possibly to a large degree. Further, domain insertion or permutation of the order of domains in a sequence can preclude alignment of the common regions in homologs, masking their relationship when measured by sequence similarity.

Families containing multidomain sequences exhibit a variety of additional processes that can affect sequence similarity. These processes yield events that act upon whole domains. That which poses the greatest challenge to family classification based upon sequence similarity is the insertion of a domain into an existing sequence. The result upon the sequence similarity network is the creation of an edge, of potentially high similarity, between the sequence from which the domain was copied, and the sequence into which it was inserted. Additionally, the recipient sequence will gain edges to any other sequence that also contains the domain. The evolutionary history of a domain does not necessarily match that of a sequence in which it is found. The edges in the sequence similarity network that arise through domain insertion from outside a family do not, by definition, represent homology of the sequences themselves. Such domains are *INCONSISTENT*.

In family classification, the dominant issues with multidomain families arise due to the *INCONSISTENCY* of some domains. Although domains are not explicitly considered in approaches that use sequence similarity, domain content strongly influences their result. In particular, the inappropriate merger of unrelated families via a series of one or more common domains plagues most family classification methods when applied to datasets that contain multidomain sequences.

In general, the application of a method to a dataset that violates the inherent assumptions of that method can generate artifacts in the result. An often-cited complication when using sequence similarity is *domain chaining* [70]. Domain chaining is an artifact of the methodology used for family classification. Figure 2.1(a) demonstrates how a series of families may be linked together by domains that each are found in at least two families. Pairs of families are each incorrectly grouped together because of at least one domain, and each of those is grouped with another family due to some other domain.

It is relevant to differentiate domain chaining, as in Figure 2.1(a), from that resulting from *INCONSISTENCY* of a single domain, as in Figure 2.1(b). With (a), absent any additional sequence information, there is no reason to believe that all pairs of sequences in the merged cluster should be grouped together. In particular, sequences in the left-most family have no domains in common with the right-most family. They are grouped only because of intermediate sequences that contain pairs of both intermediate domains. In (b), the common domain may be *INCONSISTENT*, or the clustering may be incorrect. The causes are different, and the resulting fault in classification may be mitigated by different means.

Methods based upon sequence similarity and transitive closure are prone to this mode of error. A single instance of a shared domain in a family can be sufficient for domain chaining to occur. Domain chaining can be caused by *INCONSISTENCY*, but domain chaining should not be assumed to be evidence of *INCONSISTENCY*. Further, solving domain chaining does not necessarily prevent the incorrect grouping of *INCONSISTENT* domains during family classification.

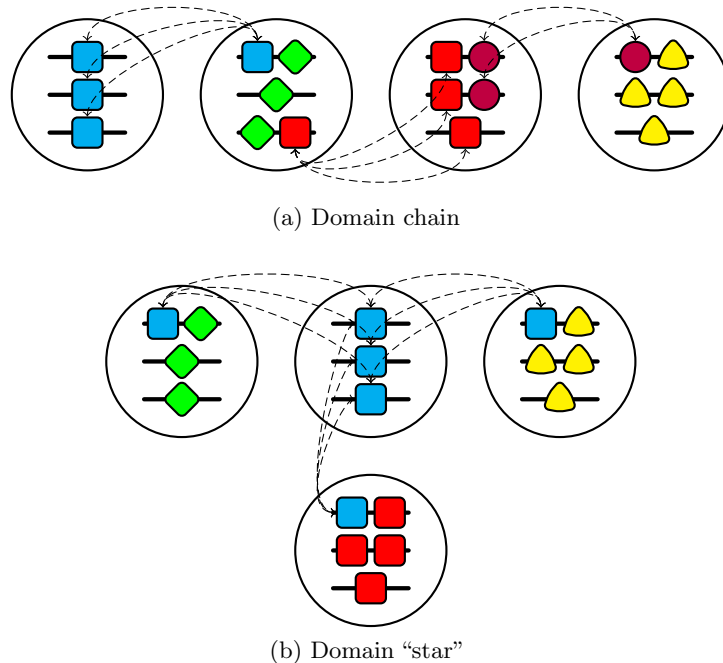


Figure 2.1: (a) A set of families grouped together due to a *chain* of domains, each found in one or more clusters. Note that no single domain is found throughout all families of the resulting cluster. (b) A set of families grouped together due to a single shared domain. Families are depicted as bounding circles, and domains as polygons on individual genes (lines).

In classification, inclusion of unrelated sequences can sometimes be cast as a problem of low precision. Ideally, one might increase the stringency of a threshold used to group families, while retaining high recall, and thereby separating them. However, many instances of domain chaining result in greater sequence similarity between non-homologous pairs than homologous pairs, for some families. For many families, such as those which are highly diverged, increasing the threshold has a propensity to discard the correct, but weak, relationships, while retaining those between the INCONSISTENT domains.

Although major gains have been made in the area of family classification overall, the problem of domain chaining remains largely unaddressed. Most work on homology prediction has focused on the problem of detecting remote homology without inappropriate inclusion of chance sequence similarity. A few heuristics to eliminate domain chaining have been proposed [75, 19, 135], but due to the lack of a gold standard, the effectiveness of these approaches was not evaluated.

2.4.3 Clustering

The underlying assumption implicit in recovering families from the sequence similarity network, G_S , is that the structure of this network does reflect the evolutionary history. If such structure does exist, the problem resolves to one of devising a method to recognize it and to identify the sets of sequences that comprise families. It is therefore helpful to consider the how the evolutionary processes that result in gene families may be expected to encode a signature upon the sequence similarity network. Insertion of a domain into a sequence from another sequence of the same family

yields a result in the similarity network whereby the recipient node gains edges to other sequences that contain the domain. When the domain is CONSISTENT, this may have a similar result as gene duplication, and increase the degree of similarity of the recipient sequence to other members of the family. Though less often observed, the insertion may also decrease the similarity of the sequence with other family members, as when the insertion disrupts a pairwise sequence alignment that covered the region into which the domain is inserted.

Domain shuffling events, and domain insertion in particular, can disrupt the community structure of the sequence similarity network, such that dense regions no longer correspond to families. Depending upon the magnitude of this disruption, a partitioning of G_S , as based on maximization of the closeness of nodes in a cluster and minimization of the similarity of disparate clusters, may not necessarily correspond to the family structure of the homology network, G_H .

While the structure of the sequence similarity network does not necessarily correspond to homology, the processes of family evolution do confer distinct structure upon the sequence similarity network. The underlying hypothesis here is that this structure may be recognized, and exploited to yield accurate predictions of homologous families.

Graph clustering algorithms used for protein classification in previous studies are guided by assumptions about the structure of the underlying similarity graph — in particular, that families correspond to dense subgraphs [49, 124]. However, few empirical surveys have been conducted to evaluate whether sequence similarity graphs, in fact, have these properties.

Unfortunately, these graph-based approaches tend to be either too permissive or too conservative. It is difficult to find thresholds that eliminate spurious edges without fragmenting families into much smaller subfamilies [95, 70, 114]. In the words of Liu and Rost [95], “...the dilemma between the Skylla of ‘restrictive thresholds yielding many small clusters’ and the Charibdis of ‘permissive thresholds yielding a few large clusters’ is a principle one.”

As further discussed in Chapter 6: Clustering and its evaluation (p.85), the data here is in the form of a similarity matrix.

2.5 Evaluation

Most empirical evaluations of methods to improve gene family classification have not used data sets designed to test sensitivity to domain chaining [6, 113, 151]. Many specifically exclude some domains and low complexity sequences. For example, the Astral data set used in [6, 113, 151], is a SCOP-based data set made up of single domain proteins. SCOP [108] and CATH [116] capture the structure single protein domains, and offer a good standard for remote homology, but will not evaluate domain chaining. Other benchmarks do include multidomain sequences, but are based on properties other than ancestry. The Gene Ontology (GO) [10] examines functional properties, but does not explicitly test evolutionary relationships.

Even when curated data is available, a lack of statistical metrics makes evaluation difficult. Chapter 6: Clustering and its evaluation (p.85) addresses this challenge.

For this problem, the biological property of interest (families that share common ancestry) corresponds to a precise mathematical property (graph transitivity). This ability to recast the problem to an objective goal guides method design, and provides a natural basis for evaluation in the absence

<i>Single Domain</i>		
Family	Size	Description
ACSL	15	Long chain acyl-CoA synthetase
FGF	46	Fibroblast growth factor
FOX	94	Forkhead box transcription factor
Tbox	32	T-box transcription factor
TNF	35	Tumor necrosis factor receptor
WNT	38	Wingless-related MMTV integration site
<i>Multidomain, Conserved Domain Architecture</i>		
Family	Size	Description
DVL	6	Dishevelled
GATA	12	GATA binding protein
KIR	13	Killer cell immunoglobulin-like receptor
Notch	8	Notch
TRAF	12	Tumor necrosis factor receptor associated factor
USP	100	Ubiquitin specific protease
<i>Multidomain, Varied Domain Architecture</i>		
Family	Size	Description
ADAM	61	A disintegrin and metalloprotease
Kinase	1057	Kinase
Kinesin	91	Kinesin
Laminin	23	Laminin
Myosin	51	Myosin
PDE	50	Phosphodiesterases
SEMA	41	Semaphorin
TNFR	56	Tumor necrosis factor receptor

Table 2.1: Curated benchmark of gene families, in the mouse and human genomes.

of a gold standard. This methodology is developed in Chapter 5: Analysis of network properties (p.65).

Notably, evaluation of the agreement of different methods is not a robust means of validating family classification. Such an approach is employed extensively for the closely related area of ortholog prediction [31, 44]. Evaluation of agreement is a reasonable basis to check whether disparate methods recapitulate the same information, but this is not a substitute for benchmarking against ground-truth data.

2.5.1 Curated family benchmark

In response to the lack of a comprehensive datasets to evaluate family classification performance, in early, collaborative work, we constructed a benchmark dataset of mouse and human families [134]. This test set was derived from the set of all 26,197 full length, mouse and human amino acid

sequences in the Swiss-Prot (version 50.9) database.

Twenty families with evidence of common ancestry were considered, including 1577 sequences in all. This set was based on a synthesis of over 70 publications by experts on specific families. The selection of families in this benchmark is necessarily limited by the availability of expert curation, though we believe this test set to be a characteristic sampling of families in the mouse and human genomes. The selected families represent single domain families, families of conserved multidomain architecture, and families of variable architecture. These families also represent a range of sequence conservation. Highly divergent single-domain families, such as the Tumor Necrosis Factors (TNF), were included to test performance on remote homology prediction. Details of the family curation procedure are given in [134].

Since initial publication of the benchmark of curated families, predictions of the set of genes in mouse and human have changed, primarily by the addition of genes. Accordingly, the benchmark dataset has been maintained to remain an accurate and complete listing of the genes in these twenty families. This dissertation adopts the Panther 7.0 [104] database of protein sequences as the set of sequences used in all analyses. Tables B.1 and B.2 list the set of genomes included in the Panther database. Of the approximately 603k sequences in this dataset, 45,491 are mouse and human sequences. Of these, 1841 sequences are a member of one of the families in this updated benchmark. Table 2.1 lists the complete set of curated families.

In addition to discussion of individual families, two aggregate sets of families will be considered. The set *ALL* is defined as the set of all sequences that are in the curated dataset. Because the Kinase family comprises a substantial fraction of the curated set, and is larger than any other family in our dataset, the set of all sequences, less kinase members is defined as *ALL-kin* to avoid bias. When considering homology, note that these combined sets do not increase the number of homolog pairs. That is, the set of homologous pairs in *ALL* is comprised of the merger of all pairs of sequences that are homologous within each family, not all pairs of sequences within the aggregate set of sequences from all families.

Network rewiring

The goal of homologous family classification is to partition genes into discrete sets of sequences, where each set comprises a single, entire gene family. As discussed in Chapter 2: Background and preliminaries (p.13), many approaches to family classification have been considered. The majority of these are based upon the principle of clustering a sequence similarity network. These are premised upon the concept that sequence similarity is an accurate estimate of homology and that, consequently, the structure of the sequence similarity network directly represents the relationship between gene family members. Under this framework, the problem of family classification reduces to one of establishing an accurate partitioning of the network (e.g., by maximizing compactness of each predicted family, and maximizing their separation) through application of a clustering algorithm.

However, sequence similarity as an estimate of homology is particularly prone to incorrect representation of family structure when multidomain sequences are considered. The primary challenge results from the property that many multidomain families contain sequences with domains that are also observed in other gene families. Compact sub-networks of high density are not observed in the sequence similarity network; rather, a common result is that groups of sequences are connected because of similarity among one or more shared domains. This lack of dense substructure tends to confound the basis of most clustering algorithms. It is also contrary to our understanding of gene family structure. This issue manifests to such an extent that direct application of a clustering algorithm to the sequence similarity network tends to fail to accurately represent datasets that contain multidomain families.

This chapter introduces and explores the consequences of a fundamentally different approach. Here, the strategy is to develop a *rewiring* method, to explicitly resolve the traces that evolutionary events encode upon the local structure of the sequence similarity network. More specifically, this approach is based upon the hypothesis that gene duplication imparts a unique signature upon the local structure of the sequence similarity network, and that this structure can be used to identify families. This inherent signal of family evolution is used to generate a new, weighted network that is an accurate estimate of homology. The resulting network is used for direct interpretation as a proxy for homology, and, in the following chapters, used as input to a clustering algorithm.

As motivation, this chapter details the limitations of sequence similarity when used as a basis for

family classification. These are demonstrated using empirical evidence from sequence similarity networks comprised of several genomes. Next, discussion focuses upon how gene family evolution imparts inherent local structure to a sequence similarity network. While the sequence similarity network does not directly mirror the homology network, the structure of this network does encode the traces of gene family evolution. This intuition is then mathematically specified, as Neighborhood Correlation, the network rewiring approach investigated here. Finally, this chapter addresses how Neighborhood Correlation may be calculated from real sequence similarity networks generated with BLAST.

3.1 Limitations of sequence similarity

The use of sequence similarity is premised on the concept that the entire length of two homologous sequences may be used as a source of evidence of their common ancestry, and that sequence similarity can detect such evidence. Consideration of the entire length is a reasonable constraint for a large fraction of the sequence universe: Many sequences are characterized by evolution exclusively through gene duplication and incremental substitution of individual amino acids. However, the evolutionary histories of many multidomain families, in particular, do not adhere to this model. Many multidomain families are characterized by sequences that have been modified by insertion of CONSISTENT domains.

Most means of sequence similarity are accomplished by aligning sites in a pair of (or, in some applications, many) sequences such that each pair of positions are presumed to have been derived from a single position in the common ancestor. A measure of sequence similarity is then derived from an evaluation of the degree to which these homologous sites have diverged since duplication from their common ancestor. Fewer sequence changes are expected to have occurred between sequences that have recently diverged or are highly conserved, and lower sequence similarity will result between more distant pairs of sequences. Further, this relies upon the concept that sequences that are not homologous do not have homologous sites, and, hence, cannot be aligned, modulo spurious regions of similar sequence composition.

The evolution of multidomain families can preclude effective alignment of sequences. The insertion of INCONSISTENT domains into disparate families can lead to high values of sequence similarity where there is no evidence of common ancestry. The occurrence of domains shared between a pair of genes frequently results in strong sequence similarity. This is true even when such genes are not homologous; the resulting similarity can be greater than that between genes that *are* members of the same family. I recognize that this is the dominant mode of errant homology detection among multidomain sequences [70, 134].

The abstract issues of domain chaining and remote homology manifest as concrete problems for datasets that are comprised of families that vary in their level of sequence divergence and the degree to which INCONSISTENT domains are included. In heterogeneous datasets, such as those with an unknown number of families, or those that include several genomes, the sequence similarity of homologous pairs of sequences is not consistently higher than those of non-homologous pairs. This may be examined empirically by comparing the distribution of scores between homologous sequences with the distribution of non-homologous pairs. A correct measurement of homology will yield higher and separable scores for pairs that are homologous, as compared to non-homologous

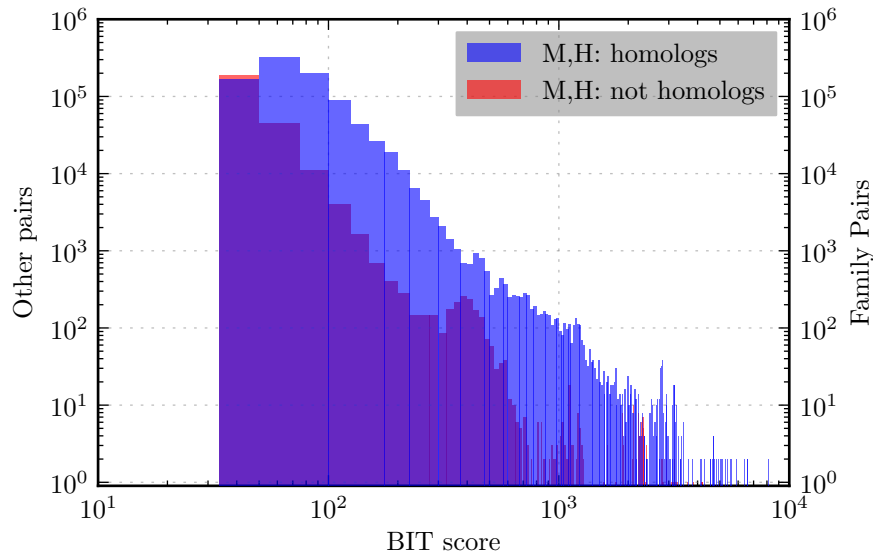


Figure 3.1: Histogram of the sequences similarity scores that result between homologous pairs of sequences (in blue, wrt. right axis) of the 20-family benchmark, in the combined mouse and human genomes.

sequences. Sequence similarity does not accomplish this.

Figure 3.1 demonstrates the result of sequence similarity for the mouse and human genomes. This figure considers all sequence pairs where at least one sequence is a member of one of the 20 homologous families in the curated benchmark. Two distributions are depicted. First, the distribution of scores between homologous sequences is shown in blue, with respect to the right vertical axis. Second, the distribution of scores between non-homologous sequences is shown in red. Here, these axes are of the same scale. In similar figures that follow, they differ, because the number of non-homologous pairs increases by the square of the number of sequences in the dataset, while the number of homologous pairs is a function of the square of the family sizes. These may be very different in magnitude. The bit-score is a measure of sequence similarity, so application of a more stringent threshold proceeds left-to-right. Clearly, the score distributions of homologous and non-homologous pairs coincide, and may not be separated on the basis of sequence similarity as calculated by BLAST, for this heterogeneous dataset of several families.

The properties of multidomain evolution that challenge sequence similarity may differ depending upon the specific family measured. With family classification, of course, the goal is to develop methods that yield consistently accurate results when applied to datasets where the families, or their evolutionary histories, are not known. Nonetheless, it is illustrative to examine specific families to demonstrate the behavior of sequence similarity more concretely. This may guide the development of methods that improve family classification.

For some families in the 20-family benchmark, sequence similarity can be an effective means of family classification. For example, Figure 3.2 shows the sequence similarity distributions of four families for which a bit-score threshold may be selected to distinguish homologous from non-homologous pairs of sequences. The families ACSL and FGF are single-domain gene families. PDE and SEMA are multi-domain. Members of the PDE family have only moderate conservation of sequence, but

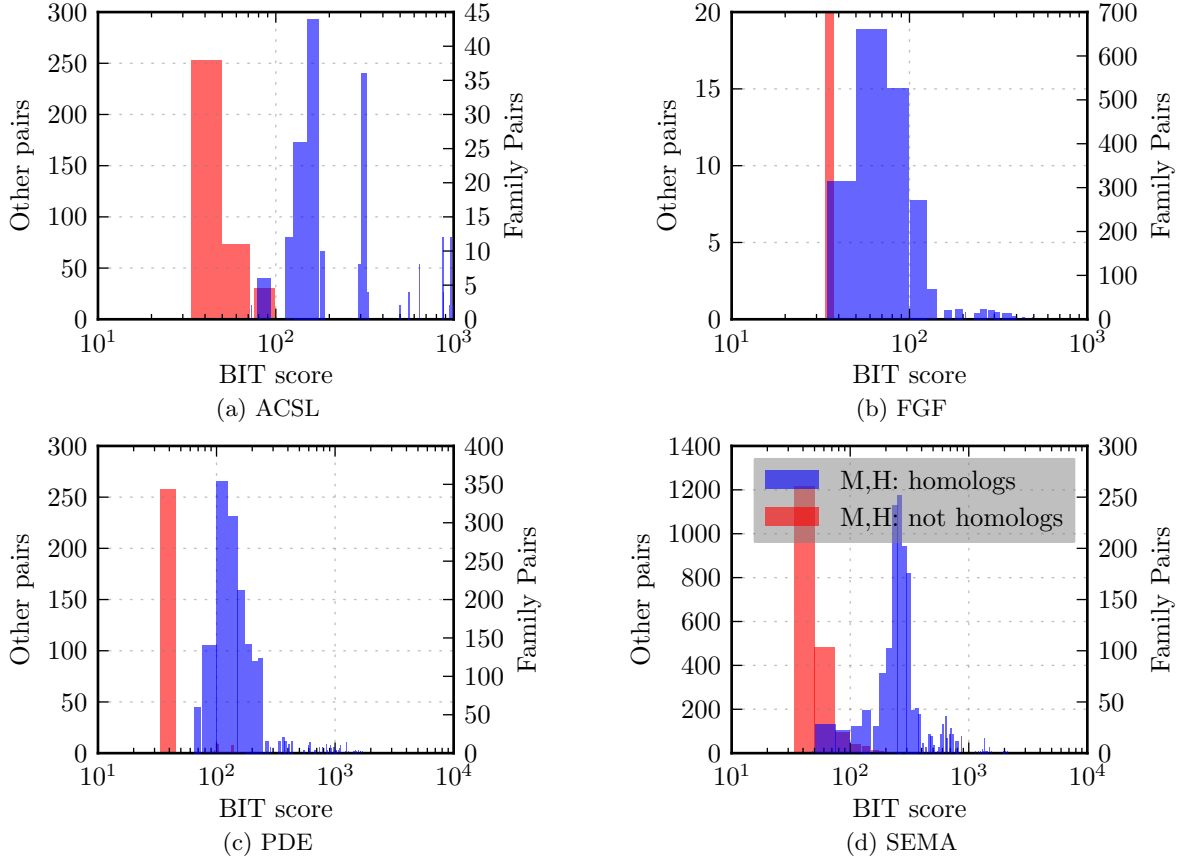


Figure 3.2: Histogram of the sequence similarity scores that result between homologous pairs of sequences (in blue, wrt. right axis), and between non-homologous pairs (in red, wrt. left axis), in the mouse and human genomes. These demonstrate that an effective bit-score threshold can be selected for some families, though no single threshold is suitable for all families.

this family contains few INCONSISTENT domains. The SEMA family has a high degree of sequence conservation and a small number of INCONSISTENT domains. For each of these families, there exists a bit-score threshold that separates homologous pairs from non-homologous pairs, to a large extent. Of these, only the PDE family is completely separable from non-homologous pairs. However, note that even in families without complex and varied domain architectures, such as these, no single threshold may be used to correctly separate homologs and unrelated sequence pairs over all families. In general, the particular threshold appropriate for classification with sequence similarity tends to be highly specific to an individual family. As a result, a sequence similarity threshold that is useful for one family may not be applied to a more general dataset.

While Figure 3.2 demonstrated that sequence similarity may be used to separate homologous and non-homologous pairs of sequences for some families, this is not possible for many other families. Figure 3.3 shows four families from the 20-family benchmark for which no bit-score threshold may be used to separate homologous and non-homologous pairs. Score distributions of all remaining families are shown in Figures C.1 and C.2. The families Kinase, Kinesin, and USP are comprised of varied and complex domain architectures. Each contains a number of INCONSISTENT domains. In

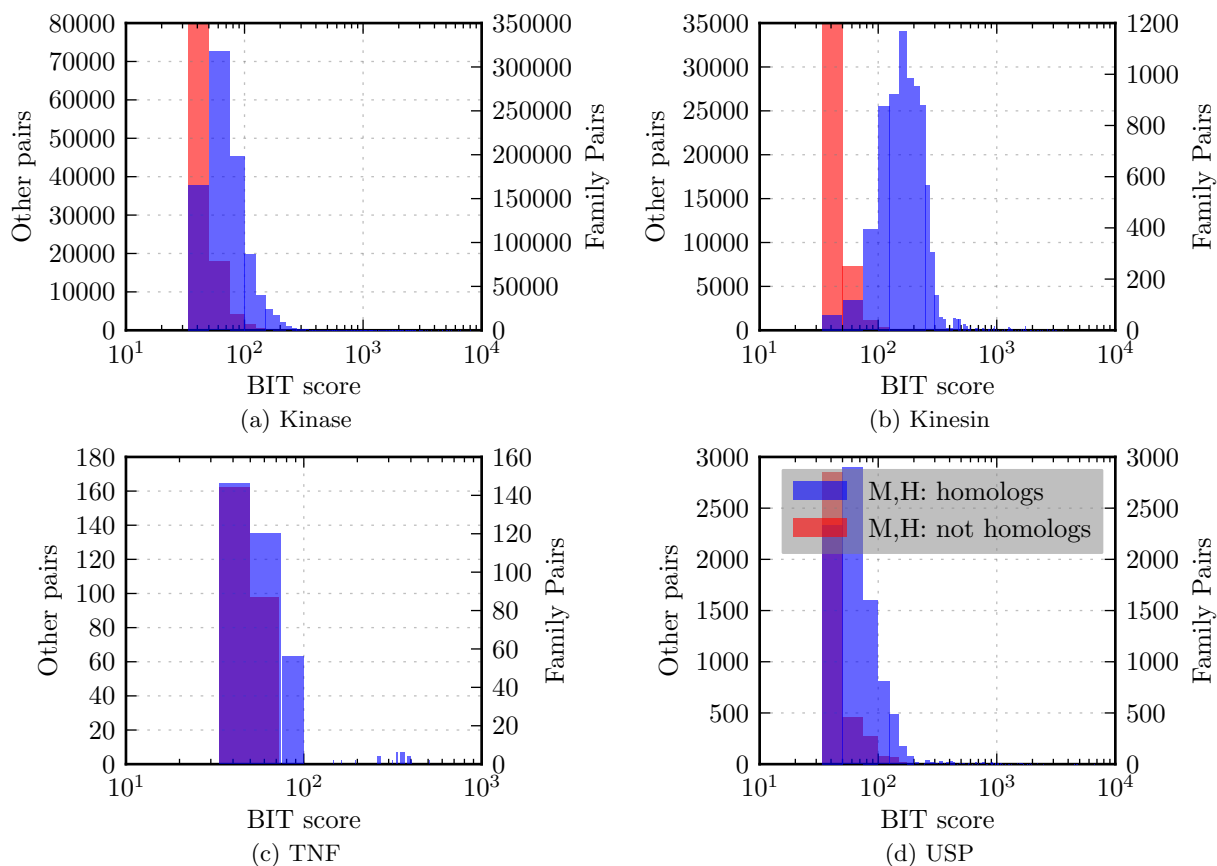


Figure 3.3: Histogram of the sequence similarity scores that result between homologous pairs of sequences (in blue, wrt. right axis), and between non-homologous pairs (in red, wrt. left axis), in the human and mouse genomes. No bit-score threshold may be selected for any of these families to effectively separate homologous and non-homologous pairs of sequences.

all cases, a more stringent threshold may be selected to include fewer non-homologous pairs, but only at the cost of substantially reducing the number of homologous pairs included. This inability to separate homologous sequence pairs is not constrained to multi-domain families. The members of the TNF family are single-domain, although conservation of sequence identity is quite low, ranging from 20–30% between pairs. Here too, no threshold may be selected to effectively separate the homologous and non-homologous pairs.

A number of heuristics have been proposed to improve the performance of family classification using sequence similarity. Many of these focus upon reducing the propensity for domain-chaining in the result, by considering the length of the alignment between pairs, or the distribution of domains [75, 19, 135]. Prior to our studies in [134], the effectiveness of these approaches had not been evaluated on heterogeneous datasets, due to lack of an effective benchmark dataset of homologous families. We found that these approaches did not improve family classification performance, and that their application was detrimental in many cases.

In particular, alignment coverage has been proposed as an additional constraint beyond sequence similarity to address domain chaining, with the assumption that alignments of greater length are

more likely between homologs. In practice, we found that while this can be effective in some cases, this approach is foiled by repeated or long domain insertions, and tends to reduce the accuracy of the family classification in the benchmark dataset [134].

PSI-BLAST [6], which computes sequence similarity in an iterative process of model construction, does improve remote homology detection. However, it is not well-suited to analyses on heterogeneous datasets because it exhibits optimal score thresholds that vary to a higher degree than BLAST.

Similarly, Song *et al.* [135] investigated the efficacy of a variety of means for explicit comparison of domain architecture to improve family classification. These suggested that another approach was required.

3.2 Network rewiring

The specific challenges demonstrated by the use of sequence similarity for family classification inspired our targeted approach to recover a more accurate estimate of homology by explicitly addressing these modes of error. The underlying premise is that gene duplication results in a specific signature in the sequence similarity network. This property is the defining feature of the mathematical concept of a homology network: a pair of duplicated genes each retain similarity to the entire set of homologous genes in the same family. Even without direct comparison of two such homologous genes, their homologous relationship can be revealed by their association with other members of the family. The assumption, stated here, and empirically demonstrated in this chapter, is that this signal is strong, and persists even when two homologous sequences have diverged to a degree beyond that readily recognized with sequence similarity.

By contrast, the process of domain insertion imparts a different structural feature to the sequence similarity network. Insertion of a domain into one sequence is likely to result in significant sequence similarity to other sequences that share the domain. At the same time, all other portions of that sequence will retain a degree of similarity to the members of the family to which it belongs. Similarly, a domain shuffling event in one member of a family will likely preclude full-length alignment of that sequence with other members of the family. That is, the degree of sequence similarity with each other member will decrease. In both cases, however, the homology may still be recognized by accounting for both the number and strength of these relationships.

Neighborhood Correlation takes a weighted network as input and calculates pairwise scores in the range $[-1, 1]$ between all pairs of nodes. Given a fully connected, weighted network, let w_x be the vector of similarity scores between x and all nodes in the network. The Neighborhood Correlation score, $\text{NC}(x, y)$, is the Pearson correlation coefficient between w_x and w_y . That is,

$$\text{NC}(x, y) = \frac{\sum_{i \in N} ((w_x[i] - \bar{w}_x)(w_y[i] - \bar{w}_y))}{\sqrt{(\sum_{i \in N} (w_x[i] - \bar{w}_x)^2)(\sum_{i \in N} (w_y[i] - \bar{w}_y)^2)}}, \quad (3.1)$$

where N is the number of sequences in the network, and \bar{w}_x is the mean of w_x .

For every pair of nodes in the sequence similarity network, Neighborhood Correlation defines a new score based upon the neighborhood of each node. Note that in Equation 3.1, for each node x , w_x is a vector of length N for a network of N nodes (the node itself *is* included). In practice, however, a

real sequence similarity network does not define a relationship between every pair of nodes. Pairs of nodes with no, or very low similarity are not included. As a result, a pseudo-score, of weight S_{\min} , is used for each pair of nodes with no defined sequence similarity score. That is, the weight of the edge between sequences x and i is

$$w_x[i] = \left\{ \begin{array}{ll} \log_{10} S(x, i) & \text{if } S(x, i) \text{ exists} \\ \log_{10} S_{\min} & \text{otherwise} \end{array} \right\}, \quad (3.2)$$

where $S(x, i)$ is the sequence similarity bit-score between sequences x and i . For real networks, the precise magnitude of S_{\min} has little effect upon the result of Neighborhood Correlation. In practice, it is set to a score of 95% of the smallest sequence similarity score.

A variant of Neighborhood Correlation, based on an unweighted input network can also be defined. This may be defined by substituting a binary value for $w_x[i]$ dependent upon whether the edge exists in the edge set, E , of the input network. That is,

$$w_x[i] = \left\{ \begin{array}{ll} 1 & \text{if } (x, i) \in E \\ 0 & \text{otherwise} \end{array} \right. \quad (3.3)$$

This formulation will be used for discussion of the behavior of Neighborhood Correlation with simple network motifs in this chapter, and during simulation, in Chapter 5: Analysis of network properties (p.65).

There are both biological and mathematical motivations for this rewiring approach. From a biological approach, the structure of the network encodes the evolutionary processes (domain insertion and gene duplication) that give rise to multidomain families. Local network structure also reflects domain architecture and sequence divergence. Intuitively, if x and y are members of the same gene family, they will tend to have similar BLAST scores when compared with other sequences in the family, resulting in high values of $NC(x, y)$. If y and z are sequences from different families that share an INCONSISTENT domain, y and z will tend to have similarity with a distinct set of sequences, outside of the shared domain. This will result in low correlation between w_y and w_z , and low values of $NC(y, z)$.

Mathematically, the rationale for Neighborhood Correlation may be understood as follows. Since gene families correspond to cliques in G_H , sequences within a family should have numerous edges to other members of the family. These relationships can be used to support edges missed by sequence comparison. This intuition suggests that a clique may be resolved from noise as long as a sufficient fraction of homologous edges are retained. Conversely, spurious edges are more likely to be in regions of low edge density and will not be supported by the surrounding local network structure.

Further, recent results by Sarkar *et al.* [126] demonstrate that link prediction based upon counts of nodes in the neighborhood common to two nodes has strong theoretical basis. The data here do not directly satisfy the constraints employed by that study; e.g., sequence similarity is not a metric, and may not be a metric in any latent space. However, the close relationship of these theoretical conclusions to the approach of Neighborhood Correlation suggests that the success of Neighborhood Correlation for family classification may not only be the result of a good heuristic, but due to a deeper relationship to network structure.

A few small examples may be used to illustrate how Neighborhood Correlation increases the transitivity of a network. Figure 3.4 illustrates three simple networks in which every edge is either zero

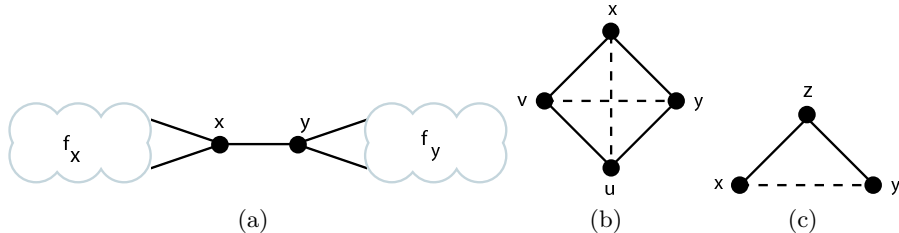


Figure 3.4: Example graph components for intuition. In (a), x and y are members of families f_x and f_y , respectively, but joined by a single edge. (b) depicts a single family missing two edges, while (c) illustrates a case where edge weights must be used to distinguish between edge addition or deletion.

or one. First, an example of two unrelated families linked by a single edge suggests an appropriate threshold for separating the cliques. The network depicted in (a) shows a connected component consisting of two subgraphs, each of three nodes or more, corresponding to families f_x and f_y . Node x is adjacent to at least two nodes in f_x , and y is similarly adjacent to at least two nodes in f_y . In this scenario, $\text{NC}(x, y)$ approaches 0.5. If x or y has more than two neighbors in its respective family, then the $\text{NC}(x, y)$ will decrease further. A threshold of $\text{NC} > 0.5$ will eliminate the spurious edge (x, y) , correctly splitting the component into two separate families. This suggests that a threshold of 0.5 will separate unrelated families in G_{NC} .

We next consider whether this threshold is low enough to restore missing edges to a clique. A family of size four is shown in Figure 3.4(b). Two additional edges, (x, u) and (v, y) , would be needed to form a clique. Here, $\text{NC}(x, u)$ and $\text{NC}(v, y)$ approach 0.6, and $\text{NC}(\cdot, \cdot) \rightarrow 0.8$ for all edges already present in the component. With a threshold of $\text{NC} > 0.5$, the existing edges will be retained and transitivity is increased by the added edges, completing the clique. In general, for any connected component of size $k > 3$, with at least $\frac{k(k-1)}{4}$ edges, more edges will be added with score $\text{NC} > 0.5$, yielding a denser component and increasing network transitivity overall.

While the unweighted model is a useful abstraction for theoretical analysis and simulation, a weighted graph based on sequence similarity scores should be used for real data. Consider the example in Figure 3.4(c). If x , y , and z represent a family, then a third edge, (x, z) should be added. On the other hand, if, say, x and y form a family of size two and z is unrelated, then (y, z) should be removed. In this case, connectivity alone provides no information to make this decision and Neighborhood Correlation can yield no additional confidence. In a weighted graph, $S(x, y)$ and $S(y, z)$ would determine whether the edge should be added or subtracted. Therefore, the information provided by edge weights should be utilized when working with a real sequence similarity network.

3.2.1 Empirical classification performance

By empirically demonstrating that rewiring according to the understanding of family evolution discussed above does yield accurate homology prediction, we demonstrate that structure captured is typical of real families.

Figure 3.5 shows the distributions (in red and blue) of Neighborhood Correlation scores for four families for which a sequence similarity threshold could be selected to partition homologous and

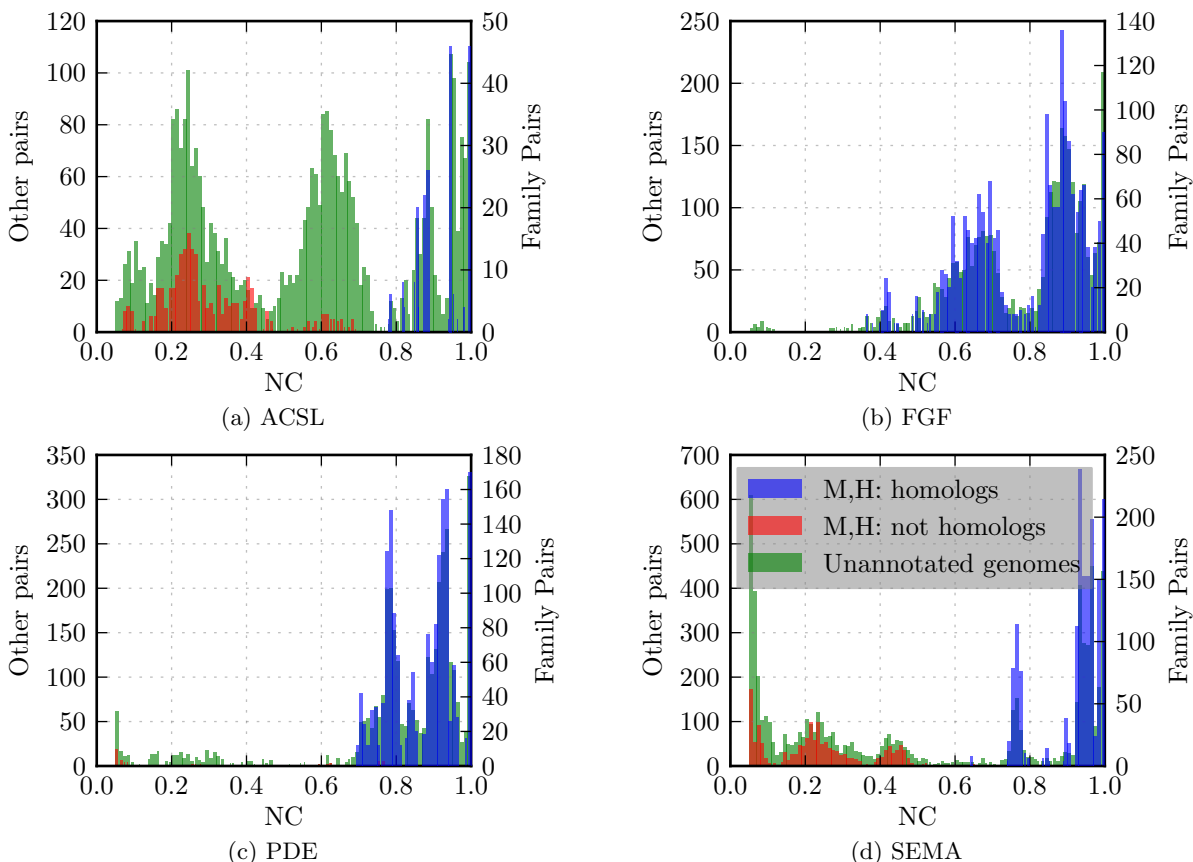


Figure 3.5: Histograms of the Neighborhood Correlation scores for curated families in mouse and human. Homologous pairs are in blue (wrt. right axis), and non-homologous pairs are in red (wrt. left axis). Additionally, in green (wrt. left axis) are scores of all pairs that include one member of a curated family, and one member from the 12-genome dataset, and not in mouse or human.

non-homologous pairs. More so than the performance of sequence similarity shown in Figure 3.2, there exists a threshold of Neighborhood Correlation scores that can accurately partition family and non-family members in each of these families. Indeed, for the FGF family, Neighborhood Correlation yields no non-homologous pairs with scores of $NC > 0.05$, the minimal score stored on disk in our databases. This success is a most basic sanity check that the application of rewiring does not degrade performance on families that may be trivially classified, at least when each family is considered alone. A valuable improvement with Neighborhood Correlation is that a single threshold now may be selected to simultaneously yield good performance on all of these families. A threshold of approximately $NC > 0.3 - 0.5$ is an acceptable compromise that will perform well here for all families here. These thresholds will be examined more rigorously in Chapters 5: Analysis of network properties (p.65) and 6: Clustering and its evaluation (p.85).

Neighborhood Correlation was designed with the goal of improving performance on complex, multidomain families. Figure 3.6 shows the score distributions of four families for which homologous and non-homologous pairs of sequences could not be partitioned by sequence similarity (Figure 3.3). This figure depicts (again, in red and blue) a marked improvement in the separation of these dis-

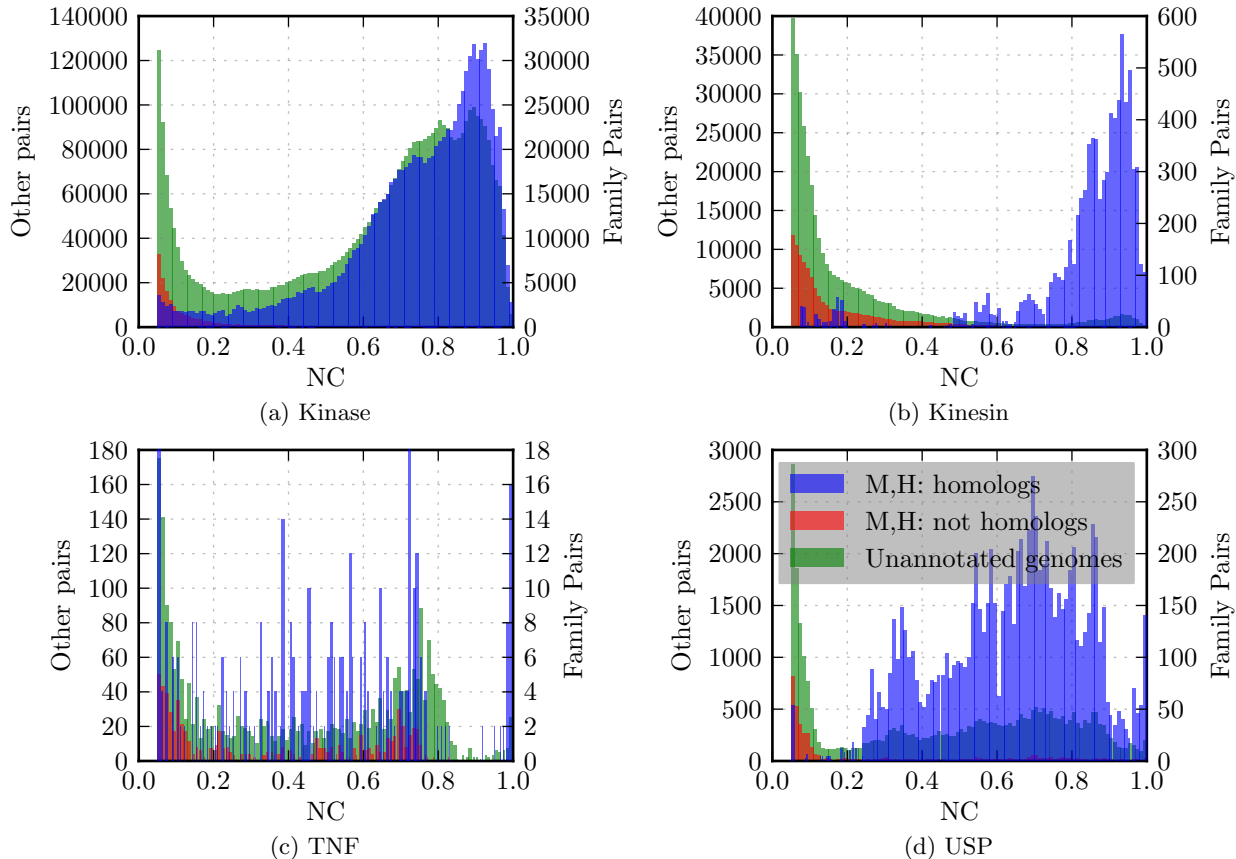
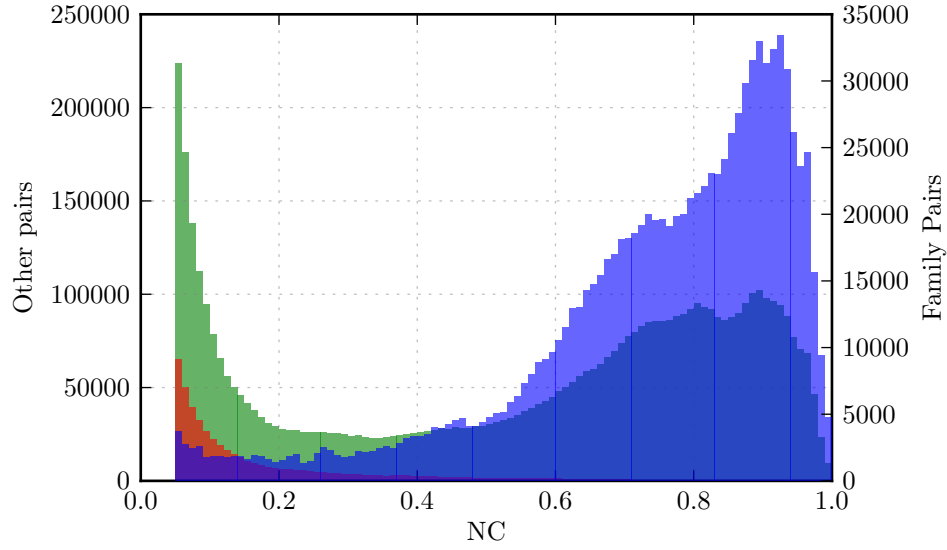


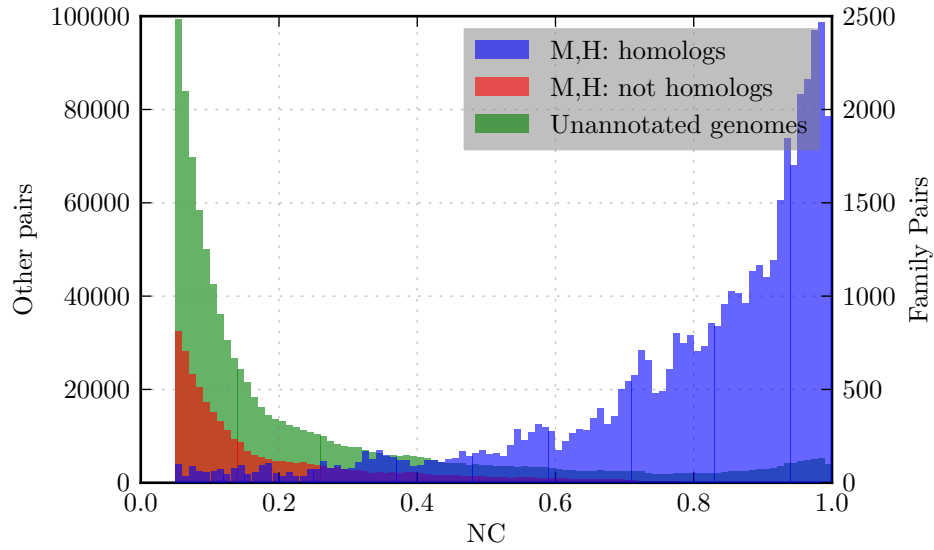
Figure 3.6: Histograms of the Neighborhood Correlation scores for curated families in mouse and human. Homologous pairs are in blue (wrt. right axis), and non-homologous pairs are in red (wrt. left axis). Additionally, in green (wrt. left axis) are scores of all pairs that include one member of a curated family, and one member from the 12-genome dataset, and not in mouse or human.

tributions. A single threshold of approximately $NC > 0.3 - 0.4$ will separate most homologous and non-homologous pairs in all of these families. The TNF family is improved over bit-score, but remains a challenging case. Score distributions for all remaining families may be found in Figures C.3 and C.4.

The above discussion considered the distributions of scores that result when Neighborhood Correlation is applied to the mouse and human genomes. The 20 families in the benchmark are curated in mouse and human, alone. That is, barring errors in curation, all members of each family are labeled when only these genomes are considered. While not so labeled, members of these families do exist in other lineages, and it is valuable to consider the distribution of scores involving these unlabeled family members. For a family F , the set of homologous pairs is comprised of $(x, y) \forall x, y \in F$. For this family, the set of all non-homologous pairs is comprised of $(x, n) \forall x \in F, n \in N$, where N is the set of all mouse and human sequences that do not belong to the family. If the dataset is expanded to include other genomes, this introduces another set of sequences, U , where family membership is unknown. Here, the set U includes all sequences in one of the genomes in the 12-genome Panther dataset (see Table B.1), excluding mouse and human. A new set of pairs may be enumer-



(a) ALL



(b) ALL-kinase

Figure 3.7: Histograms of the Neighborhood Correlation scores for the aggregate of all curated families in mouse and human. Homologous pairs are in blue (wrt. right axis), and non-homologous pairs are in red (wrt. left axis). Additionally, in green (wrt. left axis) are scores of all pairs that include one member of a curated family, and one member from the 12-genome dataset, and not in mouse or human. ALL is the set of all 20 families, while ALL-kinase is the set of 19 families, with the largest family, Kinase, excepted.

ated where one sequence is annotated, but the membership of the other sequence is unknown, or $(x, u) \forall x \in F, u \in U$.

The distribution of scores (x, u) is depicted in green in Figures 3.5 and 3.6. This distribution may be used to evaluate the degree to which the scores within the mouse and human genomes are representative of these families in other genomes. These distributions coincide to a very high degree with the distributions observed for family and non-family pairs in mouse and human. This suggests that the same scoring threshold suitable for accuracy in mouse and human would effectively distinguish families in the complete dataset.

Figure 3.7 shows score distributions for the sets (x, y) , (x, n) , and (x, u) for the combined set of all 20 families (a), as well as the set of 19 families, when Kinase is excluded (b). This figure demonstrates that the separability of family and non-family pairs observed with individual families also persists when all families are considered at the same time. Additionally, this figure demonstrates that the distributions of scores for families in the curated benchmark remains highly similar across the 12-genomes data set, as compared to that of mouse and human. The comparable distribution of sequence similarity was shown in Figure 3.1, which demonstrated that no bit-score threshold performed well over all families.

The discussion here has centered upon concrete, visual illustration of the score distributions that result in typical families, without statistical evaluation. In early collaborative work [134], these score distributions were evaluated in the context of a binary classification problem, where a particular pair of sequences is predicted to be either homologous or not. Under this framework, the area under a ROC curve (AUC) was used to determine the ability of Neighborhood Correlation to partition homologous and non-homologous pairs. In this classification problem, the number of false (non-homologous) pairs greatly outnumbers the count of true (homologous) pairs, because families are much smaller than the size of the genome, and only members of the same family are homologous. As a result, a variant of AUC that considers a fixed number of false positives was utilized. AUC was used to evaluate the performance on individual families, as well as the aggregate set of all families (ALL). Evaluated with AUC, Neighborhood Correlation performed well, and greatly exceeded the pairwise classification accuracy of all other methods considered.

This dissertation expands the goals of evaluation from a pairwise classification of homologs, to an evaluation of performance in identifying complete families as single units in the output network. The identification of discrete families is the ultimate goal of family classification, so evaluation in this context is most relevant. Further, evaluation on the basis of sets of sequences ameliorates the issues that arise from a preponderance of false results in the data. It also allows for evaluation on datasets that contain sequences not in the curated dataset. The means and result of evaluating clusters using curated families is rigorously considered in Chapter 6: Clustering and its evaluation (p.85).

3.3 BLAST sequence similarity

Neighborhood Correlation was developed with the premise that an all-against-all sequence comparison is used to generate the input sequence similarity network. I, as many others, use the NCBI BLAST [4] tool for this computation. An all-against-all computation is accomplished by generating a custom sequence database, followed by local execution of a query from every sequence against the

Parameter	blastall argument	Value	Description
Expectation	-e	$10N$	E-value significance cutoff
Search length	-Y	Y^2	Effective length of the search space
Matrix	-M	BLOSUM62	Scoring matrix
Descriptions	-v	0	Max. number of one-line descriptions returned
Alignments	-b	N	Max. number of alignments returned
Gap open	-G	11	Gap open penalty
Gap extend	-E	1	Gap extension penalty
Processors	-a	2-8	Number of CPU cores to use during search
Old engine	-V	F	Do not force use of the legacy BLAST engine

Table 3.1: BLAST parameters used for all-against-all sequence similarity calculation. N is the number of sequences in the database, and Y is the number of residues. Parameters and values correspond to `blastall`, version 2.2.16. All parameters not specified are left at default values in this version of `blast`. Some of these parameters are the default, and are included for clarity.

database. NCBI BLAST version 2.2.16 has been used for all sequence similarity computations in this dissertation. Throughout, a BLAST *run* refers to the calculation of a defined set of sequences and the particular set of parameters used for BLAST. Table 3.1 lists the full set of parameters specified for a BLAST run.

For Neighborhood Correlation, the sequence similarity score from each sequence to all N sequences are implicitly considered. Pairs for which no sequence similarity score was computed by BLAST are assigned a minimal pseudo-score of S_{\min} . Throughout the development of Neighborhood Correlation, BLAST parameters were chosen to maximize the number of similarity scores computed and returned, before resorting to use of a pseudo-score. This maximizes the amount of information encoded in the local network structure around each pair of sequences.

The Expectation and Search length parameters have been selected to embody the view that an all-against-all BLAST search is a single experiment. For a BLAST query of a single sequence, the BLAST default of $E = 10$ is often used, which specifies that all hits with an E-value of less than 10 will be returned (up to the maximum number of alignments, typically 250). The default database Search length is Y for a database comprised of Y residues. The parameters selected here are roughly equivalent to conducting N single-query BLAST searches with $E = 10$ and Y set to the number of residues in the database. Treating the all-against-all BLAST comparison as a single experiment results in symmetric E-values in the absence of low complexity filtering. We define $\theta(x, y) = \frac{E(x, y)}{10N}$ to be the expected number of chance hits per sequence in the dataset with a score equivalent to, or better than, that of the alignment of query sequence x with matching sequence y . The significance threshold of $E = 10N$ corresponds to $\theta = 10$ chance hits per sequence, in expectation.

The scores resulting from BLAST are not necessarily symmetric; i.e., the score $E(x, y)$ for a query sequence x and hit y is not necessarily equal to $E(y, x)$. Further, it is not necessarily true that both results are returned by the corresponding queries. This can result because BLAST filters low-complexity regions in a query sequence before performing a search, but equivalent filtering is not applied to database sequences. Before Neighborhood Correlation is computed, the greater of the scores between x and y is assigned to both $E(x, y)$, and $E(y, x)$ to obtain a symmetric sequence similarity score matrix.

The following chapter, 4: Optimization for large scale (p.43), discusses the data structures, algorithms, and implementation that I developed to calculate Neighborhood Correlation on large datasets. In practice, the success of these optimizations shift the computational burden to calculation of the input sequence similarity network. Calculation of sequence similarity with BLAST is an inherently computationally intensive task.

Many users of sequence data already compute all-against-all sequence similarity using BLAST. However, the parameters used there are typically very different from those we employ in conjunction with Neighborhood Correlation. Two parameters have a particularly large effect on the sequence similarity network that results. First, others typically specify a more stringent E-value threshold to reduce the number of insignificant scores that are returned. Second, the total number of hits per query is typically limited, generally to a default value of at most 250 hits per query. For interactive use, or automated uses where sequence similarity scores are directly employed as a final scoring metric, both parameters makes sense: very long lists of hits, or hits with very little sequence similarity are unlikely to be useful.

In contrast, calculation of Neighborhood Correlation between two sequences depends upon both the number and strength of edges incident upon these two sequences in the sequence similarity network. The BLAST parameters used have the result that effectively all initial hits from a query sequence against the sequence database are extended, and returned by BLAST. This exacts a large performance penalty upon the BLAST computation. The observed run-time of the NCBI BLAST implementation is highly dependent upon the number of hits returned. The number of hits returned has a greater impact on performance than the database size or the number of queries. The aggregate run-time is significant. For example, computation of all-against-all BLAST of the 48-genome Panther dataset of 600k sequences requires approximately 160-days on a single (*circa* 2010) CPU core. Fortunately, the calculation of individual sequence queries are entirely independent, therefore they may be computed in parallel in a distributed fashion with separate BLAST processes.

The task here is to consider whether use of parameters more akin to the default BLAST parameters used by others would yield a substantial difference in the Neighborhood Correlation output. This section does not aim to detail the specific changes that different BLAST parameters have upon the classification performance of Neighborhood Correlation. Instead, the purpose is to evaluate whether existing BLAST results computed for other purposes may be used as a direct substitute for a complete BLAST computation, without changing the result of Neighborhood Correlation.

Two alternate BLAST scenarios are considered. First, the maximum number of hits per query sequence is adjusted. This is equivalent to setting the Alignments BLAST parameter to a value smaller than the total number of sequences. Figure 3.8 shows a heatmap of the Neighborhood Correlation score that results for each pair of sequences, as calculated using the complete BLAST all-against-all result, compared to using only the first 500 hits from each BLAST query. (Pairs that

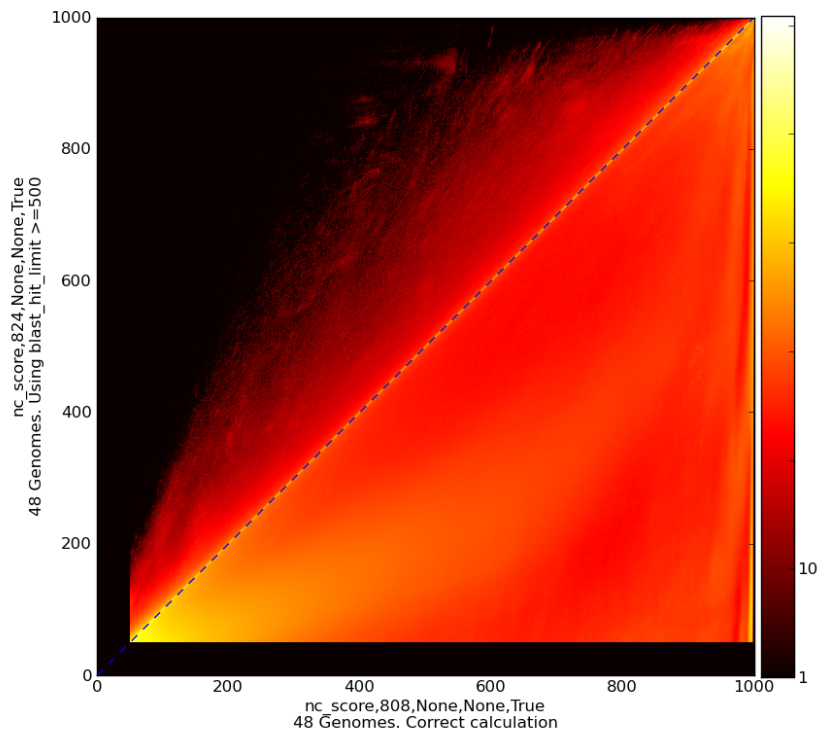


Figure 3.8: Heatmap of Neighborhood Correlation scores calculated using the complete network of sequence similarity (horizontal axis) and the first 500 hits returned by BLAST for each query sequence (vertical axis).

have a score $NC < 0.05$ by either metric are not included in this plot, because we tend to not store these due to space constraints.) Color in this plot represents the number of pairs with that particular score by each method. Were the resulting Neighborhood Correlation score of all pairs identical in each calculation, all points would lie upon the diagonal, illustrated in blue. This figure demonstrates that inclusion of neighborhoods of size greater than 500 when calculating Neighborhood Correlation does make a substantial contribution to the result. Most notably, because the distribution of scores lies in the lower half of the plot, calculation of Neighborhood Correlation from a truncated BLAST result tends to decrease the score that results between pairs in the network. Additionally, this truncation results in a substantial number of scores near one being assigned scores near zero, and effectively mixed with all other low-scoring pairs. That is, the lower-right position in this plot has a very high count of pairs.

A second evaluation of the effect of BLAST parameters concerns the imposition of a sequence similarity score threshold. This is equivalent to specifying a value for the E parameter that is less than $10N$. Figure 3.9 shows a heatmap of the scores between pairs when calculated using a sequence similarity network that consists only of pairs where the bit-score exceeds 60, which corresponds to an E-value of approximately 9×10^{-8} . This is a relatively permissive threshold for a dataset that consists of 600k sequences. As before, the calculation of Neighborhood Correlation using the complete sequence similarity network is shown on the lower axis. Again, the use of a truncated set of BLAST sequence similarity scores decreases the Neighborhood Correlation score

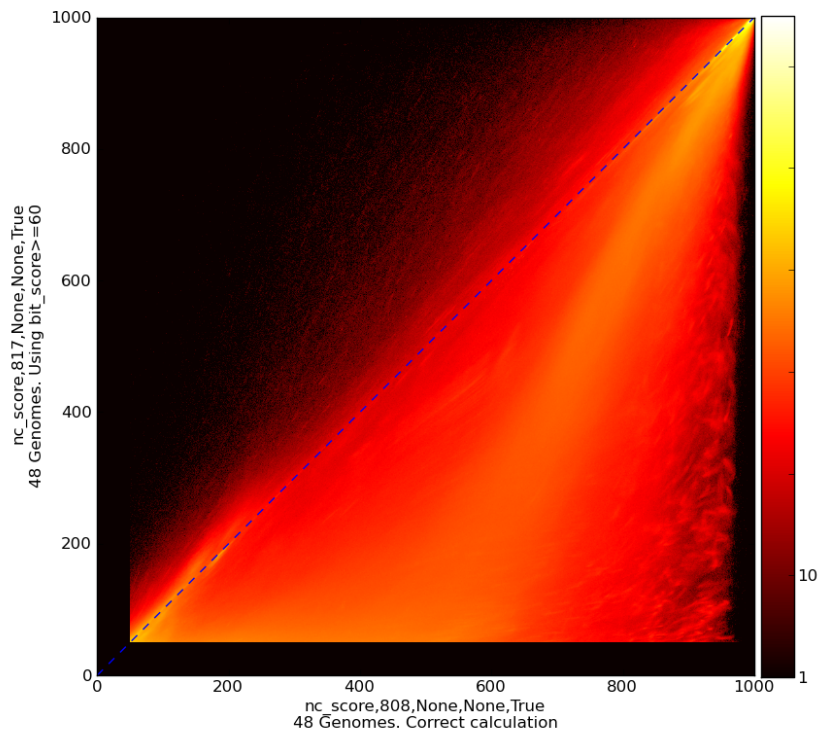


Figure 3.9: Heatmap of Neighborhood Correlation scores calculated using the complete network of sequence similarity (horizontal axis) 500-hits returned by BLAST for each query sequence (vertical axis).

of nearly all pairs of sequences. At higher scores of the complete calculation ($NC > 0.6$), the dominant effect is to reduce the score but not yield a complete reordering of the returned set. At lower thresholds, however, calculation using the truncated set of sequence similarity scores compresses a large number of pairs with scores in the range $NC = 0.0 - 0.6$ to a score of 0.1 or less. The net result is that pairs in this scoring range may no longer be distinguished.

Typical approaches to sequence similarity calculation limit the number of BLAST results using both of the above means. Most users apply a threshold that specifies the maximum number of returned hits for each query sequence, and consider only sequence similarity scores above some significance threshold. Clearly, these change the result of Neighborhood Correlation as compared to a full result from BLAST. Yet, these comparisons do not evaluate whether the classification result is likely to be substantially altered. Two properties suggest the observed changes in the scores of pairs will translate to lower performance for family classification. Both parameters are likely to introduce family-specific effects. Truncation of the size of a sequence neighborhood is more likely to change the result in a larger family, because sequences in large families may be expected to have higher degrees in the network than those in small families. Truncation of the set of neighbors of each sequence will arbitrarily reduce the number of homologous and non-homologous pairs represented by the network. Both are exploited by Neighborhood Correlation, and correct classification of such families is likely to benefit from the use of as much local structure as possible. Further, truncation of the BLAST result by either means is likely to reintroduce the limitations of sequence similarity.

In effect, truncation filters pairs in the input network with respect to the value of their sequence similarity, which we already understand to be a poor predictor of homology. As a result, I do not recommend the use of more restrictive BLAST parameters without careful evaluation of suitability on the specific dataset at hand.

Optimization for large scale

The unifying task of this work is to investigate genome evolution at a gradient of scales. These range from duplication and modification of single sequences, especially through domain shuffling, to consideration of the features of gene families, to the processes at work within a single genome, through comparison at each of these scales among disparate lineages of organisms. The scope of the questions posed necessitates a commensurate quantity of data to exploit sufficient signal over noise to support conclusions about broad relationships. As a minimal set, all sequences within a single genome may be considered; typically, an analysis is performed with the sequences of dozens (or more) of genomes. For example, the primary dataset used throughout this work is a set carefully chosen to be small, yet representative of a broad range of lineages [104]. Still, it is comprised of approximately 600k sequences from 48 genomes. The approach and methods here have been developed with an eye toward substantially larger datasets.

The approach throughout this work is data-driven, using protein sequences as input. The use of larger data sets (1) increases the phylogenetic signal that is conserved, (2) allows identification of small or disparate signals, as between lineages, and (3) lends greater confidence to the results obtained. Overall, more data can only help.

The scale of this data is not amenable to *ad hoc* approaches to computation, as when individual, “quick and dirty” programs are each used to investigate a different question from myriad data sources. This is particularly relevant in that the goal of my research has not been solely to develop stand-alone methods, but also to employ them to increase our understanding of genome evolution. Organization and planning of infrastructure become inevitable — and especially valuable — when one cannot simply re-compute an entire pipeline throughout development.

The number of sequences considered can be large, though the means of analyses present substantially greater challenges. Most of the values considered are inherently comparison-based, and reduce to an evaluation of all pairs of sequences. Homology, in particular, is a definition on pairs: two sequences are either homologous or not. In the absence of optimization, N^2 pairs of sequences must be labeled in a dataset of N sequences — a count that quickly becomes untenable. The task of homology classification is to define the relationship among all pairs of sequences under study (e.g., x, y are homologous, or not, $\forall x \in S, y \in S$, where S is the set of all sequences.) Generally, this involves implicit calculation over all pairs, at least. Consideration of gene families reduces this

complexity to sets of sequences, although these are typically found only after consideration of all pairwise relationships. Worse yet, the previous chapters have demonstrated that the direct pairwise comparison of sequences is insufficient to capture their evolutionary relationships [134]. For example, the Neighborhood Correlation calculation between two sequences uses the relationship between those sequences and all others.

Especially during exploration of methodology, it is necessary to be able to compute quickly, and without large fixed resources. As importantly, the emphasis throughout this work is to perform exhaustive, deterministic computation. The goal has not been to devise heuristics or approximations, at least as a first step. This decision was made to avoid introducing new independent variables, and to strengthen confidence in the conclusions reached. Such an approach is of particular value during exploratory research, as here. It is not necessarily evident which features of the data are most relevant, and important to support biological conclusions. For understanding during development, and for discovery of potentially weak signal, such abstraction can add a new class of uncertainty to the results, and cloud conclusions. Once the behavior of a method is well characterized on distinct datasets, it may become appropriate to consider approximations to a complete calculation.

One option toward reducing the computational and infrastructure demands might be to re-frame the problem to avoid such challenges of large data. This can become necessary. Here, this is not the case. This chapter demonstrates that it is possible to efficiently compute with the data of interest here, using practical, commodity, resources. It is not necessary to resort to sampling or heuristics. However, an integrated approach that considers all computational steps toward a usable, biologically relevant result is required. This consists of a balance of infrastructure development, design of efficient algorithms, and practical implementation. These goals are not independent, and their solutions do not stand alone.

Computation with some methods imposes fewer computational demands as more data is added, by further constraining the problem space. Others become more computationally challenging. This can depend heavily upon the real distribution of data. We can tune an algorithm and implementation for the empirical nature of the data (e.g., density).

The techniques and optimizations developed here comprise a general framework for analysis of hundreds of thousands of sequences, and exploration of source, intermediate, and derived data. These address constraints of storage, memory efficiency, and available computational resources. The approach undertaken is illustrated through examples of the infrastructure developed to organize and store sequences, networks of sequences, and families. This consists of a strong emphasis on implementation at a variety of levels, from on-disk storage, to in-memory data structures, to efficient algorithms, and to implementation of these in a manner optimized for computation on real machines, such as by ensuring good cache-locality.

Algorithmic efficiency is not the end-all solution to an efficient implementation. The means of actual implementation can be the key to practical use. The implementation can affect absolute run-time in meaningful ways, often by orders of magnitude. Further, the infrastructure used may influence the choice of a particular algorithm, and greatly alter the degree of computation required. Note that the data structures and organization of the infrastructure must not be inherently computationally difficult to initialize.

This chapter provides a high-level overview of the challenges posed by utilizing large sequence

datasets and presents the solutions developed to facilitate efficient, large-scale family classification. Many of the solutions developed in this dissertation generalize to areas of research beyond family classification. I present these with the understanding that their value is great when needed, but that their utility is best understood by means of example. I do not expect that the entire infrastructure will be directly employed elsewhere. However, the design is such that portions may be readily separated.

The architecture developed here facilitates management of input, intermediate, and output data. A variety of data types are used. This chapter details two fundamental data structures that underlie most of the methods used through this dissertation: networks and trees. It is vital that these structures may efficiently be stored on disk, be accessed in an efficient manner, consume practical amounts of program memory, and be amenable to the algorithms employed. Of course, the algorithms and data structures are inseparable; this chapter extends this dependency to permanent storage and data management.

The system architecture is such that each method (e.g., obtaining source data, BLAST, Neighborhood Correlation, clustering, ...) is a stand-alone module. A detailed discussion of the implementation of Neighborhood Correlation is used as a working example of one module. This exemplifies a need for efficiency and the means to achieve it within the infrastructure developed.

All source code is available. See Appendix A.1.1.

4.1 Architecture

I have developed a robust, flexible infrastructure for exploratory research of homologous families using protein sequences. This mode of incremental, iterative development has been accommodated by constructing what is classically termed as a “blackboard” architecture. This is a data-centric approach, using a central data store shared by many discrete tools, each developed for some task.

In this work, a relational database is used as a common data store. All source, intermediate, and derived data are interfaced to this database. This allows multiple tools to operate on the same data, and facilitates construction of a pipeline of tools, each working with the result of one or more other tools. This yields benefits for practical use: one need not re-run an entire pipeline of tools to change, for example, a parameter of a single method. The benefits for practical development are greater: one can develop or experiment with one method independent of others, while preserving a well-defined data interface. Such experimentation then carries little risk of inadvertently affecting the data stored in the database.

A further design choice not directly specified by a blackboard architecture is that of designing the data store such that new data may be appended, but existing data may neither be altered nor removed. All data is assigned a version (effectively, the date), and relationships map to all parameters that describe the data (e.g., program arguments of a method). The goals include tracking of the data source of each sequence, retention of all versions ever imported, and deduplication to facilitate updates with only incremental storage and access cost.

A major emphasis is the requirement to cross-reference all stored data, to reduce the possibility of storing inconsistent data, and to aid in verification of what is stored. The design is such that results may be interpreted separately from the implementation used to generate them. As with

```

1 CREATE TABLE prot_seq_str (
2     seq_str_id          serial,
3     sequence            text NOT NULL,
4     crc                 text NOT NULL,
5     length              integer NOT NULL,
6     molecular_weight    integer NOT NULL, -- (0 if not known)
7     PRIMARY KEY (seq_str_id),
8     UNIQUE (crc, length, molecular_weight)
9 );
10 CREATE TABLE prot_seq (
11     seq_id              serial,
12     seq_str_id          integer NOT NULL REFERENCES prot_seq_str,
13     PRIMARY KEY (seq_id)
14 );

```

Code 4.1: SQL table definitions for the storage of amino acid strings. `prot_seq_str`, short for “protein sequence string”, defines storage of an amino acid sequence. `crc`, a checksum, `length`, and `molecular weight` are stored to optimize look-up using an index when a sequence string is known, typically to check for an existing, duplicate sequence string. `prot_seq` establishes an internal sequence identifier (`seq_id`), and a pointer to a single protein sequence string. Many sequence identifiers may map to one sequence string, but every string stored in the table is unique.

source data, each “run” of some method is stored anew, without overwriting previous data.

A wide range of input, intermediate, and derived data must be handled. For example, the source data consists of sequences and meta-data about those sequences, such as data source, organism, name, description, and any functional annotations. Additional sequence data may be calculated; a particular example is the use of domain models to identify instances of domains in the source sequences. The eventual output data are predictions of gene families, each a set of sequences. The primary data type and conceptual framework used throughout this work is a weighted network.

More concretely, this architecture is implemented as a central SQL database, with all data unified through defined, and enforced relationships. That is, consistency is preserved to the greatest extent possible by the database definition, to be robust to bugs in or failures of the individual programs that interact with the database. Each method throughout this work takes some input data, such as strings representing sequences, and computes a result. The resulting data type tends to be specific to the method at hand; a separate database table is defined to accommodate any intermediate and resulting data from computation of a method. For example, separate tables are used for the networks that result from BLAST comparisons, networks from rewiring with Neighborhood Correlation, and the hierarchical trees that result from clustering.

The database is designed such that there are three logical classes of tables. The description given here of how these tables are designed and related illustrates the general strategy. A full database schema is listed in the Appendix. First, a set of definitions facilitate storage of protein sequences from some input dataset. These table definitions store the amino acid strings, associate an internal database sequence identifier, and relate these sequences to a source database are listed in Code 4.1.

```

1 CREATE TABLE prot_seq_source (
2     source_id      serial,
3     source_name    text UNIQUE NOT NULL,
4     PRIMARY KEY (source_id)
5 );
6 CREATE TABLE prot_seq_source_ver (
7     source_ver_id  serial,
8     source_id      integer NOT NULL REFERENCES prot_seq_source,
9     version        text NOT NULL,
10    date           timestamp NOT NULL,
11    PRIMARY KEY (source_ver_id),
12    UNIQUE (source_id, version, date)
13 );
14 CREATE TABLE prot_seq_version (
15    seq_id          integer NOT NULL REFERENCES prot_seq,
16    source_ver_id   integer NOT NULL REFERENCES prot_seq_source_ver,
17    primary_acc     text NOT NULL,
18    PRIMARY KEY (seq_id, source_ver_id),
19    UNIQUE (source_ver_id, primary_acc)
20 );

```

Code 4.2: SQL table definitions used to represent the data source and version from which a sequence was obtained. As is most appropriate in relational databases, separate tables are used to specifically define many-to-one relationships, such as many versions (`prot_seq_source_ver`) of the same data source (`prot_seq_source`).

The structure of these tables is such that the string that represents an amino acid is stored only once in the database and is referenced when an internal database sequence identifier `seq_id` is defined.

The sequence identifier (`seq_id`) defined in the `prot_seq` table is the basic unit of a specific biological sequence stored in the database. That is, a `seq_id` refers to a single amino acid sequence in one organism, analogous to a Uniprot identifier. A number of additional tables represent the metadata associated with that sequence identifier, such as the source, and version of the database the sequence was obtained from, or the name and description of the sequence. To illustrate, Code 4.2 demonstrates the tables and relationships used to represent the source database and version. Note that an internal sequence identifier (`seq_id`) may be assigned to multiple source database versions. This facilitates storage of multiple versions of some source database with only the incremental storage cost of new `seq_id` entities for entities that have changed in the source database. Other tables of similar construction are used to represent all other aspects of a sequence, such as the corresponding organism, predicted function, or human-readable descriptions.

A second logical class of tables in the database define sets of sequence identifiers. All of the methods described in this dissertation operate on sets of sequences; e.g., all human protein sequences, or all sequences in a collection of 48-genomes. These sets of sequences are represented by the table

```

1 CREATE TABLE prot_seq_set (
2     set_id          serial,
3     name            text NOT NULL,
4     description     text NOT NULL,
5     PRIMARY KEY (set_id),
6     UNIQUE (name, description)
7 );
8 CREATE TABLE prot_seq_set_member (
9     set_id          integer NOT NULL REFERENCES prot_seq_set,
10    seq_id          integer NOT NULL REFERENCES prot_seq,
11    PRIMARY KEY (set_id, seq_id)
12 );

```

Code 4.3: SQL table definitions for sets of sequences. `prot_seq` defines a unique identifier for the set (`set_id`) and a human-readable name and description. `prot_seq_set_member` then associates sequence identifiers with that set instance.

definitions listed in Code 4.3.

A third class of tables represent the data for specific methods employed in the family classification pipeline developed in this dissertation. Discrete tables are used to represent the output data of, for example, BLAST, Neighborhood Correlation, domain identification within amino acid sequences, and clustering. Where a method is dependent upon output from another method, this relationship is explicitly defined within the database. To illustrate this organization, Code 4.4 lists the table definitions of individual runs of BLAST (`blast_run`), and associated parameters. For BLAST, a *run* describes the computation of an all-against-all sequence comparison of a particular set of sequences, using a particular set of parameters. The figure similarly illustrates storage of instances of runs of Neighborhood Correlation (`nc_run`). Neighborhood Correlation is calculated with respect to a single BLAST sequence similarity network, so this relationship is explicitly defined via `br_id`.

The full database schema is listed in the Appendix A.1.2.

```

1 CREATE TABLE blast_run (
2     br_id          serial,
3     set_id         integer NOT NULL REFERENCES prot_seq_set,
4     date           timestamp NOT NULL,
5     num_sequences  integer NOT NULL,
6     num_residues   integer NOT NULL,
7     params         text NOT NULL,
8     comment        text,
9     blastall_path  text,
10    query_set_id    integer NOT NULL REFERENCES prot_seq_set,
11    PRIMARY KEY (br_id)
12 );
13 CREATE TABLE nc_run (
14     nc_id          serial,
15     br_id          integer NOT NULL REFERENCES blast_run,
16     date           timestamp NOT NULL,
17     e_thresh       double precision NOT NULL,
18     bit_thresh     double precision,
19     nc_thresh      double precision NOT NULL,
20     blast_hit_limit integer,
21     smin           double precision NOT NULL,
22     smin_factor    double precision NOT NULL,
23     use_symmetric  boolean NOT NULL,
24     score_type     text NOT NULL,
25     self_hits      smallint NOT NULL,
26     PRIMARY KEY (nc_id),
27     UNIQUE (br_id, date, e_thresh, bit_thresh, nc_thresh, blast_hit_limit,
28             smin, smin_factor, use_symmetric, score_type)
29 );

```

Code 4.4: SQL table definitions to represent complete “runs” of BLAST, and Neighborhood Correlation. The latter dependency upon BLAST is encoded by the table definitions. Each of these tables detail the complete set of parameters used during execution of the method.

4.2 Key data structures

The family classification approach developed in this dissertation is data-intensive. Two data structures recur throughout, and are key to the success of these methods. These two are presented because of their central importance to this dissertation, and as working examples in the algorithms described in the following section.

- 1 The first several steps of the family classification pipeline developed in this dissertation involve weighted networks of sequences. In these, a node represents a sequence, and an edge represents a derived measure of comparison, such as the sequence similarity, or the Neighborhood Correlation

network. The goal is to establish a better approximation of the true, but unknown, homology network. These networks contain large numbers of nodes and implicitly represent comparisons between every pair of nodes. The goal is to develop a single data structure that need not be translated in any significant manner between in-memory representation and native representation in an SQL database. It must be efficient for the methods of interest here.

2 Hierarchical clustering is applied to refine these sequence networks, and to provide a means of partitioning into proposed families. Hierarchical data structures are inherently difficult to store in a traditional relational database, particularly if they are to be accessed by SQL. The description in this chapter focuses upon the SQL representation of hierarchical clusters of sequences, with a particular focus on their means of construction and rapid access.

Implementation of both of these data structures requires consideration of many factors. Efficiency of computation for the algorithms and methods designed in this dissertation define their utility. Efficiency of use of real machines is equally important, and is achieved by different means, such as through ordering the computation of an algorithm to facilitate effective caching of memory pages. Due to the size of the data, compactness (both in memory, and on disk) is vital to the use of commodity hardware. Since an SQL relational database is used as a central data store, all data structures must be suitable for representation, and granular access, via SQL, in a native format to the greatest extent possible. This is distinct from, for example, storage of a data structure as a large binary object in the database, which might represent the in-memory layout of a data structure, but could not be interrogated by SQL. Finally, the goal has been to maintain a close mapping between the data structures stored in the database and their in-memory representations. This reduces the amount of translation required during loading or storage.

In this chapter, I discuss implementations for storage of graphs and of hierarchical clusters. Notably, most SQL engines¹ do not have native representations of graphs or hierarchical data structures, and these can be especially expensive to represent in SQL. Of course, these *can* be represented in SQL, but typically with great loss of efficiency in either storage or look-up. Achieving efficiency and compactness is the focus here. Description of those data structures that may be directly implemented in SQL is reserved for Appendix A.1.2.

The design goals here are to tailor storage and access optimizations to the genomic data and methods employed here — not to design general tools applicable to all manner of tasks. However, wherever possible, attention is paid to flexibility for unplanned research directions.

4.2.1 Networks of sequences

The concept of a homology network provides a theoretical context for consideration of gene families. Additionally, the use of a network is central to the approaches to family classification developed in this dissertation. The sequence similarity or Neighborhood Correlation calculations between all pairs of sequences inherently produce weighted networks. Sequences are represented as nodes, and the measure of interest is represented by a weighted edge between two sequences.

The fraction of edges in a network tends to define the mode of storage of that network, as well as the approach taken for computation when the network is constructed or used. In short, networks

¹Here, all implementation was performed in the PostgreSQL[65] database.

tend to be described as either sparse (having very few edges relative to the number of possible node pairs) or dense (having a large fraction of all edges).

The homology network is sparse by definition. Edges exist only within families, and these families are small relative to the size of the network, yielding $|E| \ll |V|^2$. The sequence similarity (G_S) and Neighborhood Correlation networks (G_{NC}) are also sparse, in practice. Implicitly, these networks represent all edges in the network; however, and importantly, most of those edges actually would have weight zero. Accordingly, they may be omitted. Typical edge densities are 0.1% for sequence similarity, and 0.4% (e.g., $1,675M/600,000^2$) for Neighborhood Correlation.

There is a small set of classical data structures used to represent weighted networks [38]. These exhibit distinct properties of mode of construction, storage space required, and efficiency for certain access and modification patterns. To better understand the decisions made and implementation developed here, it is useful to briefly review these typical data structures, and consider how they are optimized for varying modes of use.

For a network of N nodes, the most direct means of storing is a matrix, of size $N \times N$. An undirected network may be stored in $N \times N/2$ space. When using a matrix representation, the size of the data structure is independent of the number of edges represented, and sparse networks result in a great deal of wasted space. The data type of the matrix may be a binary value, to represent an unweighted network, or a scalar, such as a float to represent edge weight. A matrix is very amenable to look-up or modification of an edge; the edge between nodes i and j may be found in $O(1)$ time, at the index $[i, j]$. As a consequence, this data structure may be constructed incrementally, provided the total number of nodes is known in advance. All neighbors of node i may be identified by iterating over the row $[i, \cdot]$. However, this data structure is very inefficient for iteration over the neighbors of a node if the network is sparse. Further, when storage of an undirected network is optimized by storing half of the matrix, iteration of the neighbors of a node also requires iteration over columns, with indices $[\cdot, j]$. For a row-wise memory representation of the matrix, this is a worst-case access pattern in terms of cache locality; i.e., it is slow.

Use of a linked-list to represent the neighbors and edge weights of a node yields particular benefit to space-efficiency of sparse networks. Here, a list of nodes is maintained, each with a pointer to a linked-list of neighbors. For a network of N nodes and M edges, M space is required for the edge data (e.g., a float), plus a pointer for each of those M edges to the next neighbor in the linked-list. For sparse networks, the total space required, $M \times [\text{sizeof}(\text{pointer}) + \text{sizeof}(\text{edge})]$ is much less than $N \times N \times [\text{sizeof}(\text{edge})]$. This structure may be constructed incrementally by appending or inserting neighbors into the neighbor linked-list associated with a node, provided one does not need to check for duplicate edges. This structure lends itself to fast look-up of all neighbors of a node; however, look-up of a particular edge requires iteration over the linked-list. Iterating over a linked-list is considerably slower than over an array, primarily because the memory accesses are not contiguous, resulting in poor cache locality. The linked-list may be sorted to help this procedure, but the efficiency difference remains substantial.

A linked list may be replaced by an associative array (e.g., a hash structure), with a different set of access properties. The means of construction are similar to when a linked-list is associated with each node. The space required does depend upon the space efficiency of the associative array implementation, but the total size may approach the size of a linked-list. Additional, and potentially substantial overhead is incurred as the associative arrays are grown dynamically. The

greatest benefit of this structure is that it restores fast look-up of edges as compared to a linked-list, but without the space requirement of a full $N \times N$ matrix. Look-up of a node in an associative array is a constant-time operation, and tends to be fast, and constant on real machines. The result is that duplicate edges may be resolved efficiently during construction. However, iteration over an associative array tends to be slow, again, depending upon the implementation.

The overhead for storing the neighbors, whether due to the pointer required for a linked-list, or that of an associative array, can be ameliorated through use of an array to represent the neighbors of a node. This results in $M \text{sizeof}(\text{edge})$ memory space for a network of M edges. In this case, construction may be performed incrementally, in cases where it is not necessary to check for duplicate edges. Iteration over all neighbors of a node is extremely fast because iterating over a flat array is ideal in terms of cache locality. However, look-up of a single edge is slower, since the array must be traversed, as with a linked-list. Yet, as with iterating over all neighbors, this look-up can be extremely fast on real machines, because the list will remain in the cache and not necessitate memory accesses. This is particularly true if the list is maintained in sorted-order, where binary search may be used, for $O(\log N)$ access time. Because of these caching effects, binary search of a list is extremely fast on real machines, and can approach or exceed the speed of associative array look-ups for lists of even very long length.

There exist other means of compressed network storage, or storage schemes optimized for matrix operations. These are beyond the scope of structures required or described here. A sampling are implemented in [77].

Guided by these typical means of sparse network storage, a hybrid data structure is used here. The general data structure is an adjacency list representation, where every node is associated with two lists: one to represent the neighbors of that node, and another to represent the weight of edges to those nodes. With the data here, it is necessary to resolve duplicate edges during construction; this is a slow operation for a list-of-lists network representation. This construction overhead may be deferred as a post-processing step, and incurred once for each node, rather than for each edge insertion. Similarly, the overhead for maintaining the order of the adjacency lists may be deferred until after all edges are inserted. This data structure is referred to as a `DICTARRAY`, and is detailed in Code 4.5.

A `DICTARRAY` defines two invariants of the data. First, the adjacency list for each node contains each neighbor at most once. Second, these neighbors are represented in ascending numerical order. (Nodes are represented by unsigned integers.) These properties yield a very space-efficient data structure, and facilitate fast look-up of all neighbors of a node. However, a key benefit is that these invariants may be established after, rather than throughout, population of the data structure. This deferral results in substantial computational benefits. In particular, insertion of an edge into a `DICTARRAY` involves finding the index for the node in the adjacency list in $O(\log N)$ time, followed by a movement of all subsequent nodes by one index. This results in $O(N)$ time per insertion. Further, this assumes the list has been allocated in memory to be longer than necessary before insertion; otherwise more time may be incurred to copy and extend the list. If sorted order is maintained, this cost is incurred for each inserted edge. Instead, the sorted order may be imposed after the entire neighbor list is populated by appending to the end of a list that is grown dynamically, as detailed below.

None of the methods in this dissertation require that the partial data structure to be consistent

```

1 {
2   Node_0 : uint32 -> [Node_x, Node_y, Node_z, ... ] : uint32
3                     [E_0x, E_0y, E_0z, ...] : float64
4
5   Node_1 : uint32 -> [Node_a, Node_b, Node_c, ... ] : uint32
6                     [E_1a, E_1b, E_1c, ...] : float64
7 } : hash

```

Code 4.5: A DICTARRAY: the network adjacency list representation. The nodes in a network are represented by a set of consecutive unsigned integers. Edge weights are represented as floating-point values. A hash data structure is used to map each Node to two arrays, one a list of nodes connected, and another a list of the corresponding edge weights. These lists are in order of the Node integer.

during construction; each utilizes only a complete network. As a consequence, edge insertion may be reduced to a process of appending each edge to the adjacency list. For a given edge (i, j) , the edge to node j is appended to the adjacency list of node i , and the edge to node i is appended to the list of node j . This insertion may be done in constant time, save extension of the adjacency lists. These lists are dynamically grown by a tunable parameter, such as a 1.5-fold increase in allocated space, whenever the number of neighbors exceeds the space already allocated. This balances the amount of additional space allocated for edges that do not exist, with the a reduction in time incurred to copy the adjacency lists to new memory locations (as when `realloc()` cannot extend the memory allocation directly). Every edge is inserted twice, once for each node it is incident upon. That is, no attempt is made to store each edge only once. This is an efficiency tradeoff over space, to optimize for look-up of all neighbors of a node.

After the network is fully populated, each adjacency list is sorted, in $O(\log N)$ time, for a total construction cost of $O(E + N \log N)$ time for the complete network. This may be contrasted with at a potential cost of $O(E \times (E + \log N))$, if the data structure were maintained throughout construction. During the sort procedure, duplicate edges that may be present in the adjacency list are reconciled, with only one remaining. This latter procedure is described as symmetric sequence similarity calculation, below. Symmetric calculation would not be necessary if duplicate edges were resolved during construction, but the cost of insertion would far exceed this post-processing step.

The preceding discussion has focused upon efficiency of use as an in-memory data structure. The DICTARRAY is also amenable to storage in a relational database, and access via native SQL mechanisms. Again, it is worthwhile to consider the trade-offs of implementation to better understand the relevant issues. Here, the most direct and common scheme of storage is to represent each edge with a table row, as in Code 4.6. This may seem like an efficient means of storage, with $3 \times 4 + 8 = 18^2$ bytes required per row. However, the storage overhead of each row is large, requiring in excess of 32-bytes in MySQL, and of similar size in PostgreSQL. The size of the index (i.e., the PRIMARY KEY, here) over all rows induces additional storage overhead, proportional to the size of the data types indexed and their count. Access of a single edge, on account of the index, is fast, though

²Unsigned integers are of size 4-bytes and doubles are of size 8-bytes.

```

1 CREATE TABLE blast_hit_nc_rowwise (
2     nc_id            integer UNSIGNED NOT NULL,
3     seq_id_0        integer UNSIGNED NOT NULL,
4     seq_id_1        integer UNSIGNED NOT NULL,
5     nc_score        double NOT NULL,
6     PRIMARY KEY (nc_id,seq_id_0,seq_id_1),
7 );

```

Code 4.6: Definition of the table for storage of network edges as individual rows. Each row contains an integer identifying the network the edge corresponds to (`nc_id`), two node endpoints, and the edge weight. Beyond the size of the datatypes defined, each row in the database incurs an additional 20–40 bytes for internal representation, making this an inefficient means of network storage.

```

1 CREATE TABLE blast_hit_nc_arr (
2     nc_id            integer NOT NULL REFERENCES nc_run,
3     seq_id_0        integer NOT NULL REFERENCES prot_seq(seq_id),
4     hit_list        int[],
5     nc_score        double precision[],
6     PRIMARY KEY (nc_id, seq_id_0)
7 );

```

Code 4.7: SQL table definition for storage of network edges, directly mirroring a DICTARRAY. In contrast to `blast_hit_nc`, the row overhead as compared to the size of the data is minimal, and access to all neighbors of a node may be achieved by querying a single row.

queries to retrieve all edges in the network can require significant amounts of time, primarily as a consequence of disk speed because the stored table is large.

Despite the very large space overhead of the network storage scheme in Code 4.6, and the overhead incurred for querying the entire network, it is not necessary to forego the utility of SQL as a means of access. Rather, a DICTARRAY may be more directly represented, with the benefit that little translation is required between the in-memory data structure and that used for permanent storage. The resulting table schema is presented in Code 4.7. This listing also includes the foreign key relationships to the tables that describe individual runs of Neighborhood Correlation (i.e., `nc_id` points to table `nc_run`, described in Code 4.4, and to the base table of protein sequences (i.e., `prot_seq`), from Code 4.3). In this case, a single row is used per node, negating the row overhead for each node. The adjacency list for a sequence, `seq_id_0`, is stored as an array (`hit_list`), as is the list of edge weights (`nc_score`). Note that the array data type, `[]`, is PostgreSQL specific, though widely implemented among SQL engines. Querying for all neighbors of a node translates to a request of a single row, and is consequently a very fast operation.

The `blast_hit_nc_arr` table definition sacrifices some ease of iteration when querying with SQL. Because individual edges are not represented as single rows, they may not be directly queried. Access to single edges is most useful when working with the database in an interactive manner. Most of

```

1 CREATE OR REPLACE VIEW blast_hit_nc_rowwise AS
2 SELECT nc_id,
3        seq_id_0,
4        unzip(hit_list) AS seq_id_1,
5        unzip(nc_score) AS nc_score
6 FROM blast_hit_nc_arr;

```

Code 4.8: Definition of an SQL view to transparently emulate the `blast_hit_nc` table from the structure of a `blast_hit_nc_arr` table. The function `unzip()` transforms the array data type to a list of individual rows.

the programmatic methods throughout this work require all neighbors of a node. Much of the convenience of access to individual rows may be achieved by a view to mimic the `blast_hit_nc_rowwise` table from `blast_hit_nc_arr`. The view described in Code 4.8 emulates the table structure of a single edge per row. The only drawback of such a view is that some means of table joins may not be performed in an optimized manner.

4.2.2 Hierarchical tree storage

The family classification pipeline developed in this dissertation uses hierarchical clustering to refine and ultimately partition the sequence network. This establishes a hierarchy of nested clusters of sequences. In keeping with the goal of a central data store, this structure must be represented in SQL. The primary challenge is representing a hierarchical tree in a relational database (as opposed to in-memory representation), so the discussion here focuses upon representation in and access by SQL.

Hierarchical, or recursive data structures are not directly amenable to storage in relational databases. Recursive SQL queries can be done, but tend to rely upon an encoding of the relationships between entities as foreign key references (i.e., pointers to a value in the same or a different table). Even if the complexity of a recursive query is overcome, the database engine would still be required to traverse the height, or depth, of a tree. For example, a query to determine the leaves of a tree would necessarily traverse all nodes step-by-step, until each leaf is reached. For very deep trees, as with a binary hierarchical clustering of the 600k sequences in the 48-genome dataset used throughout this dissertation, such traversal can be extremely time consuming.

With appropriate indexing of nodes, traversal of a tree may be avoided, yielding a structure amenable to interaction via SQL. The NESTED SET structure developed by Joe Celko [30], and described in Figure 4.1 is particularly suitable for this purpose. This illustration is of a binary tree, though the representation may be generalized. The hierarchical clusters in this dissertation are binary, and the following discussion is constrained to the binary case. Here, each node n is enumerated with two integers, l_n and r_n , corresponding to the *left* and *right* indices of a node. These are assigned from a counter, i , that is incremented during a depth-first traversal from the root of the tree. At initialization, $i = 1$, and l_{root} is assigned 1, and i is incremented. For each node, the index l_n is assigned the current value of i , which is then incremented. When all children of a node have been traversed, the index r_n is assigned to be i during the back-trace, and i is,

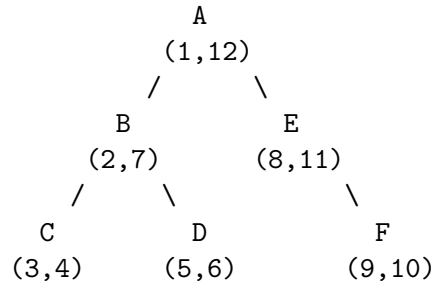


Figure 4.1: Demonstration of NESTED SET tree indexing. For each node (n) in the tree, two integers are stored. These are *left* (l_n) and *right* (r_n) indices are generated by sequential numbering of nodes during a depth first search. The value l_n is assigned when a node is first traversed, and r_n is assigned when the depth-first search revisits the node. All descendants of a node with indices (x, y) have values $(l_n > x, r_n < y)$, allowing them to be directly identified. All leaves have indices of the form $r_n = l_n + 1$.

again, incremented. This proceeds until traversal of all nodes have been completed. Note that this procedure requires a static tree; insertion into a NESTED SET is more involved, and is inherently a slow process. The number of nodes in, and structure of, the trees used in this dissertation are fixed.

The NESTED SET has a number of favorable access properties. First, all children of a node may be identified without traversal of the edges in the tree. For a node with indices $(l_n, r_n) = (x, y)$, all children that comprise the subtree beneath the node have indices such that $l_n > x$ and $r_n < y$. Additionally, all leaves may be directly identified, without graph traversal, because only leaves have indices of the form $(x, x + 1)$. The nearest common ancestor of two nodes may be identified without traversal of nodes that are not common ancestors; for two nodes, with indices (a, b) and (x, y) , respectively, all common ancestors have indices of the form $(i < \min(a, x), j > \max(b, y))$. The nearest common ancestor has the minimal value of $(r_n - l_n)$ of this set of all common ancestors.

The node data and indices of a NESTED SET may be stored in a relational table, with a single row assigned to each node. The table definition in Code 4.9 demonstrates quantities stored at each node in the hierarchical trees used in this work, as well as the nested set indices. This table definition is designed such that a tree may be defined recursively, and constructed in an incremental manner without pre-calculation of l_n (`lft`) and r_n (`rgt`). That is, every node is assigned an identifier, `cluster_id`, and the identifier of the parent of a node is stored in `parent_id`. Upon full population of a tree, and prior to any queries that rely upon the NESTED SET properties, the entire tree may be fetched from the database, l_n and r_n may be calculated, and every row in the table may be updated. No recursion is necessary when querying the entire tree.

The column `distance` corresponds to the measure of distance between the children of a node. Neighborhood Correlation scores, and sequence similarity, are similarity measures, so distance measures are derived. For average-linkage clustering of a network of Neighborhood Correlation scores, this is defined as 1.0, less the mean score of the all edges between the sets of nodes in each child. For sequence similarity, the mean score is subtracted from the maximum value in the dataset. Refer to Chapter 6: Clustering and its evaluation (p.85) for details of the hierarchical clustering procedure that generates these distances. Similarly, `parent_distance` is the distance of the parent

```

1 CREATE TABLE jj_hcluster (
2     cr_id          INTEGER NOT NULL REFERENCES jj_cluster_run,
3     cluster_id     INTEGER NOT NULL,
4     distance       DOUBLE PRECISION NOT NULL,
5     parent_distance DOUBLE PRECISION,
6     parent_id      INTEGER,
7     seq_id         INTEGER,
8     lft            INTEGER,
9     rgt            INTEGER,
10    PRIMARY KEY    (cr_id, cluster_id)
11 );
12 CREATE INDEX jj_hcluster_parent ON jj_hcluster (cr_id, parent_id);
13 CREATE INDEX jj_hcluster_nested ON jj_hcluster (cr_id, lft, rgt);
14
15 CREATE INDEX jj_hcluster_nested_seq ON jj_hcluster (cr_id, lft, rgt)
16 WHERE seq_id IS NOT NULL;

```

Code 4.9: SQL table definition to represent hierarchical clustering trees. The indices created facilitate rapid arithmetic comparison of the `lft` and `rgt` NESTED SET indices.

node. Both of these distances are stored at each node to optimize lateral cuts of the tree, where a chosen distance threshold may be used to select all nodes immediately below the threshold.

Finally, `seq_id` stores the identifier of a sequence associated with a node in the tree. Only leaf nodes represent actual sequences. The database indices listed in Code 4.9 facilitate queries based upon arithmetic comparison of `lft` and `rgt`, facilitating rapid identification of all rows that are children of a selected node. An additional index, `jj_hcluster_nested_seq` is an index only over the leaf nodes, because only leaves have a `seq_id` value.

A typical query is that of cutting the tree at a chosen distance threshold, where the returned clusters are defined to be those nodes with `distance` less than the threshold, but where the distance of the parent is greater than the threshold. Typically, the most desirable output is a set of leaf nodes, not the internal nodes, nor the structure of the subtree that comprises each cluster. This query may be performed in two steps. First, the tree nodes at the boundary of the specified distance threshold are selected. As demonstrated in Code 4.10, these nodes have distances less than the threshold, but their parent has a greater distance. Second, now that the set of internal nodes has been identified, the leaves that correspond to the subtree defined by each may be efficiently queried using the nested set indexing. This is performed as a separate query for each subtree, using the query specified in Code 4.11.

```

1 SELECT cluster_id
2 FROM jj_hcluster
3 WHERE cr_id = %(cr_id)s
4 AND distance <= %(distance)s
5 AND (parent_distance > %(distance)s
6     OR parent_id IS NULL)

```

Code 4.10: SQL query to select all tree nodes immediately adjacent to a chosen threshold, specified by the parameter `%(distance)s`. The parameter `%(cr_id)s` references a specific tree stored in the table `jj_hcluster`.

```

1 WITH node AS (
2 SELECT lft, rgt
3 FROM jj_hcluster
4 WHERE cr_id = %(cr_id)s
5 AND cluster_id = %(cluster_id)s
6
7 SELECT seq_id
8 FROM jj_hcluster
9 WHERE cr_id = %(cr_id)s
10 AND lft >= (SELECT lft FROM node)
11 AND lft <= (SELECT rgt FROM node)
12 AND seq_id IS NOT NULL

```

Code 4.11: SQL query to select all leaves under a given node, specified by parameter `%(cluster_id)s` in a given tree, which is specified by `%(cr_id)s`.

4.3 Implementation of Neighborhood Correlation

Neighborhood Correlation derives much of its utility from the network structure local to a pair of sequences to be rescored. This same property necessitates calculations that are computationally intensive. Naïvely, for a network comprised of N nodes, Neighborhood Correlation rescoring all $N^2/2$ pairs of sequences in the network. In turn, each of these calculations depends upon the set of nodes in the neighborhoods of the each pair of sequences. While an $O(N^2)$ calculation is tenable for the networks of the size considered here, the additional complexity of each computation, which is highly dependent upon local network density, results in substantially higher complexity. At least as importantly, the performance of this method on real machines can vary by orders of magnitude depending upon details of the implementation, even for the same basic algorithm. Further, these networks can consume considerable quantities of memory. Fortunately, all of these challenges may be addressed to facilitate computation of large networks on commodity hardware. This section details the algorithms and use of data structures that facilitate this result.

The primary goals are to perform the computation in a manner that achieves good algorithmic complexity, and does so in a manner that is efficient on contemporary computer architectures. The

first requires efficient reuse of the data, and a reduction of duplicated work. This is insufficient, however, if high performance on real computers is also a goal. This second requirement necessitates careful ordering of the computation to optimize memory access patterns. A final goal is to produce readable, reusable code. To balance usability and enable fast prototyping of new ideas, the Python language is used for all code. Native Python is not necessarily ideal for data-intensive loops, though it is highly suitable for further optimization where needed. First, for simplicity in relating the data structures to concrete in-memory representations, all arrays are represented as C arrays, using `numpy`[112]. Additionally, the data-intensive “inner loop” of Neighborhood Correlation is implemented as a C-extension to the Neighborhood Correlation Python code.

4.3.1 Symmetric sequence similarity

Neighborhood Correlation takes as input a weighted, undirected network. However, the input sequence similarity network, as calculated by BLAST, is effectively a directed network. The score from sequence x to y (i.e., when sequence x is used as a query) may differ from that of y to x . Moreover, in some cases, the score from either may be sufficiently low that BLAST does not return one pair at all. In this case, the network may consist of, for example, a directed edge from x to y , but lack a corresponding edge from y to x . These disparities result from the heuristic BLAST employs for identification of initial seed hits; low-complexity regions of one sequence may be masked in this initial search. (Further explanation of this effect are discussed in Chapter 3: Network rewiring (p.25).) To address this, prior to computing Neighborhood Correlation, the sequence similarity network is converted into an undirected network. An undirected edge is created if either of the two possible directed edges exist. The procedure involves creation of a *symmetric* sequence similarity network, where the score between x and y is identical to that of y and x , and that this score is the maximum similarity returned by BLAST. Where no edge exists in either query order, no edge will be added to the symmetric network.

The DICTARRAY described earlier in this chapter is central to efficient calculation of the symmetric network, with practical memory constraints. First, recall that the DICTARRAY data structure represents every edge in the network twice, to facilitate fast look-up of the complete neighborhood of a node. Construction of a symmetric network consists of insertion of every edge in the BLAST sequence similarity network into a DICTARRAY precursor, twice. That is, for every edge (x, y) , with weight w , the DICTARRAY is populated with edges (x, y) and (y, x) , each with weight w . As described with the introduction of this data structure, these insertions are performed lazily, by appending the new neighbor to the end of the adjacency list of each node, and deferring the sort of these adjacency lists. Should the BLAST result also contain the edge (y, x) , with weight z , this edge will also be inserted twice, resulting in duplicate edges (x, y) and (y, x) in the dictarray, though with weight z . Resolution of this duplicity is deferred. Upon insertion of the complete BLAST result of M edges, this lazy DICTARRAY will contain $2M$ edges, with memory requirements inflated by as much as a factor of two over the corresponding DICTARRAY. Additionally, there will exist potential space overhead from maintaining adjacency lists that may be extended dynamically, as opposed to being reallocated with each added element.

The invariants of a DICTARRAY are established after all sequence similarity edges are inserted. That is, duplicate edges are reconciled, retaining the edge with greatest weight, and order of the adjacency lists is established. This may be performed quickly, with a calculation over each adjacency list in

```

1 PyObject *resolvesymmdups( PyObject *s, PyObject *args) {
2     PyObject *pyelem, *pyvals, *pynelements;
3     int i = 0, shift = 0, nelements;
4     int *elem;
5     double *vals;
6
7     if (!PyArg_ParseTuple( args, "000", &pyelem, &pyvals, &pynelements))
8         return NULL;
9
10    elem = (int *)PyArray_DATA( pyelem);
11    vals = (double *)PyArray_DATA( pyvals);
12    nelements = PyLong_AsLong( pynelements);
13
14    while (i + shift + 1 < nelements) {
15        if ( elem[i] == elem[i+shift+1]) {
16
17            if ( isgreaterequal( vals[i+shift+1], vals[i]))
18                vals[i] = vals[i+shift+1];
19
20            shift += 1;
21            continue;
22        }
23        else {
24            if (shift > 0) {
25                elem[i+1] = elem[i+shift+1];
26                vals[i+1] = vals[i+shift+1];
27            }
28            i += 1;
29        }
30    }
31
32    return Py_BuildValue("i", shift);
33 }

```

Code 4.12: Python C-extension function to resolve duplicate edges after lazy insertion of all edges from a non-symmetric BLAST sequence similarity network.

isolation. First, the order is established by a simple $O(N \log N)$ sort. Then, duplicates are resolved in linear time, as detailed by the function listed in Code 4.12. The function `resolvesymmdups()` is written as a C-extension to Python, operating on numpy data types, which, for these purposes, may be regarded as simple C arrays. This function uses three arguments: the list of neighbors, `elem`, the list of edge weights, `vals`, and the number of neighbors, `nelements`. Because the adjacency list (`vals`) has been sorted, duplicate edges will be proximal. These are identified by a loop over all elements in `elem`. When adjacent positions in `elem` encode an identical neighbor, the weight of

the first is set to the greater of the two weights in `vals`. The following, duplicate, neighbor could then be removed from `elem` and `vals`, although this is not explicitly performed because this would needlessly result in an N^2 iteration over all subsequent neighbors to move each from position i to $i - 1$. Instead, `shift` keeps the count of how many positions should have been removed from the adjacency list. Iteration proceeds until all neighbors have been considered. Finally, `shift` is returned, so that the calling function may truncate the adjacency lists to the actual number of non-duplicate edges. The sorted order is preserved. Beyond algorithmic efficiency, this in-order, linear traversal of this procedure is exceptionally well suited to cache usage on contemporary computer architectures.

4.3.2 Calculation order

The task of Neighborhood Correlation is to rescore all of the $N \times N$ possible edges between all sequences in a network of size N . A most direct approach to calculation is to iterate over each node, x , and to calculate $\text{NC}(x, y)$ for all neighbors y , of that node. Performed in this way, all data associated with node x (i.e., the adjacency list and weights of x) is not repeatedly accessed from memory while iterating over its neighbors. Effectively, the calculation for node x involves an iteration over the complete data structure representing the network for every other node, y , which is a potentially slow endeavor. The following discussion addresses the substantial improvements that may be made over this approach.

An immediate observation is that it is only necessary to iterate over half of the sequences in the network, since $\text{NC}(x, y) = \text{NC}(y, x)$. This reduces the computation, but does not fundamentally alter the complexity involved, or the amount of data that must be considered during each calculation from each node. The inefficiency of this approach becomes especially clear when one considers that no more than 0.5-1% of the edges in a typical network of N sequences actually have defined Neighborhood Correlation scores. Recall that $\text{NC}(x, y) = 0$ if sequences x and y share no common neighbors. More graphically, nodes separated by a minimum path length of greater than two edges necessarily share no neighbors, and thus have no Neighborhood Correlation score defined, beyond the default of a score of zero. Such pairs represent over 99% of the number of possible edges in the network.

The calculation of Neighborhood Correlation scores may be greatly improved by exploiting these structural properties of typical networks with no added cost to atypical networks. Since a Neighborhood Correlation score needs to be calculated only for pairs of nodes which are connected by a path length of at most two edges, one could identify those pairs of nodes first, and then calculate their scores. Unfortunately, identifying such nodes is itself a procedure that iterates over the entire network, and following this with Neighborhood Correlation is not likely to be a net-win. The book-keeping necessary for this procedure is likely to consume a substantial quantity of memory, and inefficient cache use results from repeated iteration over the complete network. Neighborhood Correlation may instead be calculated concurrent with a search for all paths at most two edge in length.

The general scheme is to perform a breadth-first search from each node to calculate the score between a node x and all immediate neighbors. Iteration over each of those neighbors necessarily enumerates (but does not access) the set of neighbors of each of those nodes (called, the *next-neighbors*). For each node x , Neighborhood Correlation is first computed between x and all

```

1 def covariance_xy( hits_0, scores_0, expect_tup_0,
2                   hits_1, scores_1, expect_tup_1,
3                   num_seqs, logsmin):
4     (n_0, sum_0, Ex) = expect_tup_0
5     (n_1, sum_1, Ey) = expect_tup_1
6     n_01 = s01_sum = s01_prodsun = 0
7     ind_0 = ind_1 = 0
8     len_0 = hits_0.size
9     len_1 = hits_1.size
10
11     while ind_0 < len_0 and ind_1 < len_1:    # hit lists are in sorted order
12         h0 = hits_0[ind_0]
13         h1 = hits_1[ind_1]
14         if h0 < h1:
15             ind_0 += 1
16             continue
17         elif h0 > h1:
18             ind_1 += 1
19             continue
20
21         # elif h0 == h1:
22         n_01 += 1
23         s01_sum += scores_0[ind_0]
24         s01_sum += scores_1[ind_1]
25         s01_prodsun += scores_0[ind_0] * scores_1[ind_1]
26         ind_0 += 1
27         ind_1 += 1
28
29     n_others = num_seqs - (n_0 + n_1 - n_01)
30     Exy = (s01_prodsun + (logsmin**2) * n_others +
31           (sum_0 + sum_1 - s01_sum) * logsmin
32           ) / num_seqs
33     var_xy = Exy - Ex * Ey
34     return (n_01, var_xy)

```

Code 4.13: Calculation of the covariance between the adjacency lists of two sequence neighborhoods. Shown here in Python, this procedure is implemented as a C-extension of identical flow.

neighbors, y , of x . During the calculation of the correlation with each neighbor, y , the set of next-neighbors is constructed. Once the Neighborhood Correlation score to all neighbors has been computed, $NC(x, z)$ is computed, for all next-neighbors, z . The Neighborhood Correlation score between x and all neighbors, and all next-neighbors, is necessarily defined because they share at least one neighbor. This methodology computes Neighborhood Correlation for the minimal set of pairs for each node, x .

Note that this scheme accesses the adjacency lists in the network twice, because the breadth-first search is repeated from each node in the network. Additionally, this procedure is performed for every node in the network, without taking advantage of the symmetry of Neighborhood Correlation (i.e., $NC(x, y) = NC(y, x)$). The benefits of this procedure are algorithmic, and practical. Algorithmically, no book-keeping need be performed to record which edges have been calculated. A substantial practicality results: this procedure may be implemented in a highly parallel manner because the underlying data structure is not changed, and no synchronization is required between calculations from each node. Calculating the complete network is not without advantage because, ultimately, the complete network is desired when querying for the set of neighbors connected to a node x in the Neighborhood Correlation network. Here, calculation of Neighborhood Correlation may be performed twice for every pair more quickly than it would be to restore the symmetric edge through post-processing.

Discussion to this point has focused upon of the calculation of Neighborhood Correlation over the entire network. A number of optimizations are key to efficient computation of the score for an individual pair of nodes. The inner loop of $NC(x, y)$ calculates the covariance of the weights (see Code 4.13). The covariance between the neighbors of node x and the neighbors of node y must account for pseudo-edges of weight $\log(S_{\min})$, and so is more involved than a direct calculation of covariance. In this code, n_0 and n_1 are the sizes of the neighborhoods of nodes 0 and 1, respectively. `sum_0` and `sum_1` are the sum of all edge weights in the neighborhood of a node, and `Ex` and `Ey` are the mean score of these edges, accounting for $\log(S_{\min})$. During this calculation, the ordering of the adjacency lists is exploited to facilitate a linear traversal of the arrays. This function returns the size of the common neighborhood and the covariance.

Analysis of network properties

A major challenge in the development of means for family classification is a shortage of effective evaluation metrics. This dissertation develops two fundamentally different, but complementary, approaches to measuring the performance of classification methods. First, this chapter describes the framework and methodology to evaluate family classification using *intrinsic* measures of the data; i.e, by measuring the inherent properties of the input and output data that are believed to correlate with common ancestry. A second approach may be employed when curated data, such as a labeling of complete families, is available. The use of *extrinsic* measures based on labeled data is developed in Chapter 6: Clustering and its evaluation (p.85). The approach here considers the structure of a network representation of proteins, where nodes are sequences, and edges represent sequence similarity, derived measures such as Neighborhood Correlation, simulated values, or, when considering the theoretical context, true homology.

This chapter considers the scenario of evaluation in the absence of ground-truth knowledge. Of course, when evaluating a specific method with intrinsic measures, it is worth considering whether there is circularity between the method applied and those measures. It is unlikely that the methods and measures will be orthogonal, and ensuring so would be an unattainable goal. The compromise sought in this work is to develop a theoretical expectation for the structure of a homology network, and to then measure the discrepancy between real networks and this theoretical ideal.

Reliance on measures that can be calculated directly from the data yields a number of substantial benefits which are not offered by extrinsic measures based on curated families. The data upon which the evaluation is based necessarily increase in size with the input; they are the same data. In contrast, the pace of expert family curation lags far behind that of genome sequencing. The evaluation is fine-grained: It is not necessary to process the data to a form that directly corresponds to the output (e.g., clusters representing families) prior to evaluation. Instead, the structure of the initial input, intermediate, and derived networks may be measured separately, and compared. In particular, these may be compared to a theoretical expectation. Intrinsic measures may reveal *why* a method behaves as it does, more so than when only the result is examined. Finally, the approach is amenable to evaluation of targeted simulation of portions of the family classification pipeline. Without such means of incremental evaluation, it is difficult to delineate the scope of any simulation; more complex simulations necessarily introduce a great many unknowns and untested

hypotheses.

The concept of a homology network was introduced in Chapter 2: Background and preliminaries (p.13). To review, the mathematical concept is as follows. A homology network, $G_H = (V, E_H)$, is comprised of nodes that correspond to sequences. Edges are not weighted: an edge (x, y) exists in E_H iff x and y are homologous. The definition of a gene family implies certain structural properties of the network. Most notably, the network is transitive; all members of a gene family are, by definition, homologous to all other members. The members of each family comprise a clique in the network, discrete from all other families. The distribution of sizes of cliques in the network are defined by the sizes of families in the dataset. Note that this homology network may represent sequences from one or many genomes. Families that are conserved among several genomes will tend to scale in size with the size of the dataset. That is, the size of such a family relative the size of the dataset will tend to remain constant, whether the dataset is comprised of a single genome, or of many.

The homology network is unknown for real data; indeed, the goal of this body of work is to estimate it. Several networks are considered in this chapter: Common estimates of homology include use of a network of sequence similarity, $G_S = (V, E_S)$, where edges in E_S are weighted, and encode the degree of pairwise similarity between two sequences. The result of rewiring G_S with Neighborhood Correlation is encoded in $G_{NC} = (V, E_{NC})$, where the weight of an edge in E_{NC} represents the Neighborhood Correlation score between a pair of sequences. Additionally, for simulation purposes, synthetic, unweighted networks are considered.

The overall goal of measuring the properties of the graph is to evaluate how well a particular network approximates the structural properties inherent in a homology network. For this problem, the biological property of interest (families that share common ancestry) corresponds to a precise mathematical property (graph transitivity). This ability to recast the problem to an objective goal guides method design and provides a natural basis for evaluation in the absence of a gold standard. Figures of merit that assess the transitivity of a graph are appropriate internal validation methods.

An additional goal is to facilitate effective means of comparing two networks. What are the bases upon which two estimates of homology may be compared? For example, a comparison of two unweighted networks, $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$, might consist of the Hamming distance between the two edge sets, E_1 and E_2 . I investigate how meaningful comparison may also be made when the edge weights represent very different properties. The networks here are weighted, requiring methods of evaluation and comparison that take the weights of edges, not just their presence or absence, into account. Similarly, practical application to sequence networks requires means of comparison of networks in which the node sets differ, as when a network comprised of one genome is to be compared to a network comprised of many genomes.

In this chapter, I describe a set of measures that capture various aspects of network structure. These measures are designed to reflect graph transitivity, and particular attention is paid to their application to the weighted networks utilized throughout this dissertation. The first aspect of this evaluation is within simulated data, where the “true” homology network, G_H is explicitly defined. Behavior of the measures is empirically observed as the network is degraded through a simple shuffling of edges, meant as a most basic model of the effects of domain shuffling upon a sequence similarity network. This simulation provides an empirical demonstration of behavior of the network metrics, and as a sanity check for Neighborhood Correlation. The structure of the

simulated homology network, the degraded network, and G_{NC} are compared.

Next, these measures are applied to real data, to evaluate the performance of sequence similarity and compare this with the result of Neighborhood Correlation. Two sources of data are considered. The first is comprised of nine closely related yeast genomes, where little is known about families of sequences. This allows detailed examination of the behavior of Neighborhood Correlation as the size and composition of the dataset is varied. Second, the network measures are applied to the dataset comprised of the mouse and human genomes.

5.1 Network measures

The intrinsic evaluation performed here involves use of network measures designed to quantify the degree of transitivity inherent in a network, and to capture general properties of the network. They draw from a long line of fundamental measures of network properties [12]. These network measures have been selected to characterize a wide spectrum of network structure. All are intended to capture abstract properties such as the degree to which individual connected components are connected, or the degree to which they are separated from the rest of a network. The variety of measures used are intended to each capture a specific property of network structure, and together provide a perspective of the overall structural differences between different networks.

Nearly all of the real networks considered in this work are weighted. However, the meaning represented by those weights differs in each specific network. For example, an edge weight in the sequence similarity network is typically the score associated with the alignment of the sequences represented by a given pair of vertices. Throughout this dissertation, the similarity value used the bit-score reported by BLAST. The bit-score is a positive value bounded by a weighted function of sequence length, and sequence composition using an amino acid substitution matrix. The bit-score has no defined upper limit. Further, edges are defined only between pairs of sequences exceeding (1) the E-value threshold used¹, and (2) the ability of the BLAST heuristic to achieve a seed hit from one sequence to another. By contrast, a Neighborhood Correlation network may be thought to have an edge between all pairs of sequences. The weight of these edges ranges from zero to one, depending upon the local network structure. The edge weight is zero if no common neighbor exists between two sequences in the sequence similarity network.

Thus, the primary task is to compare networks that have the same set of nodes, but different edge sets, with weights that represent very different units; e.g., are defined on different numerical scales, and have different numerical distributions. Additionally, the edge weights of a network cannot be transformed by a function of edge weight to correspond to those of another network that represents fundamentally different information. For example, in the sequence similarity network, edge weights depend only upon the sequences associated with a given pair of nodes, whereas the Neighborhood Correlation calculation is derived from the local neighborhood of pairs of sequences in the network. This yields not only a different distribution of weights between sequences, but these weights also define a different ordering of edges.

Clearly, edge weight needs to be considered in evaluating a particular network and when comparing different networks. To do otherwise necessarily discards a great amount of information inherent to

¹In this work, the significance threshold is set to a minimal value, such that effectively all BLAST seed hits result in an edge. See 3: Network rewiring (p.25) for details.

the network. How the edge weights are used in these applications is not necessarily a straightforward problem. Consider graph density. In the context of an unweighted network, the fraction of edges that exist relative to the number of possible edges in the network, $D_G = \frac{|E|}{N(N-1)/2}$, has a meaning that may be directly interpreted.

A common scheme for use of a weighted network is to apply a threshold, yielding a new unweighted network comprised of the same nodes, but only edges with a weight above (or, alternately, below) a given value of edge weight. Transformation to an unweighted network enables the use of unweighted measures, and allows comparison of the structure of networks derived from different data. This is a step in the right direction. The simple strategy of counting the edges that exist in a weighted network is a special extreme of thresholding. However, thresholding still has the potential to discard important structural features. How might more of this information be preserved, while still facilitating consideration of the network with unweighted measures?

Edge weight could be utilized by performing a sum of the edge weights rather than a count. Most of the measures presented in the following could be so transformed. However, this strategy does not solve the issue of network comparison. Simply weighting any given measure, such as graph density, by the edge weight in lieu of existence is insufficient for the comparison of networks with edges that represent different values: the value that the edge weight captures is only consistent within one type of network. Even ignoring meaning or magnitude, the scale of the weights in one network type is not necessarily linearly related to the weight used in another network. Further, between networks that represent very different information, such as sequence similarity or Neighborhood Correlation, no transformation, linear or otherwise, should be expected to exist.

The strategy undertaken here is to consider edge weights as an ordering of those edges, disregarding the specific meaning inherent to the type of network it is within. This ordering is established by an implicit linear transformation of the original score range; i.e., only the order and relative distances are retained. A particular network measure may be applied to any weighted network, to produce not a single value for one static network, but rather a distribution of that network measure over all possible networks as the edges are added (or subtracted), in the order defined by their edge weight. The above procedure is similar to, but subtly distinct from, application of a threshold. It implicitly considers application of *all* possible thresholds to a network. The result is a *profile* of that network measure over the complete edge weight range of a network, from most inclusive to most stringent. These may then be compared without respect to the value that the edge weights represented.

5.1.1 Measure definitions

This section establishes the specific vocabulary used for discussion of the networks to come and defines the specific properties of interest. First, an undirected network, $G = (V, E)$ is comprised of vertices, individually enumerated as $v_i \in V$, and edges, similarly enumerated as $e_{jk} \in E$. Edges may exist between any two vertices, but no self-edges are permitted.

First, we may consider a global property of the network. The **network density** is a measure of the number edges in the network, expressed as a fraction of the possible edges in fully-connected network:

$$D = \frac{2|E|}{|V|(|V| - 1)}. \quad (5.1)$$

The network density provides a constraint upon the structure of any network. For networks of equal density, the distribution of edges between pairs of nodes defines the structural properties of each network; e.g., whether it is transitive. It is evident that a network of zero density cannot be transitive, as no edges exist, and a network with a density of 1 is maximally transitive (though uninformative). When comparing networks, network density is maintained to be equal whenever possible.

One means of directly evaluating the transitivity of a network is the **mean clustering coefficient**. The clustering coefficient, C_i is a measure of the local connectivity of the network with respect to a node i . Specifically, this is a count of the number of edges that exist between the neighbors of node i , or

$$C_i = \frac{2|\{e_{jk}\}|}{|N_i|(|N_i| - 1)} \quad \forall j \in N_i, k \in N_i, \text{ and } \forall e_{jk} \in E, \quad (5.2)$$

where $N_i = \{v_j \mid v_j \in V, \text{ where } e_{ij} \in E\}$, the set of all neighbors of node i . To obtain a single measure of the clustering coefficient over the complete network, an average of C_i is taken. In some cases, the total number of nodes in the network ($|V|$) is used as a normalization constant. However, to provide a more sensitive measure of the clustering coefficient, the convention adopted in this dissertation is to normalize only the count of nodes that have a defined clustering coefficient. That is, the clustering coefficient is a measure of how well the neighbors of a node are connected. For a node of degree 0 or 1, there are no possible edges between neighbors (i.e., $C_i = 0$, when calculated as above). To consider only the vertices that have neighbors that may or may not be connected to each other, we normalize over all nodes of degree greater than one, or

$$W = \{v_i \in V, \text{ where } |N_i| \geq 2\} \\ C = \frac{1}{|W|} \sum_{i \in W} C_i. \quad (5.3)$$

$C = 1$ iff the network is transitive.

Any network may be decomposed into a set of connected components, or sets of nodes in which each node in a component is connected by some path to every other node in the component. Nodes in different connected components, by definition, have no edge path between them. In the convention adopted here, connected components are distinguished from singleton nodes, which have no neighbors. That is, a connected component must contain at least one edge. In the following, CC refers to set of all connected components in given network. The set of nodes in a specific connected component $c \in CC$ is denoted V_c , and the set of edges in the component is E_c .

Just as we may measure the overall density of a network, we may define a more local measure of density, by considering proportion of edges present within components in the network. The **network component density**, D , is a measure of global transitivity. This is also known as the *average compactness index*. The density of a single component c , containing nodes V_c and edges E_c , is defined to be

$$d_c = \frac{2|E_c|}{|V_c|(|V_c| - 1)}. \quad (5.4)$$

To provide an aggregate measure of component density over the entire network, the densities of all components may be summed, and normalized. Because the unit of summation is a count of edges, this summation is weighted by the number of edges in each component, as follows:

$$\begin{aligned}
 D &= \frac{\sum_{c \in CC} |V_c|(|V_c| - 1)d_c}{\sum_{c \in CC} |V_c|(|V_c| - 1)} \\
 &= \frac{2 \sum_{c \in CC} |E_c|}{\sum_{c \in CC} |V_c|(|V_c| - 1)}.
 \end{aligned}
 \tag{5.5}$$

The network component density, D , is equivalent to the ratio of the total number of edges in G to the number of possible edges within the components of G . Note that $D = 1$ iff the network is transitive.

The clustering coefficient (C) and network component density (D) each increase with transitivity, reaching unity in a fully transitive graph. Although, in general, high values of C and D for a sequence similarity network, G_S , suggest that the structure of the network more closely approximates G_H , these measures can be misleading in extremely dense or sparse graphs. In a graph consisting of one, or a very small number, of dense connected components, both C and D will be close to one. However, this is not a realistic gene family model. At the other end of the spectrum, D will be unity in a graph consisting entirely of components of size two, but these, again, are not typical of gene families in real data. Moreover, the clustering coefficient is not informative for very sparse graphs, since C is not defined on connected components of size two.

The **mean connected component size** is calculated as

$$\overline{N_c} = \frac{\sum_{c \in CC} |V_c|}{|CC|}.
 \tag{5.6}$$

5.2 Interpretation of measures

As described above, the networks generated by the methods applied throughout this dissertation are weighted networks. The network measures developed here are of most value for evaluating the structure of a network as it is perturbed, either in the scenario of successive edge removal (or addition), or that of comparison of two networks of wholly different edge composition. Each measure, in isolation, provides only a limited perspective about the structure of a network. For example, consider a process where edges are continually removed from the network. That edges have been removed between steps will necessarily be reflected by a decrease in graph density. However, what might an increase in the clustering coefficient or network component density indicate? What of a decrease? In fact, example networks may be easily contrived to yield an increase *or* decrease of any of these measures, save a necessary decrease in graph density as edges are removed from the network.

Intuition about the structure of a network may be gained by considering the joint behavior of the network measures. Figure 5.1 considers the restricted case where a given component motif is modified by edge-removal. The arrow for each measure indicates whether the magnitude of that measure will necessarily increase, decrease, remain unchanged, or, as in one case, may either increase or decrease. The trends in this figure are based on the assumption that the motif is embedded in a larger network, of arbitrary construction, that does not simultaneously change.

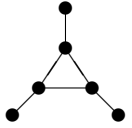
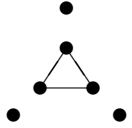
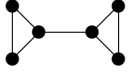
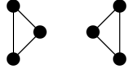






	Low Thresh	Hi Thresh	Network density	Mean clustering coefficient	# Connected Components	# Singletons	Mean CC size	Network component density
A			↓	↑	–	↑	↓	↑
B			↓	↑	↑	–	↓	↑
C			↓	–	↓	↑	↑	↓
D			↓	↑	↓	↑	↑ or ↓	↑
E			↓	↓	–	↑	↓	–

Figure 5.1: Scenarios involving various connected component motifs. The indicated change in magnitude of a network measure is reported as edges are removed from a component. Arrows indicate either an increase or decrease in the magnitude of a measure. These projections assume the component is embedded within a larger network of other components, and that larger network does not change.

The illustrated component motifs are *not* meant to be exhaustive. Further, complex, simultaneous changes in a network may yield a mixture of scenarios. However, the network measure behaviors they illustrate are representative of the real networks considered in this work. This is especially true when incremental changes are observed over a limited range of edge weight thresholds. A typical observation upon closer examination of the sequence networks considered here is that, within discrete ranges of the distribution of edge weights, a particular component motif is responsible for the structural changes observed in the network. That is, despite a lack of generality, the motifs in Figure 5.1 *are* representative of the types of networks considered in this dissertation.

With the aforementioned caveats, a number of key observations may be made about the behavior of the network measures. The scenario considered throughout is that of a continual process of edge-

removal. (The following sections will include plots of this process in simulated and real networks.) First, note that removal of edges from a network necessarily decreases graph density. The behavior of other measures of the network capture aspects of the structural changes in the network, such as whether the edge removal results in complete loss of structure (resulting in singletons) or whether components of higher density result.

Consider scenario in Figure 5.1A. Initially, a trio of nodes is maximally connected, and each node has a single edge to one of three other nodes. Suppose the three outer edges are removed during an in-order removal of edges. This necessarily increases the clustering coefficient of the network; the value C_i of each central node increases from $\frac{1}{3}$ to 1. The number of connected components is unchanged, though the count of singleton nodes increases on account of the three isolated nodes. At the same time, the density of the remaining component increases (to 1) and the size decreases. This component is representative of dense components or cliques with a few spurious edges to otherwise disconnected nodes. These are common in real sequence networks.

Figure 5.1A may be contrasted with the subtly different network measure behavior in 5.1B. The latter is representative of dense components that are connected by a small number of weaker edges. In this case, the count of connected components necessarily increases as edges are removed, with no change in the number of singletons.

In many instances, removal of a strong edge between two sequences would suggest that too many edges are being removed from the network, as with selection of too stringent an edge weight threshold, leading to loss of the inherent structure of the network. This can indicate that a less stringent threshold should be considered. For example, the datasets under consideration here frequently have pairs of genes that are very closely related. This is especially true when the dataset is comprised of two very closely related genomes. In this case, a large number of connected components of size two are observed. The scenario in 5.1C demonstrates such a case. Here, as edges are removed, the connected component is removed in entirety, and is replaced by singletons. The network component density decreases whenever a component of maximal density is removed from a network not already of density 1, which is a robust assumption here. Similarly, the mean size of components increases if the size of the degraded component was below the original mean. Here, the mean component size necessarily increases because a component of size two is the smallest possible.

Figure 5.1D represents a pervasive motif in sequence similarity networks, one that is highly unlikely to represent actual biology. Here, a series of nodes is connected as a “string”, each to two other nodes. Such a component is of minimal density, so its removal necessarily increases the network component density. However, the mean size of the components may increase or decrease, dependent upon whether the mean of the graph is above or below the number of nodes in the degraded component. Of course, the count of connected components decreases, and the number of singletons increases.

The final scenario, 5.1E, illustrates removal of two edges, and one node, from a minimally sized clique of size 3. This can be illustrative of a pair of strongly connected orthologs that are more weakly connected to a third sequence. Here, transitivity of the network is decreased, as reflected by a decrease in the clustering coefficient. The number of singletons increases. Note that the network component density remains unchanged, as the initial and remaining components are each maximally dense.

5.3 Simulation

Simulation of the types of networks under consideration here is performed to address two primary goals. First, simulation of networks with structure mirroring theoretical homology networks may be used to better understand the behavior of the network measures introduced in the previous section. Second, simulation can be used to investigate the ability of Neighborhood Correlation to restore transitivity to artificially degraded homology networks. In both cases, simulation provides a means of evaluation, where a full history of the input data is known, and allows targeted investigation of a method’s behavior in the absence of the ambiguity present in real data. Importantly, the complexity of the simulation may be chosen, and varied.

I construct an artificial, unweighted network, $G_H = (V, E_H)$, to mirror the theoretical expectation of a homology network. This network is comprised of a disjoint set of cliques, intended to emulate the structure of a set of families. The general strategy is then to degrade G_H by adding and removing edges in a process designed to mirror the effects of faulty homology detection. The changes to the network are then quantified using the network measures developed in this chapter. Finally, Neighborhood Correlation is applied to the simulated network to test its ability to restore the transitive structure of the original network of cliques.

The structure of G_H is subject to two parameters, the number of families, and the distribution of sizes of those families. Each family induces a discrete clique in the network. For the purposes of this study, I have considered networks comprised of families of varying sizes, ranging from 4 to 64 nodes each. Small cliques are more likely to be disrupted by random permutations of the network, while large cliques may be thought to have more “redundant” edges that prevent disintegration of the component upon permutation of the network. Typical family sizes may be estimated from our curated data set, though, in reality, much uncertainty exists. The clique sizes selected here are consistent with the absolute size of many known families, and their size relative to the size of the entire network.

A single mode of perturbation is used to impose noise upon G_H in the simulations presented here. Edges are selected uniformly at random from G_H , and removed from the network with a deletion probability of P_d . That is, for a network of $|V|$ nodes, and $|E|$ edges, $|E|P_d$ edges are removed, in expectation. Since the original network G_H is comprised entirely of cliques, these edges are necessarily removed from cliques. At the same time, the network density is preserved, in expectation, by a corresponding addition of edges between the original cliques. The probability of edge addition, P_a is calculated from P_d , as

$$|E|P_d = P_a \left[\frac{|V|(|V| - 1)}{2} - |E| \right]$$

$$P_a = \frac{|E|P_d}{\frac{|V|(|V|-1)}{2} - |E|}, \quad (5.7)$$

where the quantity $\left[\frac{|V|(|V|-1)}{2} - |E| \right]$ is the count of edges not present in G_H . A value of $P_d = 0.5$ represents exchange of half of the edges in G_H for an equivalent number of edges between cliques, on average.

This simulation procedure is intended to mirror the effects of faulty homology detection. The resulting network is referred to as $G_S = (V, E_S)$, akin to a sequence similarity network. Removal

of edges simulates the failure to recognize remote homology between family members. Added edges represent similarity that is not evidence of homology, especially as induced by INCONSISTENT domains. The net result in G_S is a shuffling of edges from within the cliques of G_H to between them.

Neighborhood Correlation is then applied to the simulated network G_S to generate the network G_{NC} . The definition of Neighborhood Correlation (Equation 3.1) is specified over a network that is weighted. Here, the network G_S is unweighted. The corresponding calculation of Neighborhood Correlation uses edge weights of 0 or 1 in this case, and a more succinct equation may be derived. That is, for a network of N nodes, the score between a pair of nodes is

$$\text{NC}(x, y) = \frac{NN_{xy} - N_x N_y}{\sqrt{N_x(N - N_x) \cdot N_y(N - N_y)}}, \quad (5.8)$$

where N_x and N_y are the degree of nodes x and y in the network, and N_{xy} is the size of the intersection of their neighborhoods. The network G_{NC} is weighted with the resulting score.

Throughout this chapter, networks are compared under the condition that network density is held constant. Graph density is a suitable basis for normalization, because D is directly, and C is indirectly, dependent on overall graph density. Intuitively, since the goal here is to examine structure of the network, we wish to evaluate the rearrangement of existing edges, more so than to study the behavior as edges are added or removed.

The simulation procedure produces G_S such that G_S and G_H have the same number of edges in expectation, and, hence, have identical network densities. The network G_{NC} implicitly consists of all edges, with weights in the range 0 to 1. To facilitate comparison, a Neighborhood Correlation edge threshold must be selected such that the network G_{NC} has density equivalent to G_H (and G_S). Procedurally, this is accomplished by computing G_{NC} , followed by optimization of the edge threshold such that the density of G_{NC} is within epsilon of that of G_H , or $D(G_{NC})/D(G_H) = 1 \pm \epsilon$. Here, $\epsilon = 0.001$.

Figures 5.2 and 5.3 illustrate the results of the simulation procedure for networks of 1024 nodes, and differ in the size distribution of cliques. Each figure depicts: (a) the number of connected components, (b) the number of singleton nodes, (c) the network component density, and (d) the mean clustering coefficient. Note that the network density is entirely determined by the size distribution of cliques, and, hence, is constant for all plots and values of the noise parameter, P_d , for a given network. In all figures of simulated data, the value of a measure for the original network (G_H) is depicted in red; this value is a line and does not vary with respect to the noise parameter P_d . Values for the simulated network G_S are shown in blue, and those for the network rescored by Neighborhood Correlation are shown in green. Vertical bars represent the standard error over 1000 network simulations.

First, we focus upon a network comprised of cliques of mixed size. Figure 5.2 is based upon a network of 48 cliques: 16 cliques of size 4, and 8 each of sizes 8, 16, 32, and 64. The choice of this distribution is intended to illustrate the behavior of the simulation process and network measures in components of heterogeneous sizes. The absolute sizes and sizes relative to the network size were guided by the observed family sizes in the curated mouse and human family benchmark.

Relative transitivity is assessed by comparing the values of C and D for G_S and G_{NC} . Recall that G_H is, by definition, transitive, and both C and D equal 1. A very small number of mis-assigned

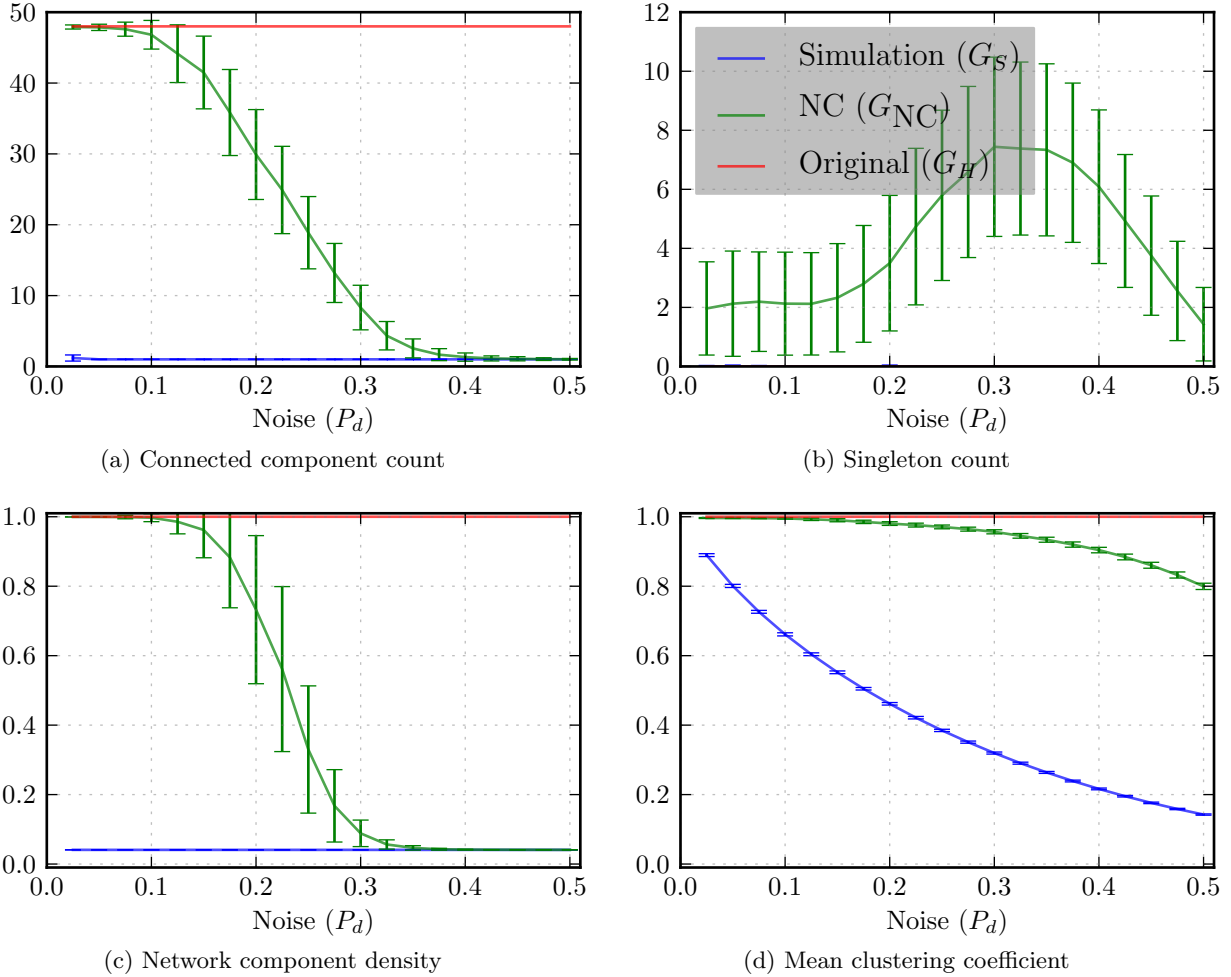


Figure 5.2: Component and transitivity measures of simulated networks of cliques degraded by noise. A network of 1024 nodes is used, comprised of 16 cliques of size 4, and 8 each of sizes 8, 16, 32, and 64 nodes. The vertical bars represent the standard error over 1000 trials.

edges is sufficient to completely disrupt the clique structure of the network G_H , as shown by the low values of both $D(G_S)$ and $C(G_S)$ in Figure 5.2. Similarly, effectively any value of $P_d > 0$ collapses the network to a single connected component. This is to be expected, because each edge added by the simulation procedure necessarily connects two original cliques. On average, the cliques contain enough edges such that removal of edges uniformly at random does not disconnect any nodes from the network, and the singleton count for G_S approaches zero.

By contrast, Neighborhood Correlation is able to completely restore transitivity to G_S , whenever $P_d \leq 0.1$. In addition, the network G_{NC} contains 48 components, for low values of noise. This is strong evidence that Neighborhood Correlation is able to reconstruct G_H perfectly, at low error rates. The number of singleton nodes (b) may be understood as follows. Rewiring with Neighborhood Correlation can never join singleton nodes with the rest of the network, but it can separate

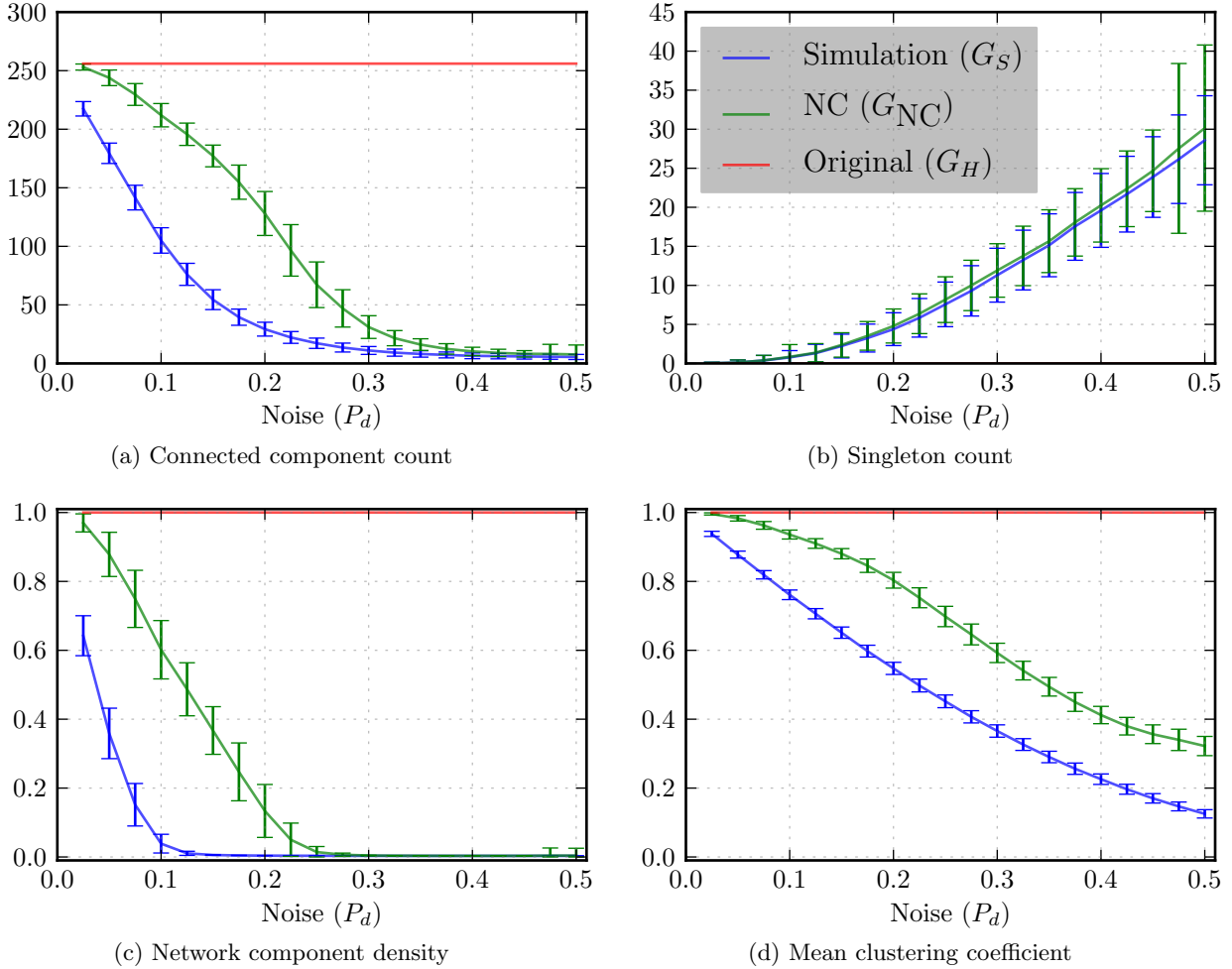


Figure 5.3: Component and transitivity measures of simulated networks of cliques degraded by noise. A network of 1024 nodes is used, comprised of 256 cliques of size 4. The vertical bars represent the standard error over 1000 trials.

weakly connected nodes. As a result, G_{NC} is observed to produce a small number of singleton nodes.

The degree to which Neighborhood Correlation restores transitivity and structure to the network decreases as G_S is subjected to more noise. However, even with P_d as high as 0.2, Neighborhood Correlation is still able to raise the number of connected components to half of the original count in G_H , and these are of high density. The clustering coefficient (d) demonstrates that even when the added noise is such that rewiring with Neighborhood Correlation does not restore a fully transitive network, the local transitivity of this network is consistently high — above 0.8 for the entire range.

The preceding simulation was performed to achieve a better understanding of behavior in heterogeneous networks. A simulated network comprised entirely of very small cliques is likely to be affected to a greater degree by noise. The number of edges in the network G_H is related to the square of

the size of each clique. As a result, for a network of the same number of nodes, fewer edges exist when G_H is comprised of a large number of small cliques, as compared to cliques of larger size.

Figure 5.3 demonstrates how a G_H constructed from cliques of size 4 is disrupted by noise. As compared to Figure 5.2, note that the network G_S does not coalesce to a single component until $P_d > 0.25$. This is the result of two factors. First, this network is comprised of 256 components, as opposed to 48. A greater number of randomly placed edges is required to connect more components. Second, the noise parameter P_d is a probability of deletion for any edge in the network. Since a network of cliques of size 4 has fewer total edges, the absolute number of edges removed and added ($|E|P_d$) differs from Figure 5.2 for a given value of the noise parameter.

As a consequence of these structural differences, the network component density of G_S decreases more gradually than in Figure 5.2. However, because fewer “redundant” edges exist in the network, the component density restored in G_{NC} is lower in this simulation, for equivalent values of noise. Similarly, because small cliques are more easily degraded to singleton nodes, Figure 5.3(b) demonstrates a creation of many more singletons in G_S and G_{NC} .

Finally, the mean clustering coefficient of G_S is observed to differ very little between a network of small cliques, and a network composition of cliques of varied, and larger sizes. The clustering coefficient is a measure of local network structure, which does not take the overall structure of components into account. Yet, the simulation procedure removes a larger relative fraction of the edges in a clique for small cliques as compared to large cliques. The clustering coefficient remains roughly equivalent in networks of small or large cliques (G_S in Figures 5.3 and 5.2, (d)). This reflects the property that the local structure does not differ much when either type of network is degraded, but that only the network with larger cliques retains the redundant information necessary to restore the structure. As a consequence, Neighborhood Correlation is unable to restore the local transitive structure to the network of small cliques as well as it does for a network that contains large cliques.

5.4 Analysis of Yeast networks

Having gained some intuition into how the network metrics presented here may behave on different network motifs and simulated networks, we can apply them to networks of real data. Consideration of yeast presents a real use case, in that there exists no comprehensive curated dataset of yeast genes that comprise evolutionary families. Additionally, there are a number of closely related yeast genomes; this data may be exploited to measure stability, and possible improvement, of the result as similar genomes are added to an analysis.

The amino acid sequences from nine yeast genomes were obtained from the YGOB, version 2 database [29]. Three groupings of these genomes were considered: (1) the set of all genes in *S. cerevisiae* alone, totaling 5616 sequences; (2) those in four genomes (*S. cerevisiae*, *C. glabrata*, *A. gossypii*, and *K. lactis*), totaling 20839 sequences; and (3) those in all nine genomes in YGOB2 (*S. bayanus*, *S. castellii*, *K. Polysporus*, and *S. kluyveri*, in addition to those above), totaling 46060 sequences.

All-against-all BLAST comparisons were carried out in each of these three datasets. Similarly, Neighborhood Correlation scores were then calculated for all pairs within each. The results of these calculations comprise three sequence similarity networks and three corresponding Neighborhood

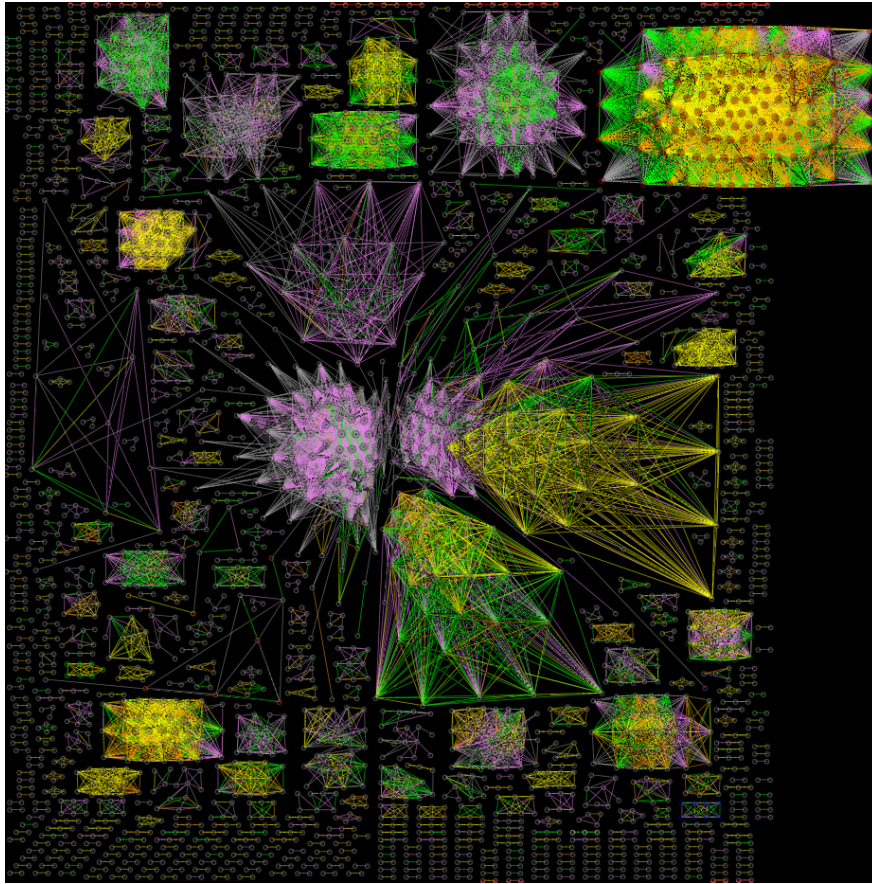


Figure 5.4: Visualization of the *S. cerevisiae* genome after rescoring with Neighborhood Correlation. Edge color signifies the Neighborhood Correlation score, where gray indicates $NC \geq 0.3$, violet ≥ 0.4 , green ≥ 0.6 , orange ≥ 0.8 , and yellow ≥ 0.9 . The dense component at top-right contains all Kinases. Singleton nodes have been omitted for clarity.

Correlation networks. To establish a common basis for comparison, these networks were then pruned to the set of nodes that represent sequences in *S. cerevisiae* only; i.e., each network has the same nodes but potentially different edge connectivity and weights. Note that, after pruning, all three sequence similarity networks actually have the same set of edges. The sequence similarity of two nodes is not dependent upon the number of nodes in the network, nor upon the connectivity of other nodes. In the following, this network is denoted as G_{S-1} , as a reminder that it contains nodes of the 5616 sequences in one yeast genome. In contrast, the calculation of Neighborhood Correlation between a pair of sequences *is* dependent upon the number of sequences in the network, and their connectivity. Three distinct Neighborhood Correlation networks result, G_{NC-1} , G_{NC-4} , and G_{NC-9} . Each represents the scores induced between all pairs of sequences in *S. cerevisiae* when the sequence similarity network is comprised of the one, four, and nine-genome datasets, respectively.

Figure 5.4 is a visual representation of the G_{NC-9} . The layout is force-based, as calculated by Neato [59], where larger edge weights (NC scores) tend toward shorter edge distances in the plotted figure. From this representation, it is apparent that the network has rich substructure, and that

Neighborhood Correlation results in a network with disjoint components. Many of these are cliques. No rigorously curated gold-standard for evolutionary families is available in yeast. As a result, the network measures are the sole basis by which the performance in yeast is evaluated in this work. However, visual inspection reveals that many of these components do correspond to groups of genes commonly considered families. For example, Actin and the seven Actin Related Proteins (ARPs) form an isolated clique, and the large cluster in the upper right-hand corner corresponds to the Kinases.

The network measures described in this chapter are used to achieve a quantitative evaluation of the yeast sequence similarity network. They are also used to evaluate the performance of Neighborhood Correlation on real data that fits a particular and typical use-case. That is, it is common to perform family classification on a smaller dataset, such as a single genome, and then expand the dataset to a larger number of genomes. Two properties are of great interest in such a scenario. First, it is desirable for the addition of data to not substantially change the output. Analysis would be very difficult if the structure (loosely, the clusters) of the network were to change with each addition or removal of a genome from the dataset. This would also cast doubt on the accuracy of the method. A second desirable property is that of increasing confidence in a result with the addition of data, since increasing the data should increase the amount of signal that may be used. Specifically, the yeast networks may be used to test the hypothesis that Neighborhood Correlation is able to perform better as data is added. Since $NC(x, y)$ effectively compares the relationship between x and other sequences with the relationship between y and other sequences, growth of the size of the network by addition of related genomes will increase the set of sequences similar to x and y , without similar increases in the strength of spurious similarity.

An assessment of transitivity of the yeast networks is presented in Figure 5.5. For each network measure, these plots illustrate the structure of the sequence similarity network G_{S-1} (in blue), and the three Neighborhood Correlation networks, G_{NC-1} , G_{NC-4} , and G_{NC-9} (in yellow, red, and green, respectively). Neighborhood Correlation is plotted with respect to the lower axis, and the bit-score of sequence similarity is plotted with respect to the upper axis. As demonstrated in the simulated data, comparison of networks is performed under the condition that network density is held constant. Here, this is accomplished by deriving a piece-wise linear scale for bit-score with respect to the NC score axis, such that the network density of G_{S-1} at bit-score threshold y has equal density to G_{NC-9} at the corresponding NC threshold on the lower axis. Intermediate positions are calculated by linear interpolation. The difference in density between the three Neighborhood Correlation networks is minimal for $NC > 0.15$, so they have not been transformed in any manner. Figure 5.5(f) illustrates the density of these networks.

Comparison of the yeast sequence similarity network with those of Neighborhood Correlation demonstrates substantial differences in the structure and transitivity of these networks. Recall that all of these networks have the same nodes. For particular horizontal position on the plot, all networks have the same number of edges. A direct measure of transitivity, the clustering coefficient, is consistently and substantially higher in G_{NC} than in G_{S-1} (Figure 5.5(e)). This demonstrates higher local connectivity for components of size three or larger. Similarly, the network component density (d) is consistently higher for Neighborhood Correlation than for sequence similarity. Taken together, these measures show that in yeast, as in simulated networks, Neighborhood Correlation increases transitivity. For all but the highest thresholds, components in G_S are less numerous (a), and more sparse than in G_{NC} . Outside of these sparse clusters, at least half of the nodes in G_S are

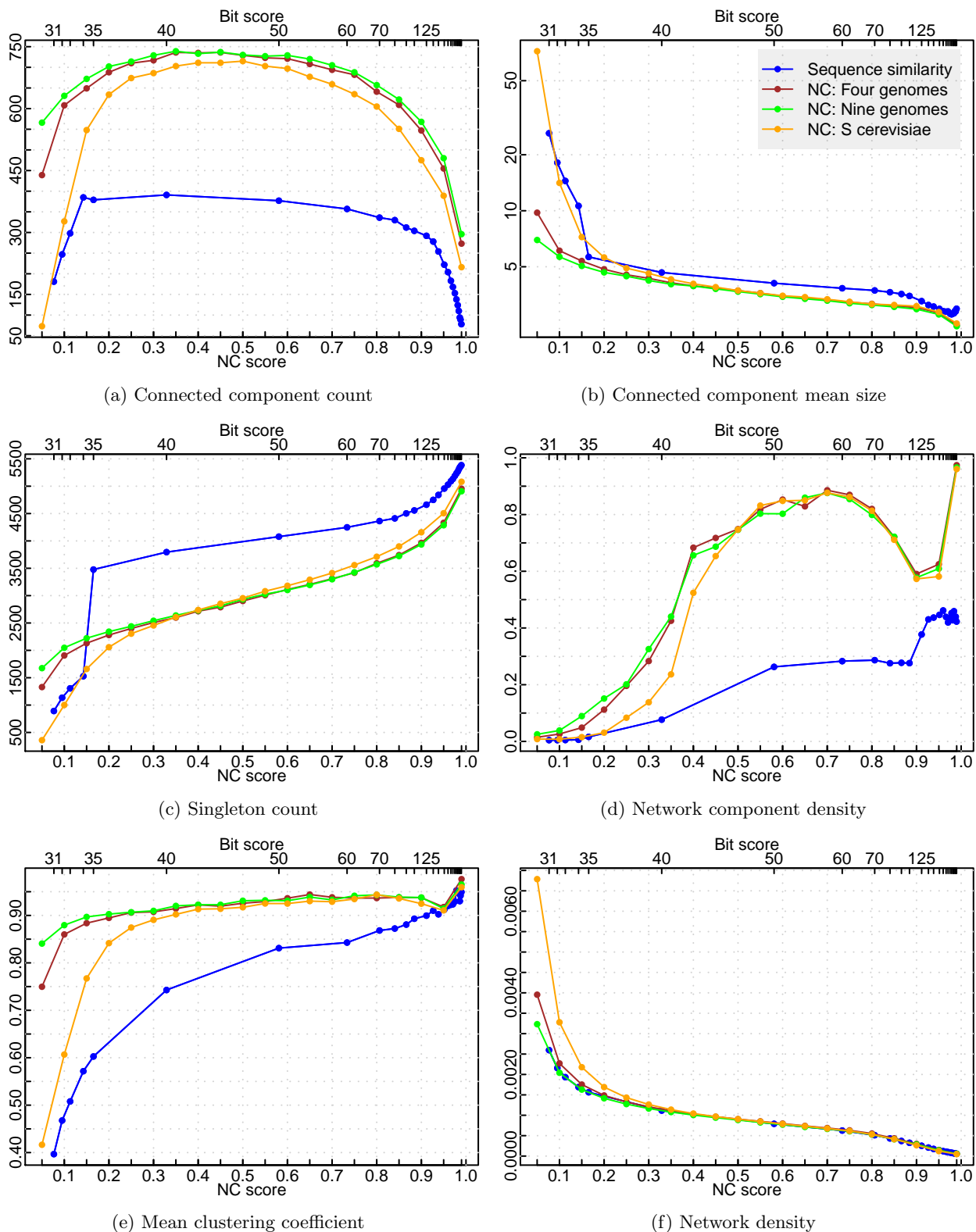


Figure 5.5: Measures of the networks comprised of the 5616 genes in *S. cerevisiae*, with edges from sequence similarity, and Neighborhood Correlation calculated with one, four, and nine yeast genomes. NC score thresholds range from 0–1. The bit-score axis ranges from 31–1000 and is aligned with NC such that the density of the sequence similarity network is equivalent to the NC network of nine genomes.

singletons, for even minimally stringent thresholds, such as bit-score greater than 34. The number of singletons rapidly increases, without the network resolving to components that are compact. Overall, this illustrates a network of structure unlike that of homology, G_H .

The overall behavior of the Neighborhood Correlation networks may be understood as follows. As more stringent thresholds are selected and edges are removed from the network, the number of connected components in G_{NC} reaches a relatively stable count between $0.15 \leq NC \leq 0.85$, and the size of these components (b) remains very stable throughout the range. (Note that (b) is plotted with a log-scale vertical axis.) The primary observation is a process of separation of the network into singleton nodes at a constant rate (c), and retention of connected components of constant size, and high density (d).

The network component density of G_{NC} decreases markedly as the threshold is changed from $NC \geq 0.75$ to $NC \geq 0.9$. First, approximately 10% of components of size two break up into singletons in this range. Since the density of a two-node component is one, loss of such pairs will substantially reduce the average value of $D(G)$. In addition, large components become sparser as the threshold becomes more stringent. For example, the largest component in the network at this range (the Kinases), decreases in density from 0.85, to 0.33. At very high stringencies, all of these networks consist primarily of singletons, two-node components, and a few very small cliques of size three or more, leading to values of $C(G)$ and $D(G)$ close to one.

The individual plots of Neighborhood Correlation demonstrate the incremental improvement yielded by the addition of related genomes to the analysis. The performance of rescoring is expected to increase as related sequences are added, because the strength and size of the common neighborhood of homologous sequences can be expected to increase as more sequences belonging to each family are added. The number and weight of edges to sequences with spurious similarity will not grow in proportion.

Comparison of G_{NC-1} , G_{NC-4} and G_{NC-9} in Figure 5.5 shows that including more genomes in the calculation of Neighborhood Correlation further increases transitivity, as measured by the network component density and the clustering coefficient. It is interesting to note that this is manifested in an increase in the number of connected components, especially at low thresholds, suggesting a network of slightly smaller, more dense components. It is reassuring to note that while the clustering coefficient and component density are higher with the addition of more genomes, the overall trends are unaffected. Moreover, most of the change is achieved when moving from G_{NC-1} to G_{NC-4} , rather than with the addition of another five genomes in G_{NC-9} . This demonstrates that the use of more data is beneficial for Neighborhood Correlation, but is not a strict necessity. It also illustrates that most of the gains are achieved early, with the addition of a small amount of data.

5.5 Analysis of human and mouse networks

The genomes of human and mouse are relatively better characterized with respect to homology, as compared to the genomes of yeast. This section examines the behavior of the network measures on two complex genomes that contain large, multidomain families. The following chapter, Chapter 6: Clustering and its evaluation (p.85), discusses the techniques and means of extrinsic evaluation using the benchmark families for evaluation in the mouse and human genomes.

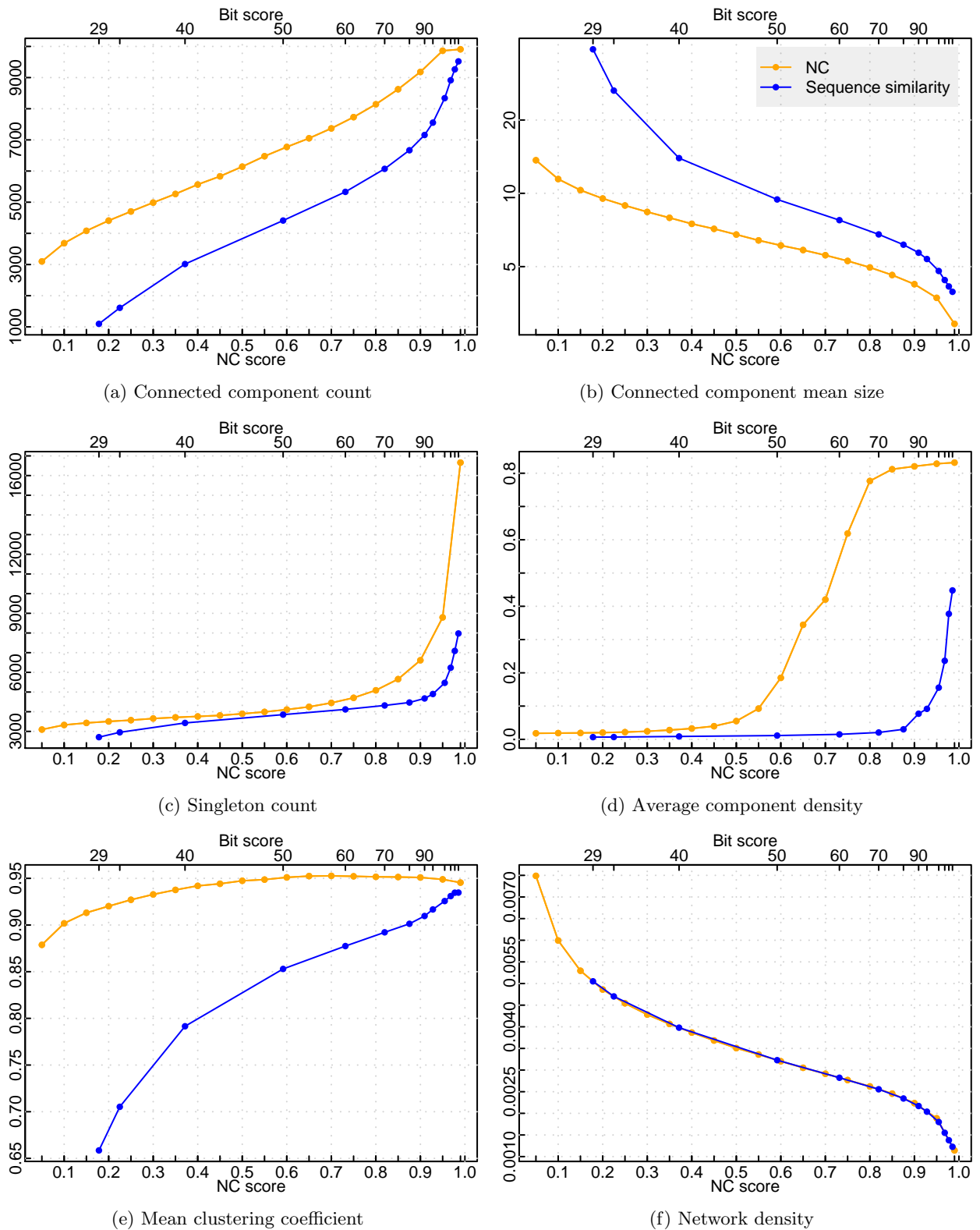


Figure 5.6: Component and transitivity measures of the sequence similarity and Neighborhood Correlation networks. These networks are comprised of all Human and Mouse sequences in the 48-genome Panther dataset, and contain 45491 sequences.

Figure 5.6 shows the network measures as applied to both the sequence similarity network (in blue) and the Neighborhood Correlation network (orange). As with the plots of yeast networks, the upper bit-score axis is aligned to the lower Neighborhood Correlation axis, such that the density of the two networks are equal at any horizontal position along these axes. The structure of these networks may be understood as follows. As compared to Neighborhood Correlation, the sequence similarity network is consistently structured of large (b), but weakly connected components (d). For sequence similarity, there are many fewer components for the same number of edges, ranging from 25% to 75% as many connected components. Throughout nearly the entire scoring range, the sequence similarity network is substantially less transitive than the Neighborhood Correlation network (e).

For a scoring metric that captures family structure, it would be desirable for the network to coalesce into a large number of dense components. For sequence similarity, this does occur to an extent; the number of connected components increases, the mean component size decreases, and the mean clustering coefficient increases. However, the consistently low network component density (d) illustrates that while a large number of small, transitive components exist in the network, there remain one or a few very large components of extremely low density. That is, a “hairball” remains amid some structure. As the bit-score threshold is increased further yet, the network rapidly degrades to one dominated by singletons. By a bit-score of 80, more than 50% of the network is comprised of singleton nodes. The components that do remain are smaller (b), but the density of these components remains low (d). The non-monotonic behavior of the sequence similarity network above a bit-score of 200 is the result of sensitivity to the few components that do not become singletons. At no point does the network resolve to a set of small, but dense components.

The Neighborhood Correlation network has structure considerably more akin to a homology network. First, the network exhibits a high transitivity of 0.90–0.95 throughout the full range of scores (Figure 5.6(e)). There are 1.5 to 4 times as many connected components than in the sequence similarity network. For most of the scoring range, the components in the Neighborhood Correlation network have half as many nodes, on average (b). The density of these components is consistently higher; more than half of the edges exist in any component for $NC > 0.75$. This occurs without rapidly degrading to singletons or minimally-sized components: until the very highest thresholds, the network is comprised of approximately 5 to 15% singleton nodes. Notably, there are no marked points of sharp inflection for any of the network measures as the threshold is changed. This suggests that the components are relatively stable within local ranges of edge weight.

The structure of the network observed for the human and mouse dataset differs from that of yeast, considered in the previous section. This can be understood in light of what is known about the domain content of these genomes. The human and mouse networks present a challenging scenario for sequence similarity, and highlight the ability of Neighborhood Correlation to recover the structure of the homology network amid greater degrees of remote homology, spurious similarity, and domain shuffling. For illustration, the combined set of the human and mouse genomes is comprised of 45,491 sequences, 19,330 of which have a domain recognized by a PFam [16] domain model (42%). Of these, 7513 sequences have more than one domain (38.9% of these, 16.0% overall). Those sequences with an identifiable domain have an average of 2.01 domains. Further, 1508 different domains are represented, with an average of 1.36 different domains per sequence containing an identifiable domain.

In contrast, 2093 of 5616 sequences in *S. cerevisiae* have an identifiable domain; of these, 565 sequences are multidomain (27.0% of these, 10.0% overall). Sequences with at least one recognizable domain have a mean of 1.39 domains, and 1.29 different domains. In total, 922 different domains are represented.

Clustering and its evaluation

The preceding chapters have discussed the challenges that multidomain families pose for successful family classification and introduce a network-based means of reasoning about homology. Chapter 3: Network rewiring (p.25) described a method to exploit the structure of the sequence similarity network, and rewire it under this framework, to yield a more accurate estimate of homology. Chapter 5: Analysis of network properties (p.65) considered the structural network features that guide this approach, and examined how these features can be used to compare a rewired network with the structure of the theoretical, unknown homology network. Ultimately, the goal of family classification is to establish a discrete partitioning of sequences into families, in which, by definition, all members share a common ancestor. This chapter describes a strategy to cluster the rewired sequence network, and details an approach to quantify the accuracy of the resulting clusters using the curated benchmark of 20 families.

The use of the term *clustering* in this dissertation refers to the partitioning of data, specifically of the nodes that comprise a network of pairwise scores. The approach undertaken here is to use clustering as a means of establishing a “final” partitioning after building a sequence similarity network, and rescoreing it using Neighborhood Correlation. These steps encode the biological data (sequence similarity) into the structure and weight of a network, and serve to enhance the signal that results common ancestry, the biological property of interest to family classification. General-purpose clustering methods are employed; that is, specific biological features of the data are captured using sequence similarity and Neighborhood Correlation, to make the data amenable as input to existing algorithms.

Clustering the rewired network serves two purposes. First, clustering establishes a discrete partitioning of the sequences, with a granularity that may be optimized in concert with evaluation. Second, it serves to refine the network that results from rewiring with Neighborhood Correlation. The process of clustering can derive additional inferential power from the structure of the network, beyond that reflected by individual network edges. As a consequence, the result of clustering can be different from that which might be obtained by simply applying a score threshold to the sequence network, and predicting families based on the connected components that arise. For a network that imperfectly represents the underlying structure of the data, application of a clustering algorithm can improve our ability to resolve that structure.

A major emphasis of this chapter is the development of metrics to evaluate clusters using ground-truth data of gene families. Collectively, these metrics are referred to as *extrinsic* measures, whereby accuracy of the result is assessed by comparison to benchmark data. These measures may be contrasted with the *intrinsic* measures discussed in Chapter 5: Analysis of network properties (p.65), which require no additional data about ground-truth. The most concrete means to evaluate family classification is validation with known families. As has been mentioned, there is a paucity of such annotated gene families, so the utility of this approach is necessarily limited. The evaluation result may be expected to be representative of the curated families in the benchmark. The 20-family benchmark, discussed further in Chapter 2: Background and preliminaries (p.13), encompasses approximately 4% of the sequences in the mouse and human genomes. These families have been selected to be representative of the variety of evolutionary histories that we believe to be typical of these genomes. In particular, the benchmark contains families with a broad spectrum of sequence conservation, families comprised exclusively of single-domain sequences, and, most targeted by this dissertation, those with sequences that are multidomain, of varied or conserved domain architectures.

Measurement of the overlap between curated families and clusters is a natural fit for Precision and Recall. Such an approach is detailed by Brohée and van Helden for a different application [23]. The empirical evaluation of the clustering result on biological data has been considered at length, as surveyed in [68]. In many cases, the evaluation approach must be tailored to the properties of the problem, and the data used to address it. A typical bifurcation is that of whether the clusters, or their labeling may overlap. For example, in studies of protein function, several proteins may share a particular function, and a single protein may have multiple functions [133]. Families are discrete, and a protein belongs to only one family, framing the clustering approach. At the same time, the available data, a set of benchmark families, covers a small fraction of the entire set of sequences in mouse and human, and a substantially smaller fraction of all 48 genomes in the Panther 7.0 dataset.

Many statistics, such as the direct application of Precision and Recall, do not address a means of evaluating partially labeled data. One typical approach is to restrict their application to datasets that are completely labeled, or, restricting the dataset to include only labeled sequences. The circumstance here is such that the curated data does not cover the entire dataset, but that each curated family labels all sequences that do belong to the family, to our knowledge. For example, consider a cluster that contains all members of a curated family, as well as other, unlabeled sequences. Because the curation procedure was designed to label all sequences that comprise a family, this clustering result would necessarily be incorrect. However, exclusion of the unlabeled sequences prior to analysis would suggest a perfect result.

Further, how may the evaluation be performed so that it captures properties of families that are not clustered perfectly, as when members are assigned between several clusters, rather than one? One approach is to consider only the most-represented, or best, cluster for a single family, though this discards valuable information about the performance of the clustering methodology. It is useful to distinguish between, for example, a case where a family is partitioned between two or more clusters, but not co-mingled with other, unrelated sequences in any cluster, from the case where that family is split between several clusters, each with many other unrelated sequences. Both cases reflect an undesirable quality, although the latter is due to a clustering result that has little correlation with families. The former may instead be a reflection of family sub-structure, and could be rectified

by selection of clustering parameters that yield a clustering with granularity that is more coarse – rather than a wholesale restructuring of the result.

The remainder of this chapter is organized as follows. First, a set of extrinsic evaluation metrics is developed to quantify the degree to which a clustering result corresponds to ground truth data as provided by a set of curated families. The second section discusses the properties that best define the suitability of a clustering algorithm to the family classification task. Finally, this chapter benchmarks the performance of the complete family classification pipeline developed in this dissertation: sequence similarity, Neighborhood Correlation, and clustering.

6.1 Evaluation metrics

The case considered here is a dataset comprised of all sequences in one or more genomes. Curated data consists of m families, where each family is a set of the sequences that comprise the family. By definition, a sequence may belong to at most one family. Not all sequences in the genomes considered are assigned to a family; the family membership of unannotated sequences is unknown. A clustering of the data consists of a discrete partitioning of all sequences into n clusters. Each sequence in the genome is in exactly one cluster.

The overlap between families is described by a contingency table, T , of m families and n clusters,

$$T = \begin{pmatrix} t_{1,1} & t_{1,2} & \cdots & t_{1,n} \\ t_{2,1} & t_{2,2} & \cdots & t_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ t_{m,1} & t_{m,2} & \cdots & t_{m,n} \end{pmatrix}. \quad (6.1)$$

Each value, $t_{f,c}$ in this matrix is the number of sequences that exist in both family f and cluster c .

Marginal sums may be defined as follows. Let M_f be the size of family f , or

$$M_f = \sum_{c=1}^n t_{f,c}. \quad (6.2)$$

Let N_c be the size of cluster c . Note that N_c is *not* the sum of a column in T , because clusters may contain sequences that are not annotated as belonging to a curated family. The value O_C is defined as the count of all annotated sequences within the cluster c , or

$$O_c = \sum_{f=1}^m t_{f,c}. \quad (6.3)$$

The Precision, P is the fraction of sequences in each cluster that are also members of a family. That is, the elements of the matrix P , for family f and a cluster c , are

$$p_{f,c} = \frac{t_{f,c}}{N_c}. \quad (6.4)$$

Similarly, the Recall, represented by the matrix R is the fraction of sequences in each family that are found in a cluster, with elements

$$r_{f,c} = \frac{t_{f,c}}{M_f}. \quad (6.5)$$

The F-statistic, F , is the harmonic mean of Precision and Recall, and is a measure of the compromise that may be obtained between maximizing each. Elements of the matrix F , for a family f and cluster c , are defined as

$$f_{f,c} = \frac{2p_{f,c}r_{f,c}}{p_{f,c} + r_{f,c}}. \quad (6.6)$$

One goal here is to consider performance per family, over all clusters that represent it. Family-wise aggregates of Precision, Recall, and the F-statistic are calculated as the mean of the corresponding statistic, weighted by the representation of the family in a particular cluster. In these calculations, it is useful to enumerate the clusters that contain at least one member of the family f . The set of such clusters is defined as

$$U_f = \{c \in \{1..n\} \mid t_{f,c} > 0\}. \quad (6.7)$$

The **family-wise Precision**, P_f , for a family f is then defined as the weighted average of $P_{f,c}$ over all clusters, or

$$\begin{aligned} P_f &= \frac{1}{M_f} \sum_{c \in U_f} t_{f,c} P_{f,c} \\ &= \frac{1}{M_f} \sum_{c \in U_f} \frac{(t_{f,c})^2}{N_c}. \end{aligned} \quad (6.8)$$

The **family-wise Recall**, R_f , for a family f is similarly defined:

$$\begin{aligned} R_f &= \frac{1}{M_f} \sum_{c \in U_f} t_{f,c} r_{f,c} \\ &= \frac{1}{(M_f)^2} \sum_{c \in U_f} (t_{f,c})^2. \end{aligned} \quad (6.9)$$

Finally, the **family-wise F-statistic**, F_f is

$$F_f = \frac{1}{M_f} \sum_{c \in U_f} T_{f,c} F_{f,c}. \quad (6.10)$$

The family-specific measures capture the classification performance on individual families. The mean of the family-wise statistic, weighted by the size of the each family, gives an aggregate performance measure over multiple families. The **overall Precision, Recall, and F-statistic** of a set of families, S , are defined as follows.

$$P_S = \frac{\sum_{f \in S} M_f P_f}{\sum_{f \in S} M_f}. \quad (6.11)$$

$$R_S = \frac{\sum_{f \in S} M_f R_f}{\sum_{f \in S} M_f}. \quad (6.12)$$

$$F_S = \frac{\sum_{f \in S} M_f F_f}{\sum_{f \in S} M_f}. \quad (6.13)$$

6.2 Clustering methodology

The purpose of family classification is to partition sequences, where all members of each cluster are homologous. The approach taken here is to use the result of Neighborhood Correlation, G_{NC} as input. The problem then resolves to that of partitioning a sequence network, where the edge weight between two sequences is presumed to be an accurate estimate of homology. Clustering is used to establish a discrete partitioning of the sequences in the network, and to compensate for errors that remain in the network.

The aim of this work is not to design a new clustering algorithm. Instead, this dissertation separates an explicit step of resolving biological signal with Neighborhood Correlation, from a final step of clustering. A wide variety of clustering algorithms exist. In choosing an algorithm, it is useful to again consider the theoretical structure of the homology network, and the structure of the network that results from Neighborhood Correlation. The true homology network is comprised of one clique for each family. Modulo any errors, the Neighborhood Correlation network is comprised of components that each correspond to a family, with many edges and strong weights, joined by fewer, and weaker edges to other components. The quality most needed in a clustering algorithm is to extract the dense components that represent families without falsely joining families because of spurious network edges.

Several properties of the application here guide selection of the class of clustering algorithm to be used. Many clustering methods are based upon a euclidean feature space, and cluster upon those features. Implicitly, a Euclidean distance may be calculated in such a context, but a majority of clustering algorithms depend upon the embedding, and a matrix may not be substituted. Of clustering algorithms that are not inherently reliant upon a feature space, many of their implementations preclude use of a matrix. For this application, the data is in the form of a pairwise matrix, requiring a method that can use this directly as input. This greatly constrains the choice of clustering algorithms suitable for this problem.

All clustering algorithms have one or more parameters that determine the granularity of the result; i.e., that determine whether many, small clusters are returned, or whether fewer, large clusters result. In this problem, the correct number and size of clusters is unknown. It is useful, generally, and especially here, for the clusters that result at coarse granularity to be a super-set of the clusters produced by parameter values that specify a result that is more fine-grained. This nesting of the clusters at varying parameters can be used to fine-tune the result, without broad changes in the structure of the clustering. Considered another way, the partitioning established with parameter values that yield a coarse clustering is stable as more stringent values are applied.

Here, it can be recognized that gene family evolution is inherently the result of a hierarchical evolutionary process. As a consequence, the use of a clustering algorithm based upon a hierarchical model is a good conceptual match. There are a number of additional practical benefits for hierarchical output when working with gene families. While the tree that results from the clustering may not directly correspond to the phylogeny of a family, its structure can be used to better understand family substructure. For the development of methodology, it is also extremely beneficial to be able to directly examine how and why the partitioning of the sequences would change when the hierarchical tree is cut above or below a given threshold.

6.2.1 Agglomerative, hierarchical clustering

The general procedure of agglomerative, hierarchical clustering algorithms is to initialize a set of N clusters that each represent a single element – here, a sequence. Clustering then proceeds by iteratively merging the nearest pair of clusters, and repeating, until only a single cluster remains. The result is a nested hierarchy of clusters that may be represented by a binary tree. Leaves consist of the singleton clusters that each represent one element. Every other, internal, node represents the merger of two child clusters. Each node is associated with the distance between the two child clusters that were merged into a single cluster by that node. As a result, the hierarchy may be *cut* with respect to this value of distance, to yield a partitioning of sub-trees, each of which includes a distinct set of sequences.

This methodology is predicated upon a measure of distance (or similarity) between individual elements, as well as between the sets of elements represented by clusters. A sequence network, or similarity matrix of all pairs of sequences, necessarily encodes the measure between sequences. However, neither this, nor the algorithm framework above, specifies how one might calculate the distance between two clusters. A wide array of agglomerative hierarchical clustering methods have been proposed, using a variety of measures of the calculation of closeness between clusters. Day and Edelsbrunner [40] review how these methods reduce to a single algorithm, differing only in the definition of distance between clusters, and propose an efficient algorithm that is common to all approaches.

Depending upon the structure of the input data, the choice of the distance measure to be used between clusters can have a significant impact upon the clustering result. First, consider one of the most direct, and well-studied approaches to defining the distance: the single-linkage, or nearest-neighbor approach [53]. Here, the distance between two clusters is defined to be the shortest distance between any pair of sequences where one sequence is in each cluster. Use of the shortest distance represents an extreme, in that a single distance between pairs defines the distance between two clusters, irrespective of the shape of those clusters. Conceptually, the single-linkage approach will not distinguish between the merger of two clusters in the sequence network that have a single edge between them, from two clusters that each comprise half of the nodes in a single clique. A cut of the hierarchy produced via single-linkage at a given closeness threshold is exactly equivalent to identification of connected components in the input sequence network, when only network edges above the same threshold are retained.

An opposite extreme is specified by the complete-linkage, or furthest-neighbor, approach. Here, the distance between two clusters is defined to be the maximal distance between any sequence in one cluster, and any sequence in the other cluster. Again, this measure bases the distance between two clusters upon a single distance between pairs. Here, however, the shape of two clusters with respect to one another does impact the distance between the clusters. Because the maximal distance between sequences in different clusters is considered, this approach favors the merger of clusters only when the result is compact, and the extremes of distance within the cluster are minimal.

A variety of other distance definitions incorporate the distances of other pairs of sequences between two clusters. The average-linkage, or UPGMA, approach defines the distance between clusters as the mean of all input pairwise distances from any sequence in one cluster to any sequence in the other cluster; i.e., the mean over all pairs of sequences that span the two clusters [132]. This is subtly distinct from the mean of all pairs of sequences in the resulting, merged cluster.

The computational complexity of agglomerative, hierarchical clustering is large. Day and Edelsbrunner [40] showed that all methods for pairwise cluster distance can be implemented asymptotic run-time of $O(n^2 \log n)$ and space of $O(n^2)$. The single-linkage method may be implemented with $O(n^2)$ run-time, and $O(n)$ space, by taking advantage of the property that the distance between two merged clusters and all other clusters is the lesser of the two original distances [130]. For the same reasons, the complete-linkage method may be implemented in $O(n^2)$ run-time and $O(n)$ space [42].

When the input is a matrix, the average-linkage method may be computed in $O(M \log N)$ time and $O(M)$ space, where M is the number of edges in the network, or distances in the sparse input matrix, and N is the number of sequences [96]. Further, Loewenstein *et al.* developed an algorithm, coined MC-UPGMA, that computes the average-linkage result by incrementally loading the matrix into memory. This facilitates efficient parallel computation, as well as decreased memory requirements. They also provide an efficient implementation, which is used for average-linkage clustering in this dissertation.

6.3 Results

Agglomerative hierarchical clustering was performed for the 48-genome, 600k sequence dataset used throughout this dissertation. Three variants of cluster distance were used for this hierarchical clustering: single, average, and complete-linkage. As described in the previous section, the single-linkage method is equivalent to identifying the connected components that remain in the network comprised of all edges above a chosen score threshold. When the resulting clusters are compared to families, this is, therefore, a direct reflection of the degree to which the edge weights represent homology. As a consequence, this is expected to be suitable only when the network weight and structure is an accurate estimate of homology. By contrast, the complete-linkage method is particularly well-suited to a network comprised exclusively of dense components, or cliques.

With the average-linkage method of calculating cluster distance, multiple edges are considered between pairs of clusters. The intended result is that the network structure is refined, to yield a more accurate partitioning with respect to homology than any single edge would represent. It should be more robust to spurious, or a few missing edges. Height in an average-linkage tree is a mean of weights between the two children of a cluster. As a consequence, this height is a transformation of the original score, and is not equivalent to selection of a single edge threshold in the network.

The accuracy of clustering has been evaluated using the benchmark dataset of 20 curated families. The metrics developed earlier in this chapter are relevant to a partitioning of all sequences in the human and mouse genomes, because the benchmark comprises a labeling of family members that exist in these genomes.

Two fundamentally different sets of data are considered as input to the clustering applied here. First, and most direct, the set of sequences in the human and mouse genomes is used as input to the clustering algorithm. The input score matrix consists of pairwise scores between sequences in these genomes, as calculated by either sequence similarity, or Neighborhood Correlation. The hierarchical clustering tree then consists only of sequences in human and mouse.

An alternate approach, considered later in this chapter, is to construct a hierarchical clustering tree that consists of *all* sequences in the 48-genome dataset, with a score matrix of all pairwise

relationships between the 600k sequences. Here, the clustering result is a tree with 600k leaves, each corresponding to a sequence. The partitioning of human and mouse sequences may then be considered by extracting the subset of sequences in these genomes, but using the structure induced by the complete tree. Similar to the dependence of Neighborhood Correlation upon other sequences in the network, the structure that results from clustering can depend upon other sequences in the input, even if these are not members of the subset of human and mouse sequences that are used as the basis for evaluation.

The following discussion is organized to first introduce the evaluation performed, by considering a clustering of the sequence similarity network. This methodology is then employed to demonstrate a comparison with the result of Neighborhood Correlation.

6.3.1 Sequence similarity

Figure 6.1 shows the performance of clustering the sequence similarity network using the single-linkage measure of inter-cluster distance. This figure depicts an evaluation based upon the entire benchmark (a), as well as evaluation using the set of all families, excepting kinase (b). Because the granularity of the clustering becomes more fine left-to-right in these plots, Recall necessarily decreases monotonically, and Precision increases monotonically.

Because the single-linkage method is equivalent to partitioning the network using an edge weight threshold, the bit-score thresholds discussed here are directly comparable to the edge weights described in Chapter 5: Analysis of network properties (p.65), as well as the score histograms in Chapter 3: Network rewiring (p.25).

On the ALL set of families, in Figure 6.1(a), it can be observed that clustering the sequence similarity network using the single-linkage method does not yield particularly good performance, at any score threshold. As the hierarchical clustering tree (and, equivalently, the network) is partitioned into smaller clusters, the Precision increases markedly in the range of 41 – 56. Those smaller clusters mix fewer family and non-family members. However, this threshold range also displays a sharp decrease in Recall, such that, overall, the one or more clusters that represent a family each contain a very small fraction of the family. (Remember that these measures are weighted averages over all clusters that contain a family member.) When both measures are considered using the F-statistic, it is clear that no single threshold can yield good performance, and F never exceeds 0.35. When the kinase family is excepted (b), Recall decreases more gradually. This result may be understood because kinase, as a very large family, is more prone to being split among several clusters, each of which represent a small fraction of the family. Overall, an F-statistic of approximately 0.6 is achievable for the bit-score range of 41 – 56.

The performance of clustering the sequence similarity network using the average-linkage method is considered in Figure 6.2. As compared to the single-linkage method, this greatly increases the performance of clustering the sequence similarity network. A substantial fraction of this improvement may be attributed to more accurate clustering of the kinase family. The maximum F-statistic achieved on the ALL dataset is approximately 0.8 (a), as compared to 0.35 for the single-linkage method. The ALL-kin dataset (b) exhibits a corresponding increase from approximately 0.61 to 0.83.

Here, the thresholds reported are the distance threshold used to laterally cut the hierarchical

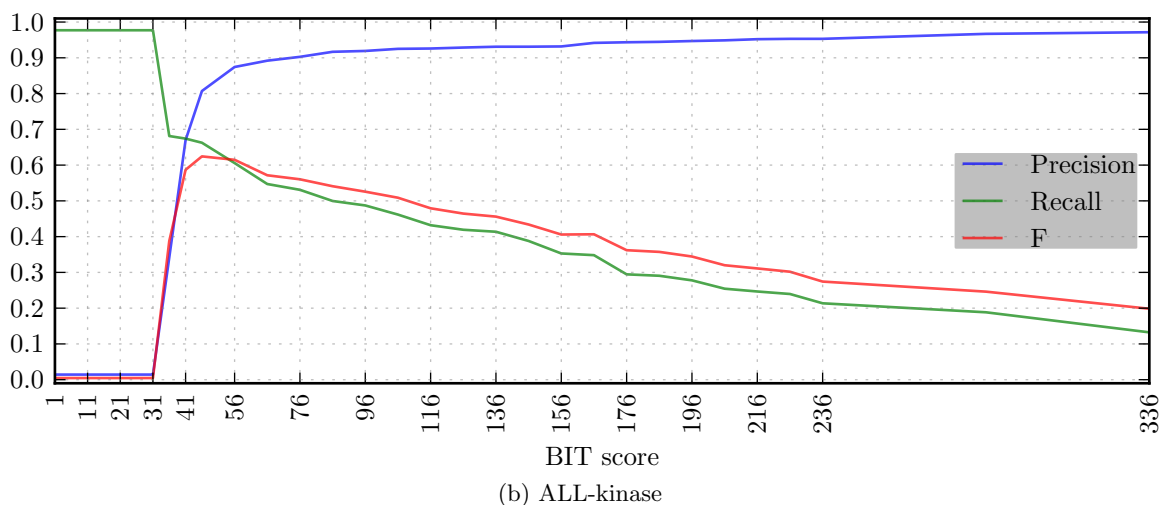
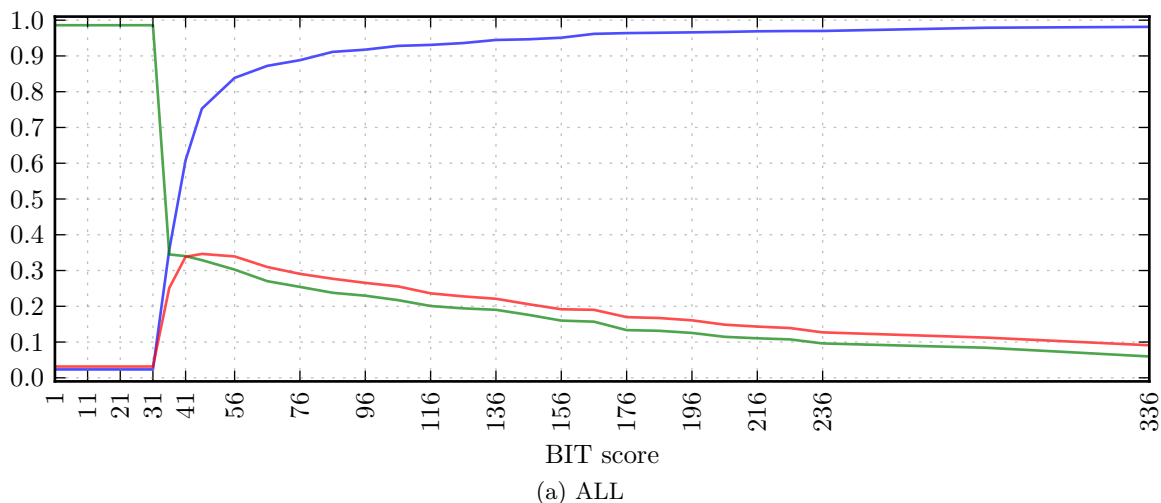


Figure 6.1: Performance of hierarchical clustering of sequence similarity, using the single-linkage method. This figure depicts the Precision, Recall, and F that result from selection of the partitioning induced by cutting the tree at a range of cluster distance thresholds.

clustering tree. The distances between children in this tree are derived from the mean bit-score of edges between clusters, *not* the bit-score of any particular edge, or the mean score of all edges in the merged cluster. As a consequence, the average-linkage threshold here is not directly comparable to the bit-score threshold of the corresponding sequence similarity network, nor with the score histograms in Chapter 3: Network rewiring (p.25).

The preceding figures considered the aggregate performance over all families in the curated benchmark. It is illustrative to consider performance on individual families, to better understand the clustering result. Figure 6.4 depicts a heatmap of the F-statistic for each of the 20 families, as well as the two aggregate datasets already considered. Color in this heatmap represents the value of the F-statistic, ranging from 0 (blue) to 1 (red). Refer to Figure 6.3 for a color legend that corresponds to all heatmaps in this chapter. These families are ordered top-to-bottom roughly in terms of complexity. The families ACSL–WNT are single-domain families. The families DVL–USP are mul-

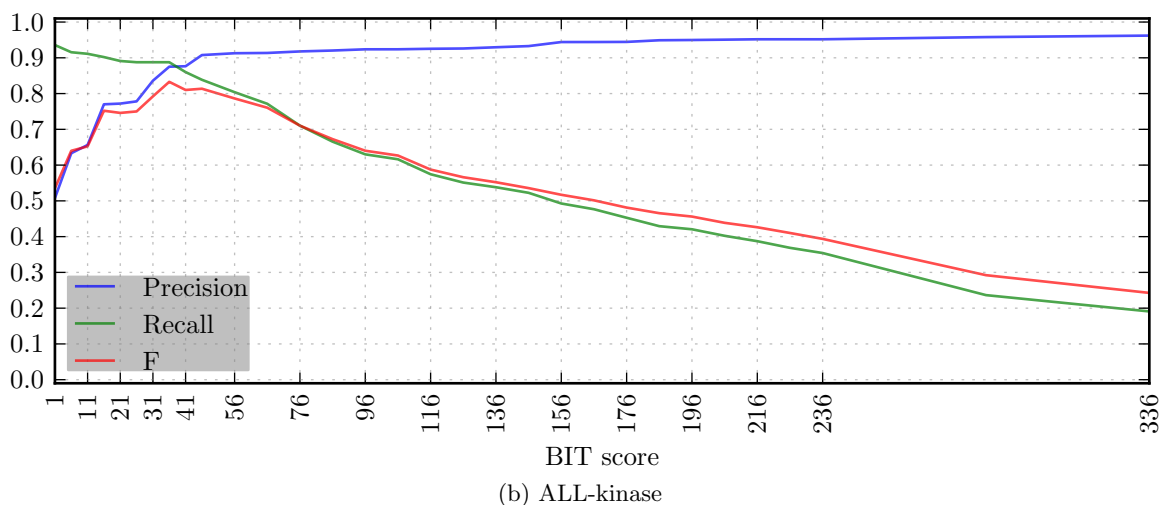
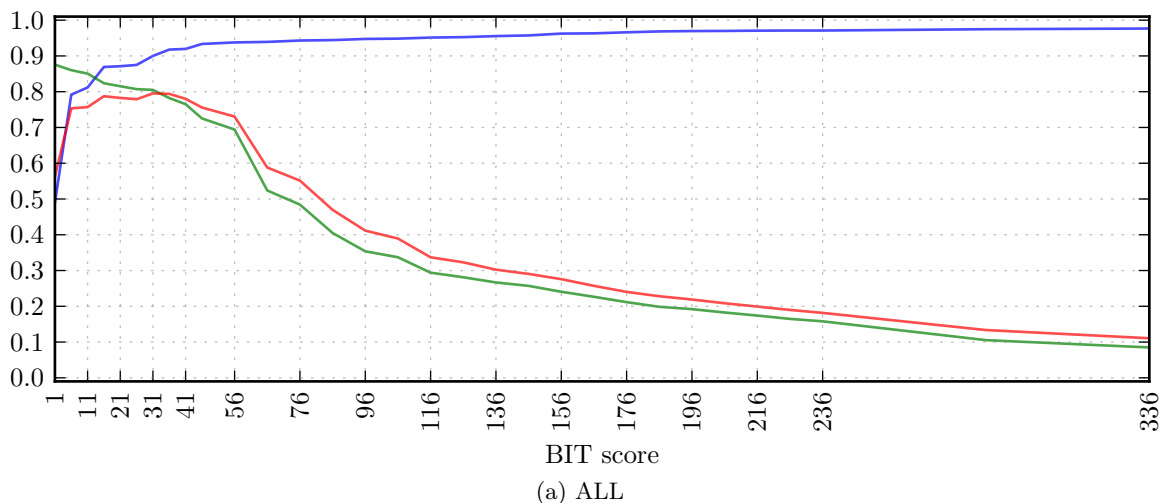


Figure 6.2: Performance of hierarchical clustering of sequence similarity, using the average-linkage method. This figure depicts the Precision, Recall, and F that result from selection of the partitioning induced by cutting the tree at a range of cluster distance thresholds.

tidomain families in which the domain architecture of all sequences are relatively constant. These families contain very few INCONSISTENT domains. Finally, the families ADAM-TNFR are multidomain families with variable domain architectures. These contain many INCONSISTENT domains. In addition to the influence of domain architecture, sequence divergence can have a substantial impact upon the family classification result. The families, FGF, TNF, TNFR, and USP each have a particularly high degree of sequence divergence.

Clustering by single-linkage is highly dependent upon how well the pairwise scoring metric reflects homology. Figure 6.4 illustrates that application of single-linkage clustering to the sequence similarity network can achieve good performance, for a few families. These families are most consistent with the observation that single-domain families, and those with highly-conserved multidomain architectures, are a good fit for the use of sequence similarity as a proxy for homology. In particular, note that for most families in the top half of the single-linkage plot, there exists a threshold that

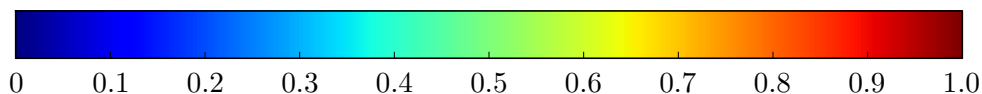


Figure 6.3: Color legend for all heatmaps in this chapter

can achieve a good clustering. However, as observed in Chapter 3: Network rewiring (p.25), no single threshold can yield good performance on all of these families.

As expected from the performance of average-linkage clustering on the aggregate dataset, Figure 6.4 demonstrates that the application of average-linkage clustering greatly improves the partitioning of families. In particular, the performance on single-domain families remains high, and the optimal threshold range for each becomes considerably more broad. Additionally, these threshold ranges coincide for a majority of single-domain families. The most notable performance gains are found with multidomain families. While a particular threshold may be used to partition each family with relatively high accuracy, the optimal threshold is highly specific to each family.

In addition to the single and average-linkage methods of inter-cluster distance, Figure 6.4 shows the performance achieved by using the complete-linkage method. Its performance is nearly identical to that of single-linkage, when applied to sequence similarity networks.

Figures 6.5 and 6.6 show the Precision and Recall, respectively, for the benchmark of families. These may be used to better understand which factors play into the F-statistic shown in Figure 6.4.

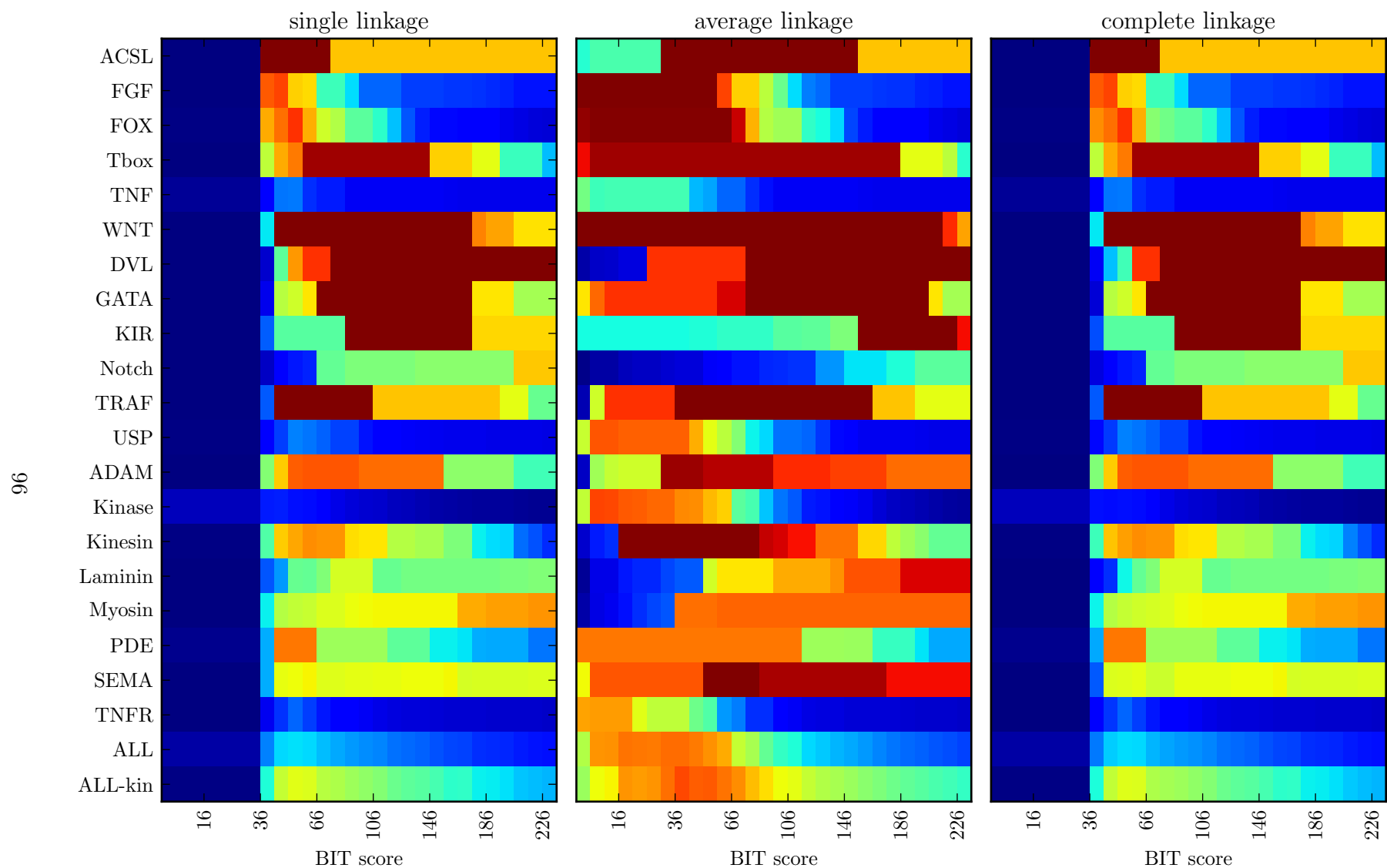


Figure 6.4: Heatmap of the F-statistic for all 20 benchmark families. Hierarchical clustering, by the single, average, and complete linkage methods has been performed for the set of mouse and human sequences using bit-score.

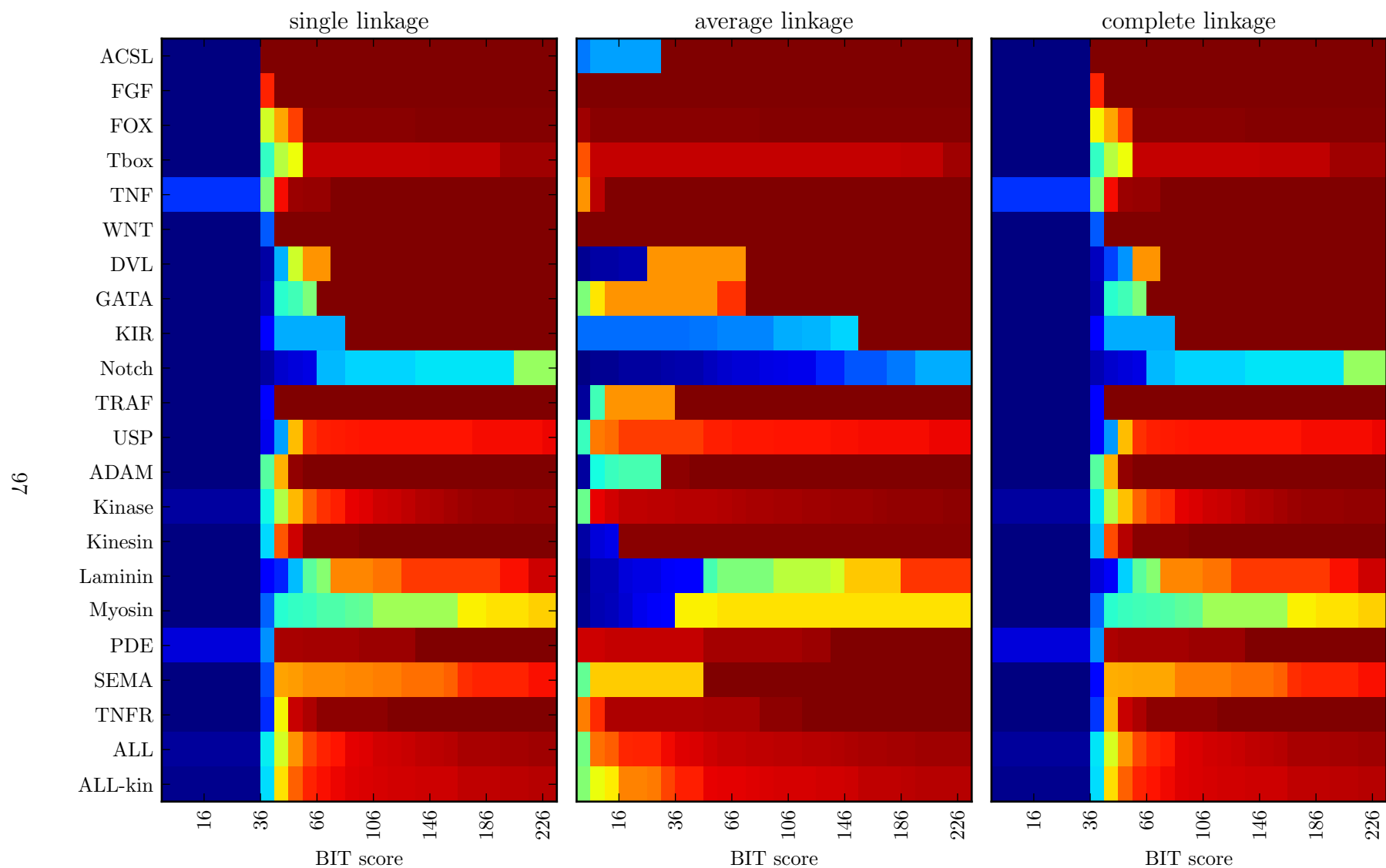


Figure 6.5: Heatmap of Precision for all 20 benchmark families. Hierarchical clustering, by the single, average, and complete linkage methods has been performed for the set of mouse and human sequences using bit-score.

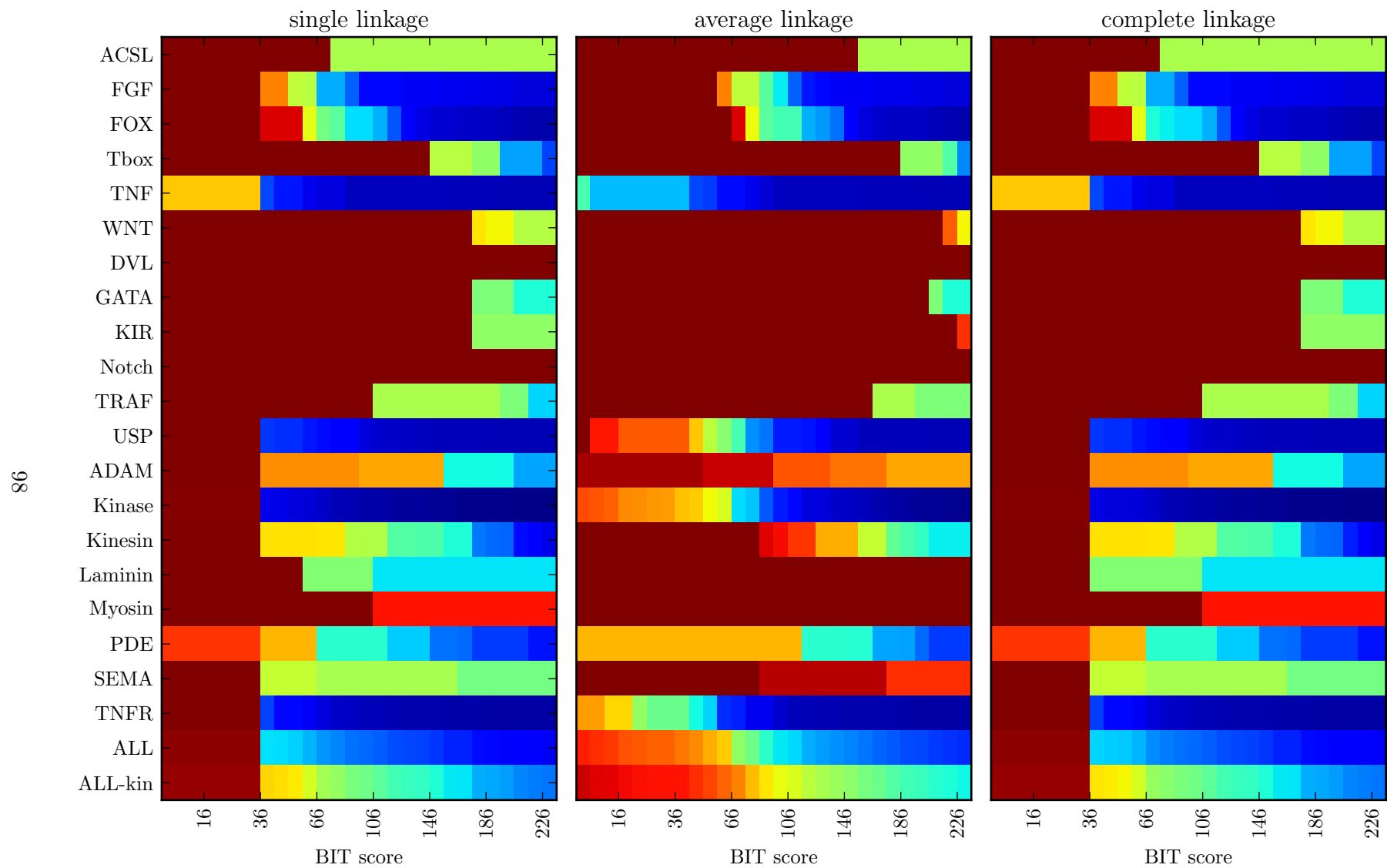


Figure 6.6: Heatmap of Recall for all 20 benchmark families. Hierarchical clustering, by the single, average, and complete linkage methods has been performed for the set of mouse and human sequences using bit-score.

6.3.2 Neighborhood Correlation

The family classification result of Neighborhood Correlation is examined using a methodology that mirrors that used to consider sequence similarity, in the preceding subsection. First, Figure 6.7 shows the performance of the single-linkage on the Neighborhood Correlation network of human and mouse sequences. Here, it is notable that Neighborhood Correlation scores are a very accurate estimate of homology for the benchmark of 20 families. On both the ALL (a) and ALL-kin (b) aggregate sets of families, the F-statistic is greater than 0.74. Notably, the maximum F-statistic is highest for the ALL dataset, suggesting that many of the increases in performance are the result of better estimates of homology on the kinase family, a particular challenge to sequence similarity. These observations are consistent with the increases in local network density observed in Chapter 5: Analysis of network properties (p.65).

The application of the average-linkage method to the Neighborhood Correlation network yields further gains in clustering performance. On both aggregate sets, the maximum F-statistic is approximately 0.85 (Figure 6.8). Notably, the optimal choice of Neighborhood Correlation threshold broadens significantly in this figure, with an F-statistic greater than 0.7 throughout the range of $0.025 \leq \text{NC} \leq 0.70$ for the ALL set (a) and $0.15 \leq \text{NC} \leq 0.725$ for the ALL-kin set (b). This suggests that selection of a threshold within a broad range will yield accurate clusters of families.

Figure 6.9 shows the F-statistic for individual families. Here, with the exception of the TNF family, it is evident that Neighborhood Correlation yields an accurate estimate of homology for single-domain families, and that the application of average-linkage clustering serves to broaden the range thresholds that will perfectly partition these families.

As compared to sequence similarity, Neighborhood Correlation yields good estimates of homology for multidomain families as well. In particular, the single-linkage plot of Figure 6.9 indicates that a threshold of NC 0.85 partitions most of the multidomain families with high accuracy, and, consequently, yields good performance on the aggregate of all families. It is interesting to note that although we developed Neighborhood Correlation specifically to address the domain chaining problem, Neighborhood Correlation exhibits great improvement over sequence similarity in cases of remote homology. This is evidenced by increased performance on highly diverged families such as TNF and USP.

The application of the complete-linkage method to the Neighborhood Correlation network yields a result that differs from the single-linkage method. Overall, the result is a broadening of the optimal scoring threshold. The optimal F-statistic on the ALL dataset decreases from that observed with single-linkage, while that of ALL-kin increases. While the complete-linkage method would not be preferred over the average-linkage method, here, its behavior is illustrative of the structure that results from Neighborhood Correlation. The complete-linkage method is particularly well suited toward identifying cliques in a network. That the optimal score ranges decrease for complete-linkage, with respect to single-linkage, suggests that the cliques and dense components in the Neighborhood Correlation network are connected, on average, by much weaker edges than are necessary to connect families.

Figures 6.10 and 6.11 show heatmaps of the Precision and Recall, respectively, of clustering the Neighborhood Correlation network.

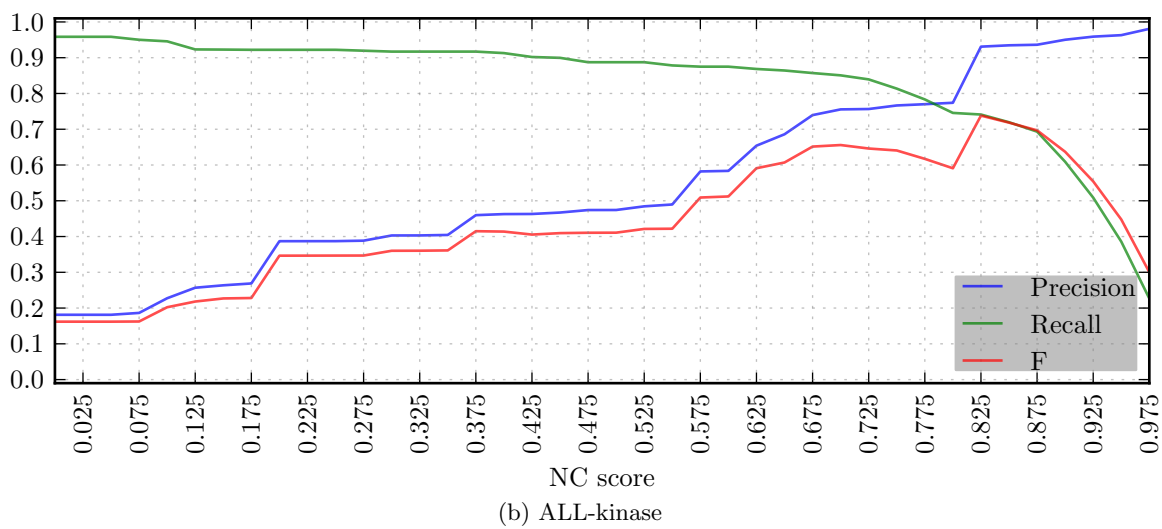
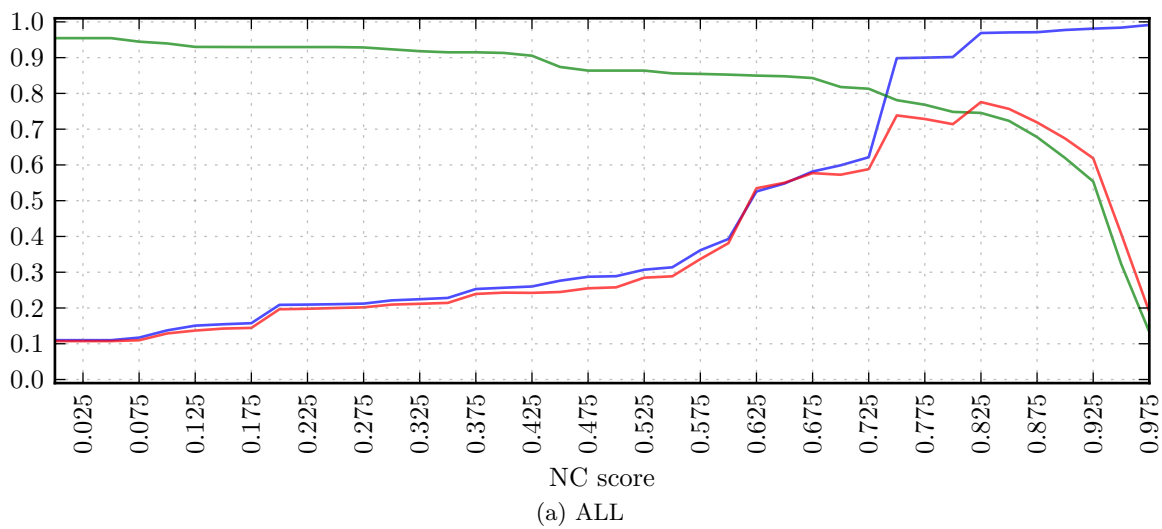


Figure 6.7: Clustering performance of single-linkage clustering of the Neighborhood Correlation network.

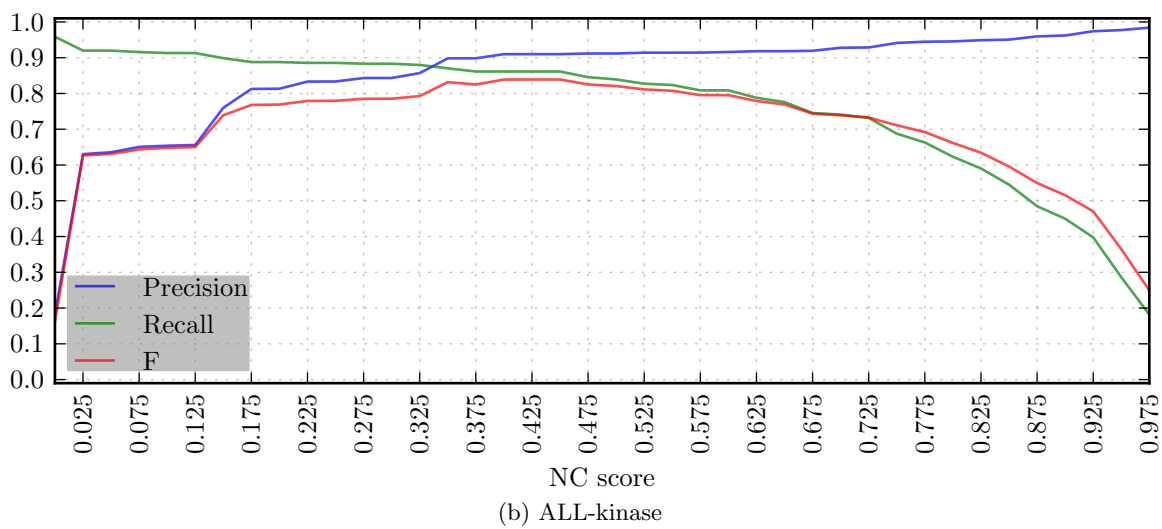
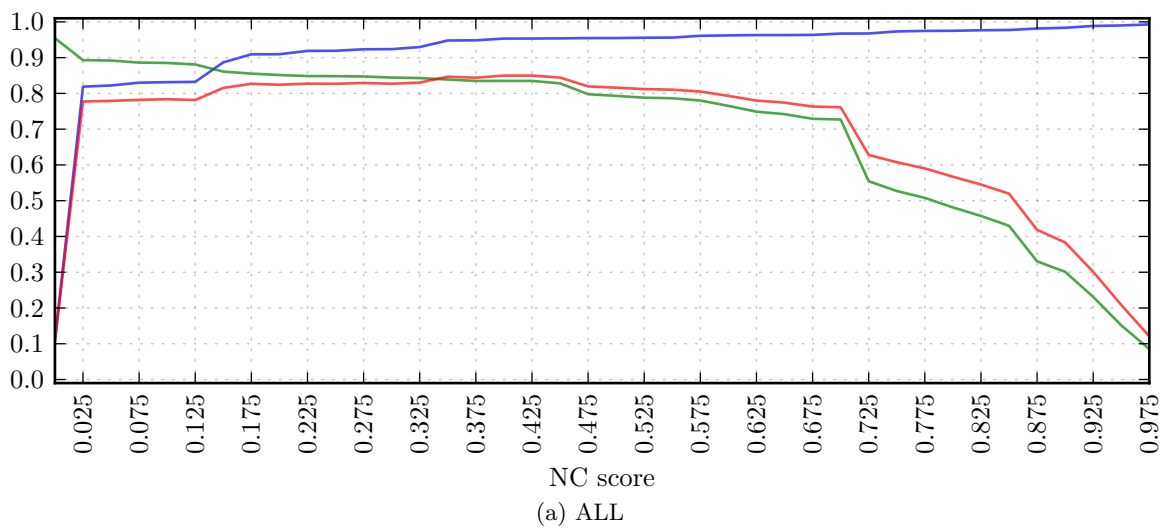


Figure 6.8: Clustering performance of average-linkage clustering of the Neighborhood Correlation network.

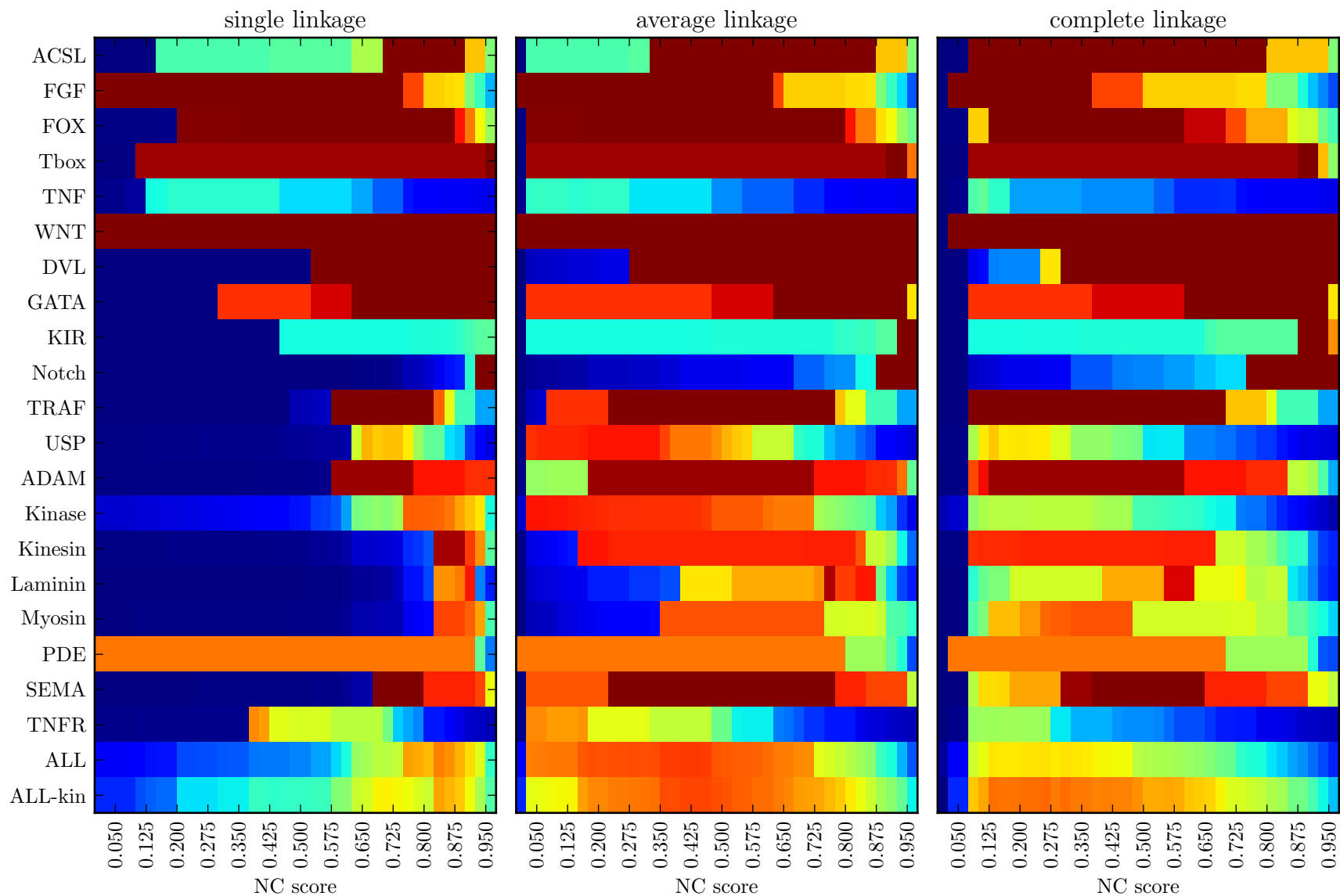


Figure 6.9: Heatmap of the F-statistic for all 20 benchmark families. Hierarchical clustering, by the single, average, and complete linkage methods has been performed for the set of mouse and human sequences using Neighborhood Correlation scores computed with the 48-genome dataset.

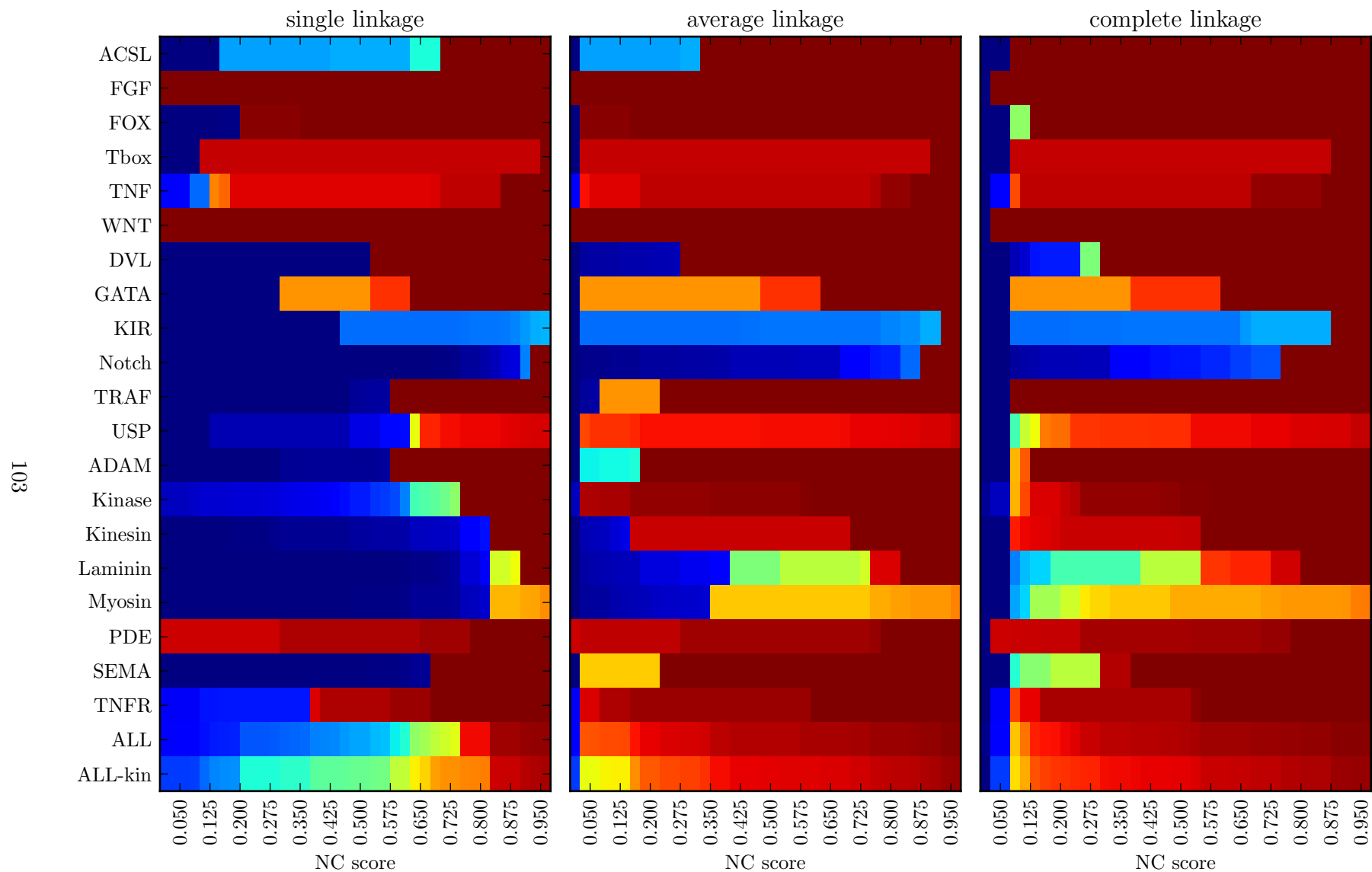


Figure 6.10: Heatmap of Precision for all 20 benchmark families. Hierarchical clustering, by the single, average, and complete linkage methods has been performed for the set of mouse and human sequences using Neighborhood Correlation scores computed with the 48-genome dataset.

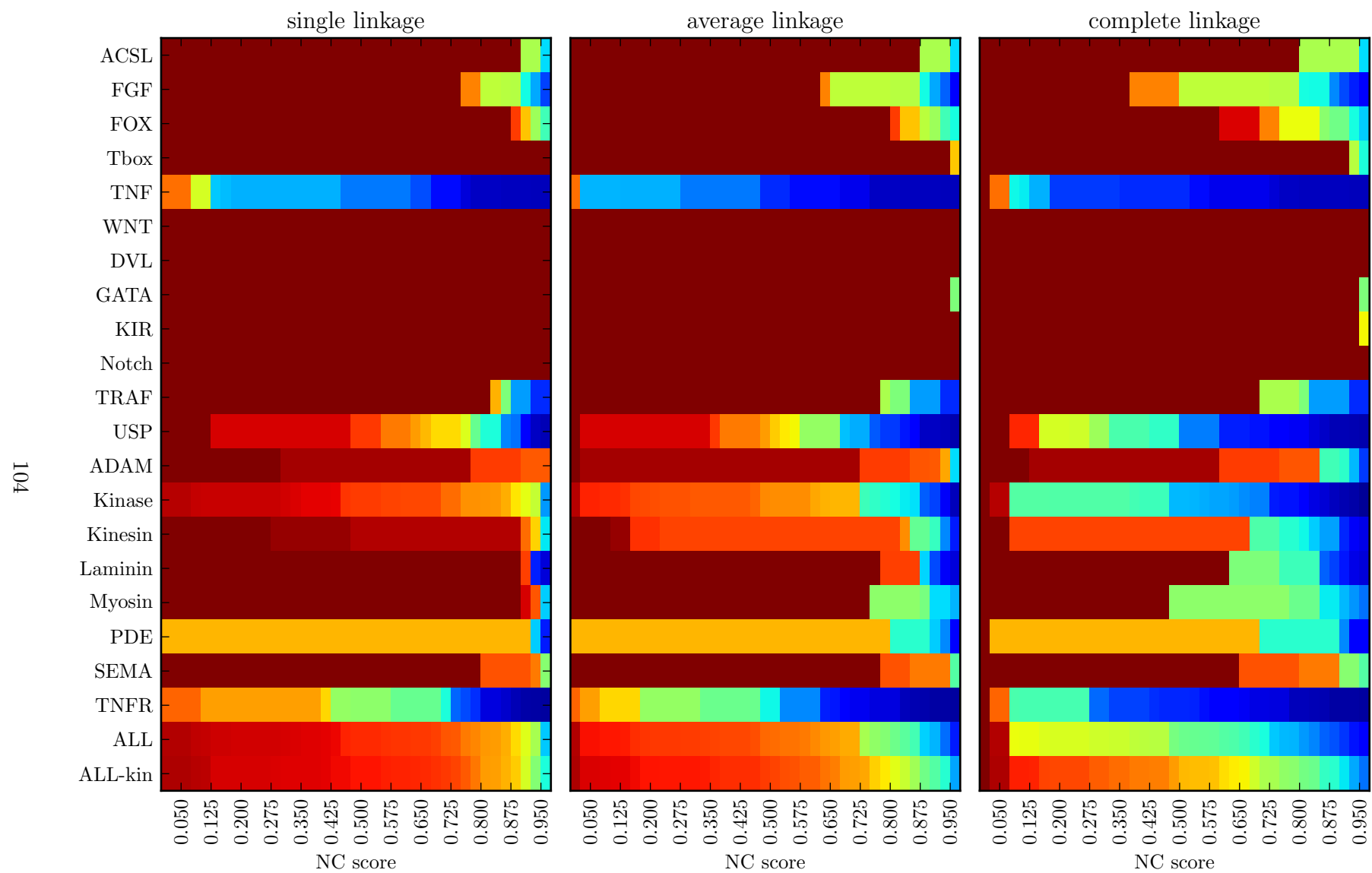


Figure 6.11: Heatmap of Recall for all 20 benchmark families. Hierarchical clustering, by the single, average, and complete linkage methods has been performed for the set of mouse and human sequences using Neighborhood Correlation scores computed with the 48-genome dataset.

6.3.3 Influence of additional data

The preceding analyses considered the hierarchical tree that results from clustering the dataset comprised of the mouse and human genomes. Because the curated benchmark includes only sequences from mouse and human, this provides a natural basis for evaluation. An alternate scenario can be considered where additional genomes are clustered concurrently, yielding a tree with leaves from mouse, human, and several other genomes. Evaluation is then performed using the curated mouse and human sequences, by examining the structure induced upon the subset of leaves that are mouse and human sequences.

The analysis provides an opportunity to examine any differences that result from clustering when other sequences are added to the dataset. Most notably, is the clustering of the curated families stable when other family and non-family members are added to the dataset? Beyond stability, the addition of sequences could be expected to decrease clustering accuracy, as if intermediate sequences lead to the merger of families, or increase it as more data strength correct relationships and reduce the effect of spurious network edges.

An effective test of the behavior of the result as more data is either added or removed, is, of course sensitive to *which* sequences are included in the dataset. That is, a test of stability can be evaluated through sampling, but the test can be highly dependent upon the sampling process employed. For genome data, the structure of families is largely uncharacterized, so it is difficult to devise an approach to sample individual sequences from one or more genomes. For example, uniformly random selection of sequences in a dataset is particularly prone to inaccurate representation of small families, particularly because their size relative to other families may differ greatly in the discrete case. The addition (or exclusion) of complete genomes is a natural sampling of the sequence space. Additionally, the inclusion several of complete genomes mirrors the approach taken for most large-scale analyses.

The remainder of this section considers average-linkage clustering of all 48 genomes in the Panther 7.0 dataset, comprised of approximately 603k sequences. Figure 6.12 shows the Precision, Recall, and F-statistic for the clustering of 48 genomes using sequence similarity. Compared to the corresponding performance when only the sequences of mouse and human are clustered, shown in Figure 6.2, it is evident that the performance is nearly identical. The maximum F-statistic, and the range of scores over which it is achieved, are effectively identical, for both the ALL and ALL-kin benchmark sets. The most notable difference in each is a more smooth behavior of Precision as the bit-score threshold is increased from minimal values. This data illustrates that the addition of 46 more genomes to the data set of mouse and human sequences does not alter the clustering result of sequence similarity when evaluated with the curated benchmark.

The average-linkage clustering performance of Neighborhood Correlation on the curated benchmark families improves to a very small degree when the complete 48-genome dataset is clustered, as compared to a clustering result with only the mouse and human sequences as input. Figure 6.13 shows the Precision, Recall, and F-statistic for the ALL and ALL-kin benchmark sets. As compared to Figure 6.8, the clustering performance on the ALL dataset (a) is virtually unchanged. Within the ALL-kin dataset (b), the use of all genomes in the clustering yields an increase in Precision at low values, $NC \leq 0.350$, with a resulting increase in the F-statistic. The net result is that, with the 48-genome dataset, an F-statistic of at least 0.8 is achieved in the range of 0.225 through 0.575, as compared to a lower limit of 0.325 when only the mouse and human sequences are clustered.

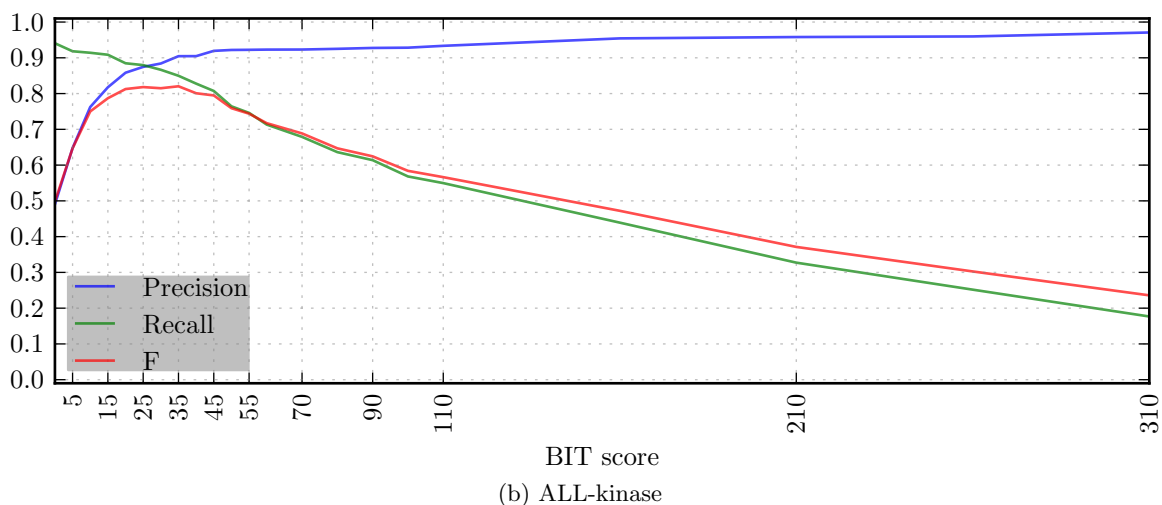
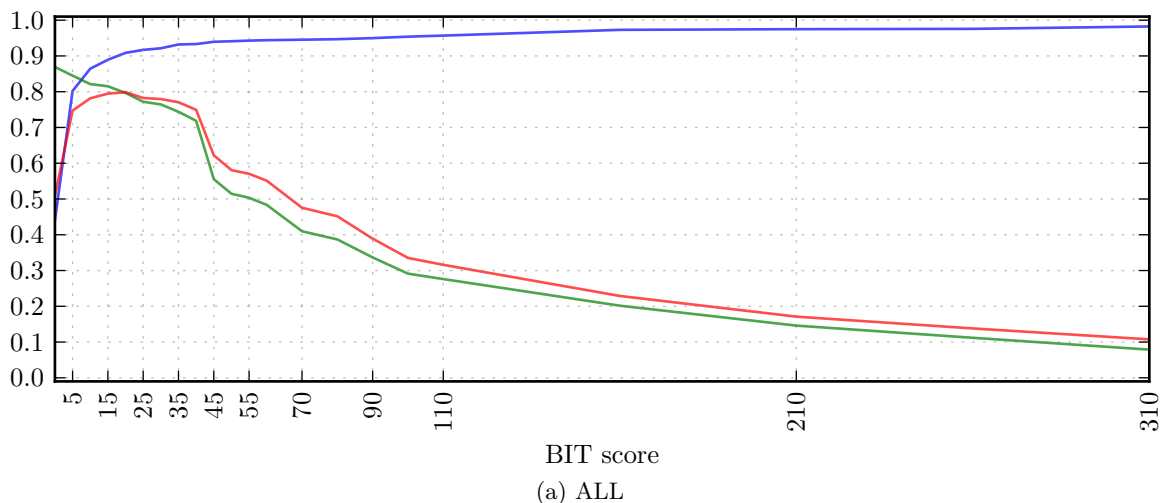


Figure 6.12: Clustering performance of average-linkage clustering of the sequence similarity network of all 48 genomes in the Panther 7.0 dataset.

Overall, this comparison suggests that average-linkage clustering of Neighborhood Correlation is very stable as the input dataset is changed from the mouse and human genomes, alone, to a dataset of 48 genomes.

Evaluation using aggregate sets of families from the curated benchmark, above, suggested that the clustering accuracy is relatively independent of the number of genomes over which the clustering is performed. While remaining small in magnitude, differences are observed for individual families. Figure 6.14 shows heatmaps of the F-statistic for each family, for both sequence similarity (left) and Neighborhood Correlation (right), when 48 genomes are used as clustering input. As with earlier heatmaps, the bottom rows depict the ALL and ALL-kin aggregates.

First consider the accuracy of sequence similarity, as compared to the central heatmap of Figure 6.4, which shows the F-statistic that results from average-linkage clustering of mouse and human sequences. For the majority of families, the performance as measured by the F-statistic is nearly

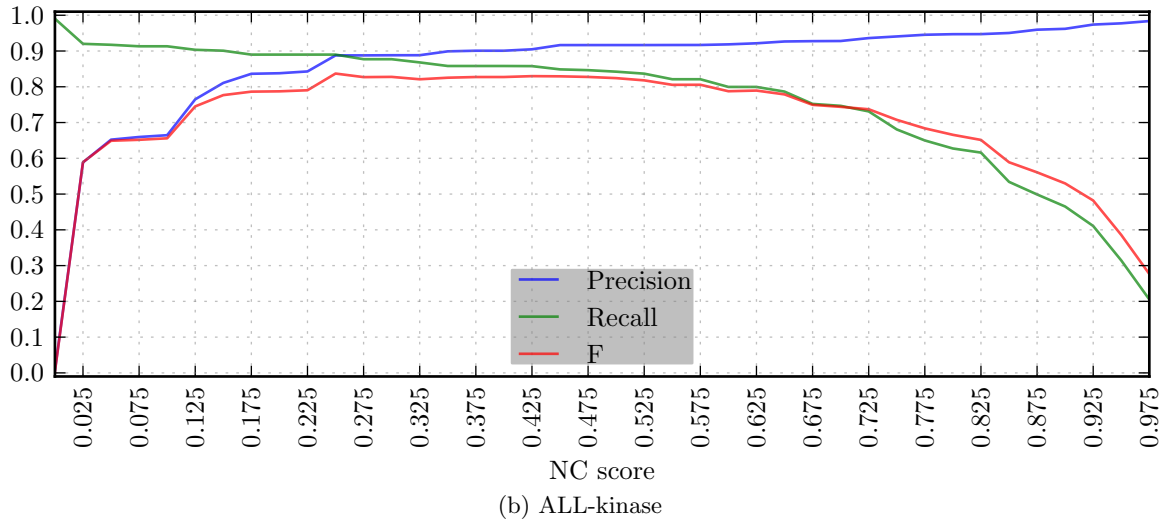
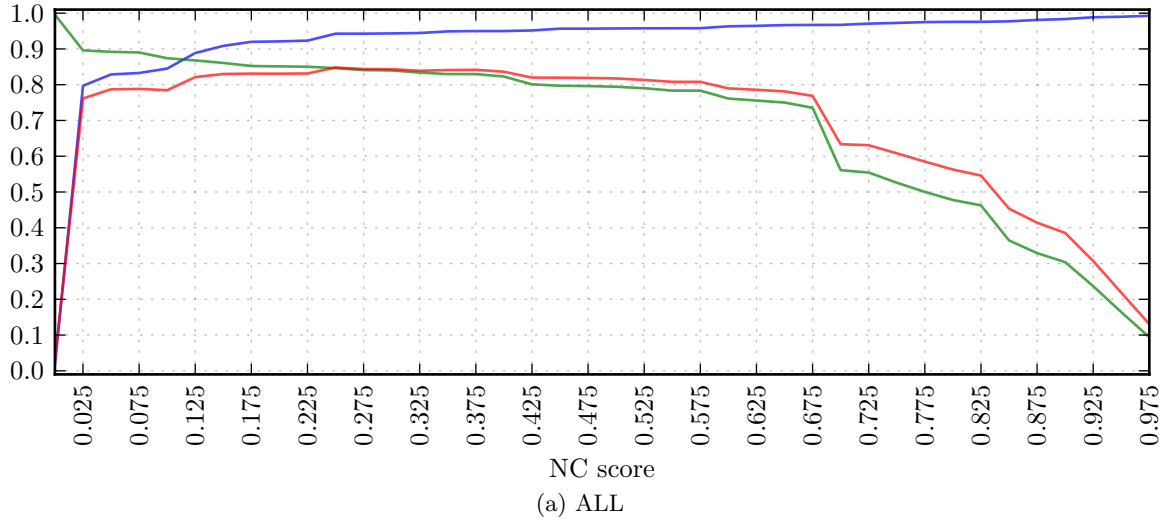


Figure 6.13: Clustering performance of average-linkage clustering of the Neighborhood Correlation network of all 48 genomes in the Panther 7.0 dataset.

identical. For other families, an increase in the maximum value of the F-statistic does occur. This includes the families DVL and USP, for which the ideal range of bit-score thresholds becomes more broad. For other families, such as KIR, Laminin, and SEMA, the ideal scoring range appears to become more narrow. As previously examined, the net performance on ALL and ALL-kin remains unchanged between the two clustering input datasets.

The F-statistic for individual families changes to a small degree when average-linkage clustering of Neighborhood Correlation scores is performed for the 48-genome dataset as compared to that of mouse and human sequences. The above discussion considered a slight increase in F-statistic on the ALL-kin dataset. When compared to the average-linkage clustering result in Figure 6.9, no marked change in performance occurs for any single family. Of those for which a change in the maximum or breadth of the range of F-statistic is observed, the magnitude of this change is small. Most notably, the lower threshold range achieves a higher F-statistic for the DVL family,

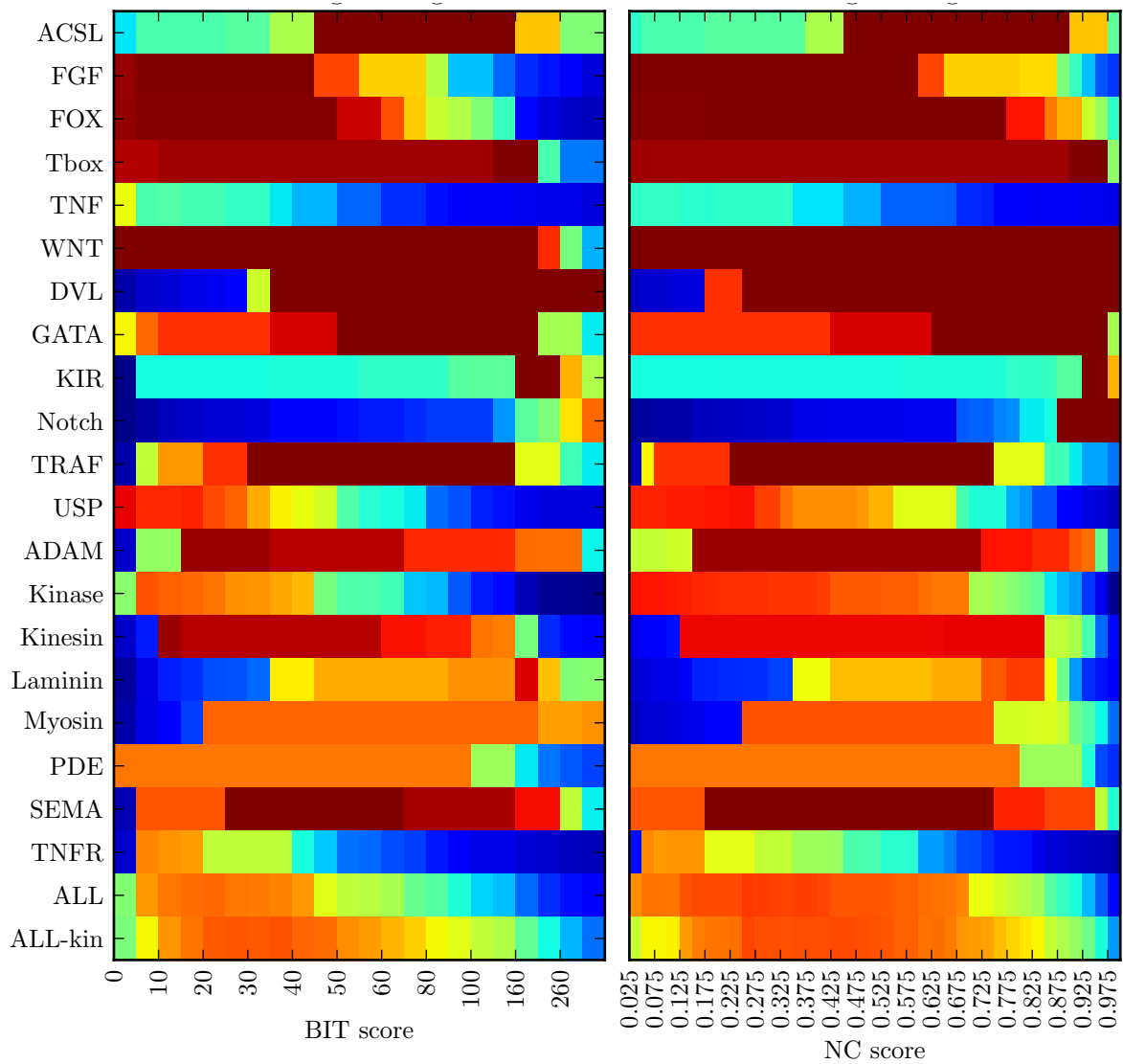


Figure 6.14: Heatmap of the F-statistic for all 20 benchmark families when clustering is performed on the full 600k, 48-genome dataset. Results for hierarchical clustering using the average-linkage method is shown for both sequence similarity, and Neighborhood Correlation.

because of greater Precision in that range. Conversely, while the range of high scores changes very little for the Laminin family, the maximum F-statistic achieved does decrease, from an F-statistic of approximately 1.0 to 0.8.

The relationship between domains and clusters

7.1 Introduction

The focus of this chapter is investigation of the interplay of domain content and the sequences that comprise gene families. The intent is to provide insight into the occurrence of domains among gene families in one or several genomes, and additional means of evaluating the quality of a family classification result. Although domains are not explicitly considered in approaches that use sequence similarity, there is reason to believe that domain content influences the outcome of such clustering. An information theoretic approach will be employed to characterize domains, and address these relationships.

Because most gene families are unknown, this study necessarily considers the interdependence of the family prediction methodology with the extent to which domain content has played a role in the evolution of predicted families. That is, when evaluating domain evolution within the context of families, it is important to consider that families proposed through clustering are putative. In earlier chapters, annotated gene families have been used as a means of evaluating and tuning the performance of family classification. Here, those same families will be used to examine properties of domain content. Similarly, domain content can be used to evaluate classification performance, but only so well as one can identify domains, or combinations thereof, that correspond to individual gene families. To clarify discussion, the term *family* will be used when describing a known or hypothetical set of sequences related through common ancestry. *Cluster* will be used to refer to a set of sequences that are predicted to comprise a family. Finally, a *clustering* refers to a complete dataset, partitioned into clusters.

Throughout this chapter, the set of sequences in the human and mouse genomes of the Panther 7.0 dataset will be considered. These have been partitioned using a pipeline of Neighborhood Correlation, followed by hierarchical clustering by the average-linkage method, as described in Chapter 6: Clustering and its evaluation (p.85). The particular portioning here is derived from the clustering of all sequences in the 48-genome Panther dataset, and imposition of a tree threshold of 0.425, which was shown to perform well when evaluated with the curated family benchmark (see

Figure 6.14).

The task of considering these factors will be decomposed as follows:

First, this chapter investigates the influence of domain content upon a clustering. Specifically, the degree to which protein domain content influences the clustering is measured. Key questions include: Can this clustering be completely explained by the domain content of the clustered sequences? If only the domain content of sequences, rather than their sequence similarity, were used as input to a family classification pipeline, could the same partitioning be obtained as by clustering the sequence network? An information theoretic approach is used to address whether the domain content is sufficient to explain the family classification result.

Second, a family classification may be used to identify domains that are specific to individual families, and provide a way of examining the evolutionary history of INCONSISTENT domains, which occur in many families. A correct family classification would group family members together, while avoiding effects such as domain chaining. Can INCONSISTENT domains be identified by their discordance with the clustering? Some domains are known to be INCONSISTENT among annotated families, providing a means of initial validation. Questions of interest include: Are most clusters characterized by a single domain? Do most domains occur in only one cluster? Is there a clear signal that differentiates domains consistent with the clustering from those that are found in many clusters? Which domains exhibit evidence of CONSISTENCY or INCONSISTENCY?

Third, I will consider how domain information may be used as a means of evaluating the quality of a clustering. There is an obvious relationship in that domain content can influence the sequence similarity measures that underlie our approach to family classification. Domain chaining is one dominant effect we have sought to eliminate, and one that is readily evidenced by a large “hairball”. A cluster characterized by very many domains, each of low count relative to the size of the cluster, does not fit our understanding of gene family evolution. The methods developed here are suitable for simultaneously evaluating the co-occurrence of domains and clusters.

7.2 Mutual Information Formulation

Here I describe an approach to addressing the above questions by considering the mutual information of domains and a set of proposed or annotated families. Using an information theoretic approach is a suitable way to capture the complexity of a partitioning, and directly measure to what degree the correspondence of other features (here, domain presence) can be used to explain that complexity.

Consider a clustering of sequences. Additionally, each sequence is associated with (i.e., contains) zero or more known domain types. (Note that excepting intrinsically disordered proteins [45], a sequence that contains no identifiable domains almost certainly still folds to a structural unit that is not yet described in domain databases.) We will not consider the number of instances of each domain type in a sequence. Similarly, domain order will not be captured. Our interest is to capture two properties of a protein p : whether p is in cluster i , and whether p has one or more copies of domain j .

The following notation is used:

- C_k : Number of sequences in cluster k
- D_l : Number of sequences with domain l
- $D_{l,k}$: Number of sequences with domain l and in cluster k
- N : Total number of sequences
- N_C : Total number of clusters

The probability that a sequence is in cluster k is

$$P_C(k) = \frac{C_k}{N}. \quad (7.1)$$

Similarly, the probability that a sequence has domain l is

$$P_D(l) = \frac{D_l}{N}. \quad (7.2)$$

The **entropy of the clustering** is then

$$H(C) = - \sum_{k \in \text{clusters}} P_C(k) \log_2(P_C(k)), \quad (7.3)$$

and the **entropy of a domain** is

$$H(D) = -[P_D(l) \log_2(P_D(l)) + (1 - P_D(l)) \log_2(1 - P_D(l))]. \quad (7.4)$$

The entropy of a clustering captures the complexity of that partitioning, from the perspective of how uncertain one is about the assignment of a single sequence to a cluster, and how many bits are required to optimally encode the assignment of each sequence to a particular cluster. This is based solely upon the distribution of cluster sizes (only on $P_C(k)$), independent of the specific counts. Taken to extremes, representing cluster assignments in a partitioning that contains only a single cluster requires no information, and $H(C) = 0$. Conversely, the maximal entropy of the clustering is achieved when the probability that a sequence p is assigned to a cluster i is the same for all clusters. In this case, $H(C) = \log_2(N_C)$.

The mutual information of two variables may be defined as

$$I(X; Y) = \sum_{x \in X} \sum_{y \in Y} \text{SMI}(x, y), \text{ where}$$

$$\text{SMI}(x, y) = p(x, y) \log_2 \frac{p(x, y)}{p_X(x)p_Y(y)}. \quad (7.5)$$

Considering two random variables, X and Y , mutual information can be cast as the degree to which knowledge of a particular value of X reduces the uncertainty of knowing the value of Y . Note that mutual information is a symmetric measure. If X and Y are independent, $I(X; Y) = 0$, indicating

that knowledge of one variable provides no information about the values of other variables. By contrast, if values of X and Y were identical, or entirely dependent, then $I(X;Y) = I(X;X) = H(X)$. the mutual information of x and y is bounded above by the lesser of the entropies of x and y . One way to understand this is to consider the formulation:

$$\begin{aligned} I(X;Y) &= H(X) + H(Y) - H(X;Y) \\ &= H(X) - H(X|Y), \end{aligned} \tag{7.6}$$

where

$$H(X|Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log_2 \frac{p(x)}{p(x,y)}. \tag{7.7}$$

That is, mutual information is the reduction of entropy ($H(X)$) obtained through knowledge of Y . When $X = Y$, $H(X|Y) = H(X|X) = 0$, and $I(X;Y) = I(X;X) = H(X)$. Conversely, when Y does not explain X , $H(X|Y) = H(X)$, and $I(X;Y) = 0$.

Here, I will primarily consider the **mutual information** between the clustering, C , and a single domain, l ,

$$I(C;l) = \frac{1}{N} \sum_{k \in C} D_{l,k} \log_2 \frac{D_{l,k}}{D_l C_k} + (C_k - D_{l,k}) \log_2 \frac{C_k - D_{l,k}}{D_l C_k}. \tag{7.8}$$

Mutual information between pairs of domains will also be considered where noted.

7.3 Working Example of Framework

This methodology can be best understood by following a concrete example. Consider a clustering of $N = 14$ sequences into two clusters. Cluster x has 8 single-domain sequences, with domain architectures: (a, a, a, a, b, b, e, e). Cluster y has 6 sequences with architectures: (cd, cd, cd, cd, e, e).

The particular partitioning of sequences and domain assignments in this example are intended to facilitate a rough intuition of cases where a domain is found only in a single cluster, estimated families but possibly not in all of the sequences of a cluster. Additionally, it illustrates a case (domain e) where a domain is distributed among many clusters, but in slightly varying proportions of the respective cluster sizes. This is interesting because such a domain, despite uniform distribution among the clusters, does have a non-zero mutual information with the clustering. This is because it comprises a larger fraction of the smaller cluster, y . Typically, more skew will be present in real data.

The following is a table of the domain content of clusters x and y :

		Domain				
		a	b	c	d	e
Cluster	x	4	2			2
	y			4	4	2
		4	2	4	4	4

Note that while this table summarizes the domain content of clusters, it is not a contingency table. Because a sequence may simultaneously contain more than one domain type, the sum of a row may exceed C_k , and the sum of all values may exceed N . Column sums *are* equal to D_l , the number of sequences in the cluster that have domain l , as noted.

Instead of directly using this table, we can construct a separate contingency table for each domain over all clusters. This describes the count of sequences with the presence (1) or absence (0) of each domain, tabulated by cluster. Marginal sums are as shown:

		Domain <i>a</i> Presence						Domain <i>b</i> Presence			
		0	1					0	1		
Cluster	x	4	4	8	C_x	Cluster	x	6	2	8	C_x
	y	6	0	6	C_y		y	6	0	6	C_y
		10	4	14	N			12	2	14	N
		$N - D_a$	D_a					$N - D_b$	D_b		
		Domain <i>c, d</i> Presence						Domain <i>e</i> Presence			
		0	1					0	1		
Cluster	x	8	4	8	C_x	Cluster	x	6	2	8	C_x
	y	2	0	6	C_y		y	4	2	6	C_y
		10	4	14	N			10	4	14	N
		$N - D_c$	D_c					$N - D_e$	D_e		

Here, $H(C) = 0.9852$, or just under one bit, owing to the slight imbalance in cluster size. The maximum entropy would be achieved when the probability of assignment of a given sequence to each of the clusters is equally probable. In that case, the entropy would be $\log_2(N_C) = \log_2(2) = 1$. In the current example, the sizes are slightly unequal, so the entropy is slightly lower.

The entropies of the domains above are

$$\begin{aligned}
 H(a) = H(c) = H(d) = H(e) &= 0.8631 \\
 H(b) &= 0.4917.
 \end{aligned}$$

From this, we can read that no single domain has sufficient power to fully describe the partitioning into two clusters (all domains have lower entropy than $H(C)$). Here, each domain represents a relatively high fraction of the clustering entropy. In practical cases, where more than two clusters exist, it is not possible for the existence or absence of a single domain in each cluster to explain the structure of the entire clustering.

To quantify the degree to which each domain can be used to explain the clustering, we can calculate mutual information from each domain’s contingency table:

$$\begin{aligned}
 I(C; a) &= 0.2917 \\
 I(C; b) &= 0.1281 \\
 I(C; c) = I(C; d) &= 0.4696 \\
 I(C; e) &= 0.0060
 \end{aligned}$$

Here, it is evident that those domains which one would intuitively observe to be correlated with one cluster (a with x , c and d with y) exhibit higher mutual information, and those that comprise a large fraction of a cluster are most informative. Domain b , though unique to cluster x , is less informative than a , c , and d due to there being fewer instances of it.

Domain e might be expected to have zero mutual information because it has the same number of instances in both clusters. However, it has mutual information greater than zero because it comprises a larger fraction of cluster y than x . As mentioned above, such skew is likely to be more dramatic in real data.

Another way to consider the magnitude of mutual information of a domain with respect to a clustering is to compare it with the maximum mutual information that would result from existence of a domain in all sequences in a single cluster, and only that cluster. Consider a hypothetical domain α that occurs in all sequences of cluster x , and β in those of cluster y :

$$\begin{aligned}
 I(C; \alpha) &= 0.9852 \\
 I(C; \beta) &= 0.9852
 \end{aligned}$$

This mutual information *cluster maximum* will be reported for families in the analysis that follows.

The specific mutual information (SMI) is the contribution of the joint probability between two particular values of the two variables considered. Here, a domain is either present or not in each cluster. Both the presence and absence of a domain contributes to the mutual information. To isolate the contribution of the presence of a domain in a particular cluster, the *presence SMI* will be used to refer to the SMI of a cluster and the presence of a domain in that cluster (PSMI ($c, 1, a$), for existence of domain a in cluster c). Do note that the absence of a domain from other clusters will typically contribute a positive quantity to I , so PSMI should not be considered to be the only positive, or even dominant, component of I .

To fully illustrate the contribution of individual values of a contingency table to mutual information, consider the SMI of all positions of the contingency tables of the domains below. Recall that the PSMI refers to the “1” column only, for each cluster.

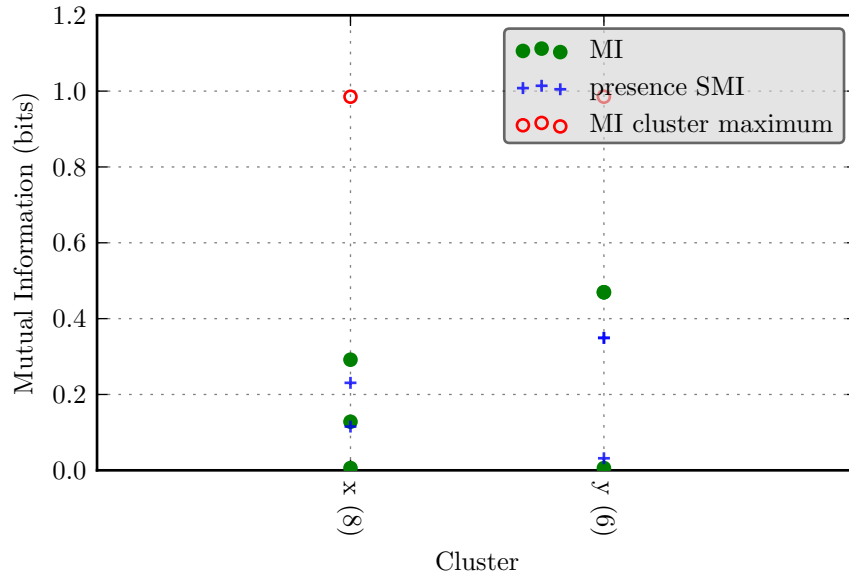


Figure 7.1: Mutual information of all domains in the working example, separated by cluster. For each domain found in a cluster, there are two points in the column of that cluster: MI, the mutual information of this domain with the entire clustering, and the PSMI of this domain and the particular cluster. Additionally, the MI cluster maximum is the maximal mutual information attainable for a hypothetical domain that occurs in every sequence of that cluster and no sequence outside of the cluster.

		Domain <i>a</i> Presence				Domain <i>b</i> Presence	
		0	1			0	1
Cluster	x	-0.147	0.231	Cluster	x	-0.083	0.115
	y	0.208	0		y	0.095	0
		Domain <i>c, d</i> Presence				Domain <i>e</i> Presence	
		0	1			0	1
Cluster	x	0.277	0	Cluster	x	0.030	-0.028
	y	-0.157	0.349		y	-0.028	0.032

The PSMI of domain *a* is 0.231 in cluster *x* and 0 in cluster *y*. This domain is unique to cluster *x*, and the high PSMI reflects this. To again caution against an excess of interpretation of PSMI, note that the absence of domain *a* in cluster *y* also yields a strong positive contribution to $I(C; a)$. Also note that the negative component of SMI of domain *a* in cluster *x* reduces the value of $I(C; a)$ from what might be inferred if only the presence in *x* and absence in *y* were considered. Domain *e*, distributed uniformly among clusters *x* and *y*, has a PSMI of -0.0275 with cluster *x* and 0.0318 with cluster *y*. These values are not each zero because of the discrepancy of cluster sizes.

Figure 7.1 illustrates a summary of the values derived above. Each column in this figure represents a single cluster, and depicts all domains found within it. Similar construction will be used to describe clusterings of real data, where there are many more clusters than the two here.

Cluster x contains three domains, a , b , and e . These have mutual information values of 0.2917, 0.1281, and 0.0060, respectively, and are represented by the three green circles in the column of cluster x . For each domain, it is useful to be able to consider whether the mutual information attained by a domain found in a cluster is due primarily to its association with that cluster. For each domain in a cluster, the pSMI (blue crosses) convey this information. For example, in cluster x , domain a has a pSMI value at 0.231.

The value of this figure is greatest when one can identify which blue cross and green dot correspond to the same domain. The construction of this figure does not unambiguously illustrate this relationship. However, often the pairing is obvious, and a number of relevant measures may be inferred. In practice, one or a few domains stand out for each cluster, with high mutual information and high pSMI, allowing clear visual association of which correspond to single domains, in the absence of intervening points. In all instances, the distribution of pSMI values for the domains found in a cluster indicate whether any are highly associated with that cluster.

Cluster y also contains three domains, c , d , and e , yielding three mutual information points in that column, but only two are visible because domains c and d have identical mutual information values of 0.4696, and overlap. Notice that domain e occurs in both clusters, so the same mutual information point occurs in both columns. However, the pSMI of domain e does differ between clusters x and y . In cluster y , the pSMI of domain e is 0.0318 and is larger than its mutual information, 0.0060. Typically, the pSMI is below the mutual information, though this is not always the case. The measured values, here, of domain e can be seen largely as an artifact of having very, few (two) clusters, allowing a single domain to be evenly distributed. In real clusterings, there are almost always fewer sequences of a particular domain than the number of clusters, which effectively eliminates this behavior.

7.4 Areas of focus

A number of biological and methodological questions will be considered using this framework. These fall into three major areas: (1) characterization of clusters and families, (2) characterization of the relationship between domains and those clusters and families, and (3) measurements of the co-occurrence, or exclusion, of pairs of domains. The following is a brief guide to the results presented in the following sections, and the questions that motivate their consideration.

The central goal of this study is to investigate the relationship between domains and families. The results here are presented with an emphasis upon distinguishing a number of *views* into the data, accompanied by a discussion of the relevant variables and conclusions.

7.4.1 Structure of predicted families

There is need for a framework that allows comparison among single methods applied to different datasets, and among different analyses and methods on the same dataset. Chapters 5 and 6 investigated the use of structural properties of the sequence network and evaluated the ability of this network to recapitulate families. To what extent does entropy facilitate evaluation of the structure of a clustering?

The structure of biological families may change between datasets. Closely related genomes might

be expected to have roughly the same set of gene families, with similar relative sizes and domain content. (For example, see Ye and Godzik [154].) In contrast, when comparing distantly related genomes, one might expect to see families that are unique to each genome. Families that occur in both genomes might have different size distributions and greater differences in domain content.

A typical case in practice is first to consider a small set of data (e.g., one or two genomes) before expanding to a larger super-set (i.e., many genomes). In the case of closely related genomes, the addition of genomes would be expected to increase the number of sequences in each family proportionate to the number of added genomes. Because the cluster sizes, relative to each other, would differ very little, and the entropy of the clustering depends only upon these relative sizes, the entropy of the clustering would change little between the smaller and larger datasets. A particularly useful side effect of this property is that the mutual information values of domains against the two clusterings may be directly compared.

A set of predicted families may change due to use of a different clustering method, different parameters, or coverage of the data set to which it is applied. Practical scenarios include adding more genomes to an analysis, comparing different clustering methods, or switching taxonomic lineages.

When the sequences in two datasets are very different, or are clustering performed at widely varying granularity (e.g., sub-families versus families in a correct clustering), the entropy can vary widely. Such changes are an indication of a very different clustering or very different family structure, which should be investigated independently. The entropy of a clustering provides an indication of when one should investigate differences in the clusterings to be considered, before moving on to a detailed examination of the mutual information of domains. When the entropy of two clusterings are similar, the clusterings are of similar granularity—even if quite different in cluster content—and it is useful to compare domain content of those clusters.

Note that a domain found in several clusters, but occurring in most or all sequences in those clusters will have high mutual information. When a clustering is too coarse, grouping many families together, or entirely incorrect, partitioning the dataset independent of families, the mutual information of such a domain will be lower. Conversely, a (correct) partitioning that is too fine will split families into separate clusters, but those clusters will each remain comprised of related sequences. Domains that correspond to each of those clusters will retain high mutual information. This yields robustness: The mutual information of a CONSISTENT domain decreases when a clustering does not correspond to family boundaries, but far less so when a family is split into sub-families. This also provides a means of calibrating the granularity of a clustering.

7.4.2 Characteristics of domains

Exploration of the interplay of domains and family structure is of primary interest in this study. A number of canonical examples of domain and family correspondence motivate this approach. Single domain families are an obvious case of direct association of a domain and a family. Among multidomain families, INCONSISTENT domains introduce an additional layer of complexity. It is an open question whether families typically have a single domain that occurs in all members of the family, to which additional domains contribute auxiliary function, such as a specific binding mode. There are a number of examples of this case, including the Kinases. Absent such a defining domain, are there families defined by the co-occurrence of several domains?

Rather than looking for specific cases of domain content, the intent with this approach is to examine how typical they are in real genomes, and to remain flexible enough to be able to identify and characterize other interesting paradigms of domain occurrence. Questions that will be addressed include: Does a family contain one or more domains that can be used to define that family (e.g., all members have the domain, and no other sequences do)? Conversely, does a given domain define one or more families (e.g., the domain occurs in two families, possibly sub-families, and never elsewhere)? May a family be defined by co-occurrence of a combination of domains? Is a given domain CONSISTENT?

The mutual information of each domain and a clustering of a particular dataset can be used to compare the occurrence of different domains within that clustering. In particular, the above formulation and examples exhibit how domains with close correspondence to families, predicted or real, have high mutual information. This highlights domains that “correlate” well with the partitioning (as domain *c*, in the example, which has high mutual information) as well as those that do not, and are INCONSISTENT (e.g., domain *e*, which has low mutual information).

Additionally, mutual information is useful for comparing domain content across different clusterings and datasets. Consider, first, different clusterings of a single dataset. Here, the number of sequences (N) remains constant, but the partitioning of sequences may differ, and the distribution of the counts of sequences in clusters may vary widely. In an extreme example, one clustering may have many clusters all of equal size, and another may assign nearly all sequences to a single, large cluster. These clustering will have very different entropies. Further, observed mutual information of domains will change greatly. The clustering that groups nearly all sequences together will exhibit very low mutual information for all domains, since no domain would be in most of the sequences. When partitioned more finely, domains that do correspond to clusters will show higher mutual information than other, INCONSISTENT, domains in the same set.

It is useful to consider the range of mutual information with respect to family structure. The maximum attainable mutual information for a domain is limited by two primary factors. Numerically, the total number of sequences in which a domain occurs limits the entropy of that domain, and, accordingly, the mutual information of the domain and the set of families. A domain that occurs in very few sequences will necessarily have low mutual information, but such a low value does not necessarily reflect promiscuity. Comparison with the entropy of that domain can determine whether this is the case. Further, the pSMI of a domain and family indicates whether a domain has high mutual information because of a particular family. When a domain is uniquely associated with a cluster or family, even if in only a subset of sequences, the pSMI of a domain will be below the corresponding mutual information of that domain because the absence of a domain in other clusters also contributes to the mutual information.

A second situation is when a domain occurs among many families. Such a domain *is* INCONSISTENT. This property is captured by the mutual information. A domain that is found in most sequences of one cluster will have high mutual information. Similarly, a domain that is found in several clusters, but occurs in most sequences of those clusters, will retain high mutual information. (This scenario is more typical of a clustering that is too fine-grained than a INCONSISTENT domain.) By contrast, a domain that is scattered among many clusters, but not found in the majority of sequences in those clusters will have low mutual information.

7.4.3 Domain co-occurrence

Two other patterns arise for which the mutual information of a single domain with the clustering may not be sufficiently expressive to capture of the properties of interest. Many families are characterized by a “primary” domain that appears almost exclusively in members of the family, often in various combinations with other, auxiliary domains. A particularly notable example is the kinase family. The above example captures some aspects of such families.

An interesting case is that where a particular combination of domains is found to characterize a cluster, but where each domain, singly, is found in many other clusters. A concrete example of this is the Laminin family. It comprises 23 sequences in Mouse and Human and contains seven distinct domains. All sequences have unique architectures. Only one domain, Laminin_EGF is found in all Laminin sequences, but it also is present in 41 unrelated sequences. None of the other six domains in the Laminin family are present in all 23 members. Only two of these are unique to the Laminin family, but there are Laminins without either of these. This is a situation where consideration of at least three domains would be necessary to correctly group the Laminin family using only domain content.

A combination of domains need not strictly correspond to the presence of the domains, but could instead refer to relationships such as the *XOR* of two domains. The Kinase family provides an illustration of this scenario. Based on annotations from the PFam database, members of this family are characterized by the presence of either the Pkinase-C domain or the Pkinase-Tyr domain, but never both. Here, to group the Kinase family together using domain content alone, this exclusivity would need to be captured. Note that in Kinase, this situation arises due more to convention in the database than to disparate domains. The two Pkinase domains are very similar, and can be considered classes of the same domain.

A multivariate formulation of mutual information would be necessary to simultaneously consider two or more domains and the clustering. However, this will introduce complexity in interpretation of the result and a risk of over-fitting the limited, discrete data available. To gauge the necessity of this complication, I have structured the results given here with an eye toward determining (1) whether a single domain is sufficient to characterize every cluster, and to what degree, and (2) whether the joint consideration of multiple domains (multivariate MI) could be expected to improve this characterization for individual clusters.

One approach to the latter is to consider a loose upper bound on the (joint) mutual information of all domains and the clustering. This may be considered by summing the pairwise mutual information of all domains, each against the clustering. This sum accurately represents the mutual information of all variables, together, only if the domains are each independently distributed, which is not the case here. Thus, this bound may be very loose. Similar summation of pairwise mutual information values among many variables has been employed, for example, in image registration as an effective estimate of the scale (i.e., not the numerical value) of a multivariate mutual information [84].

Without full development of a multivariate formulation of the framework here, specific questions that will be addressed include: Which domain pairs tend to co-occur? Which pairs never co-occur?

7.5 Data

To investigate the effectiveness of this approach, I have applied a series of analyses to two clusterings, one of curated families, the other a result of a clustering pipeline stemming from sequence similarity.

The sequence data considered here is the combined set of Human and Mouse sequences from Panther 7.0, comprising 45,491 sequences in all. Domains have been annotated by performing a search using domain HMM profiles from a recent version of the PFam database.

The influence of domains upon family classification can most directly be evaluated when a ground-truth clustering is known, using curated families. Use of independent evidence for curation ensures that the signal of domains may be measured independently of how sequences are associated with predicted families. However, it is limited to consideration of sequences, and families, that have been curated. This is typically a small fraction of the genome.

Conversely, an automated clustering of the dataset facilitates analysis of the full dataset, but introduces the variable of whether a clustering accurately corresponds to the evolutionary families we aim to identify. Families and partitions generated by automated clustering are complementary, and related: selection of clustering parameters, and, indeed, design of the clustering pipeline has been informed by our set of known families. A critical interest is whether we can evaluate how typical our curated families are of the entire genome.

Here, two datasets representing these extremes are considered. First, the curated benchmark of families is considered alone. As detailed in Chapter 2: Background and preliminaries (p.13), this dataset is comprised of 1841 sequences, spanning 20 families. The sequences in this set cumulatively contain 203 unique domains. Second, the set of sequences in the mouse and human genomes are considered. This dataset contains annotations for 4304 distinct domains.

7.6 Clustering entropy

It is useful to consider the *granularity* of a clustering; i.e, into how small of groups the sequences are partitioned. Biologically, this is akin to the concept of grouping at a given level of specificity, such as families, or subfamilies. In the sense of hierarchical clustering, this may be intuited as cutting the tree at different levels. Entropy can be used to capture this property. It is dependent upon the distribution of cluster sizes in a clustering, but not the particular assignment of sequences to those clusters, the specific number of sequences in the clusters, or even the number of sequences in the dataset. These latter properties are of particular benefit when working with varied data considered here.

Within a particular dataset, it is useful to consider the overall behavior of a clustering method with different parameter values, or with different methods entirely. Entropy is a reasonable means of identifying pathological cases such as a one cluster ($H = 0$) or a clustering of singletons ($H = \log_2 N$), as well as cases between. Similar values of entropy between clusterings indicate a similar granularity, while large changes indicate wholesale differences in the structure of a clustering.

It is frequently desirable to compare clusterings across different datasets. Recall that a typical use-case is exploration of a small dataset followed by study of a larger, inclusive set. Here, a lack of dependence upon the specific cluster or dataset sizes is particularly useful.

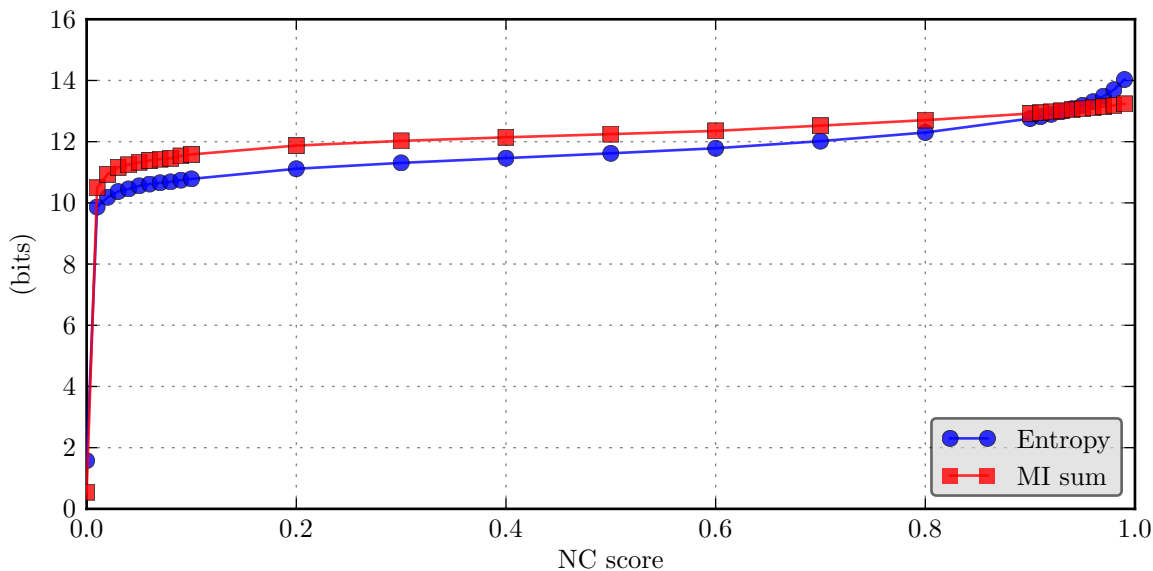


Figure 7.2: Clustering entropy as a function of the threshold used to partition the Neighborhood Correlation average-linkage clustering result. The entropy of each partitioning is in blue. The MI sum is the sum of the mutual information, over all domains. This latter value is a loose upper bound of the combined joint mutual information of all domains, calculated as in [84].

Figure 7.2 illustrates the entropy of Neighborhood Correlation, after hierarchical clustering (blue line). Specifically, it displays how the entropy of the clustering varies across the range of all possible clustering thresholds. The entropy of the clustering necessarily increases as the clustering becomes finer. At $NC = 0$, all sequences are part of one cluster, and $H = 0$; $NC = 1$ separates nearly all sequences into singletons.

The primary conclusion from these data is that the granularity of the clustering changes slowly throughout the full scoring range, and that substantial changes to the granularity of the clustering are constrained to the first 0.01 and the very high values, above 0.9. This suggests that the structure of the clustering is such that, as the threshold is increased, a single “hairball” cluster disintegrates (inflection around 0.01), separating into a number of smaller clusters, but retaining a structure beyond singletons. As the threshold is increased further yet, some of these smaller clusters are split, and some sequences do become singletons. Only at the highest thresholds (inflection around 0.98) does the clustering degrade entirely to very small clusters and singletons.

An identifiable region of relative stability is of most value for predictions. Within such a broad region, a range of thresholds can be selected to yield clusterings of similar structure, but with tunable specificity. Small changes within this range of thresholds will not result in dramatic changes in output.

Plotting entropy as a function of threshold can distinguish this behavior from an alternate scenario in which increasing the threshold incrementally removes singletons from such a hairball, but never recovers intermediate structure. Such data would not be hierarchical in nature.

Consideration of the entropy of the clustering can provide some guidance about the significance of the mutual information of a domain with respect to that clustering. As no single domain can

be expected to represent the entire clustering, the mutual information of domains will typically be small compared to the entropy of the clustering. A rough way to address how well *all* domains may jointly characterize the clustering is to sum the mutual information values of all domains, each with respect to the clustering. In the case where domains are distributed over sequences independent of all other domains, this sum would represent the maximum information gained by knowing the assignment of domains to sequences. The sum would not exceed the entropy of the clustering; indeed, the entropy is also an upper bound on the joint mutual information.

When the MI sum is less than the entropy, the domain content of sequences is certainly not sufficient to account for the clustering.

7.7 Relationship of domains and clustering

Notable questions of interest addressed by this construction include (1) whether one or more domains is unique to each cluster, (2) whether there exists a broad distribution of mutual information of domains within each cluster (i.e., are most clusters characterized by a single domain, or are there generally several domains exclusive to each cluster), and (3) to what degree the presence of a domain in a cluster contributes to high values of mutual information, as compared to its absence.

This will be approached by considering a number of perspectives to identify properties of each the set of domains and the set of families, as well as their joint contribution. First, to facilitate a detailed look at the behavior of the mutual information formulation provided here, Figure 7.3 illustrates the mutual information of every domain found in each of the 20 families. Similarly, Figure 7.4 illustrates the same content for each cluster in the NC Average Linkage clustering. These figures are constructed in an identical manner as Figure 7.1.

It is helpful to first understand the mechanics of what is presented:

For each domain found in a family, there are two points in the column of that family: (1) MI, the mutual information of that domain with the set of all families, and (2) the PSMI of this domain in the family. Additionally, the MI Cluster Maximum for each family is the maximal mutual information attainable for a hypothetical domain that occurs in every sequence of only that family.

This plot separates the contribution to the mutual information of a domain by family, allowing consideration of the data from the perspective of domains (e.g., how many families contain domain X; are they defined by it?), as well as by family (e.g., how many domains occur in family F; do any define it?). At a glance, this provides information about how many distinct domains are found within a family, by counting the number of MI points in the corresponding column. The MI value of a domain illustrates how closely that domain corresponds to the set of all families; those with higher values are associated with a few specific families, not spread among all.

Note that a domain may have a mutual information *higher* than the MI maximum for a family owing to close association with one or more other families. For example, in Figure 7.4, note that several domains, including, from the top, 7-trans-membrane-1 (MI 0.27) and Pkinase (MI 0.11) occur in several clusters.

For each MI point (green dot) representing a domain found in a family, a corresponding PSMI point (blue cross) shows the specific contribution of occurrences within that family to the mutual information. Blue crosses closely aligned to or just below green circles almost certainly represent

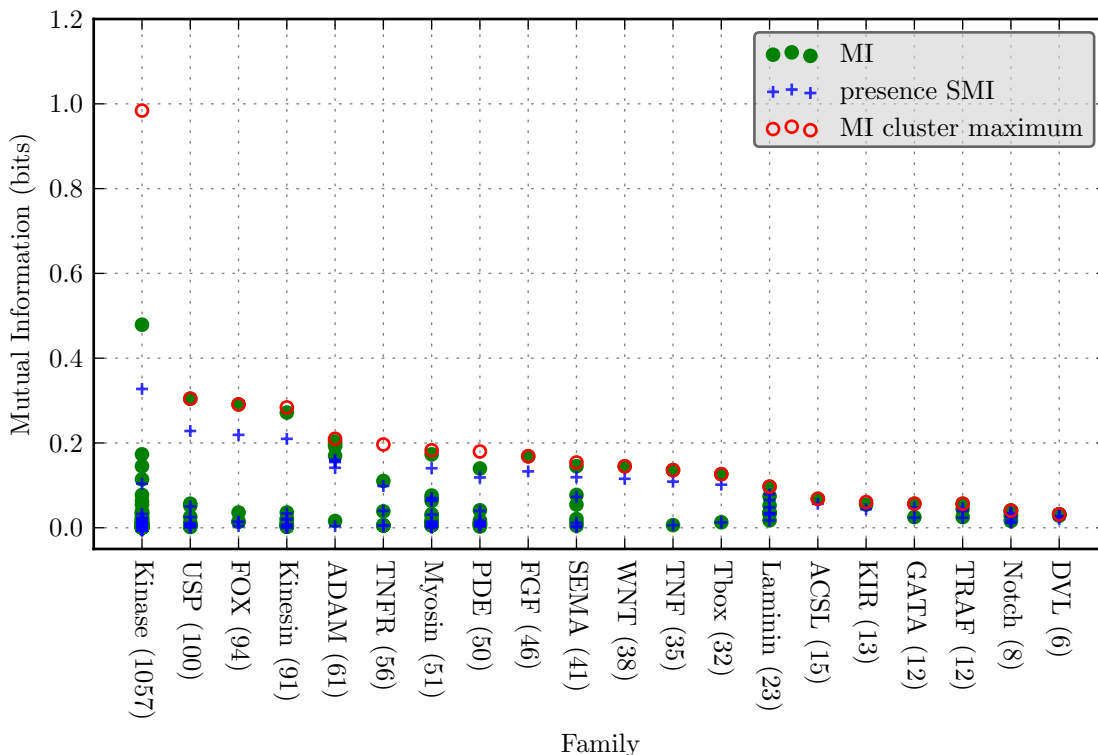
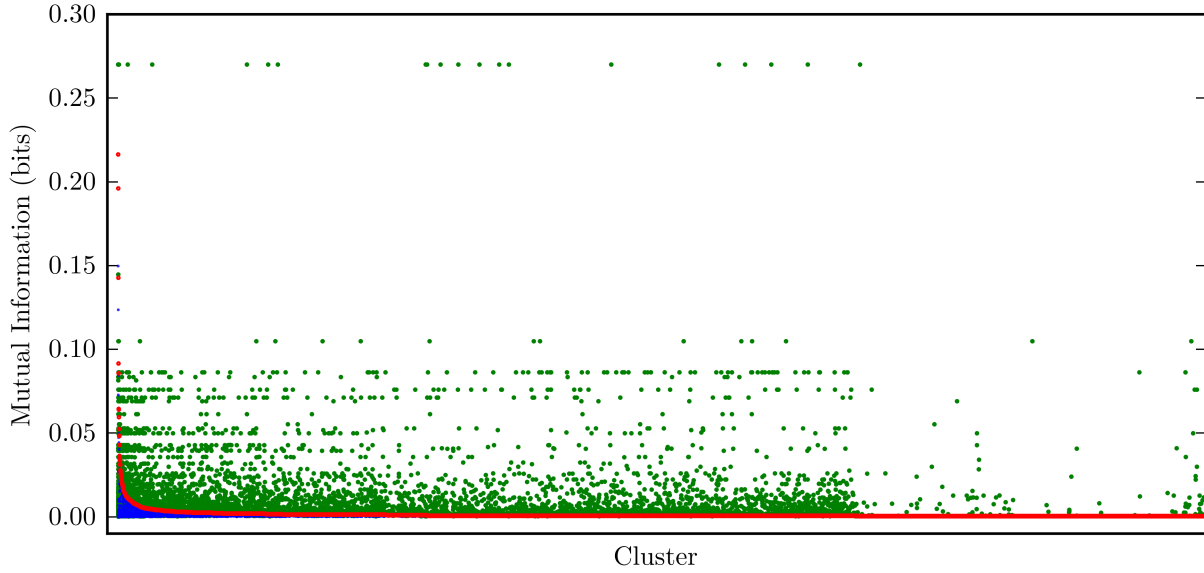


Figure 7.3: Mutual information of all domains in the 20-family benchmark set, separated by family. Families are ordered by descending size, indicated in parentheses. For each domain found in a family, there are two points in the column of that family: MI, the mutual information of that domain with the set of all families, and pSMI of this domain in the family. Additionally, the MI cluster maximum for each family is the maximal mutual information that would be obtained by a hypothetical domain that occurs exclusively in every sequence of that family.

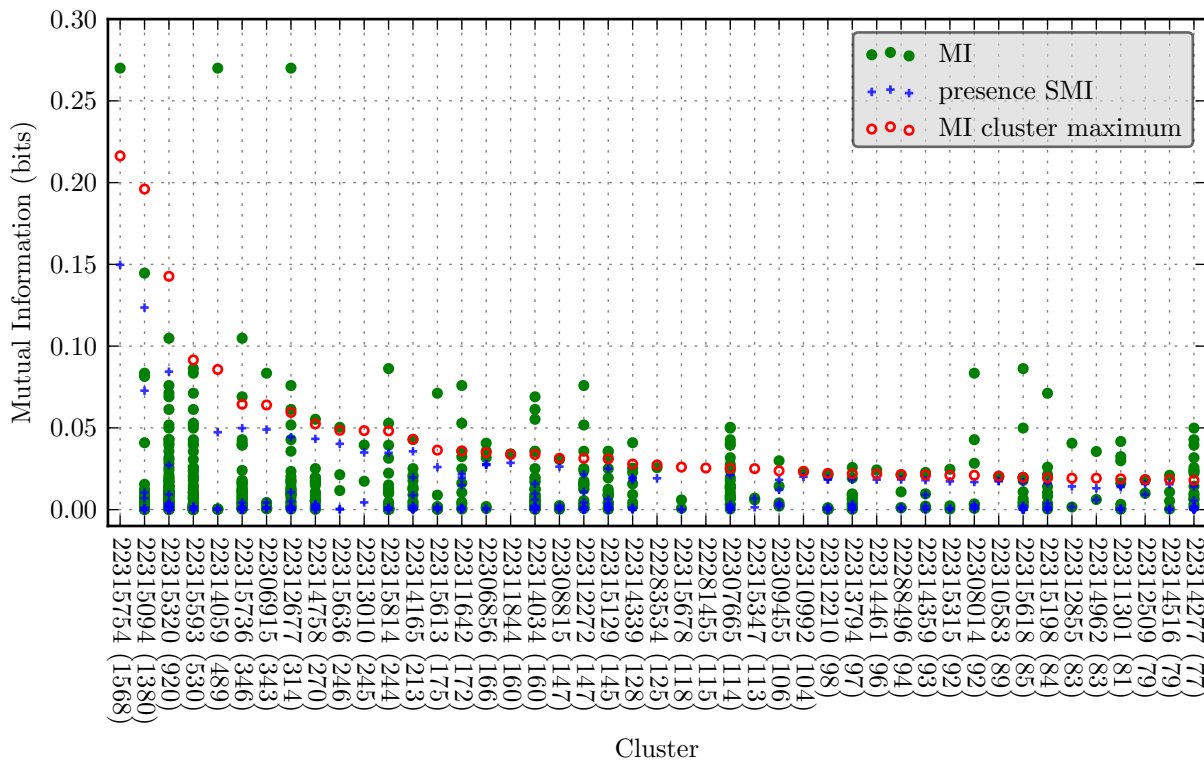
the same domain, though matching the corresponding points can be ambiguous when several domains have similar mutual information. The pSMI of a domain will typically be lower than the corresponding mutual information of that domain because the absence of a domain in other clusters also contributes to the mutual information.

Focusing upon the 20 families (Figure 7.3), nearly every family exhibits a single domain that overlaps or is very near to the MI maximum for that family. A notable exception is the Kinase family, in which all sequences are characterized by presence of either the Pkinase-C or Pkinase-Tyr domains. These domains never co-occur in the same sequence. These are the top two green points, respectively. These domains are also represented by the top two pSMI points. Recall that the distinction between the Pkinase-C and Pkinase-Tyr domains is primarily one of annotation convention, rather than distinct biology. These domains are very similar, though are maintained as separate entities by the PFam database.

Note that very few domains occur in more than one family, *in this dataset*. Many of these domains are found elsewhere in the genome in members of other, un-curated families. However, these sequences are not part of this dataset. As a result, with the analysis constrained only to our annotated families, most domains are CONSISTENT relative to the data set.



(a) All clusters



(b) Largest clusters

Figure 7.4: Mutual information of all domains in clusters of human and mouse sequences. (a) contains the full set of 9000 clusters. (b) depicts only the largest clusters, of size indicated in parentheses. Clusters are ordered by descending size. The properties of each contained domain are represented by two points in the column of a cluster: MI, the mutual information of that domain with the entire clustering, and pSMI of this domain in the cluster. Additionally, the MI cluster maximum is depicted.

In contrast, consider the full set of sequences in the Mouse and Human genomes (Figure 7.4). Here, it becomes obvious that many domains do occur in more than one cluster. The apparent horizontal bands are domains that occur in multiple clusters. Additionally, very many clusters contain domains that have mutual information values greater than the maximum attainable by association with only that particular cluster. These values are the result of close association with larger clusters.

This figure illustrates clusters where there exists a domain that occurs in every sequence of that cluster. Such domains are perfectly correlated with the clustering: all sequences that contain it comprise a single cluster. These domains have mutual information equal to the cluster maximum, and a pSMI value immediately below the mutual information. Recall that the difference in value between mutual information and pSMI of such perfectly correlated domains is due only to consideration of the absence of the domain in all other clusters with the mutual information.

Also illustrated are clusters that contain one domain that has high mutual information, relative to the MI cluster maximum, and which has a high pSMI value. The assignment of sequences to such clusters could be performed by considering only the presence or absence of the domain of interest. When the cluster represents a family that is single-domain, this is a trivial case.

7.8 Mutual information of domains versus entropy

In the preceding section, we considered the data from a perspective of family or cluster structure. Now, the analysis will target the behavior of domains with respect to a clustering.

Figures 7.5 and 7.6 show the entropy of the distribution of that domain over all sequences on the horizontal axis. The vertical position gives the mutual information of that domain and the clustering, or family structure. The objective of this construction is to identify how closely domains correspond to a clustering as a whole. Comparison of the mutual information and entropy of each domain is a measure of the degree to which sequences that contain that domain are clustered together. It is important to approach this from a perspective that is invariant to the number of instances of a domain. Domains that occur in very few sequences can contribute less knowledge of relationships among the full set of sequences. (Each domain can define at most one bi-partitioning of the sequences.) Most domains occur in relatively low count, however; taken together, it is these domains that most define families.

More formally, recall that the upper bound of mutual information of two variables is the lesser of the entropies of those variables (see page 112). Here, the entropy of a domain is directly related to the number of sequence instances. The entropy of the clustering is much greater than that of any domain in both the benchmark families ($H(C) = 2.6$) and the Neighborhood Correlation clustering ($H(C) = 11.2$). The mutual information of a domain, d , is equal to the entropy of d if and only if the clusters containing d do not contain any sequences that lack domain d .

Beyond identifying domains that have mutual information tending toward their entropy, which indicates a perfect correspondence with one or more clusters, this construction can be used to identify whether there exists a set of domains that are INCONSISTENT and disparate from the clustering. These would be identified by a mutual information much less than the entropy.

This figure is intended to address (1) whether domains tend to have mutual information tending

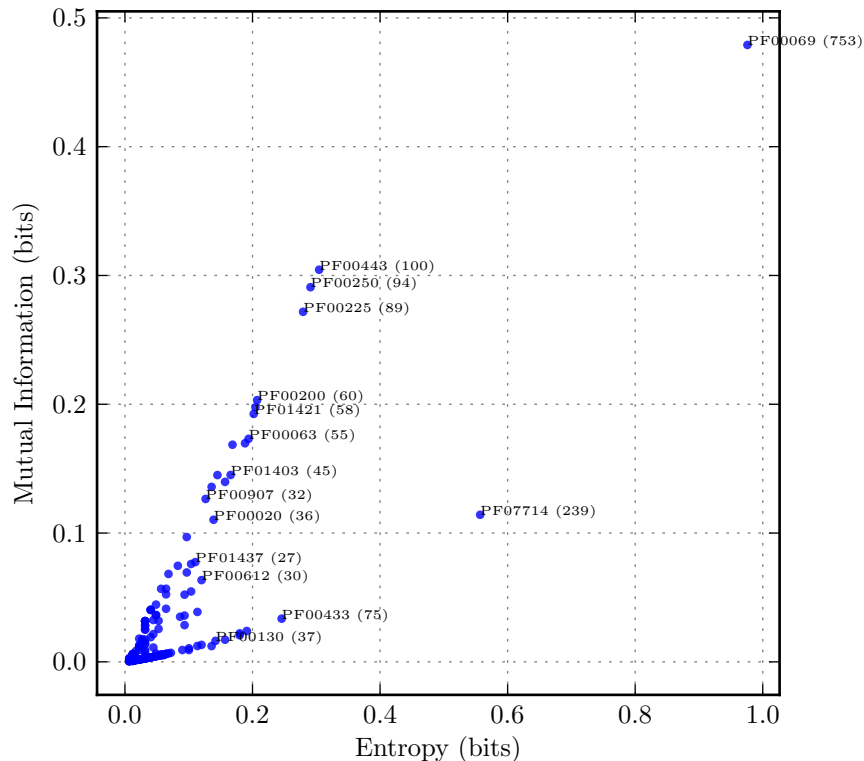


Figure 7.5: Mutual information as compared to the entropy of each domain in the family benchmark. The most numerous domains are labeled by Pfam identifier, with the number of sequence instances in parentheses.

toward their entropy, which would indicate perfect correspondence with one or more clusters, and (2) whether there is a set of INCONSISTENT domains that are disparate from the clustering, and identifiable by having much greater entropy than mutual information.

In the 20 Families, the majority of the 203 domains lie on or very close to the diagonal (note the different axis scales), indicating perfect association with the structure of the families. This could be anticipated from Figure 7.3. A notable lower line of domains is apparent, and interesting. This set of domains includes PF00041 (FN3), PF00433 (Pkinase_C), PF07714 (Pkinase_Tyr) and PF00069 (Pkinase). These domains exist only in the Kinase family, but are limited in mutual information because they occur in only some of the sequences. Note that the domain FN3 is not believed to be a CONSISTENT domain, but no other families in the benchmark have an instance of it. Domains that are found in low count in large clusters, but perfectly associated with one cluster, will have lower I/H than those that are found in smaller clusters, even if those found in smaller clusters are not CONSISTENT.

This effect appears to be an artifact of limiting the dataset under study. Recall that few domains are actually INCONSISTENT between our curated families, but do occur in un-curated sequences in the genomes. Including those other sequences ameliorates the effect, and the distinction does not persist in the clustering of all human and mouse sequences.

Figure 7.6 represents the mutual information and entropy of domains in the clustering of all mouse

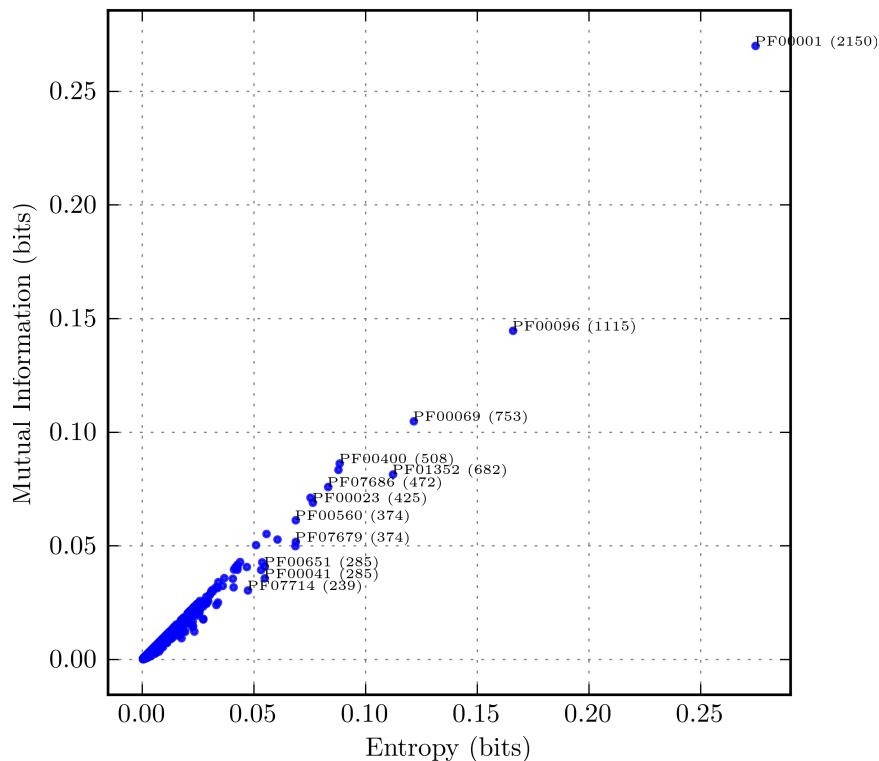


Figure 7.6: Mutual information as compared to the entropy of each domain in the clustering of mouse and human sequences. The most numerous domains are labeled by Pfam identifier, with the number of sequence instances in parentheses.

and human sequences. This figure demonstrates, again, that most domains lie upon the $I = H$ diagonal, and thus are maximally informative. This may be interpreted to mean that most instances of an individual domain occur in clusters where nearly all sequences contain that domain. This could correspond to a single cluster, or several clusters, each of which have many instances of the domain.

Notable, highly informative domains include PF00001 (7-transmembrane-1), PF00096 (zinc-finger C2H2), and PF00069 (Pkinase). From other analyses, we expect all curated families, especially the Kinase family, to be very well reconstructed with the average linkage threshold used here. It is an open question whether the 7-transmembrane-1 and zinc-finger domains are INCONSISTENT domains. This data, and the following analyses suggest that irrespective of this question, they do not, however, lead to domain-chaining and the inclusion of sequences with other domains when Neighborhood Correlation is used.

Interestingly, and perhaps contradictory, PF00400 (WD40) occurs in 75 clusters and 508 sequences, which would suggest INCONSISTENCY, a conclusion consistent with other evidence about the WD40 domain [134]. The combined size of all 75 clusters is 714 sequences, suggesting that while the domain may be INCONSISTENT, it does tend to be found in nearly all sequences within the clusters where it does occur.

Factor	Aggregate function	Description
<code>num domains</code>	mean, sum	Number of domains in a cluster, on average per sequence, or in total.
<code>seq length</code>	mean	Amino acid length of sequences in a cluster, on average per sequence.
<code>cluster max</code>	-	Highest possible mutual information for a domain that occurs exclusively in all sequences of that cluster.
<code>domain mi</code>	maximum	Mutual information of the domain in that cluster with greatest mutual information.
<code>domain psmi</code>	median	PSMI of each domain in a cluster.
<code>domain mi psmi</code>	median	For each domain in a cluster, the difference between the mutual information and its PSMI.
<code>clmax less psmi</code>	median	For each domain in a cluster, the difference between <code>cluster max</code> and its PSMI.
<code>clmax less mi</code>	median	For each domain in a cluster, the difference between <code>cluster max</code> and its mutual information.

Table 7.1: Features of clusters.

7.9 Characterization of clusters by domain content

Beyond the ability to accurately classify families, as covered in earlier chapters, it is valuable to identify defining features of those predicted families. For example, the families in Figure 7.3 and clusters in Figure 7.4 have been ordered by descending size. In previous chapters, they have been organized with respect to the number and variety of domains observed in the family. Each of these figures suggest that rich structure results from the information-theoretic approach pursued here. This section investigates whether this structure can be used to reliably group families and predicted families into classes that mirror our intuition about the modes of family evolution.

To this end, the results presented in Figures 7.3 and 7.4 have been used to derive a range of features about the domain content of individual clusters. The goal at this stage is not to ascertain whether any particular feature is biologically relevant, but whether it, in combination with the other features, can be used to explain the structure observed in the domain mutual information data. A second step of considering the biological relevance of these features will be discussed as they are evaluated in the following pages.

The complete set of features considered is listed in Table 7.1. The purpose of these features is for each to result in a single value for each cluster or family. The majority of these features are calculated for individual domains in every cluster or family, which results in as many values as there are domains. A single value for a cluster or family is achieved by applying an aggregate function, such as the sum, mean, or median, over all instances in the cluster. Other features, such as the number of domains or the sequence length, are not based on the domain information theoretic values derived in this chapter.

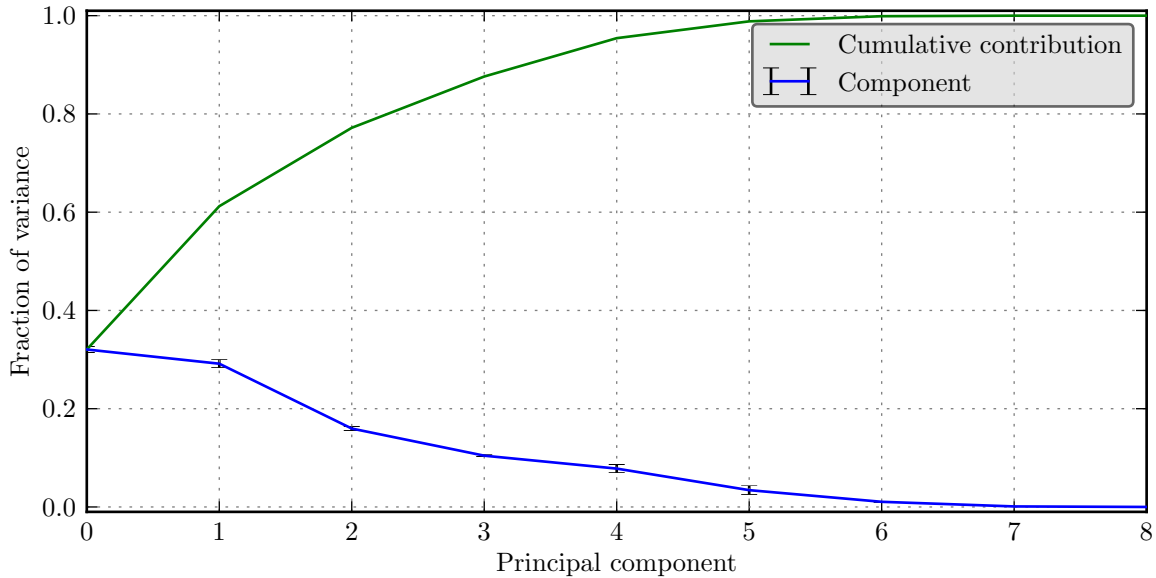


Figure 7.7: Fraction of variance explained by successive PCA components. The first two components, alone, explain more 60% of the total data variance. The first three components explain approximately 75% of the variance.

Many possible features are highly correlated. For example, the mean and median are unlikely to differ greatly for any of these features. To reduce the set of features considered here, this table lists the aggregate functions that were not observed to be highly correlated, as described in the following section. That is, this restricted set is as explanatory as the inclusion of all aggregate functions for each of these factors.

7.9.1 Principal component analysis

The calculation of features for each cluster yields a $k \times N$ matrix, for k features and N clusters. Principal component analysis (PCA) [73] has been performed upon this matrix to accomplish two goals. First, PCA reduces the dimensionality of this feature space. Second, it facilitates the investigation of which individual features are most explanatory. The space derived from PCA lacks discrete, identifiable features, reducing one's ability to intuit about what a value in a dimension might mean. If a small number of features are equivalent in explanatory power as the derived PCA component space, the use of the features directly would allow greater intuition of the result.

The result of PCA is the construction of a new parameter space where each dimension, or component, is a vector in the original feature space. Each vector is aligned along the axis of maximum variance of the data, such that the first component accounts for the greatest variance, followed by successive components. For the dimensionality of the space to be reduced in an effective manner, the desired result is that the first few components explain the majority of the variance in the original feature space. The amount of variance explained by individual components is given in Figure 7.7. The standard deviation of the fraction explained by each component is indicated by individual error bars, and are effectively of magnitude zero. These have been calculated over 1000 trials where 75% of the data is sampled. This *Scree* plot indicates the fraction of variance explained by each successive component in the transformed PCA space. The primary observation to be made is that

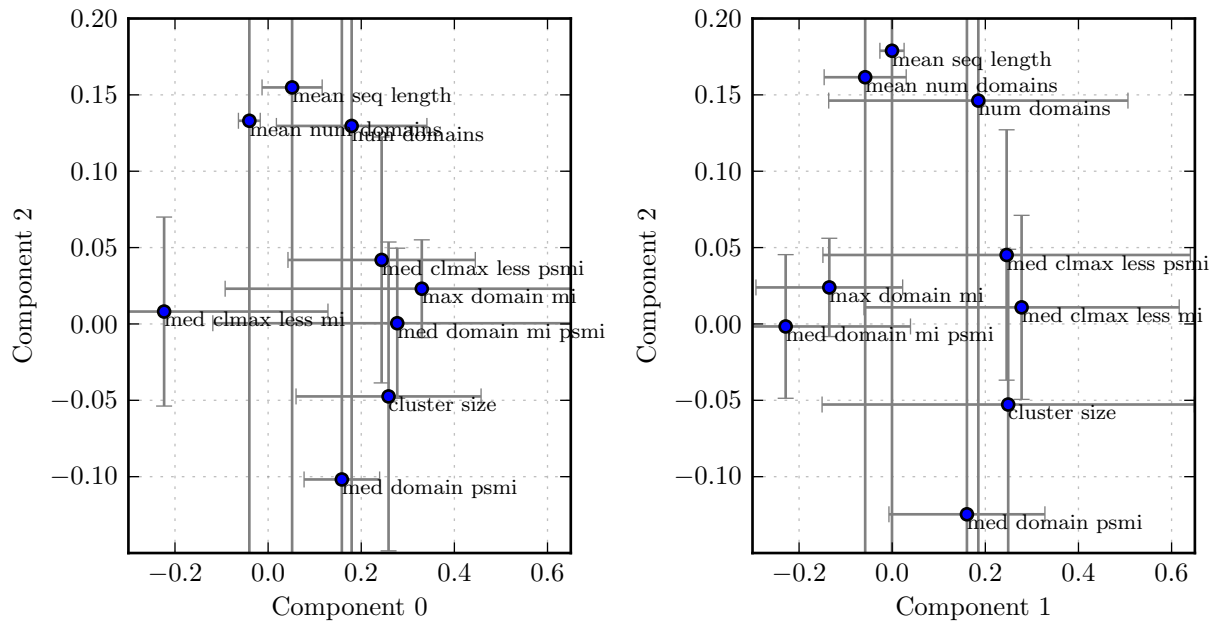
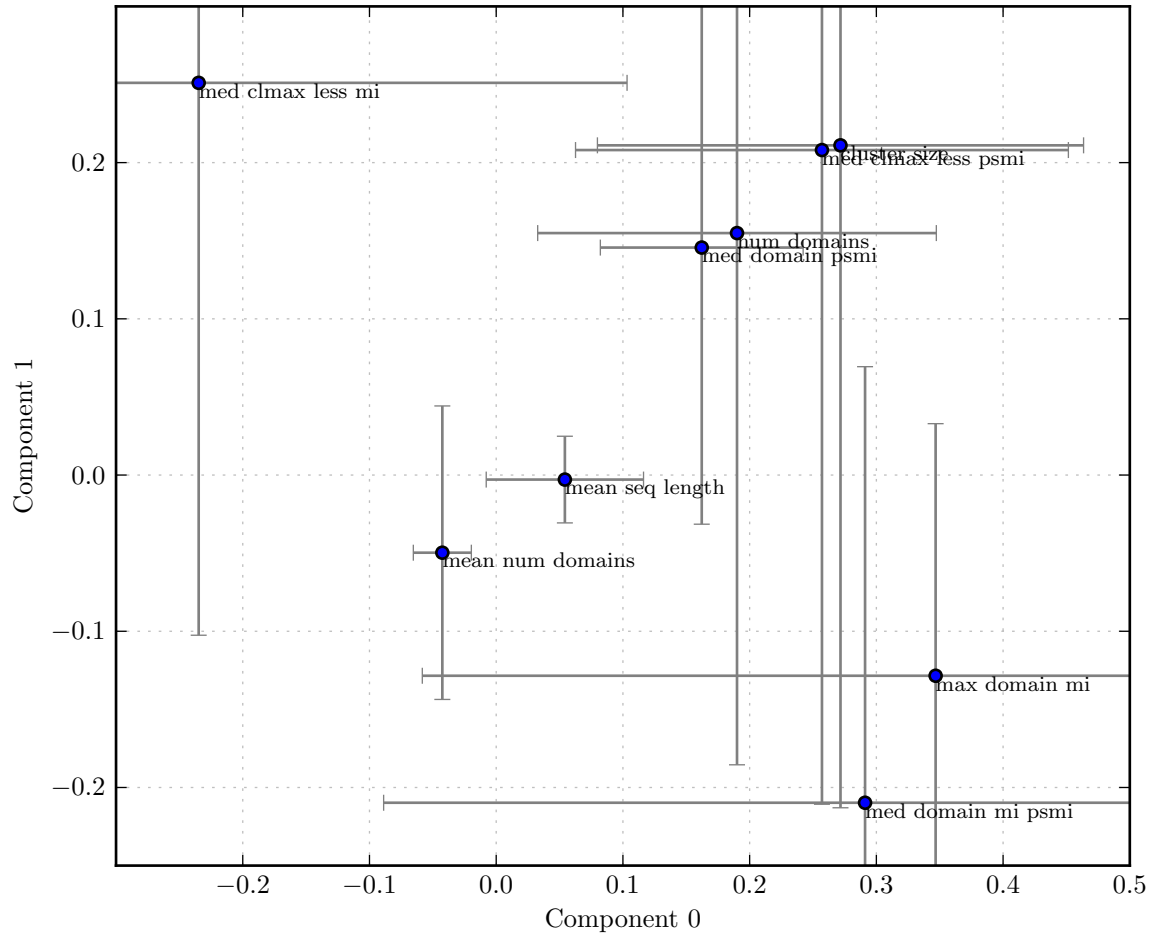


Figure 7.8: Coefficients of individual features in the PCA component space. Error bars indicate the standard deviation over 1000 samplings of 75% of the clusters in the clustering.

the first two components account for in excess of 60% of the variance in the feature space, and the first three explain greater than 75% of the variance. As a result, the remainder of this section considers the first three components.

Figure 7.8 shows the contribution of individual features to the PCA component space. Points indicate the component values when computed on the entire dataset. Error bars again indicate the standard deviation over 1000 trials where 75% of the clusters are used as input to the PCA procedure. A low standard deviation of the magnitude to which a feature contributes to a component indicates that the component is stable with respect to variation of the input data. A high standard deviation can result from several causes, and does not necessarily indicate that the features that contribute to a component are sensitive to the data. For example, consider that the sign of the coefficients of a component may be reversed, without changing the significance of component, or the features that contribute to it. Oscillation of sign would contribute to a high standard deviation. This effect appears to be responsible for the majority of the standard deviations measured here. Overall, the PCA component space is observed to be stable.

Figure 7.8 also provides insight into the relationship between individual features. For example, `cluster-size` and `med-clmax-less-psmi` are co-located in components 0 and 1, suggesting that their values are very similar, at least with respect to the approximately 60% of the variance that these components capture. The omission of either feature would not qualitatively alter either of the first two components, or the total proportion of variance that they explain. Component 2 does separate these features widely.

Other features, such as `med-clmax-less-mi` have large magnitudes and are distinctly separate from the other features, in both components 0 and 1. Such features may be interpreted to measure a property that is not captured by any other feature. Additionally, the contribution of features derived from values other than the information theoretic measures developed in this chapter may be investigated. Notably, the features `mean-seq-length` and `mean-num-domains` are found in a region of components 0 and 1 that does not substantially overlap with the coefficient of any other feature.

7.9.2 Projection of clusters

A primary goal of the information theoretic measures developed in this chapter is to characterize the properties of clusters. Figure 7.4 considered the individual domains that comprise each cluster, but that presentation does not directly facilitate the comparison or grouping of clusters into classes related to their biological properties. The projection of clusters into the PCA coordinate space allows one to group clusters with respect to features derived from the domain information measures.

Figure 7.9 represents a projection of all clusters in the human and mouse clustering into the PCA component space. These figures do not directly represent biological properties, though empirical examination of individual clusters provides some insight into the dominant structural properties captured by the PCA coordinate space. The following discussion is a sampling of notable clusters.

The most notable feature of Figure 7.9(a) is a bifurcation of clusters, approximately along the lines $y = x$ and $y = -x$. This separation does appear to correspond to a biologically relevant separation. In particular, clusters that lie approximately along the upper line, from coordinate (0,0), tend to be very large in size (Figure 7.10), and contain a great many domains (Figures 7.11). The domain

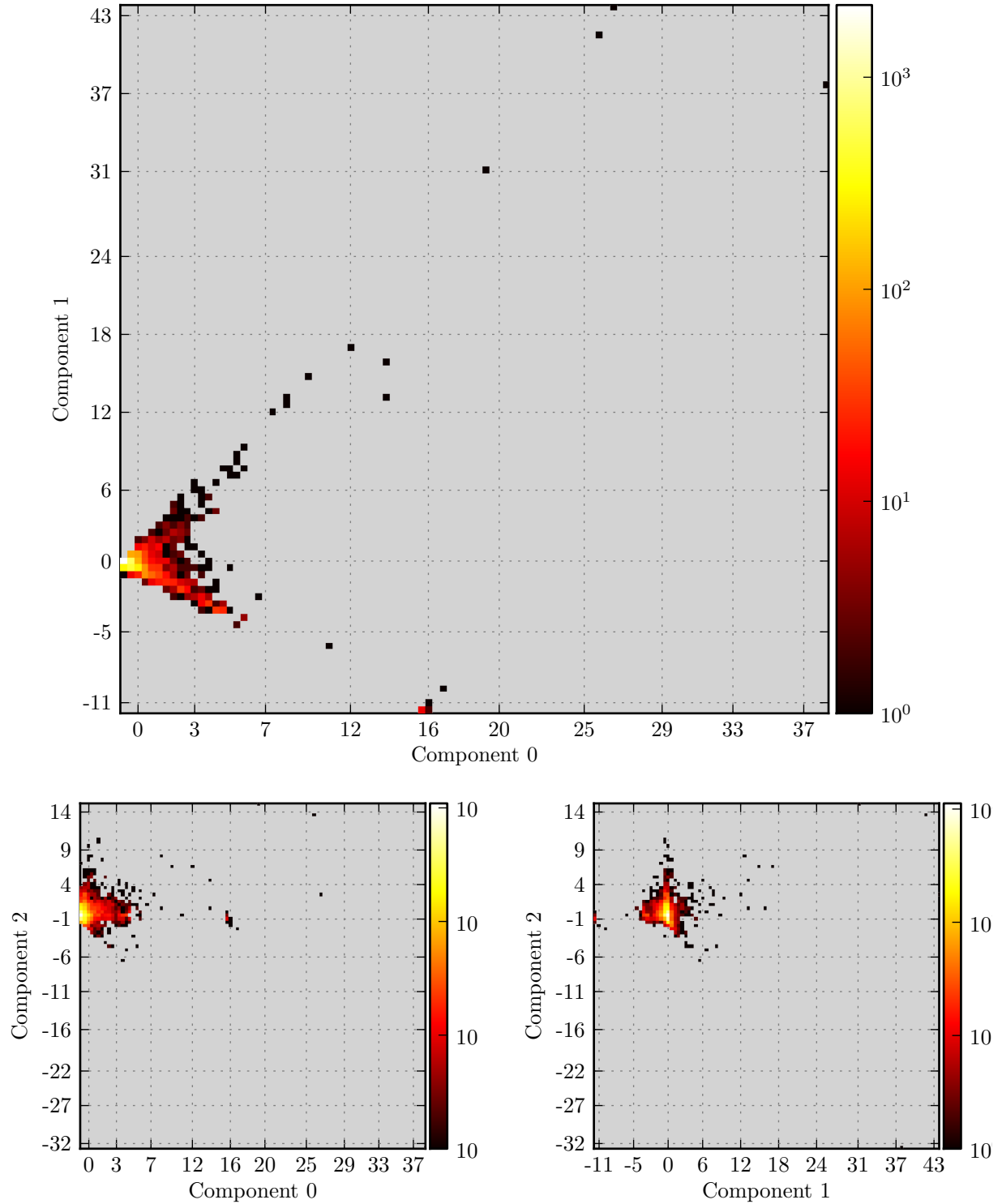


Figure 7.9: Projection of clusters into PCA component space. This figure is a heatmap of 100x100 bins, where black points indicate a single cluster, and increasingly bright points represent as 3000 clusters at that coordinate.

content of these large clusters may also be observed in Figure 7.4(b).

For example, the two clusters near coordinate (26,42) are large. The upper cluster contains 1380 sequences, and 21 different domains. This cluster contains all of the 1115 zinc-finger C2H2 domains in human and mouse, as well as the majority (662 of 682) of KRAB domains. The other cluster contains 920 sequences and a total of 89 different domains. This cluster corresponds to the curated Kinase family. It contains most (696 of 753) Pkinase domains in these genomes.

The cluster at coordinate (38,37) is comprised of 1568 sequences, nearly all of which (1552) contain a 7-transmembrane-1 domain, representing a majority of the total of 2150 sequences that have this domain. None of these sequences contain any other domain found in the PFM database.

Note, however, that not all clusters in this area of the PCA coordinate space necessarily have a single domain that is found in the majority of the sequences that comprise a cluster. The cluster near coordinate (19,31) contains 530 sequences, and 93 different domains, including SH3, CH, and Spectrin. While a large number of different domains are represented, no single domain occurs in greater than 15 sequences of this cluster.

Consistent with the large cluster sizes observed along the line $y = x$, the three clusters nearest to coordinate (13,15) range in size from 314 to 489 sequences, with 6 to 40 different domains in each cluster. Sequences within each of these clusters predominantly contain a single domain. Top to bottom, these clusters are characterized by (1) the Ankarin repeat and SOCS box domains (2) the majority (489) of the 7-transmembrane-1 domains not accounted for above, and (3) the Leucine Rich Repeat 1 and I-set domains.

Continuing further down the $y = x$ line toward the origin, cluster sizes decrease (Figure 7.10), with clusters of approximately 100 sequences near the coordinate (3,3). Clusters near this position contain 1-15 different domains (Figure 7.11), with a single domain found in the majority of sequences in each cluster.

Clusters along the line $y = -x$ in Figure 7.9(a) are very small. Near coordinate (0,0), most clusters are pairs of sequences. Continuing downward, toward (3,-4), clusters are comprised almost entirely of singletons (Figure 7.10). The clusters below a value of -6 for component 1 are small clusters, ranging in size from 1 to 42 sequences. Interestingly, all of these clusters contain the 7-transmembrane-1 domain, and these sequences account for nearly all of the remaining instances of this domain not already noted above. Only one cluster, at (10.6,-6.35), contains any other domain. That domain, Sp100, is found in a single sequence.

Correspondence with biological features

Having considered the general trends of cluster size and number of contained domains when clusters are projected into the PCA component space, it is useful to consider how the separation of clusters in this space corresponds to our intuition of categories of families. For example, the 20-family benchmark used throughout this dissertation is most directly categorized into families of single-domain architectures and those that are multidomain, with either conserved or variable domain architectures. Further intuition suggests that the majority of families are characterized by a single domain found in nearly all members of the family, and other domains contribute auxiliary or specific function to individual sequences.

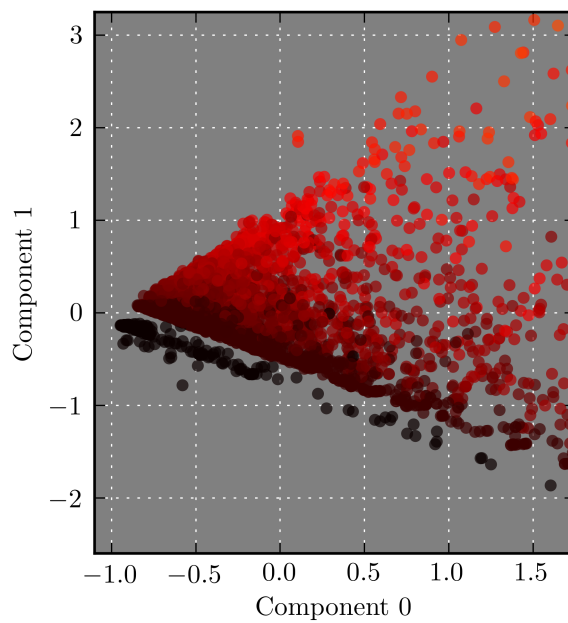
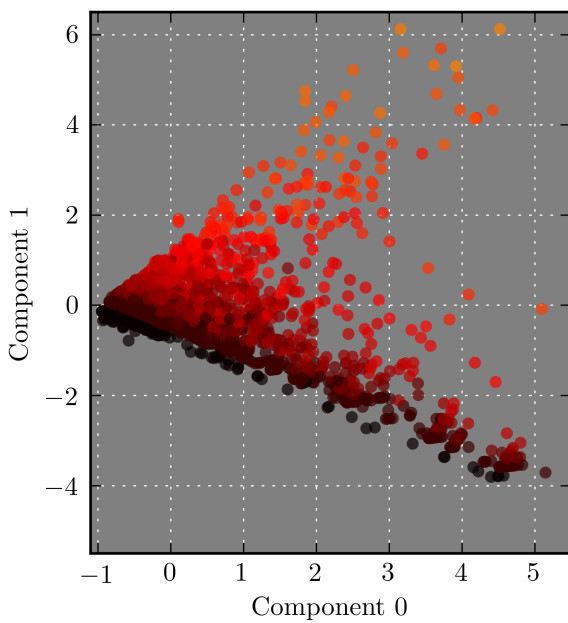
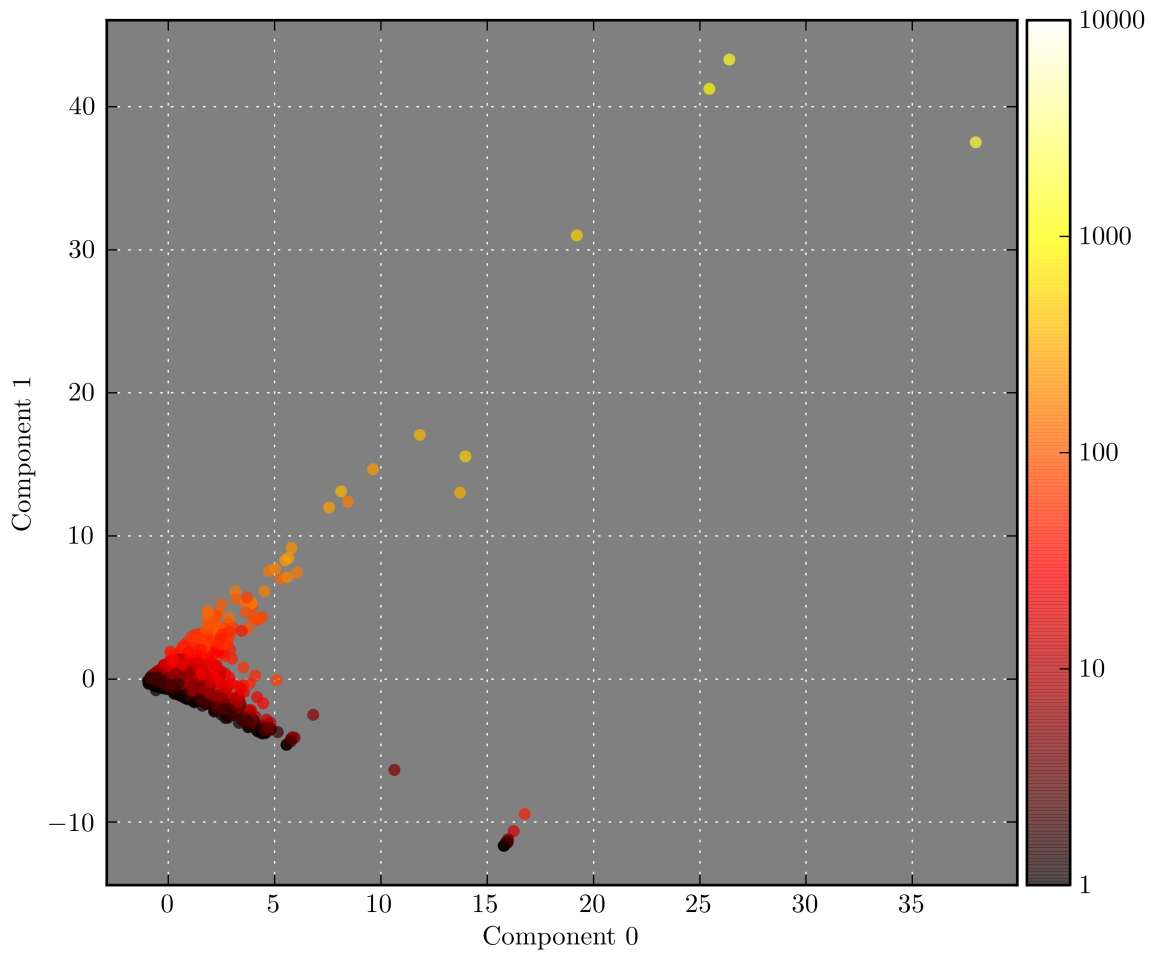


Figure 7.10: Projection of clusters into PCA component space. Clusters in this figure are colored according to the number of sequences in found in that cluster.

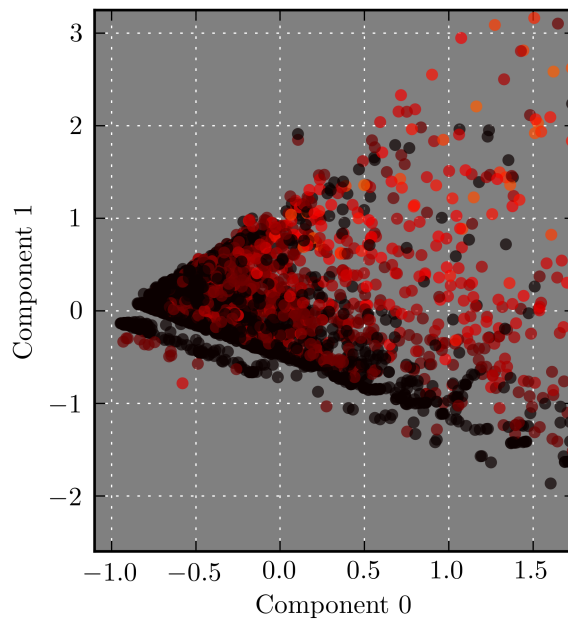
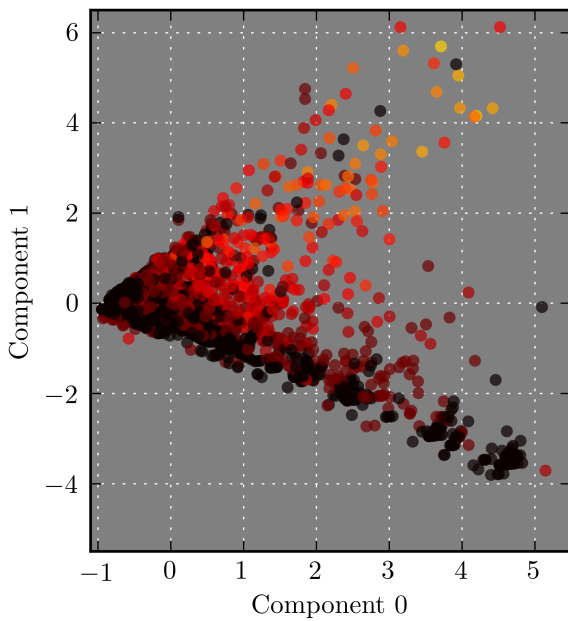
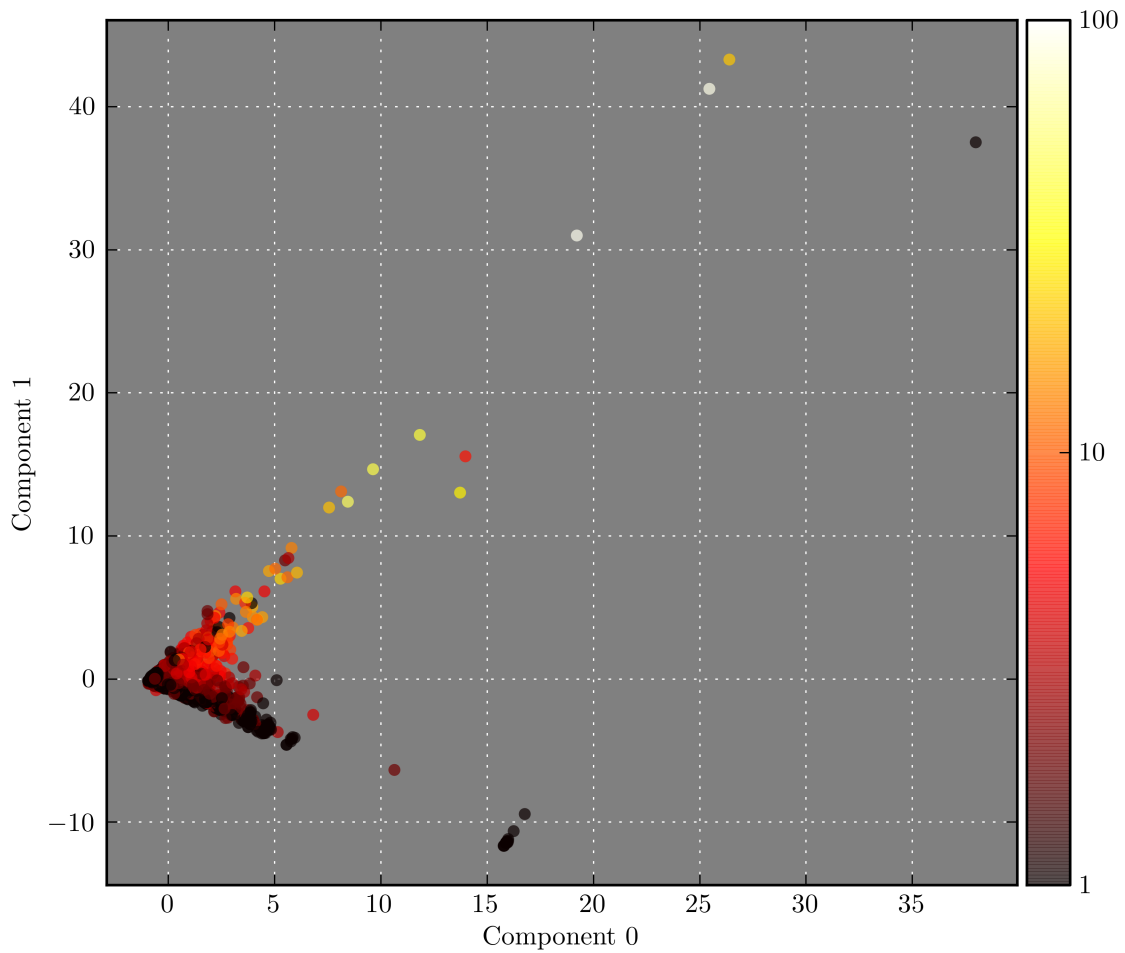


Figure 7.11: Projection of clusters into PCA component space. Clusters in this figure are colored according to the total number of distinct domains found in that cluster.

Does the PCA coordinate space highlight other classes of families that are not represented by our family benchmark? To address this question, Figure 7.12 overlays the curated families atop a scatter-plot of clusters the space of PCA components 0 and 1. Families have been colored by the categories listed in Table 2.1, with single-domain families in red, multidomain families with conserved architectures in cyan, and those with variable architectures in orange. Because the clustering result does not necessarily assign all members of a family to a single cluster, more than one cluster may have sequences of a particular family. In those cases, the most representative cluster is depicted as a solid color, and each other cluster is outlined.

Considering the entire space, the most notable observation is that a very few families share characteristics with the Kinase family. Domain content of the clusters surrounding the most representative Kinase cluster were discussed above. A second dominant observation is that clusters which are not highly representative of a family, but contain some members, are widely separated from more accurate clusters, and lie along the lower $y = -x$ line in the space of components 0 and 1. These clusters tend also to be very small, containing only approximately 1 – 5 sequences per cluster. With respect to known families, this characteristic suggests that clusters along this line may represent sequences that were inappropriately partitioned from the containing family.

Looking closer, with the lower two plots in Figure 7.12, the clusters that best represent families appear to separate to some degree by category. Excepting the TNF family, which presents particular challenge to the clustering performed here, all single-domain families are localized to a small region near coordinate (1,2), though the FOX family is at coordinate (2,4), most likely because it is a very large in comparison to the other single-domain families in the benchmark.

Multidomain families with conserved architectures are found distinctly lower with respect to component 1. Note that the Notch family is particularly poorly represented by the use of the Neighborhood Correlation threshold chosen here. The F-statistic, shown in Figure 6.9, illustrates that a very high threshold would be required to correctly classify this family. The 8 members of Notch are found in a single cluster at coordinate (8.4,12.4), though these sequences are commingled with 160 sequences in total, and 47 domains. That is, this cluster does have composition as would be seen in a large family of variable domain architecture, but happens not to correspond to what we believe is a correct, curated family. Similarly, members of the KIR family are found in a larger cluster that contains other unrelated sequences.

The USP family cluster appears separate from other families with conserved architectures. Here, the clustering result is quite accurate. While the USP family is regarded as having a conserved architecture, the family does contain domains that are found only in a small minority of sequences in the family. The cluster at (3.2,5.6) contains 86 of the 100 USP family members, and 11 sequences that are not known to be part of USP. Each of the remaining clusters are of size 1–4 and each contain only members of USP. As may be observed from Figure 6.9, the threshold selected here is slightly too stringent for this family.

Multidomain families with variable architectures are found more distant from the origin than either single-domain or multidomain families with conserved architecture. Nearly all are in the region of coordinate (3,3). As can be expected from the size distribution observed in Figure 7.10, larger families are found more distant from the origin along the line $y = x$.

Overall, these data suggest that the curated benchmark has broad coverage of families with at least

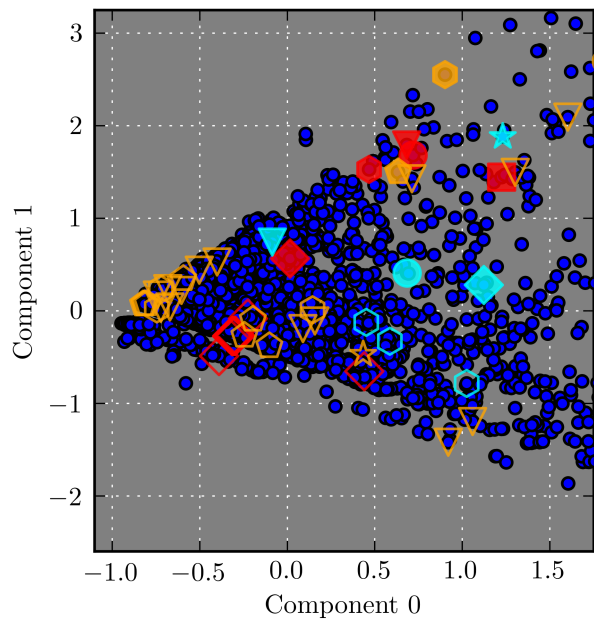
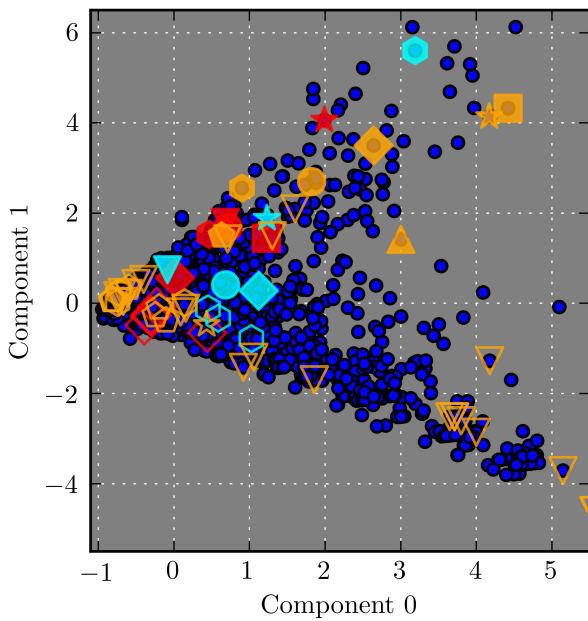
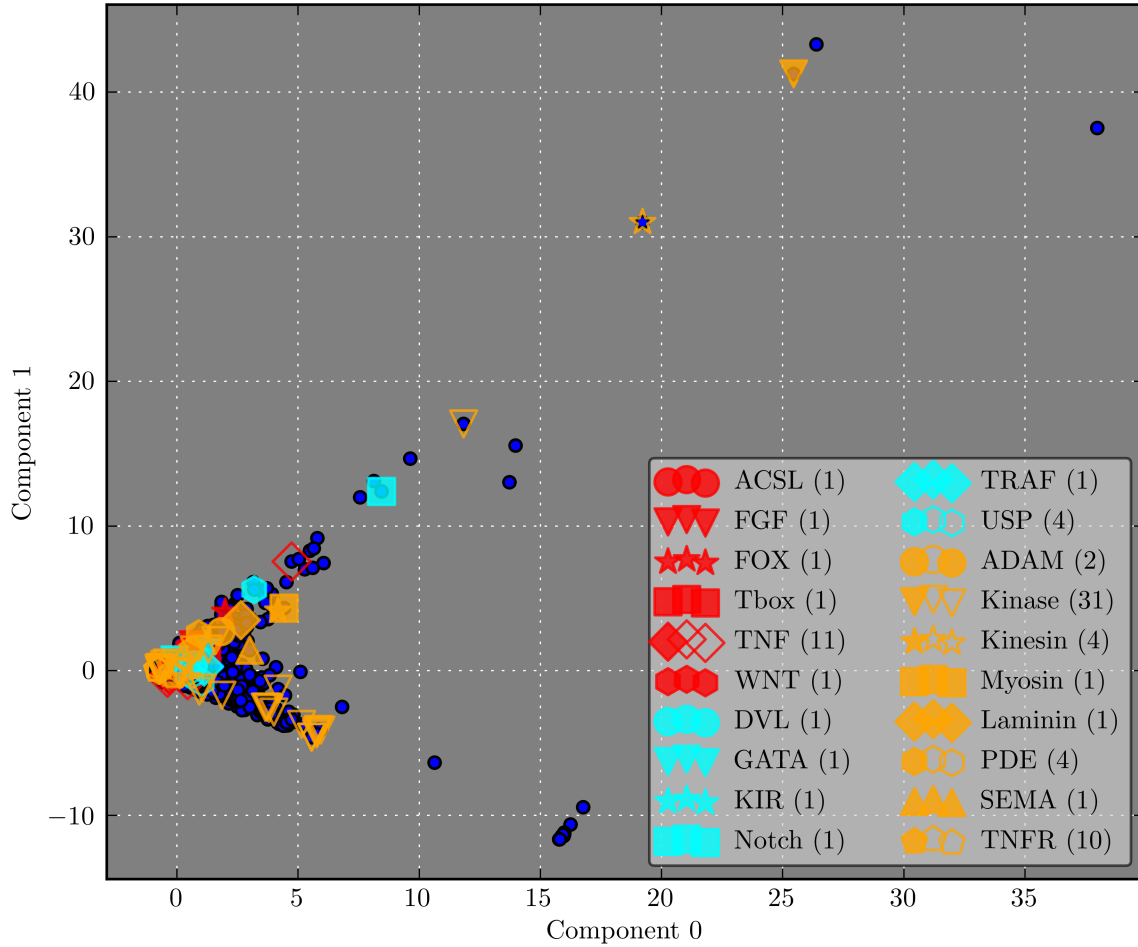


Figure 7.12: Projection of clusters into PCA component space. Clusters that contain at least one curated family member are outlined. The most representative cluster is shaded, and the number of clusters is in parentheses.

approximately 10 members. Among larger clusters, there do not exist broad regions of the PCA component space where no family is found. However, no family is found near the origin, or below the coordinate of 0 in component 1. As seen in Figure 7.10, these clusters contain fewer than 20 sequences each. It remains to be seen whether the curated benchmark therefore fails to represent the class of families found in this region, or whether they result from an artifact of clustering. Those clusters at the lowest extreme, along $y = -x$, are most likely to be fragments of larger families, such as the Kinase family members observed in this region.

Correspondence with information features

The preceding discussion concerned the characterization of the PCA component space with respect to (1) features that correspond directly to intuitive measures, such as the size of a cluster, or the number of domains found within it, and (2) known families. This section considers features that directly correspond to the information measures developed in this chapter.

Figure 7.13 depicts clusters, annotated by `med-clmax-less-psmi`. The value `med-clmax-less-psmi` represents the maximum possible cluster mutual information less the presence specific mutual information of each contained domain. To recall, the specific mutual information is the quantity that the presence of a domain in a specific cluster contributes to the summation for the mutual information of a domain with respect to all clusters. The difference between this value and the maximum possible mutual information for a cluster is used as a means of normalizing for cluster size.

This figure demonstrates that values of `med-clmax-less-psmi` are relatively constant along any line $y = -\frac{2}{3}x$, and increase in a nearly monotonic fashion from the lower left coordinate of this plot to the upper right. First, this illustrates that neither component directly recapitulates the variance of `med-clmax-less-psmi`. Indeed, looking back to Figure 7.8, such a result may be expected from the large coefficients of this value in both components 0 and 1.

Figure 7.14 is of similar construction to the previous figure. Here, the difference between the maximum possible mutual information and the mutual information of each domain is considered. Values in this figure are observed to increase monotonically with increasing values of component 1. A notable exception to this trend is the cluster found at (38,37), which, as discussed above, contains only sequences with the 7-transmembrane-1 domain. Our understanding of families suggests that that cluster is an oddity, and is unlikely to be a true family.

Clusters in the lower portion of this figure tend to have negative values, meaning that they contain domains that have large values of mutual information, but *not* due to their membership in these clusters. This provides further evidence that the lowest clusters with respect to component 1 are likely to correspond to partial families.

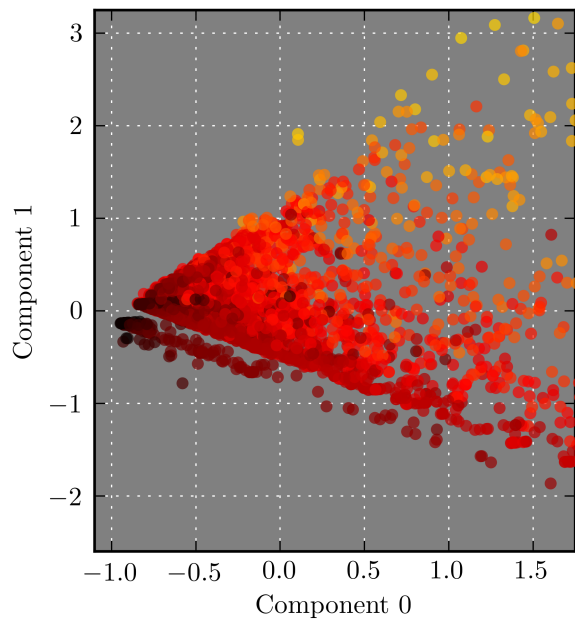
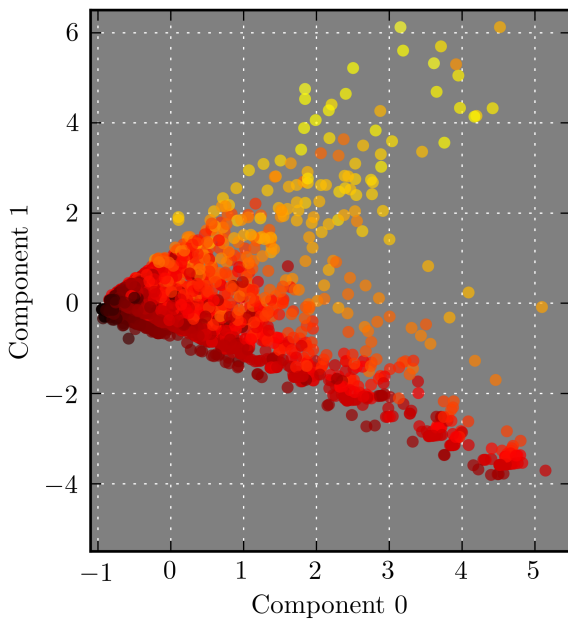
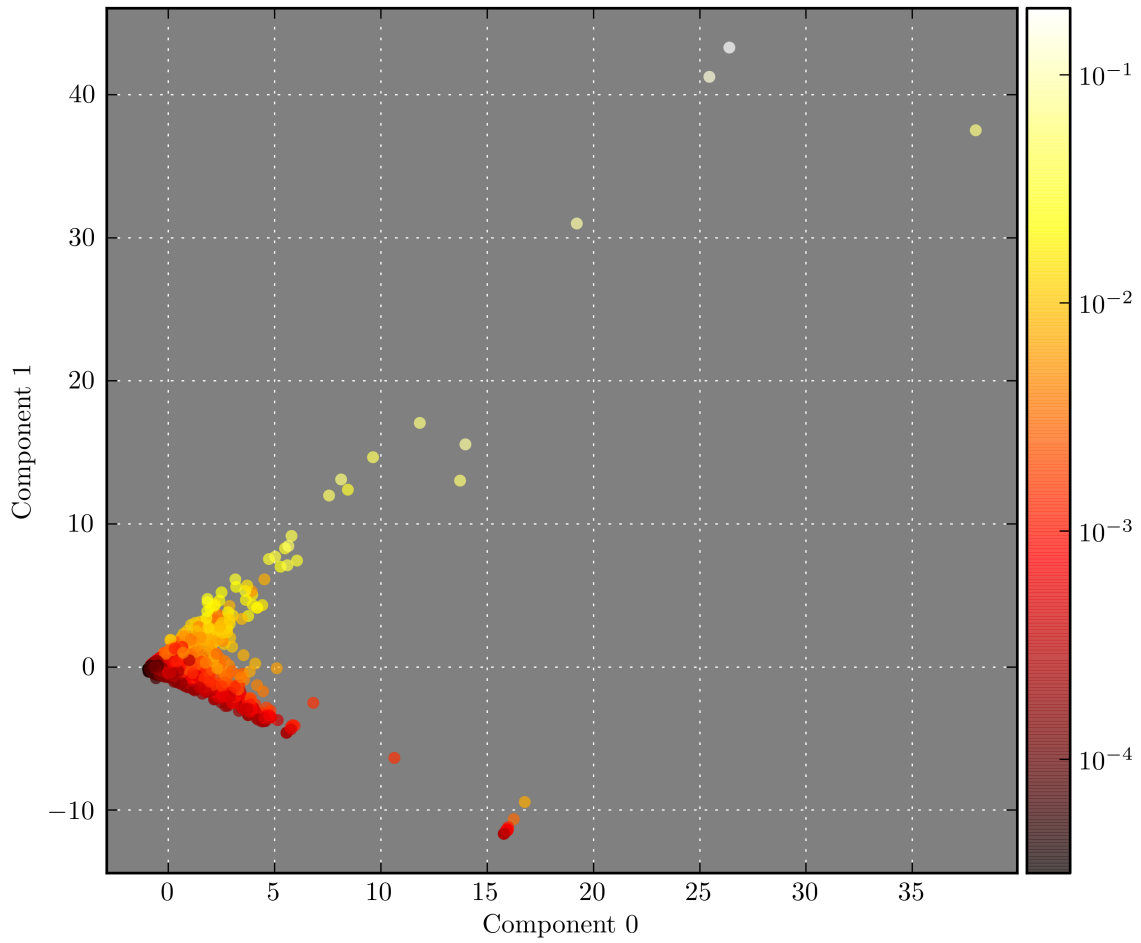


Figure 7.13: Projection of clusters into PCA component space. Clusters in this figure are colored according to med-clmax-less-psmi.

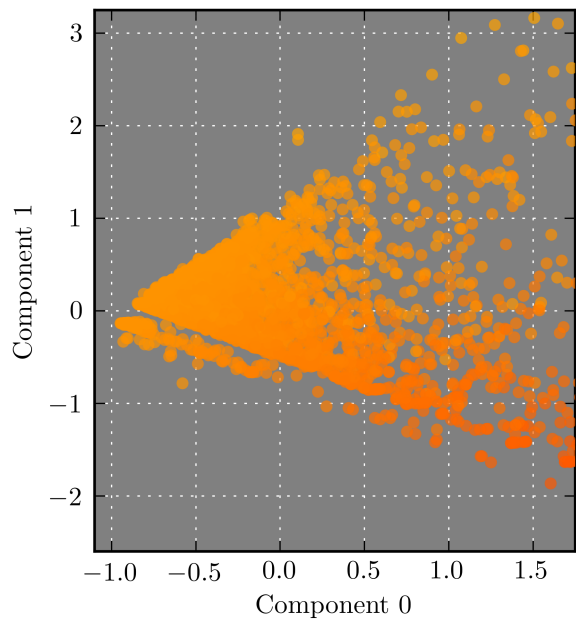
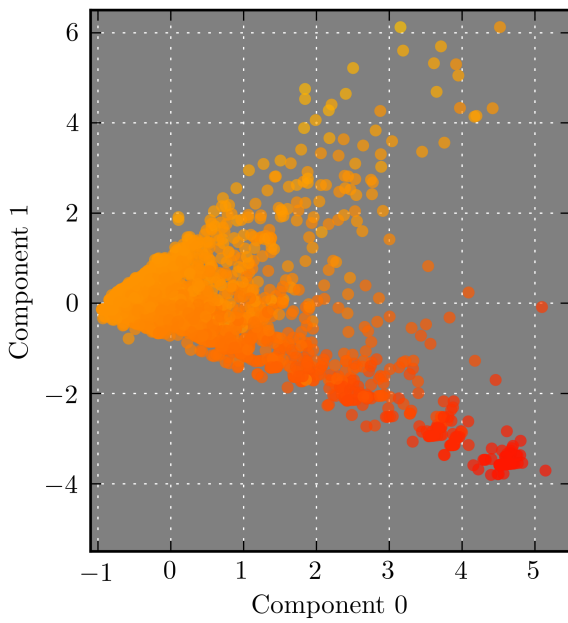
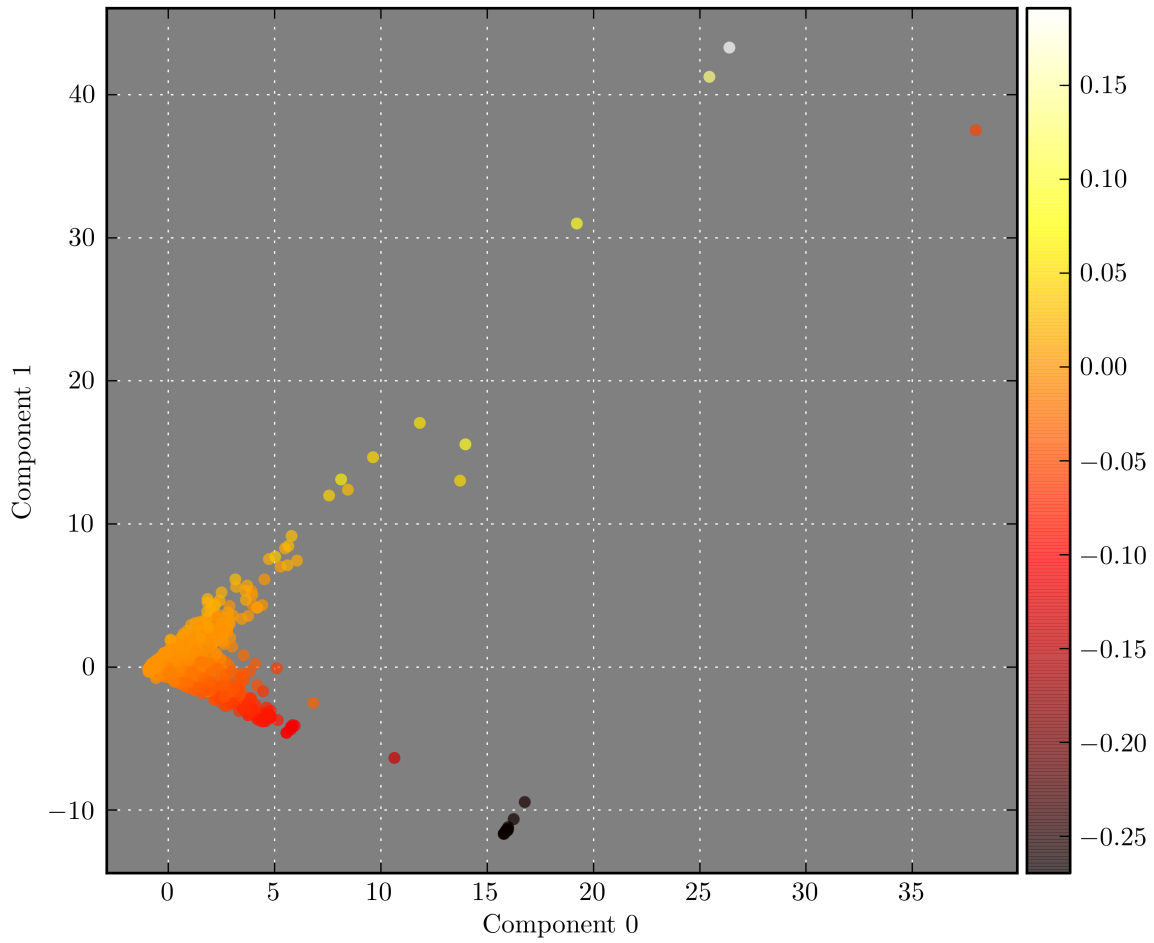


Figure 7.14: Projection of clusters into PCA component space. Clusters in this figure are colored according to med-clmax-less-mi for that cluster.

Conclusions and future directions

This dissertation has presented a theoretical context for family classification, and has demonstrated effective means for performing classification on large data sets. A specific summary of results is presented in Section 1.5. This chapter addresses the broader implications of the work.

The underlying rationale throughout this dissertation is that the history of evolution is reflected in the structure of the sequence similarity network. Here, this structure has been exploited to yield an effective family classification methodology, particularly by distinguishing the traces of gene duplication from those of domain shuffling. This suggests that network structure is indeed a reflection of the evolutionary processes. It leaves open the intriguing question of what other evolutionary processes may be identified using a similar framework.

A lack of knowledge about genome-wide properties of gene families has posed a challenge for the development and validation of effective family classification methods. A complicating factor in considering those existing methods designed to detect homology is that some do not appear to use the definition of homology, which is an evolutionary relationship. Often the definition used is only implicit and ambiguous. In other cases, where the definition used is explicit, statements such as “...we use the term ‘phylogenetic’ in the sense of ‘relatedness of biological functions’” [83] are not uncommon. Disagreements in definitions affect the data used to evaluate the accuracy of a method, and weaken the basis for biologically motivated conclusions. The definition of homology developed in this dissertation encompasses multidomain gene families, and defines of a concrete goal for family classification.

A rigorous definition of the goal of family classification also facilitates evaluation. Validation of the result of family classification is of critical importance if these methods are to receive broader use. To this end, two primary means of validation have been introduced. First, a statistical framework for *extrinsic* evaluation has been presented, and the accuracy of the family classification methodology developed here has been concretely demonstrated. The observed performance on a curated benchmark suggests that the methodology may be well-suited for classification of families in other data sets. Additionally, a framework for *intrinsic* validation using features of the data itself has been developed. This is key to use when little is known before classification about the family structure of the data set — a typical scenario.

The combination of family classification methods and approaches for validation suited toward large

data sets addresses a critical niche in the contemporary analysis of genomic sequence data. For much of the past decade, these data have increased exponentially in size, at least as fast as increases in computational capacity. Beyond the value of addressing the engineering challenges, the ability to use large data sets expands the range of possible studies. The availability of such data makes it practical to ask questions and pose hypotheses that may be addressed only at large scale. For example, the investigation of evolutionary differences between lineages tends to require data sets that are large enough to encompass the lineages under study, and most likely, other related lineages. Other, novel biological properties may only be evident when a very large quantity of data is used, sometimes because the signal is weak, and, in other instances, because the property is only observed across broad evolutionary distances.

Efforts to develop large public databases of gene families has illustrated that the biological community has a need for such data [72, 103, 119, 146]. Toward fostering broader use, and addressing the practical engineering challenges of large-scale classification, I have collaborated with the Princeton Protein Orthology Database [72] to bring more accurate family predictions to their database. The PLAZA database [119] has also shown interest in the classification pipeline I have developed.

The relationship between domain content and gene function has long been a focus of study in this field. This work extends the range of study to include gene families, a natural grouping of evolutionary properties. For example, it is widely believed that family members tend to share biological function. The accurate prediction of families can be used to evaluate the generality of this hypothesis in families that as yet have not been identified, or predicted. Additionally, the information-theoretic approach developed in this dissertation comprises an initial step toward characterizing the domain content of predicted families. The results in Chapter 7 support the conclusion that domain content, alone, cannot be used to recapitulate the family structure predicted by the methods developed in this thesis. Rather than a detriment, this quality suggests that domain content may now be used to better describe the predicted families. Predicted families may be better understood when they may be organized based on such description. Toward this end, a set of features have been used to group predicted families into classes that facilitate intuition about biology similarities, and differences.

In the greater context, this dissertation comprises the groundwork for large scale studies of the biological properties of gene families. Currently, studies of biological properties have been partitioned into two extremes. Detailed studies of individual families have provided a wealth of knowledge of the evolutionary processes that give rise to gene families. These have provided many examples of families, and have guided the methodology developed in this dissertation. At the same time, large-scale studies of complete genomes have focused upon individual entities, such as domains or sequences. However, an inability to accurately identify gene families has stymied inference with respect to the evolutionary background of gene families in large-scale data.

Infrastructure

A.1 Software

Please refer to http://www.jjoseph.org/phd_dissertation for repositories of my software and data.

A.1.1 Existing tools and packages

Much of the software implementation in this dissertation has been produced in the Python [143] programming language. A variety of additional Python packages have been employed, including the following. The numerical packages, Scipy [77] and Numpy [112], have been used to facilitate efficient numerical computation. The NetworkX [67] has been used for calculation of network metrics. The Biopython [34] package has been used to parse BLAST and source database XML formats. Most plots in this document have been produced with Matplotlib [13]; R [123] has been used for the remainder.

The relational database structure described in this document has been implemented using PostgreSQL [65].

The implementation by Lowenstein *et al.* [96] has been used for average-linkage, hierarchical clustering.

A.1.2 Developed software

All software developed as part of this dissertation is supplied under the terms of the GNU General Public License, Version 3 [57]. This includes implementations of all methods described in this document, the schema of the relational database, and all code used to produce the data presentation (i.e., figures) in this document. Copyright to all of the above is retained by myself, Jacob Joseph, and Carnegie Mellon University.

Additionally, and central to the use of this work, I will supply the contents of the relational database used as the central data source in this dissertation work. A variety of data sources comprise the source data used in this database, including NCBI, SwissProt, Ygob, and Panther. I place all derived data in the public domain. However, not all of these sources provide a clear license for reuse or

distribution. Nonetheless, to the best of my knowledge, these data sources exist to facilitate broad use.

A.2 SQL Schema

```
----- durandlab2_schema.sql -----
1 CREATE TABLE taxon (
2     tax_id      integer,
3     name        text NOT NULL,
4     common_names text[],
5     PRIMARY KEY (tax_id)
6 );
7
8 CREATE TABLE prot_seq_source (
9     source_id    serial,
10    source_name   text UNIQUE NOT NULL,
11    PRIMARY KEY (source_id)
12 );
13
14 CREATE TABLE prot_seq_source_ver (
15     source_ver_id serial,
16     source_id     integer NOT NULL REFERENCES prot_seq_source ON DELETE RESTRICT,
17     version       text NOT NULL,
18     date          timestamp NOT NULL,
19     PRIMARY KEY (source_ver_id),
20     UNIQUE (source_id, version, date)
21 );
22
23 -- these entries should be unique. Unfortunately mysql cannot have a
24 -- unique entry longer than 767 bytes (if sequence were a varchar) or
25 -- include a text field. BEWARE of (crc,length) conflicts! This
26 -- restriction may later be removed if it poses a problem. Length
27 -- need not be stored, but it could help in the event of conflicts.
28 -- adding molecular_weight eliminates all conflicts so far
29 CREATE TABLE prot_seq_str (
30     seq_str_id    serial,
31     sequence      text NOT NULL,
32     crc           text NOT NULL,
33     length        integer NOT NULL,
34     molecular_weight integer NOT NULL,
35     PRIMARY KEY (seq_str_id),
36     UNIQUE (crc, length, molecular_weight)
37 );
38
39 CREATE TABLE prot_seq (
40     seq_id        serial,
41     seq_str_id    integer NOT NULL REFERENCES prot_seq_str ON DELETE RESTRICT,
42     PRIMARY KEY (seq_id)
43 );
44
```

```

45 CREATE TABLE prot_seq_version (
46     seq_id            integer NOT NULL REFERENCES prot_seq ON DELETE RESTRICT,
47     source_ver_id    integer NOT NULL REFERENCES prot_seq_source_ver ON DELETE RESTRICT,
48     primary_acc       text NOT NULL,
49     PRIMARY KEY (seq_id, source_ver_id),
50     UNIQUE (source_ver_id, primary_acc)
51 );
52
53 CREATE TABLE prot_seq_accession (
54     seq_id            integer NOT NULL REFERENCES prot_seq ON DELETE RESTRICT,
55     source_id         integer NOT NULL REFERENCES prot_seq_source ON DELETE RESTRICT,
56     acc_order         smallint NOT NULL, -- 0,1,2,...
57     accession         text NOT NULL,
58     PRIMARY KEY (seq_id, acc_order)
59 );
60
61 CREATE TABLE prot_seq_set (
62     set_id            serial,
63     name              text NOT NULL,
64     description       text NOT NULL,
65     PRIMARY KEY (set_id),
66     UNIQUE (name, description)
67 );
68
69 CREATE TABLE prot_seq_set_member (
70     set_id            integer NOT NULL REFERENCES prot_seq_set ON DELETE RESTRICT,
71     seq_id            integer NOT NULL REFERENCES prot_seq ON DELETE RESTRICT,
72     PRIMARY KEY (set_id, seq_id)
73 );
74
75 -----
76 -- Swiss-Prot specific info
77 -----
78 CREATE TABLE sp_gb_tax_id (
79     seq_id            integer NOT NULL REFERENCES prot_seq ON DELETE RESTRICT,
80     gb_tax_id         integer NOT NULL,
81     PRIMARY KEY (seq_id)
82 );
83
84 CREATE TABLE sp_feature_key_str (
85     feature_key_strid serial,
86     feature_key       text UNIQUE NOT NULL,
87     PRIMARY KEY (feature_key_strid)
88 );
89
90 CREATE TABLE sp_feature (
91     seq_id            integer NOT NULL REFERENCES prot_seq ON DELETE RESTRICT,
92     feature_key_strid integer NOT NULL REFERENCES sp_feature_key_str ON DELETE RESTRICT,
93     from_pos          text NOT NULL,
94     to_pos            text NOT NULL,
95     comment           text NOT NULL,

```

```

96         ftid                text NOT NULL
97         -- PRIMARY KEY (seq_id, feature_key_strid, from_pos, to_pos, comment, ftid)
98     );
99     CREATE INDEX seq_id_key on sp_feature(seq_id);
100
101     CREATE TABLE sp_keyword_str (
102         keyword_strid    serial,
103         keyword          text UNIQUE NOT NULL,
104         PRIMARY KEY (keyword_strid)
105     );
106
107     CREATE TABLE sp_keyword (
108         seq_id           integer NOT NULL REFERENCES prot_seq ON DELETE RESTRICT,
109         keyword_strid   integer NOT NULL REFERENCES sp_keyword_str ON DELETE RESTRICT,
110         PRIMARY KEY (seq_id, keyword_strid)
111     );
112
113     CREATE TABLE sp_xtra (
114         seq_id           integer NOT NULL REFERENCES prot_seq ON DELETE RESTRICT,
115         -- data_class          enum('STANDARD', 'PRELIMINARY') NOT NULL,
116         entry_name       text NOT NULL,
117         gene_name        text NOT NULL,
118         description      text NOT NULL,
119         -- created             datetime NOT NULL,
120         -- sequence_update     datetime NOT NULL,
121         -- annotation_update   datetime NOT NULL,
122         PRIMARY KEY (seq_id)
123     );
124
125     -- store swissprot cross references
126     -- since the DR line does not seem well defined, simply store the
127     -- db name and next 4 semi-colon delimited fields
128     CREATE TABLE sp_cross_ref (
129         seq_id           integer NOT NULL REFERENCES prot_seq ON DELETE RESTRICT,
130         source_id        integer NOT NULL REFERENCES prot_seq_source ON DELETE RESTRICT,
131         id_0             text NOT NULL,
132         id_1             text NOT NULL,
133         id_2             text NOT NULL,
134         id_3             text NOT NULL,
135         --PRIMARY KEY (seq_id, source_id, id_0, id_1, id_2, id_3)
136     );
137     -- a primary key of all columns is very large. Just put a key for faster lookups
138     CREATE INDEX sp_cross_ref_seq_id on sp_cross_ref (seq_id, source_id);
139
140
141     -----
142     -- BLAST
143     -----
144     CREATE TABLE blast_run (
145         br_id            serial,
146         set_id           integer NOT NULL REFERENCES prot_seq_set ON DELETE RESTRICT,

```



```

147     date            timestamp NOT NULL,
148     num_sequences   integer NOT NULL,
149     num_residues    integer NOT NULL,
150     params          text NOT NULL,
151     comment         text,
152     blastall_path   text,
153     query_set_id    integer NOT NULL REFERENCES prot_seq_set ON DELETE RESTRICT,
154     PRIMARY KEY (br_id)
155 );
156
157 CREATE TABLE blast_lock (
158     br_id            integer NOT NULL REFERENCES blast_run ON DELETE RESTRICT,
159     query_set_id    integer NOT NULL REFERENCES prot_seq_set ON DELETE RESTRICT,
160     set_id          integer NOT NULL REFERENCES prot_seq_set ON DELETE RESTRICT,
161     chunk_size      integer NOT NULL,
162     max_chunk       integer NOT NULL,
163     chunk           integer NOT NULL,
164     start_time      timestamp,
165     end_time        timestamp,
166     hostname        text,
167     PRIMARY KEY (br_id, chunk)
168 );
169
170
171 CREATE TABLE blast_hit_arr (
172     br_id            integer NOT NULL REFERENCES blast_run ON DELETE RESTRICT,
173     hit_rank         smallint NOT NULL,
174     seq_id_0         integer NOT NULL REFERENCES prot_seq(seq_id) ON DELETE RESTRICT,
175     hit_list         int[],      -- int array. allows indexing and fast searching
176     bit_score        double precision[],
177     e_value          double precision[],
178     seq_start_0      integer[],
179     seq_end_0        integer[],
180     seq_start_1      integer[],
181     seq_end_1        integer[],
182     identities       integer[],
183     gaps             integer[],
184     positives        integer[],
185     low_complexity   boolean[],
186     coverage         double precision[],
187     PRIMARY KEY (br_id, hit_rank, seq_id_0)
188 );
189
190 -- store symmetric scores.
191 CREATE TABLE blast_hit_symmetric_arr (
192     br_id            integer NOT NULL REFERENCES blast_run ON DELETE RESTRICT,
193     seq_id_0         integer NOT NULL REFERENCES prot_seq(seq_id) ON DELETE RESTRICT,
194     hit_list         int[],
195     bit_score        double precision[],
196     e_value          double precision[],
197     PRIMARY KEY (br_id, seq_id_0)

```

```

198 );
199
200 CREATE TABLE nc_run (
201     nc_id          serial,
202     br_id          integer NOT NULL REFERENCES blast_run ON DELETE RESTRICT,
203     date           timestamp NOT NULL,
204     e_thresh       double precision NOT NULL,
205     bit_thresh     double precision,
206     nc_thresh      double precision NOT NULL,
207     blast_hit_limit integer,
208     smin           double precision NOT NULL,
209     smin_factor    double precision NOT NULL,
210     use_symmetric  boolean NOT NULL,
211     score_type     text NOT NULL,
212     self_hits      smallint NOT NULL,
213     PRIMARY KEY (nc_id),
214     UNIQUE (br_id, date, e_thresh, bit_thresh, nc_thresh, blast_hit_limit,
215            smin, smin_factor, use_symmetric, score_type)
216 );
217
218 CREATE TABLE blast_hit_nc_arr (
219     nc_id          integer NOT NULL REFERENCES nc_run ON DELETE RESTRICT,
220     seq_id_0       integer NOT NULL REFERENCES prot_seq(seq_id) ON DELETE RESTRICT,
221     --num_match_seq_0 integer NOT NULL,
222     hit_list       int[],
223     -- Using a composite type for the record means that indexing
224     -- the array takes O(i) time, as postgres assumes a variable
225     -- length element, and does a scan through the array. Using
226     -- fixed types allows O(1) access
227     -- record_list  nc_record[],
228     nc_score       double precision[],
229     --num_match_seq_1 integer[],
230     --num_match_seq_both integer[],
231     PRIMARY KEY (nc_id, seq_id_0)
232 );
233
234 CREATE TABLE nc_lock (
235     nc_id          integer NOT NULL REFERENCES nc_run ON DELETE RESTRICT,
236     chunk          integer NOT NULL,
237     chunk_min      integer NOT NULL,
238     chunk_max      integer NOT NULL,
239     start_time     timestamp,
240     end_time       timestamp,
241     hostname       text,
242     PRIMARY KEY (nc_id, chunk)
243 );
244
245 CREATE TABLE prot_seq_taxonomy (
246     seq_id         integer NOT NULL REFERENCES prot_seq ON DELETE RESTRICT,
247     gb_tax_id      integer NOT NULL,
248     PRIMARY KEY (seq_id)

```

```

249 );
250
251 -----
252 -- Pfam
253 -----
254 CREATE TABLE prot_seq_pfam (
255     seq_id          integer NOT NULL REFERENCES prot_seq ON DELETE RESTRICT,
256     align_start     integer NOT NULL,
257     align_end       integer NOT NULL,
258     envelope_start  integer NOT NULL,
259     envelope_end    integer NOT NULL,
260     hmm_acc         text NOT NULL,
261     hmm_name        text NOT NULL,
262     hmm_type        text NOT NULL,
263     hmm_start       integer NOT NULL,
264     hmm_end         integer NOT NULL,
265     hmm_length      integer NOT NULL,
266     bit_score       double precision NOT NULL,
267     e_value         double precision NOT NULL,
268     significance    integer NOT NULL,
269     clan            text NOT NULL,
270     PRIMARY KEY (seq_id, hmm_acc, align_start, align_end)
271 );
272
273
274 -----
275 -- Generic (FASTA insertion)
276 -----
277 CREATE TABLE fa_descr (
278     seq_id          integer NOT NULL REFERENCES prot_seq ON DELETE RESTRICT,
279     descr           text,
280     PRIMARY KEY (seq_id)
281 );
282
283 -- Allow mapping from FASTA sequence id to another id (typically
284 -- Uniprot with annotations)
285 CREATE TABLE fa_map (
286     seq_id          integer NOT NULL REFERENCES prot_seq ON DELETE RESTRICT,
287     seq_id_1        integer NOT NULL REFERENCES prot_seq(seq_id) ON DELETE RESTRICT,
288     PRIMARY KEY (seq_id)
289 );
290
291 -- Annotation of FASTA sequences, primarily for those from PPOD
292 CREATE TABLE fa_annotate (
293     seq_id          integer NOT NULL REFERENCES prot_seq ON DELETE RESTRICT,
294     synonyms        text,
295     descr           text,
296     PRIMARY KEY (seq_id)
297 );
298
299

```

```

300 -----
301 -- Derived data
302 -----
303 -- PFAM domain promiscuity
304 CREATE TABLE sp_promisc (
305     set_id          integer NOT NULL REFERENCES prot_seq ON DELETE RESTRICT,
306     id_0            text NOT NULL,
307     cnt             integer NOT NULL,
308     promiscuity     double precision NOT NULL,
309     PRIMARY KEY (set_id, id_0)
310 );

```

family_schema.sql

```

1 CREATE TABLE family_set (
2     fam_set_id     serial,
3     name           text NOT NULL,
4     description    text NOT NULL,
5     PRIMARY KEY (fam_set_id),
6     UNIQUE (name, description)
7 );
8
9 CREATE TABLE family (
10    family_id      serial,
11    name           text NOT NULL,
12    abbrev         text NOT NULL,
13    description    text,
14    PRIMARY KEY (family_id)
15 );
16 -- case insensitive unique
17 CREATE UNIQUE INDEX family_abbrev_unique on family (lower(abbrev));
18
19 CREATE TABLE family_member (
20    seq_id         integer NOT NULL REFERENCES prot_seq ON DELETE RESTRICT,
21    fam_set_id     integer NOT NULL REFERENCES family_set ON DELETE RESTRICT,
22    family_id      integer NOT NULL REFERENCES family ON DELETE RESTRICT,
23    PRIMARY KEY (seq_id, fam_set_id)
24 );

```

cluster_schema.sql

```

1 CREATE TABLE jj_cluster_run (
2     cr_id         serial,
3     date          timestamp not null,
4     br_id         INTEGER REFERENCES blast_run ON DELETE RESTRICT,
5     nc_id         INTEGER REFERENCES nc_run ON DELETE RESTRICT,
6     stype         text,
7     set_id_filter INTEGER REFERENCES prot_seq_set ON DELETE RESTRICT,
8     use_symmetric BOOLEAN,
9     score_threshold DOUBLE PRECISION,
10    params         text,
11    comment        text,

```

```

12         PRIMARY KEY (cr_id)
13     );
14
15 CREATE TABLE jj_hcluster (
16     cr_id          INTEGER NOT NULL REFERENCES jj_cluster_run ON DELETE RESTRICT,
17     cluster_id     INTEGER NOT NULL,
18     distance       DOUBLE PRECISION NOT NULL,
19     parent_distance DOUBLE PRECISION,
20     parent_id      INTEGER,
21     seq_id         INTEGER,
22     lft            INTEGER,
23     rgt            INTEGER,
24     PRIMARY KEY    (cr_id, cluster_id)
25 );
26 CREATE INDEX jj_hcluster_parent ON jj_hcluster (cr_id, parent_id)
27 CREATE INDEX jj_hcluster_nested ON jj_hcluster (cr_id, lft, rgt);
28
29 CREATE INDEX jj_hcluster_nested_seq ON jj_hcluster (cr_id, lft, rgt)
30 WHERE seq_id IS NOT NULL;
31
32 CREATE TABLE jj_flatcluster (
33     cr_id          INTEGER NOT NULL REFERENCES jj_cluster_run ON DELETE RESTRICT,
34     cluster_id     INTEGER NOT NULL,
35     seq_id         INTEGER NOT NULL REFERENCES prot_seq ON DELETE RESTRICT,
36     PRIMARY KEY    (cr_id, cluster_id, seq_id),
37 );

```

Appendix B

Data

12-genome subset

Organism	NCBI taxonomy id	Number of sequences
Arabidopsis thaliana	3702	27005
Caenorhabditis elegans	6239	19811
Danio rerio	7955	20886
Dictyostelium discoideum	44689	12445
Drosophila melanogaster	7227	13399
Escherichia coli K-12	83333	4119
Gallus gallus	9031	18204
Homo sapiens	9606	19372
Mus musculus	10090	26119
Rattus norvegicus	10116	27636
Saccharomyces cerevisiae	4932	5813
Schizosaccharomyces pombe	4896	4943

Table B.1: Table (part 1 of 2) of the 48-genomes included in the Panther 7.0 data set [141] used throughout this dissertation.

All other genomes

Organism	NCBI taxonomy id	Number of sequences
Anopheles gambiae	7165	12433
Aquifex aeolicus VF5	224324	1556
Bacillus subtilis	1423	4103
Bacteroides thetaiotaomicron	818	4775
Bos taurus	9913	20767
Bradyrhizobium japonicum	375	8256
Canis lupus familiaris	9615	19268
Chlamydia trachomatis A/HAR-13	315277	919
Chlamydomonas reinhardtii	3055	14272
Chloroflexus aurantiacus J-10-fl	324602	3850
Ciona intestinalis	7719	14172
Deinococcus radiodurans	1299	3164
Emericella nidulans	162425	9528
Entamoeba histolytica	5759	8032
Eremothecium gossypii	33169	4716
Geobacter sulfurreducens	35554	3414
Gloeobacter violaceus	33072	4409
Leishmania major	5664	8003
Leptospira interrogans	173	4690
Macaca mulatta	9544	21880
Methanosarcina acetivorans	2214	4468
Monodelphis domestica	13616	19468
Neurospora crassa	5141	9804
Ornithorhynchus anatinus	9258	17948
Oryza sativa Japonica Group	39947	26844
Pan troglodytes	9598	19776
Plasmodium yoelii yoelii	73239	7302
Pseudomonas aeruginosa PA7	381754	6246
Saccharomyces cerevisiae	4932	5813
Streptomyces coelicolor	1902	8039
Strongylocentrotus purpuratus	7668	28574
Sulfolobus solfataricus	2287	2922
Takifugu rubripes	31033	18517
Tetrahymena thermophila	5911	25921
Thermotoga maritima	2336	1857
Xenopus (Silurana) tropicalis	8364	18015

Table B.2: Table (part 2 of 2) of the 48-genomes included in the Panther 7.0 data set [141] used throughout this dissertation.

Appendix C

Family score distributions

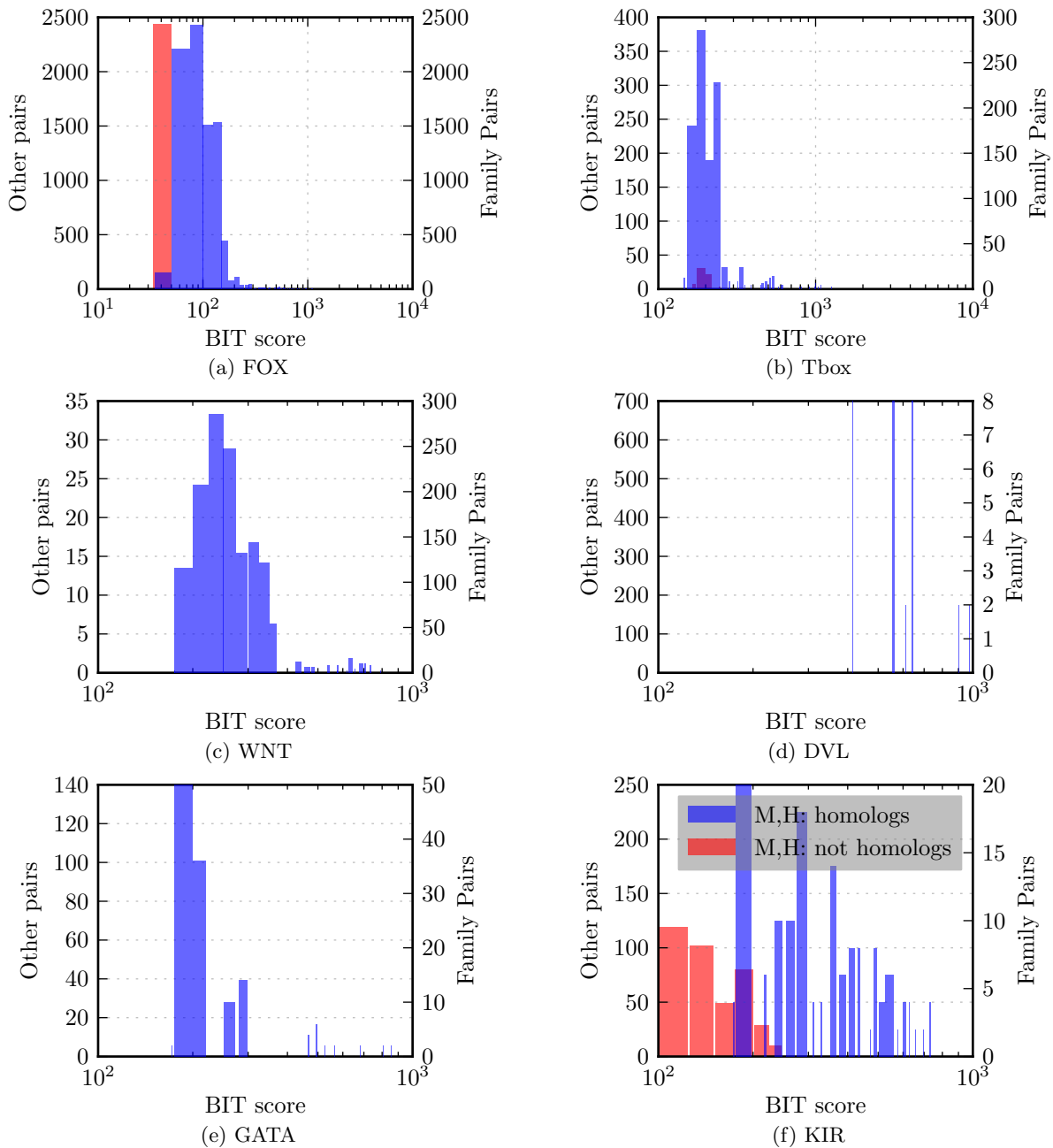


Figure C.1: Histogram of the sequence similarity scores that result between homologous pairs of sequences (in blue, wrt. right axis), and between non-homologous pairs (in red, wrt. left axis), in the mouse and human genomes. These demonstrate that an effective bit-score threshold can be selected for some families, though no single threshold is suitable for all families. See Figures 3.2, 3.3, and C.2.

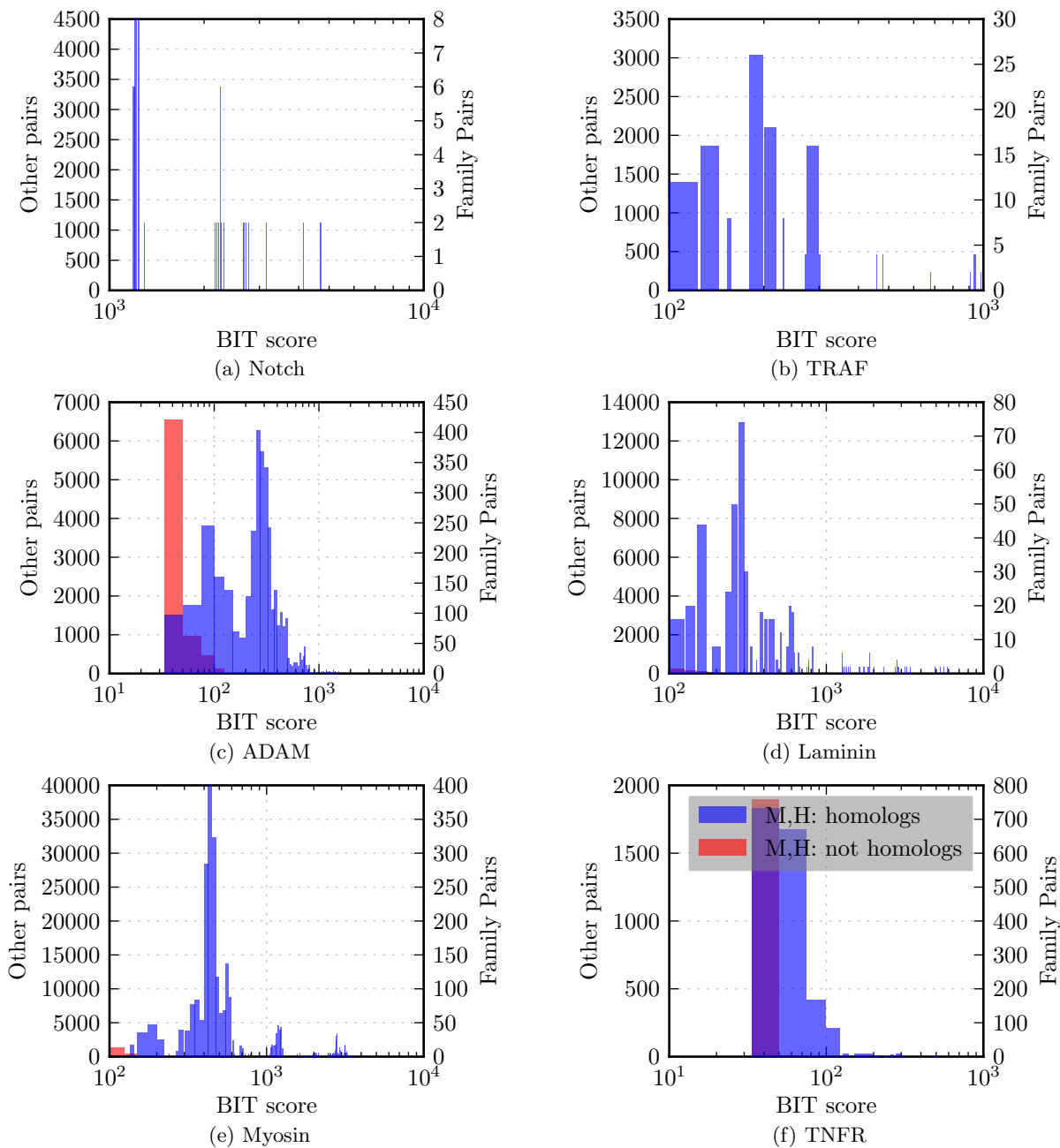


Figure C.2: Histogram of the sequence similarity scores that result between homologous pairs of sequences (in blue, wrt. right axis), and between non-homologous pairs (in red, wrt. left axis), in the mouse and human genomes. These demonstrate that an effective bit-score threshold can be selected for some families, though no single threshold is suitable for all families. See Figures 3.2, 3.3, and C.1.

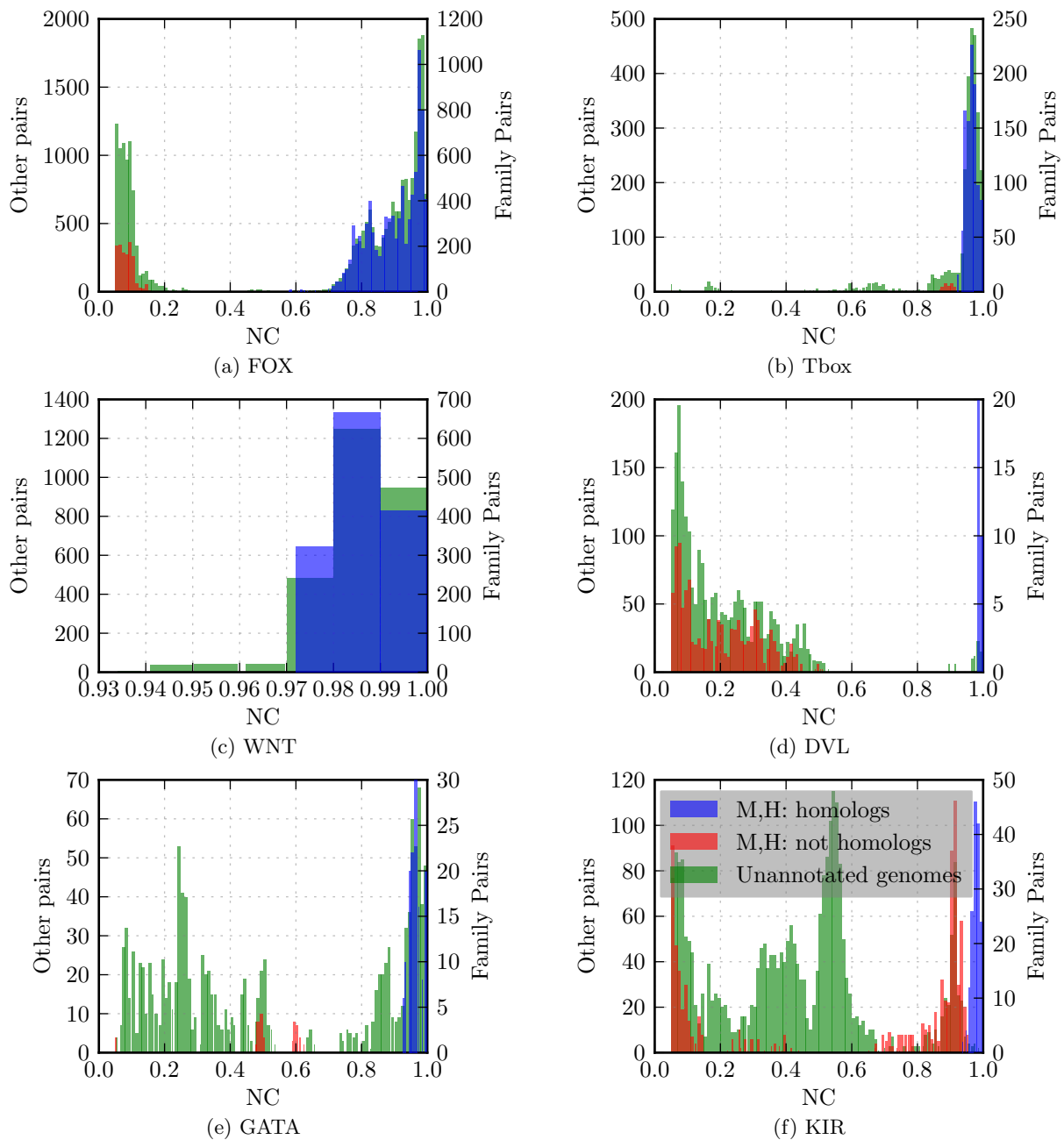


Figure C.3: Histograms of the Neighborhood Correlation scores for curated families in mouse and human. Homologous pairs are in blue (wrt. right axis), and non-homologous pairs are in red (wrt. left axis). Additionally, in green (wrt. left axis) are scores of all pairs that include one member of a curated family, and one member from the 12-genome dataset, and not in mouse or human. See Figures 3.5, 3.6, and C.4.

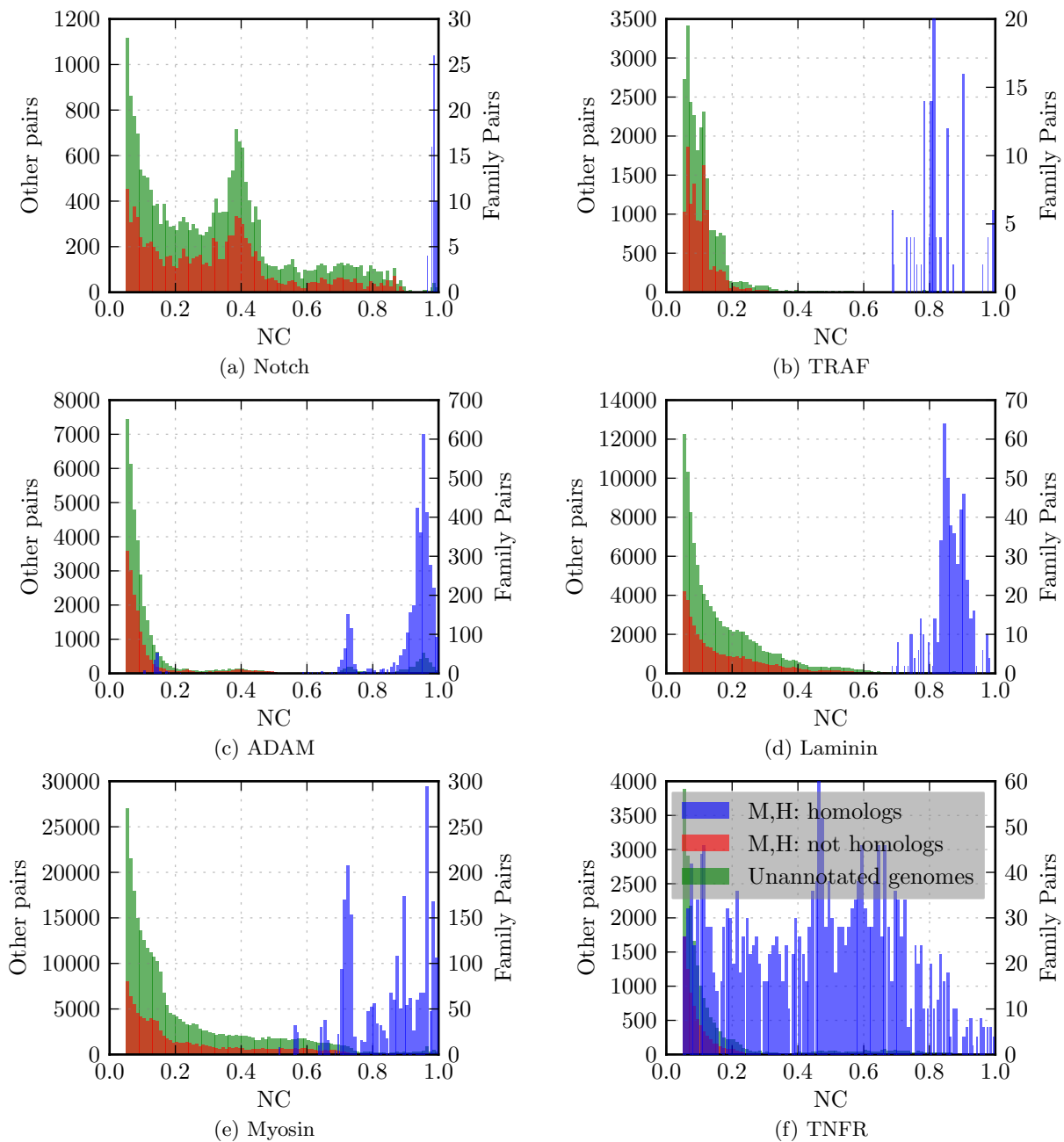


Figure C.4: Histograms of the Neighborhood Correlation scores for curated families in mouse and human. Homologous pairs are in blue (wrt. right axis), and non-homologous pairs are in red (wrt. left axis). Additionally, in green (wrt. left axis) are scores of all pairs that include one member of a curated family, and one member from the 12-genome dataset, and not in mouse or human. See Figures 3.5, 3.6, and C.3.

Domains and clusters – supplementary data

D.1 Examination across lineages

The analysis in Chapter 7: The relationship between domains and clusters (p.109) considered the domain content of human and mouse sequences. The clustering of those sequences was derived from construction of a hierarchical clustering result that contained all 600k sequences in the Panther dataset. The data here illustrates that the trends observed in human and mouse persist in other lineages. In particular, this appendix presents domain data for four organisms at increasing phylogenetic distance from human and mouse. The same hierarchical tree of 600k sequences was used in this data, and, as with human and mouse, the clustering of sequences within an organism is obtained by selecting leaf nodes only from the chosen organism.

The following figures show the domain and PCA data for *S. purpuratus*, *D. melanogaster*, *C. elegans*, and *S. cerevisiae*.

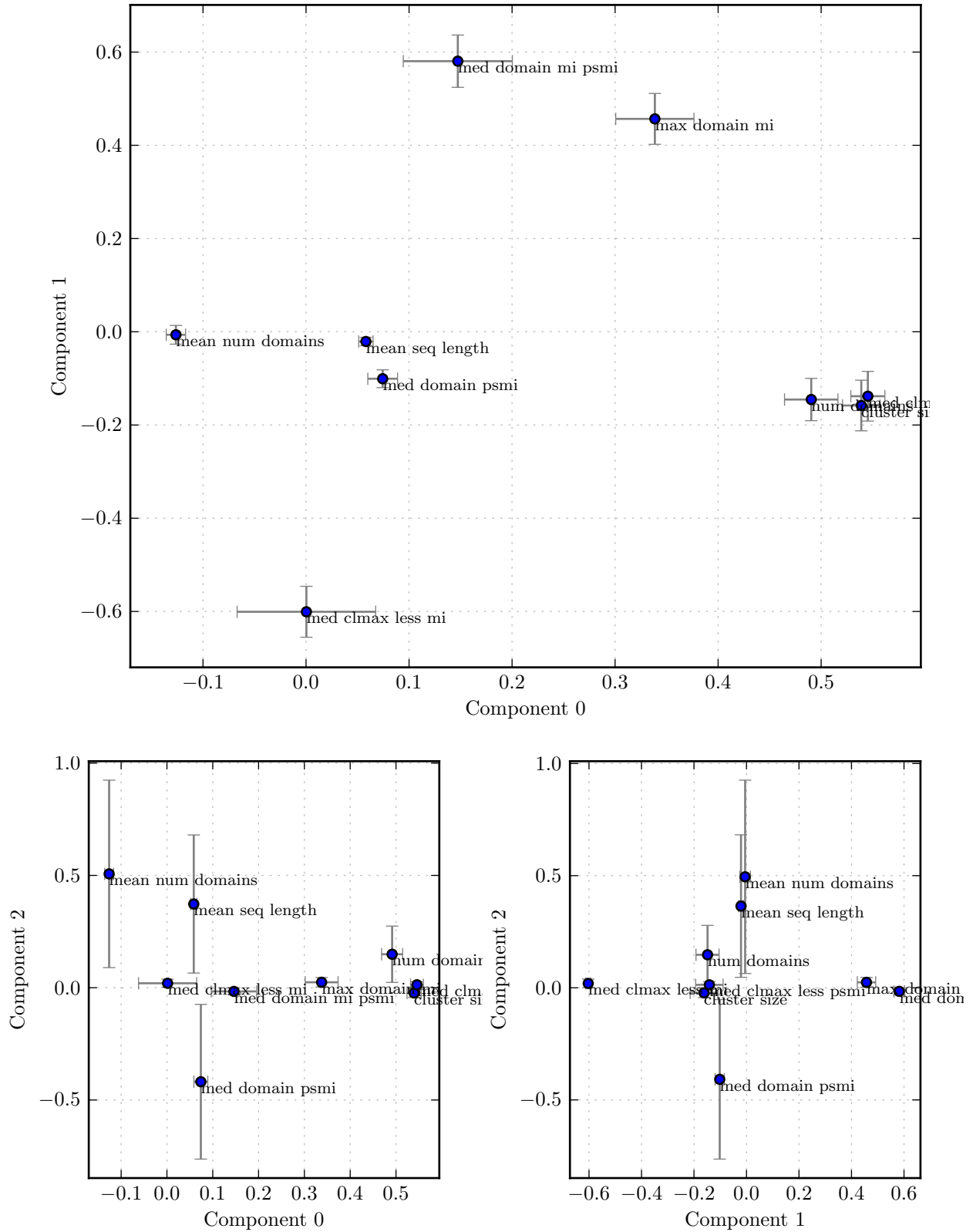


Figure D.1: *S. purpuratus*: Coefficients of individual features in the PCA component space. Error bars indicate the standard deviation over 1000 samplings of 75% of the clusters in the clustering.

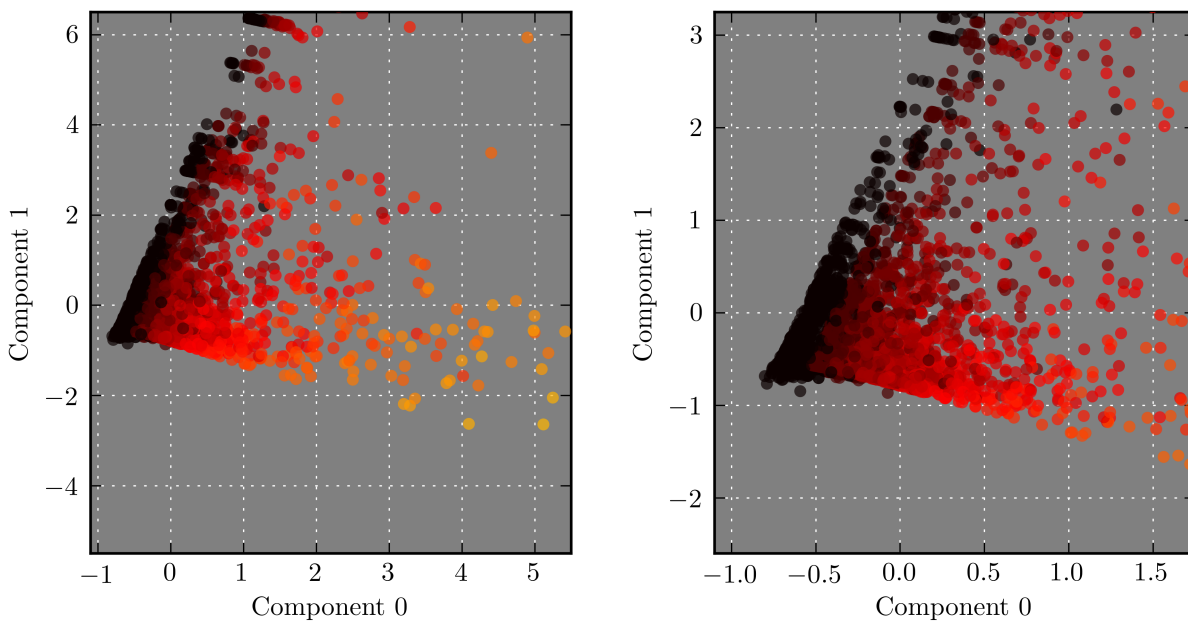
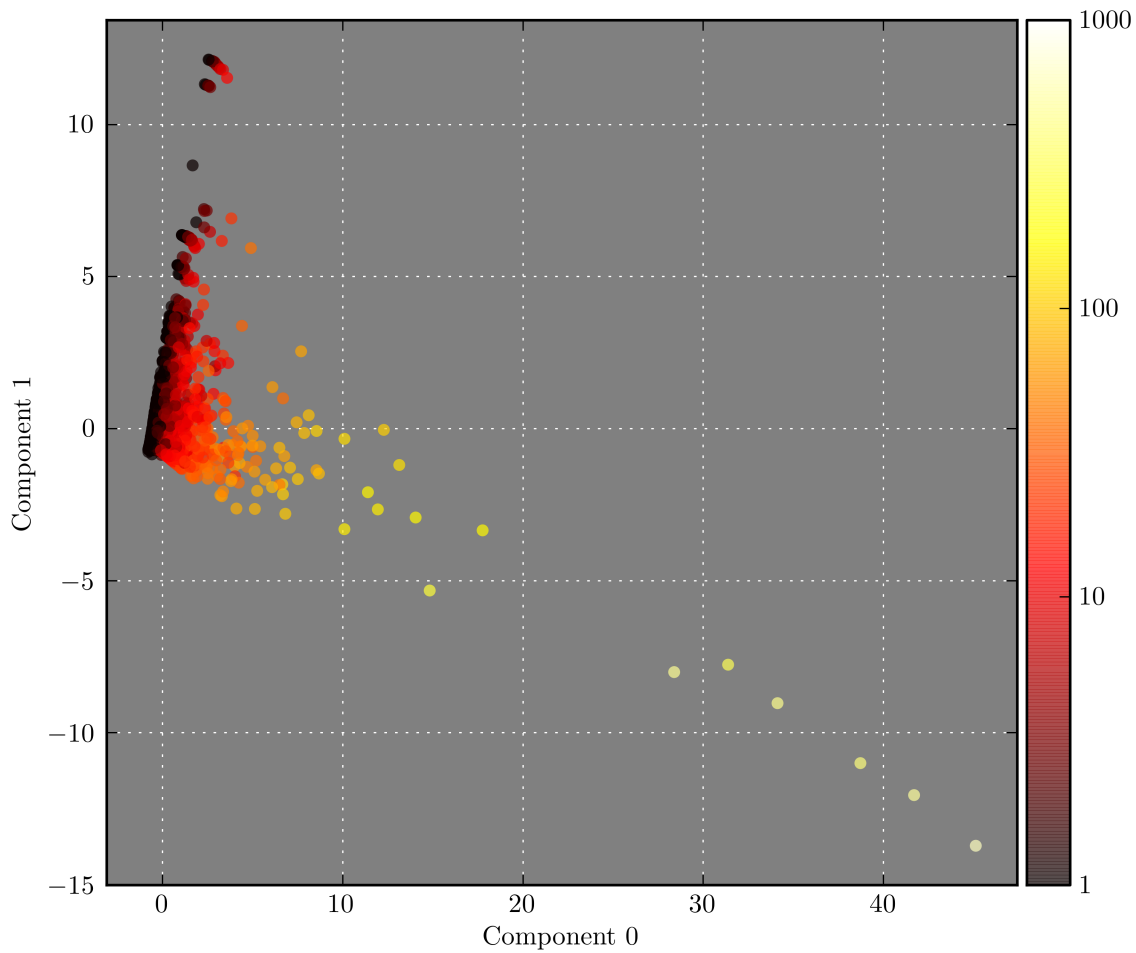


Figure D.2: *S. purpuratus*: Projection of clusters into PCA component space. Clusters in this figure are colored according to the number of sequences in found in that cluster.

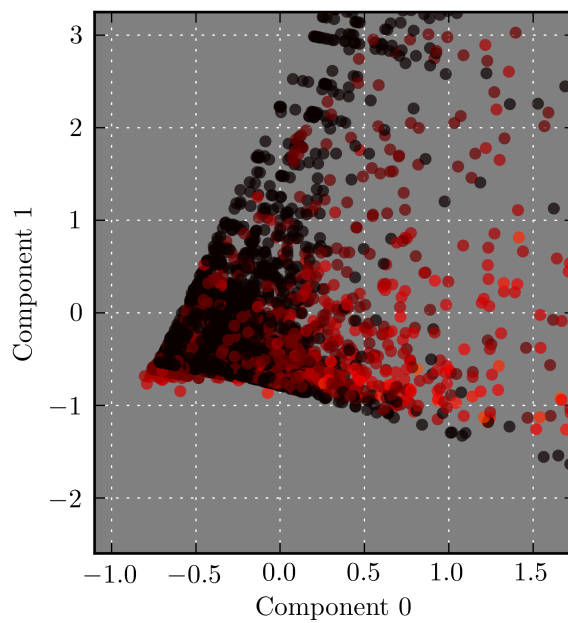
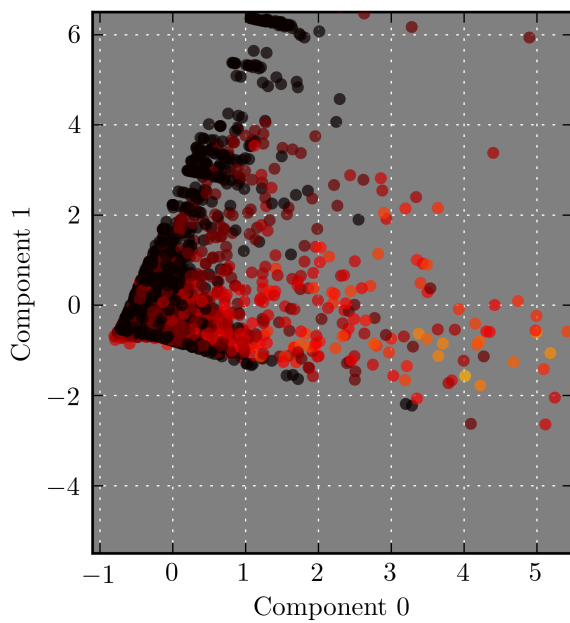
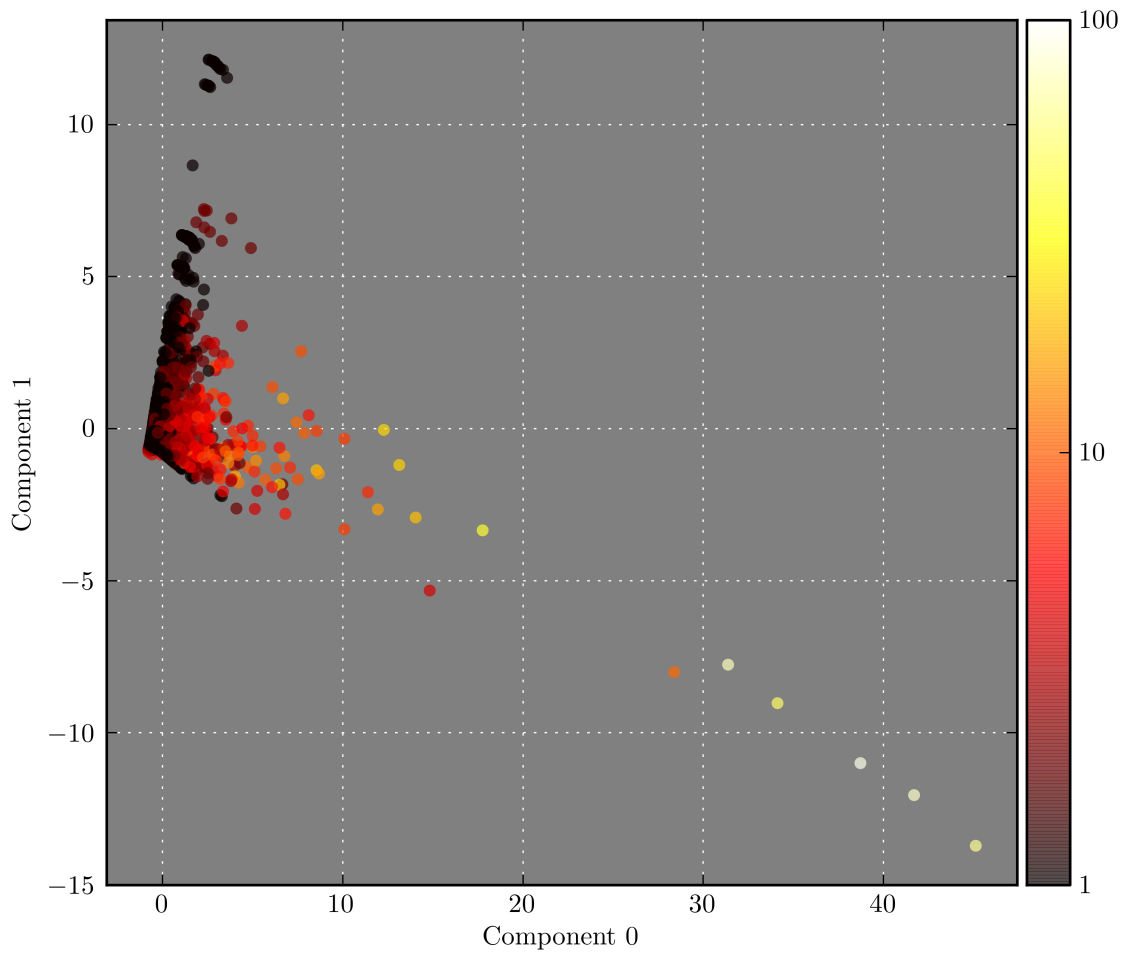


Figure D.3: *S. purpuratus*: Projection of clusters into PCA component space. Clusters in this figure are colored according to the total number of distinct domains found in that cluster.

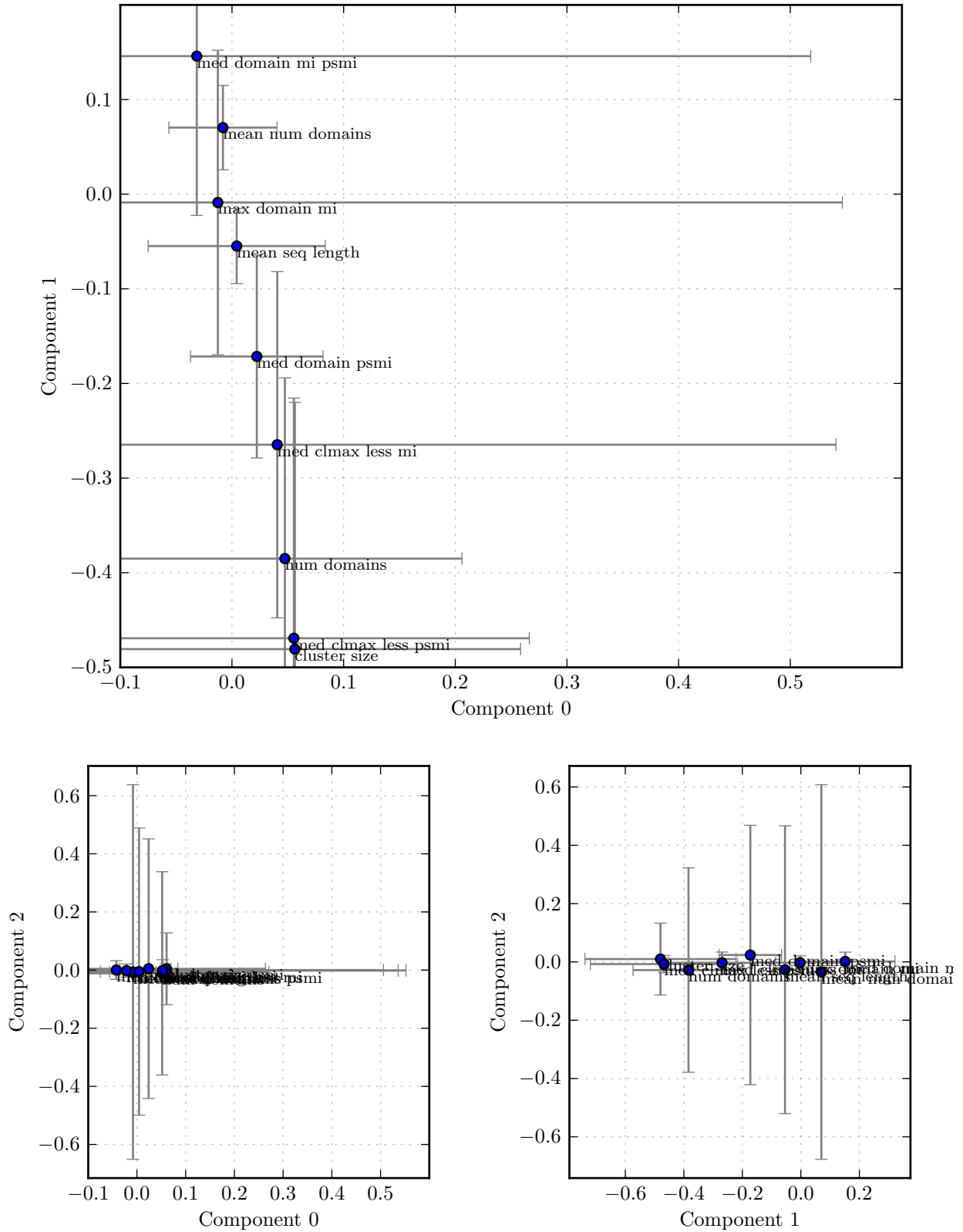


Figure D.4: *D. melanogaster*: Coefficients of individual features in the PCA component space. Error bars indicate the standard deviation over 1000 samplings of 75% of the clusters in the clustering.

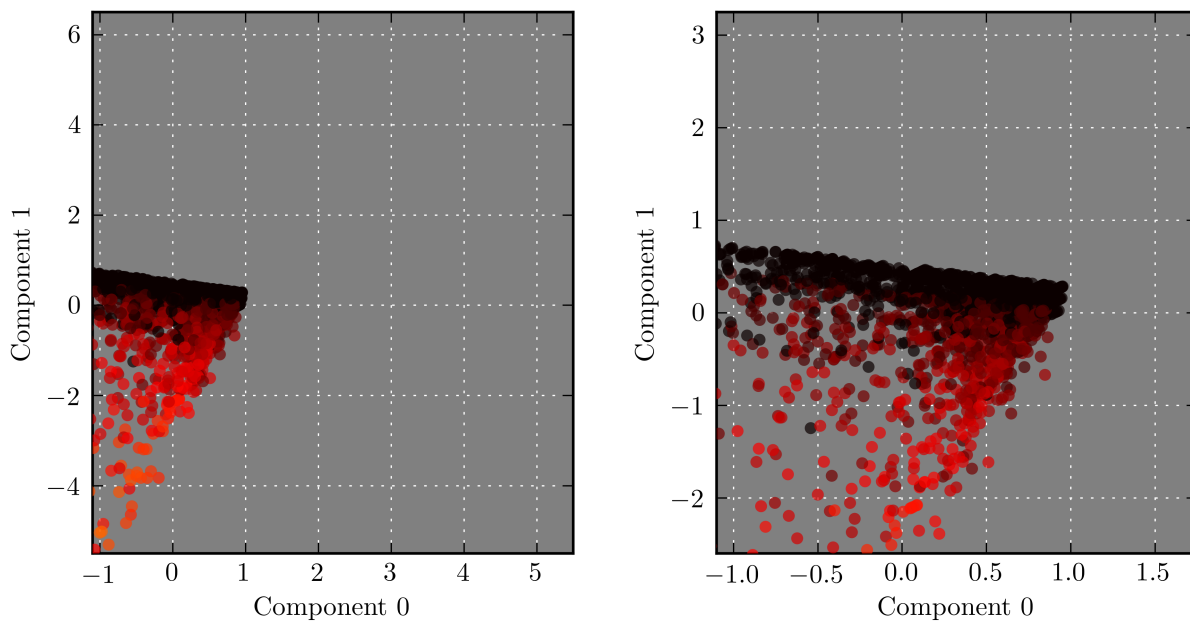
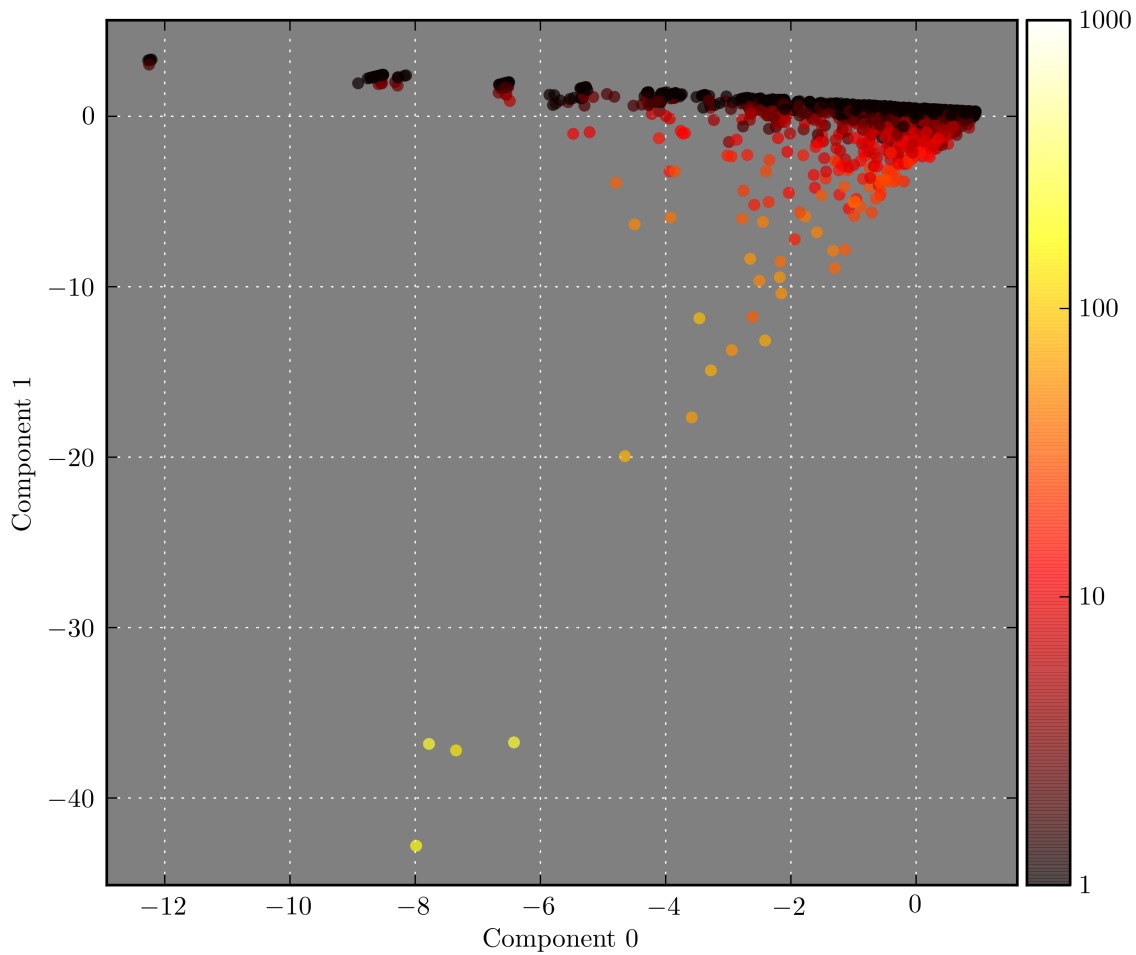


Figure D.5: *D. melanogaster*: Projection of clusters into PCA component space. Clusters in this figure are colored according to the number of sequences in found in that cluster.

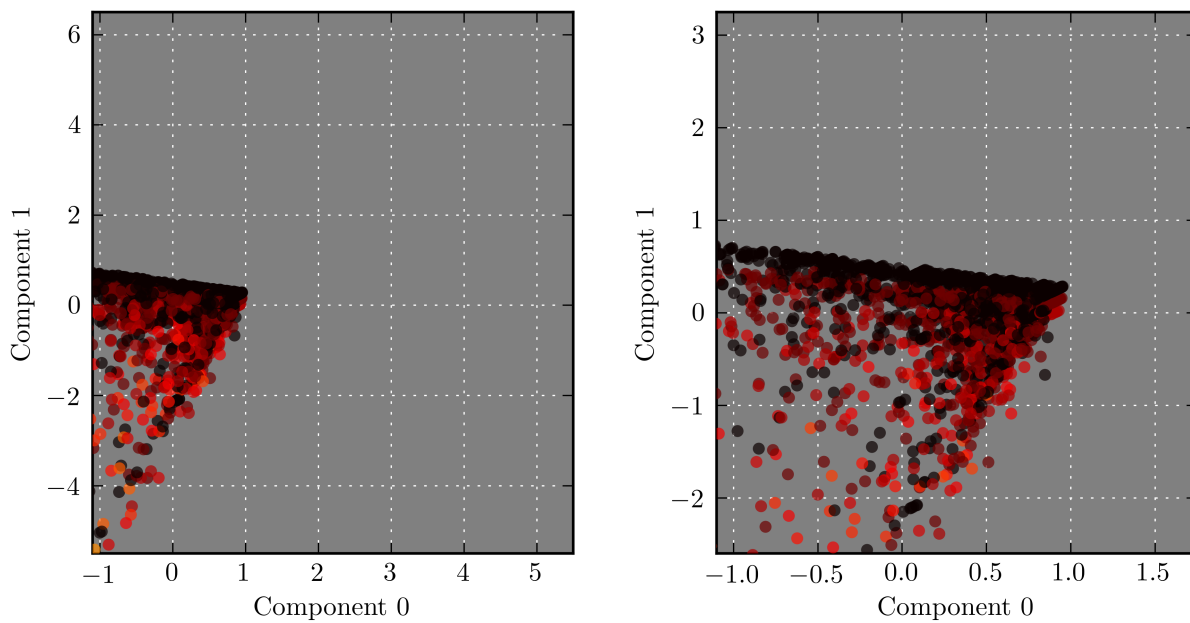
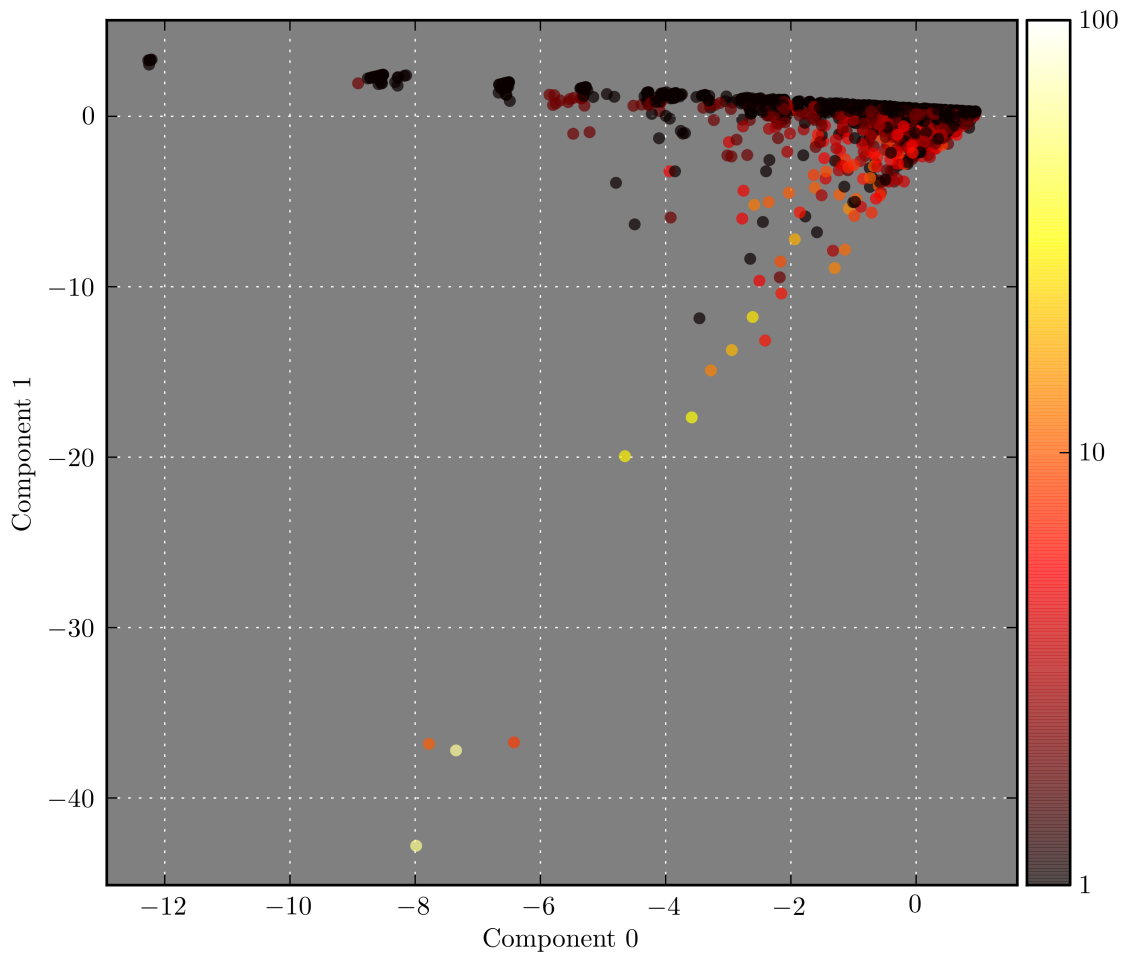


Figure D.6: *D. melanogaster*: Projection of clusters into PCA component space. Clusters in this figure are colored according to the total number of distinct domains found in that cluster.

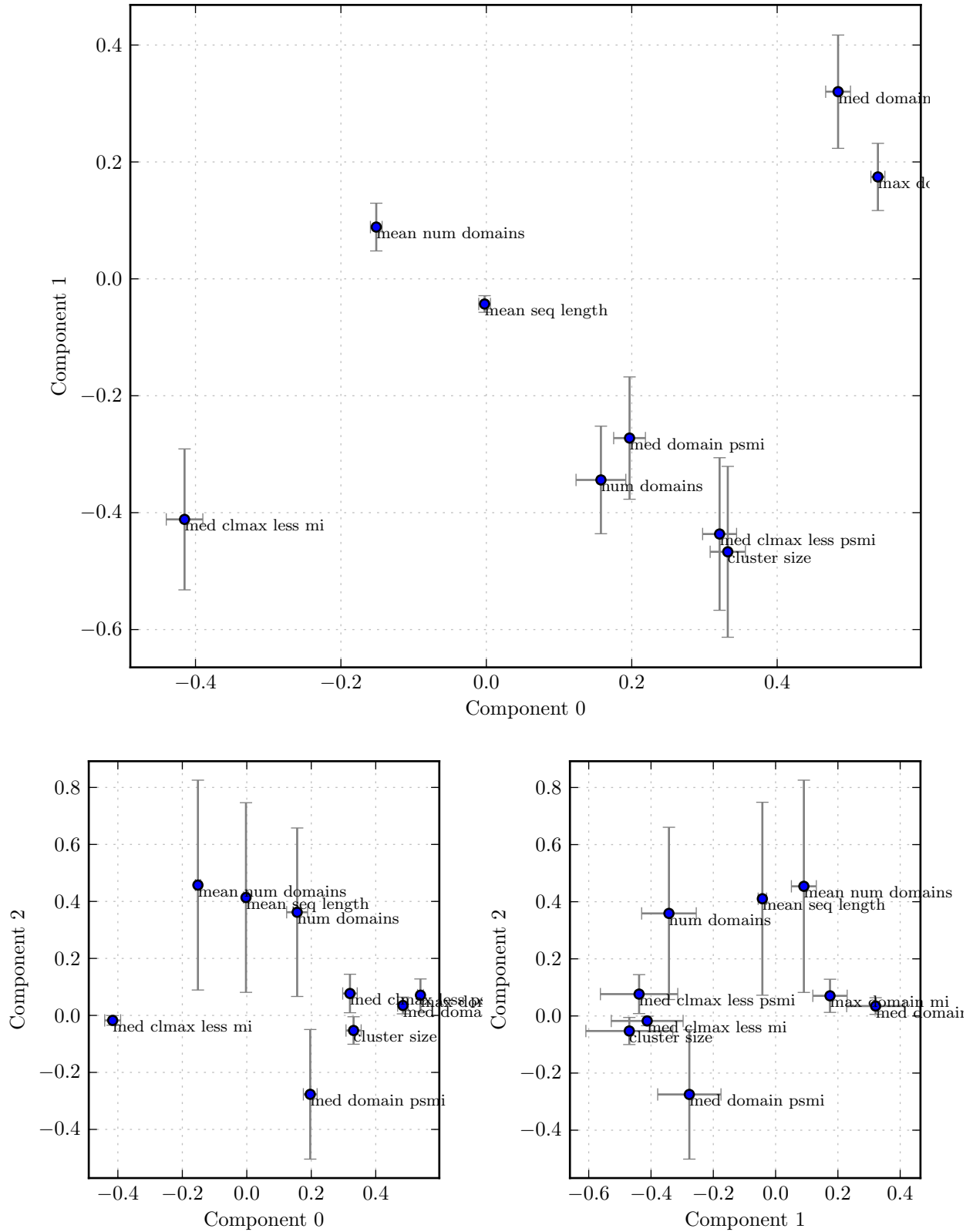


Figure D.7: *C. elegans*: Coefficients of individual features in the PCA component space. Error bars indicate the standard deviation over 1000 samplings of 75% of the clusters in the clustering.

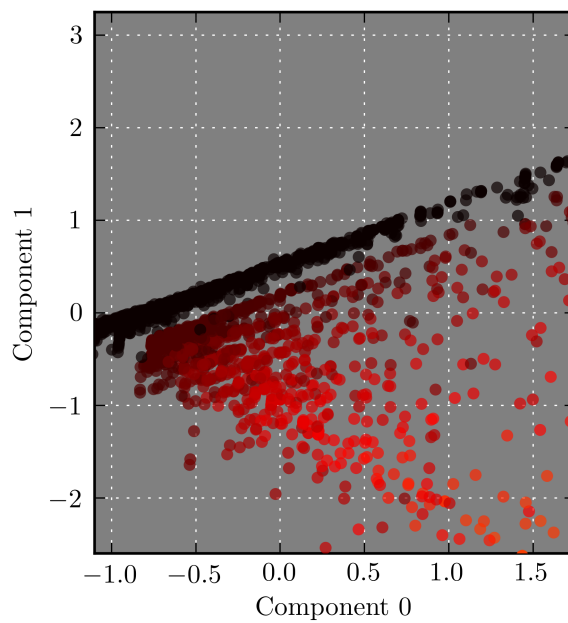
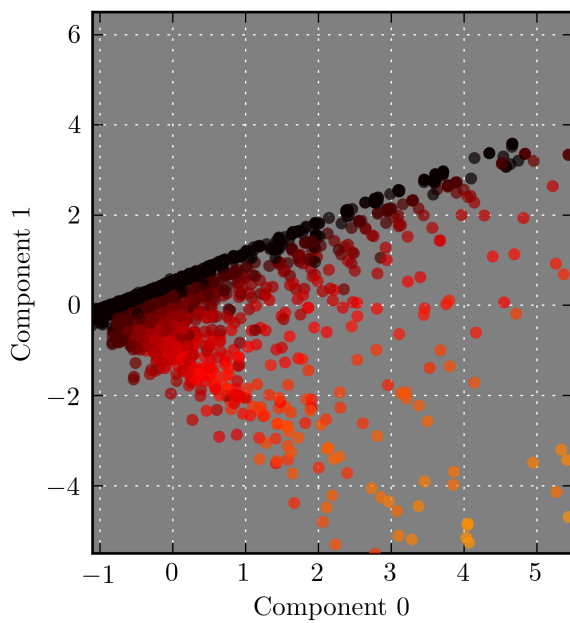
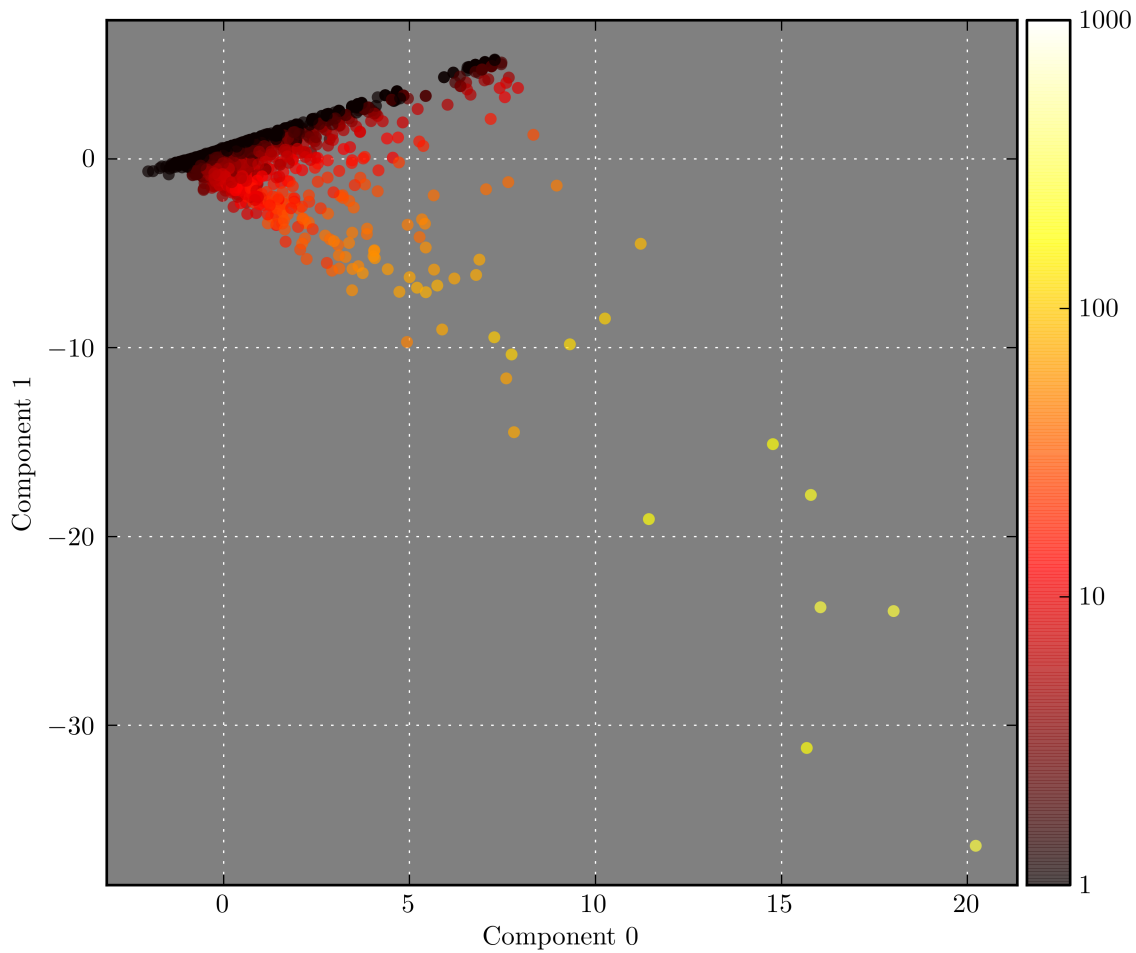


Figure D.8: *C. elegans*: Projection of clusters into PCA component space. Clusters in this figure are colored according to the number of sequences in found in that cluster.

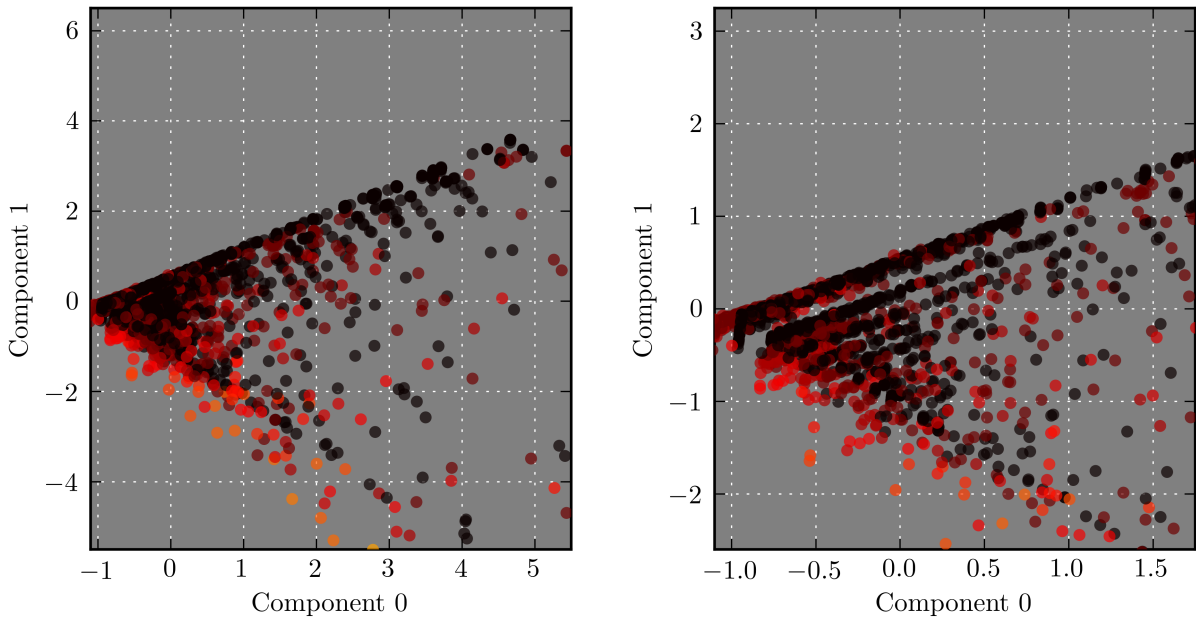
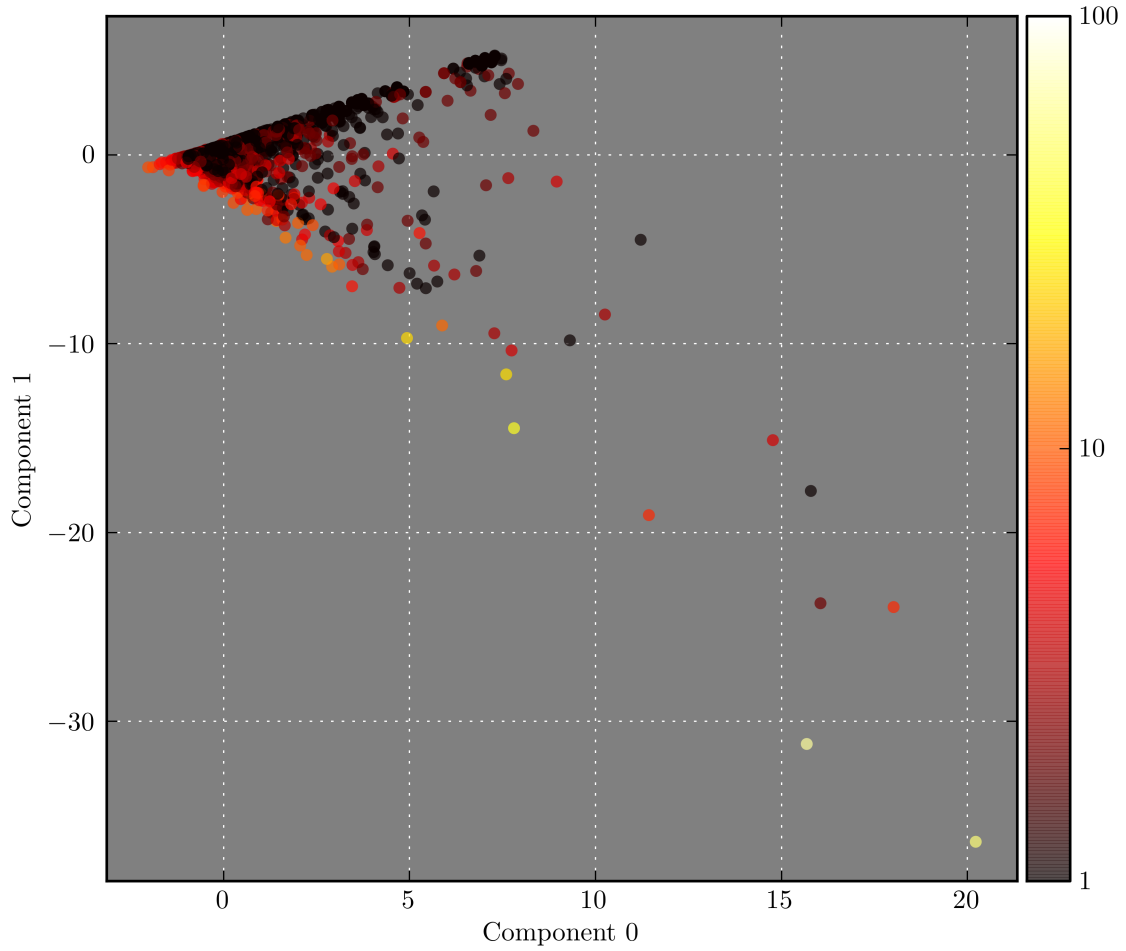


Figure D.9: *C. elegans*: Projection of clusters into PCA component space. Clusters in this figure are colored according to the total number of distinct domains found in that cluster.

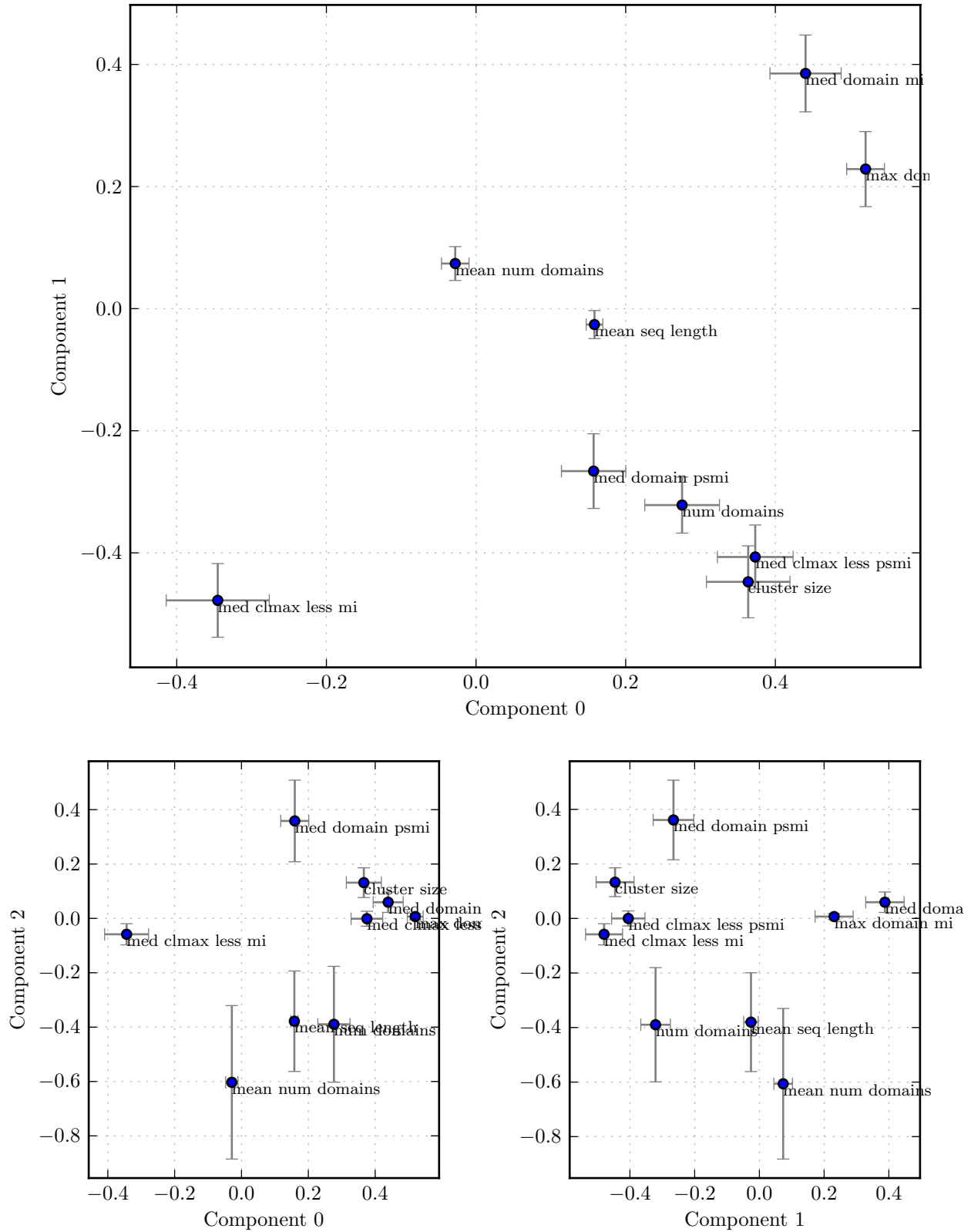


Figure D.10: *S. cerevisiae*: Coefficients of individual features in the PCA component space. Error bars indicate the standard deviation over 1000 samplings of 75% of the clusters in the clustering.

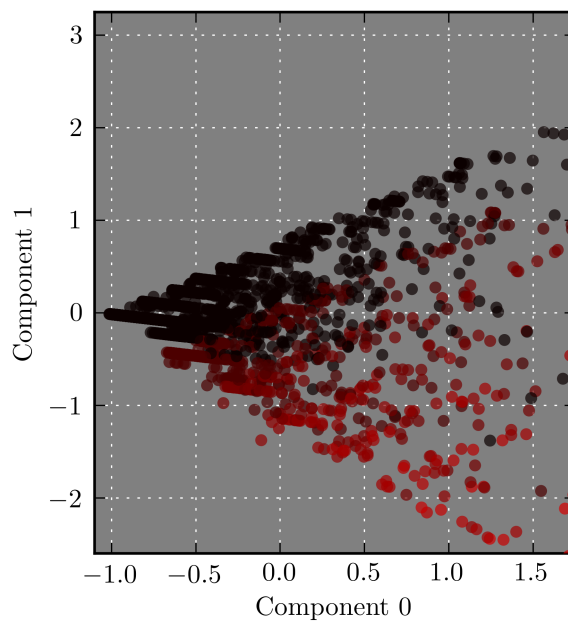
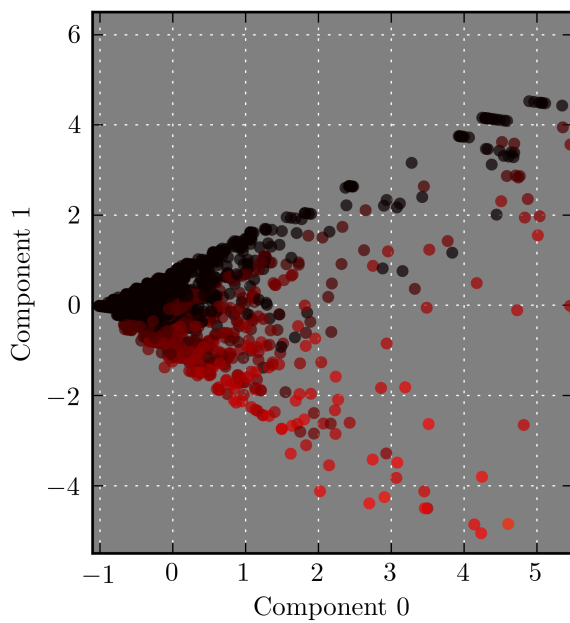
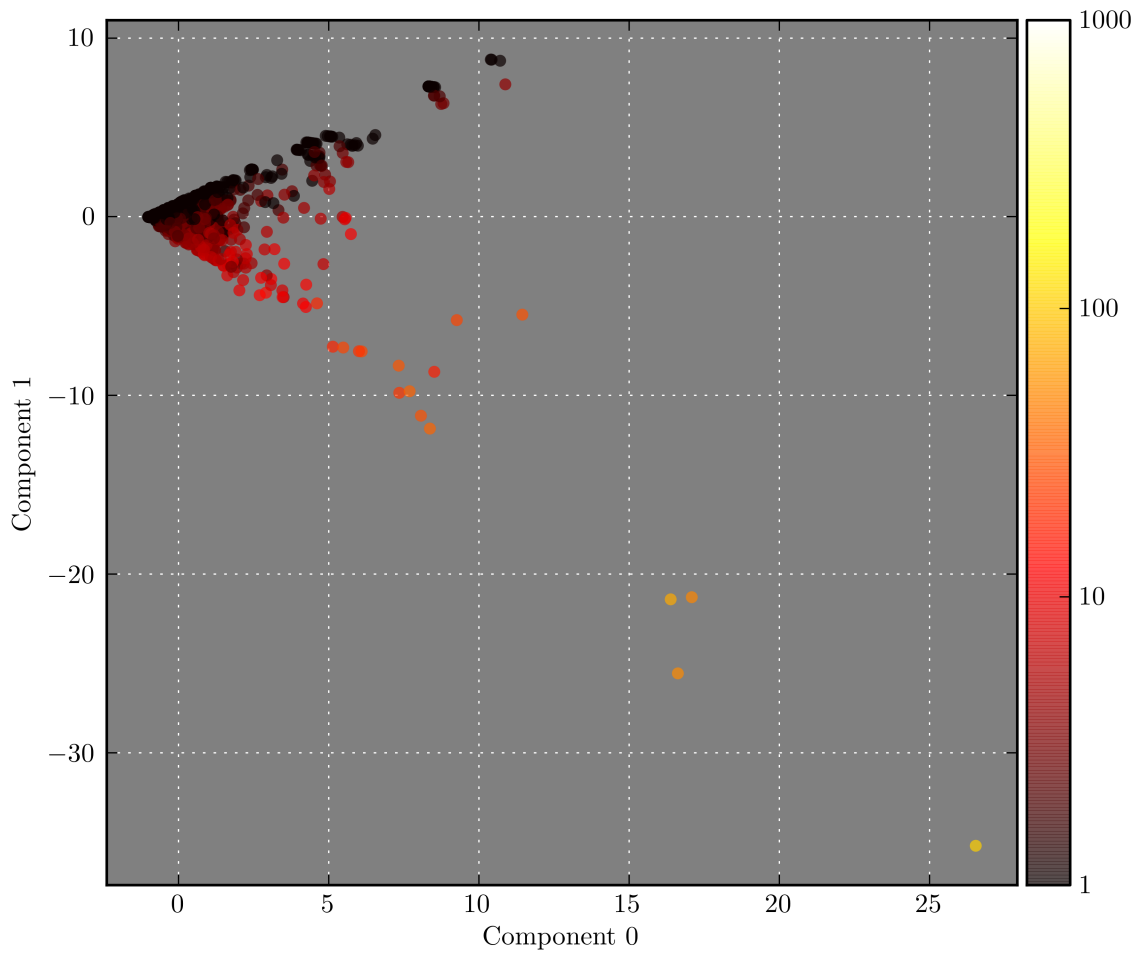


Figure D.11: *S. cerevisiae*: Projection of clusters into PCA component space. Clusters in this figure are colored according to the number of sequences in found in that cluster.

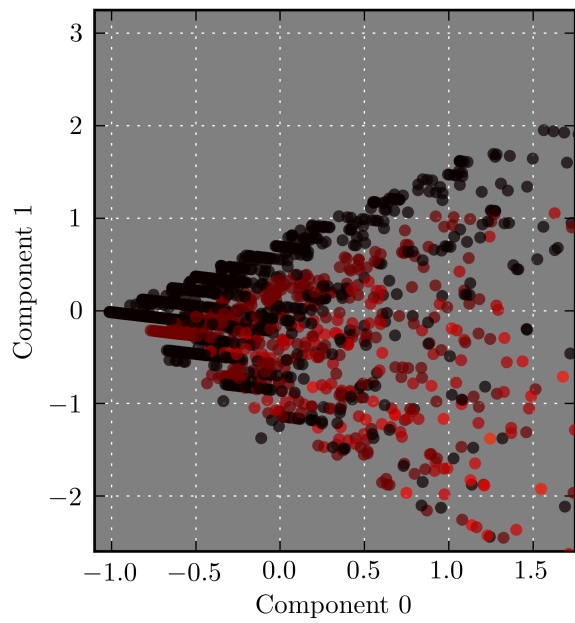
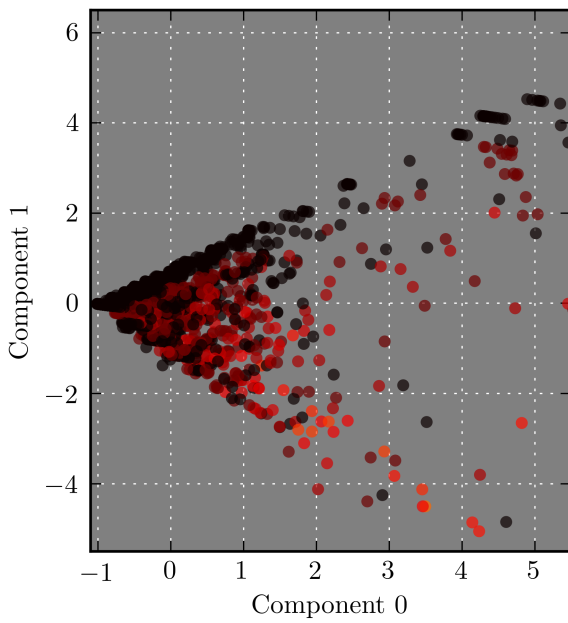
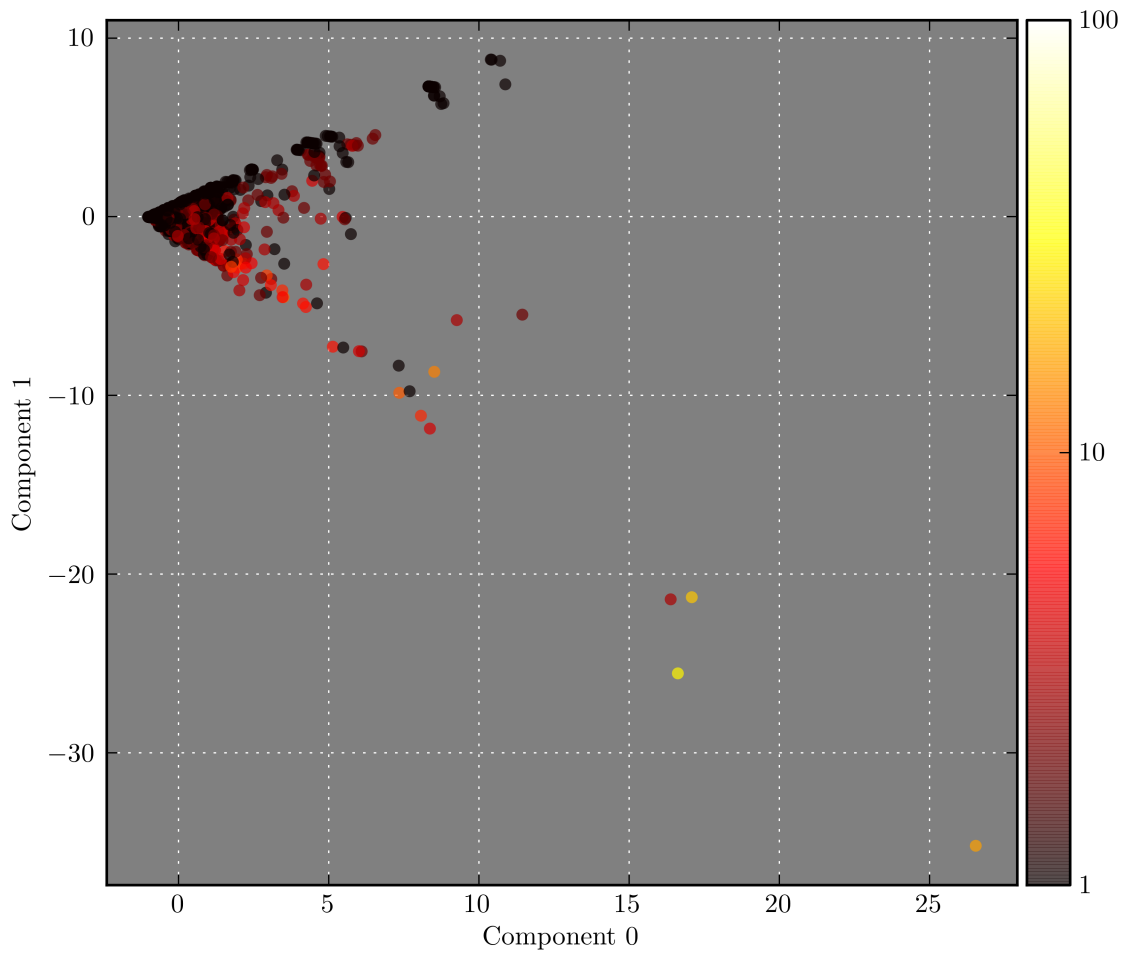
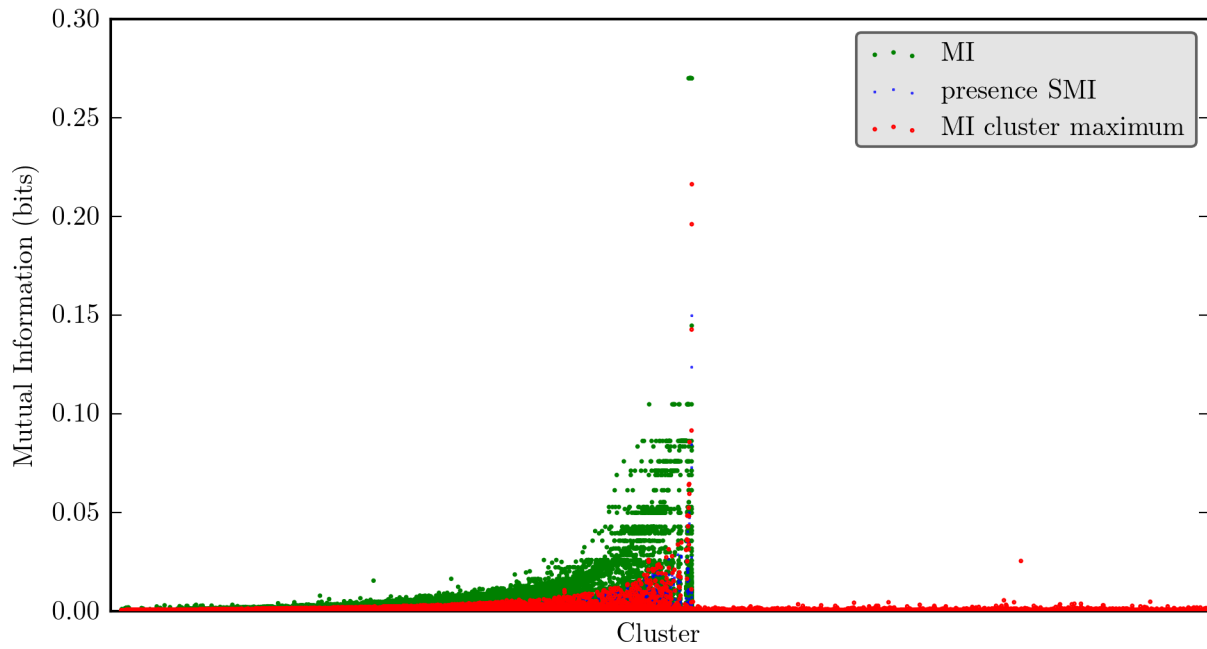
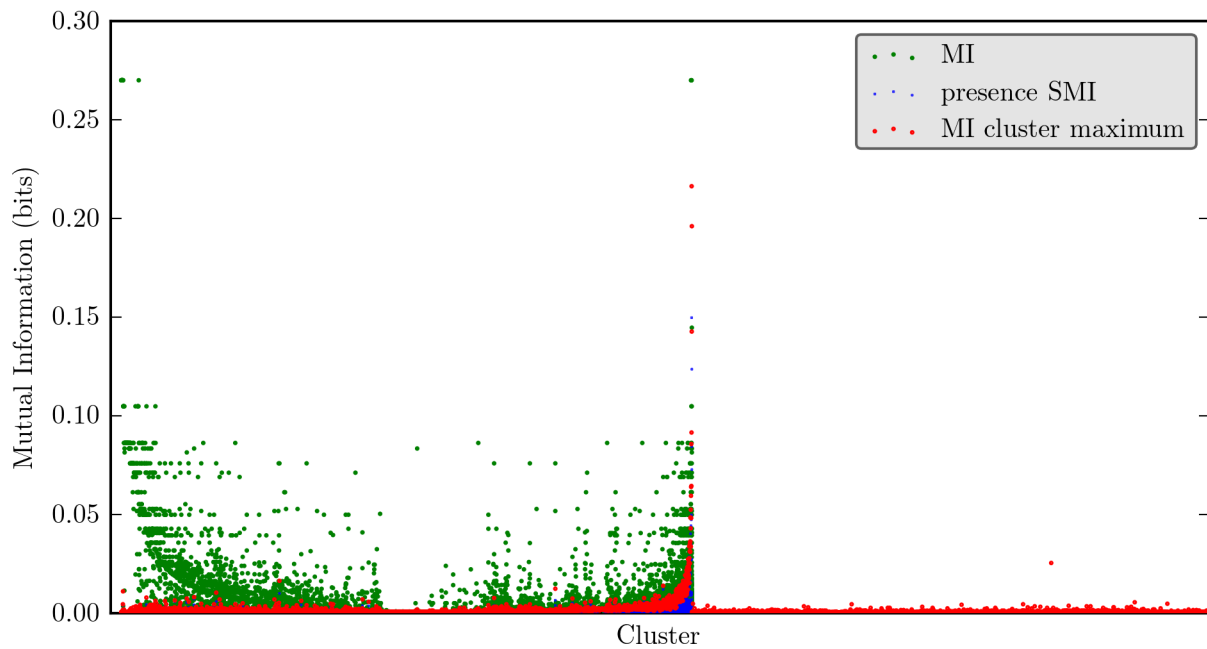


Figure D.12: *S. cerevisiae*: Projection of clusters into PCA component space. Clusters in this figure are colored according to the total number of distinct domains found in that cluster.

D.2 Ordering of clusters by PCA component



(a) PCA component 0



(b) PCA component 1

Figure D.13: Ordering of clusters in mouse and human by their position within PCA components 0 (a) and 1 (b).

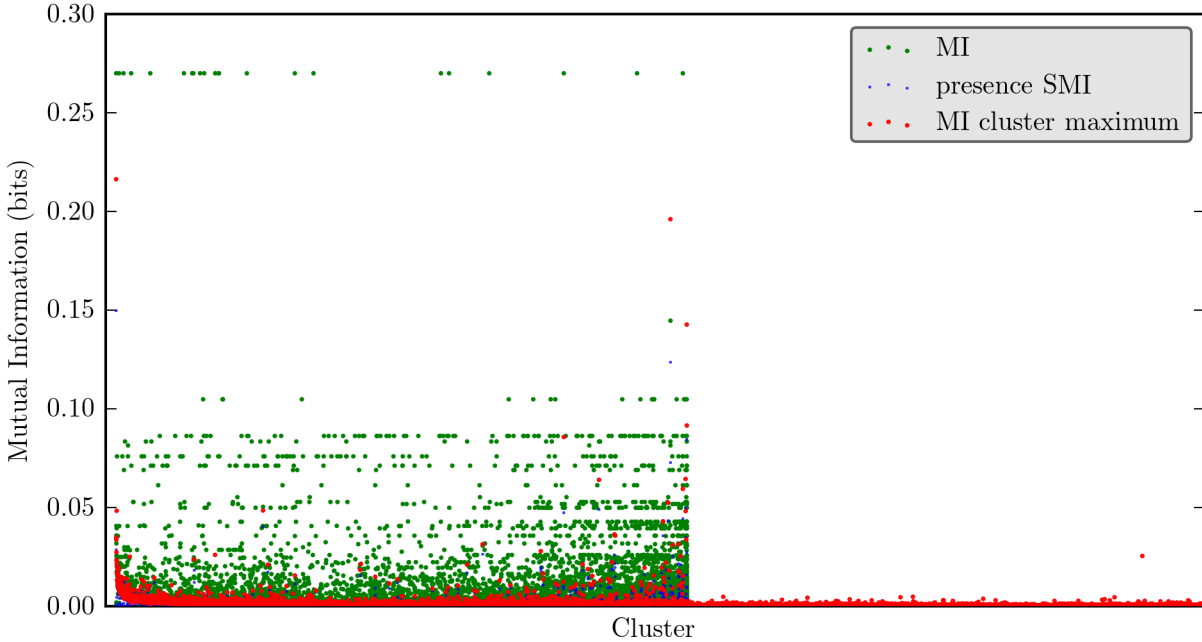


Figure D.14: Ordering of clusters in mouse and human by their position within PCA component 2.

D.3 Stability in single genomes

The analyses in Chapter 7: The relationship between domains and clusters (p.109) and Section D.1 were based upon Neighborhood Correlation and hierarchical clustering of all 600k sequences in the Panther dataset. The partitioning of sequences from one or several organisms was achieved by cutting this tree at a particular height ($NC \geq 0.425$) to induce a partitioning of the leaves, followed by selection of leaves that represent the sequences of interest.

This section considers whether similar results would be obtained if the set of input sequences to the family classification pipeline consisted only of the set of sequences that were used to filter the set of leaves. The same $NC \geq 0.425$ tree threshold was applied in all cases. The following figures demonstrate the results when the set of sequences is constrained to sequences from (1) human, (2) mouse, and (3) human and mouse.

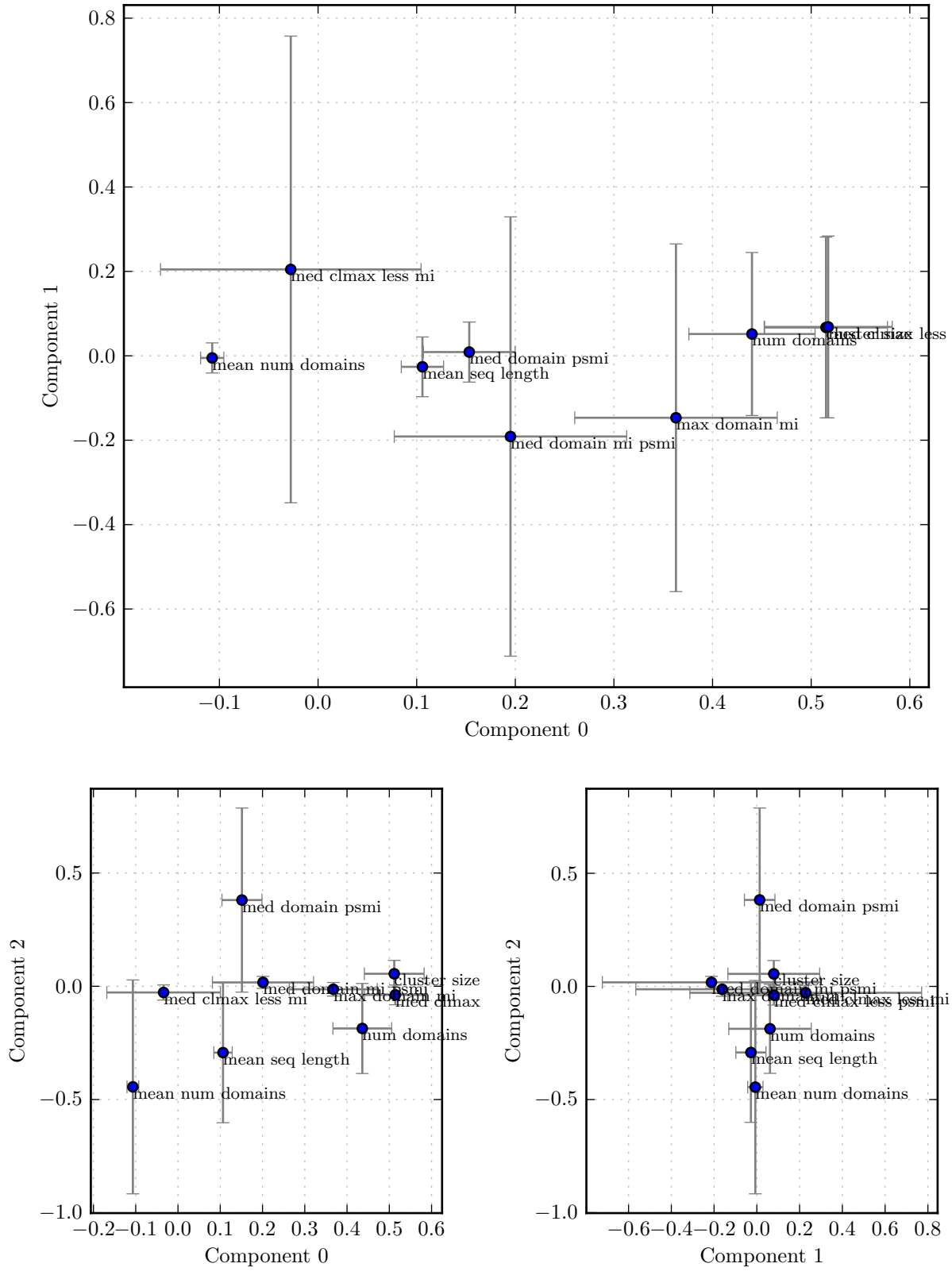


Figure D.15: **Human:** Coefficients of individual features in the PCA component space. Error bars indicate the standard deviation over 1000 samplings of 75% of the clusters in the clustering.

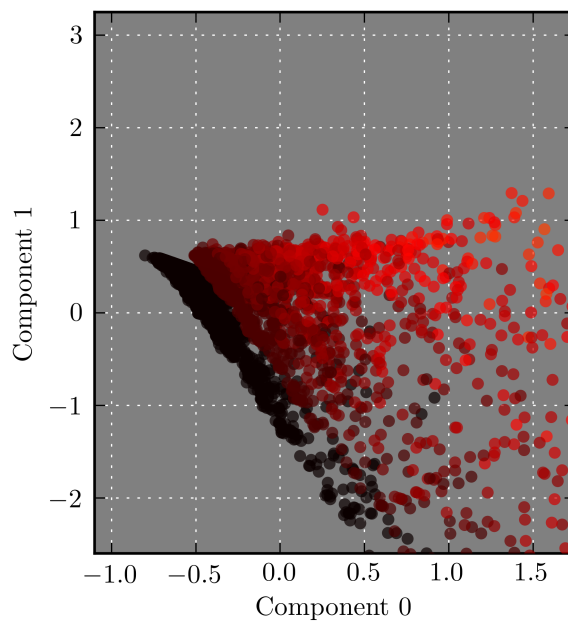
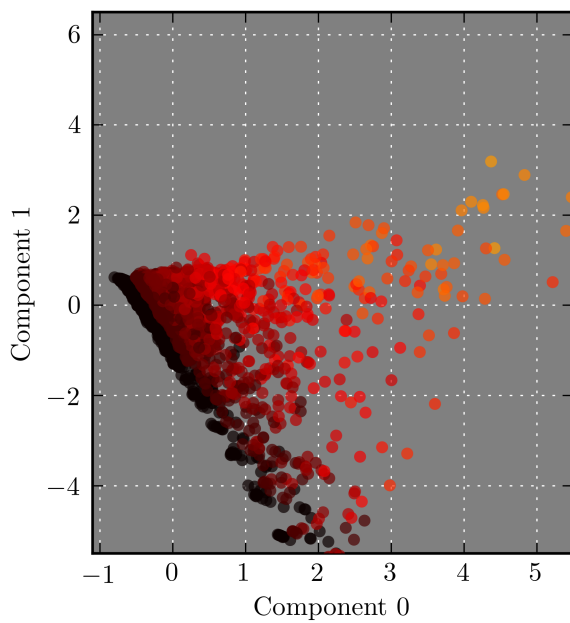
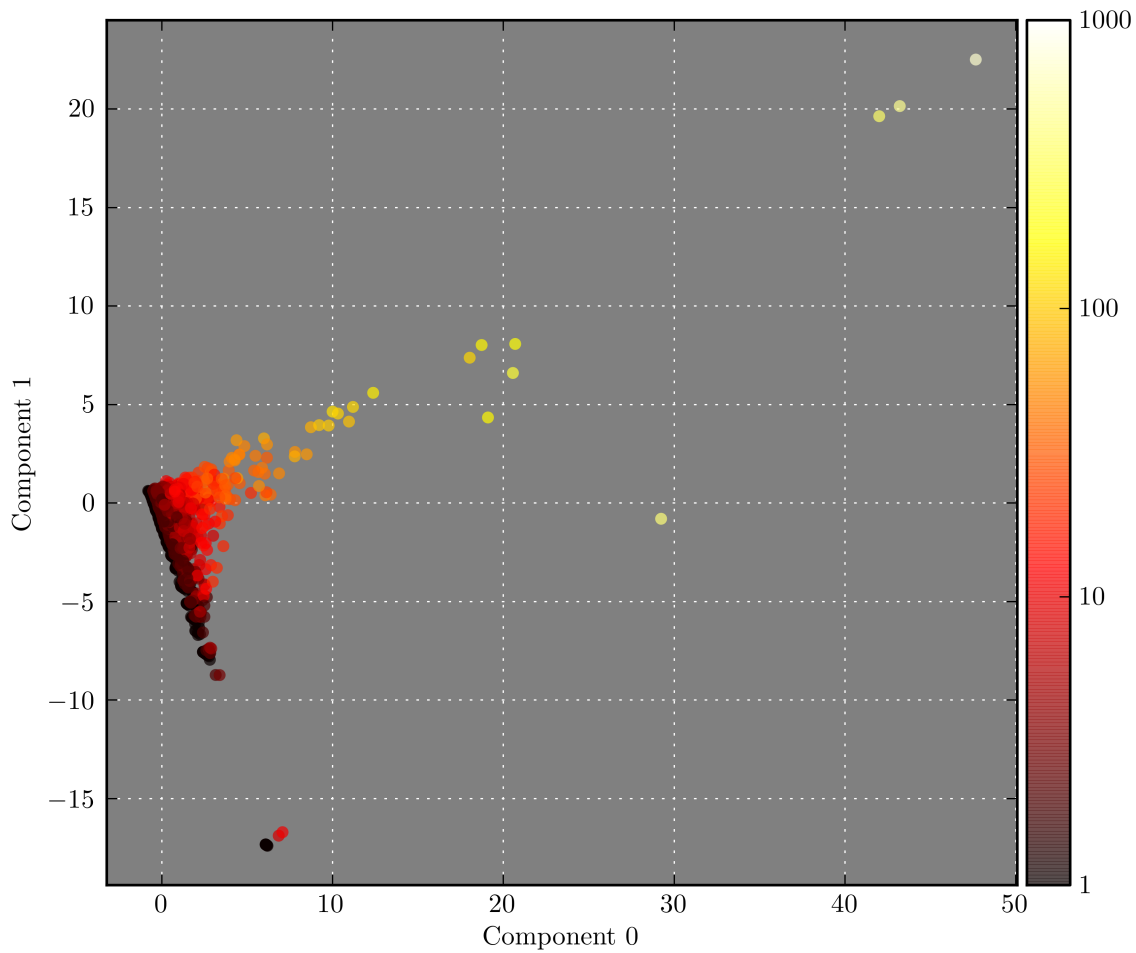


Figure D.16: **Human:** Projection of clusters into PCA component space. Clusters in this figure are colored according to the number of sequences in found in that cluster.

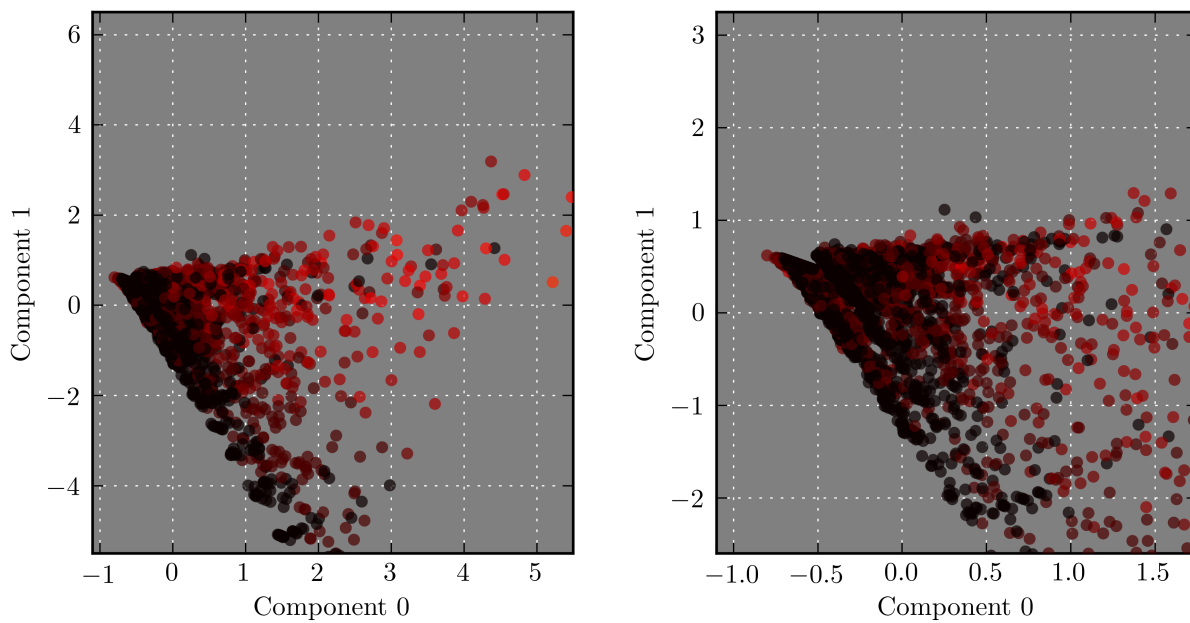
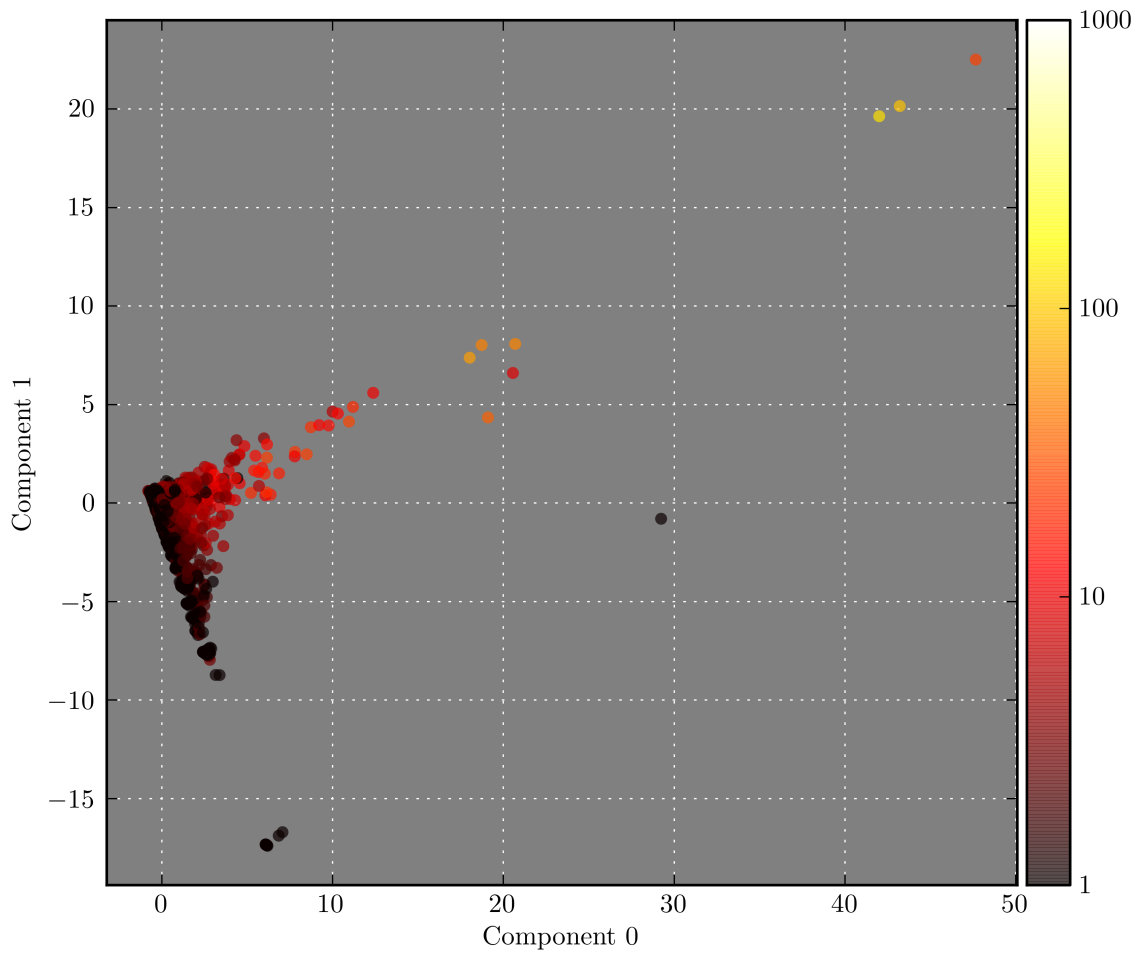


Figure D.17: **Human:** Projection of clusters into PCA component space. Clusters in this figure are colored according to the total number of distinct domains found in that cluster.

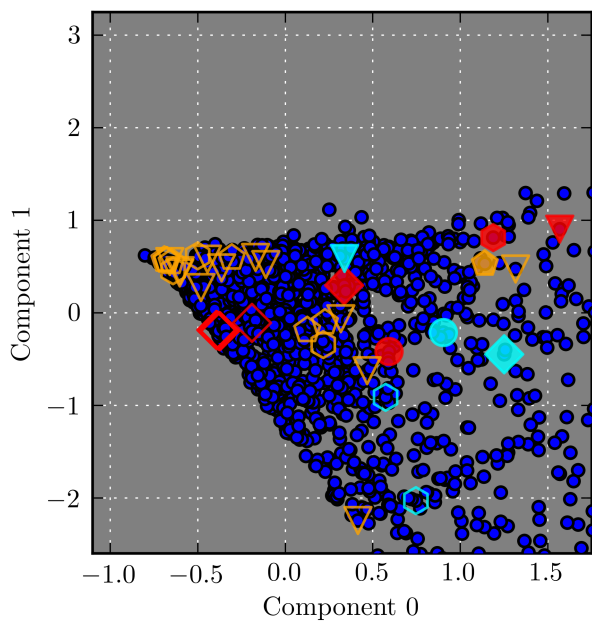
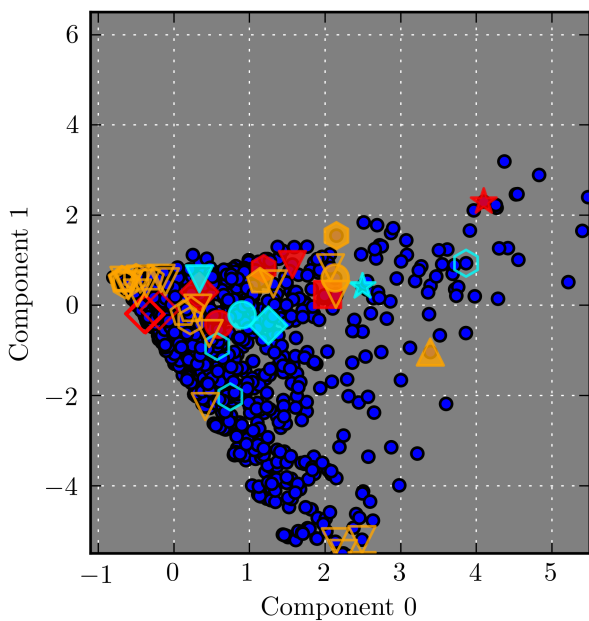
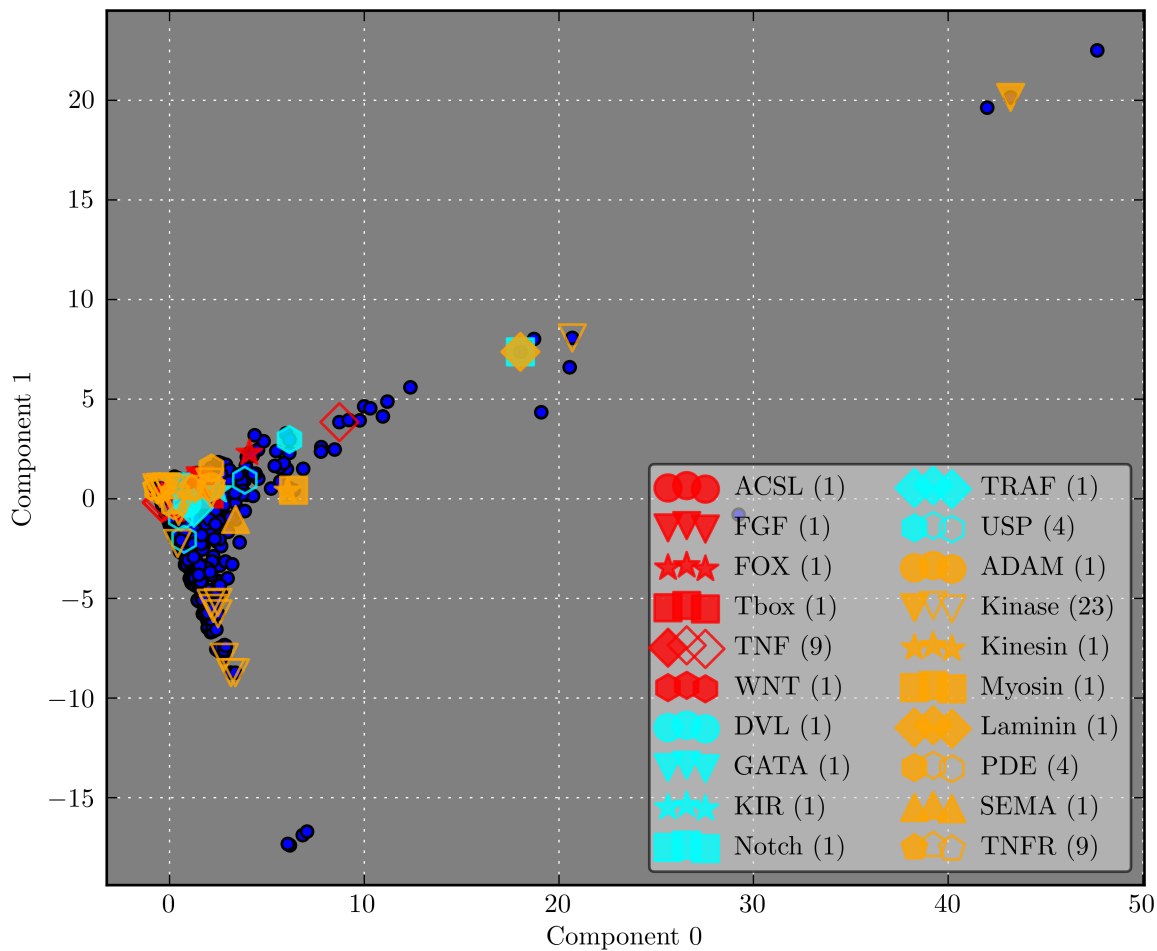


Figure D.18: **Human:** Projection of clusters into PCA component space. Clusters that contain at least one curated family member are outlined. The most representative cluster is shaded, and the number of clusters is in parentheses.

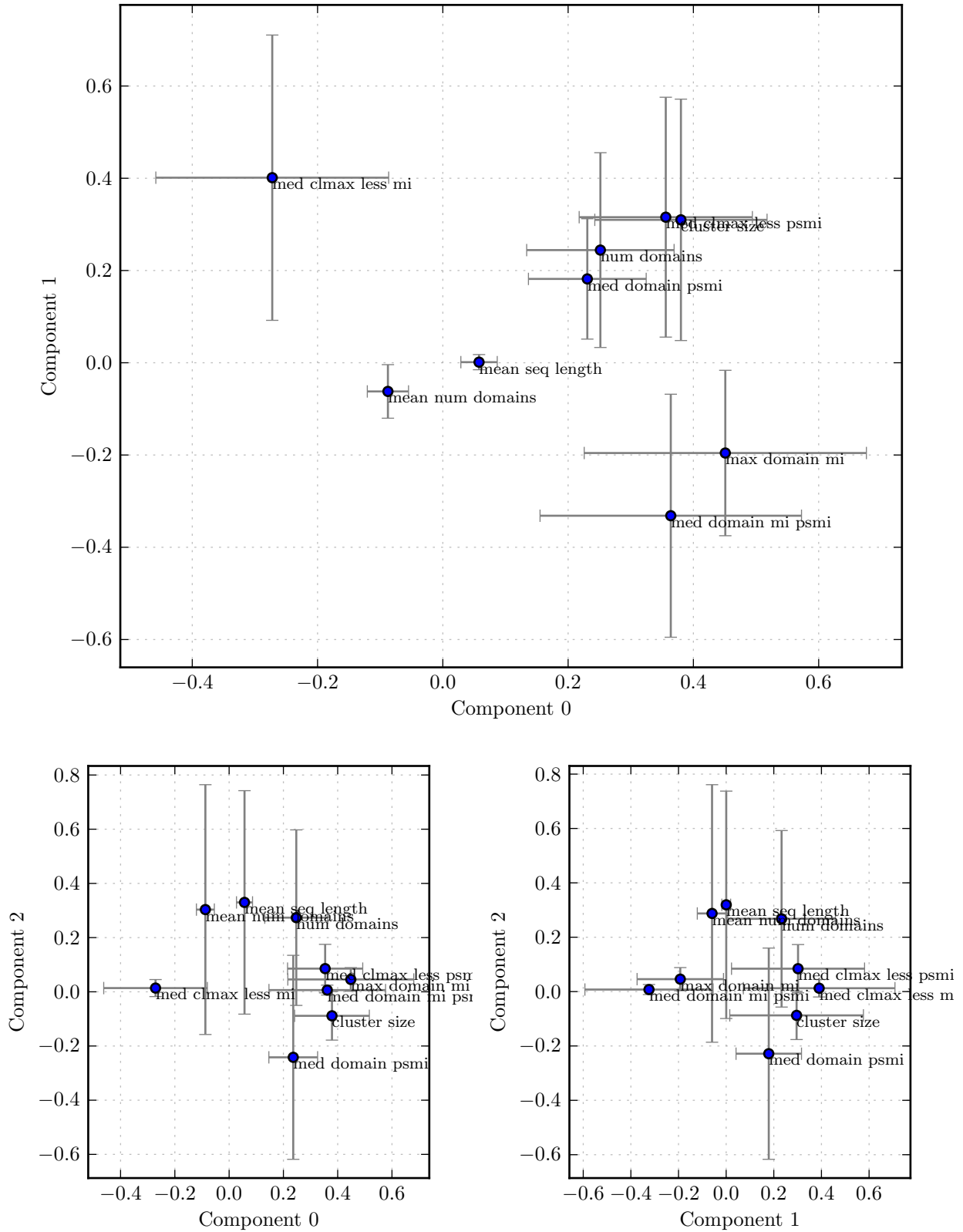


Figure D.19: **Mouse:** Coefficients of individual features in the PCA component space. Error bars indicate the standard deviation over 1000 samplings of 75% of the clusters in the clustering.

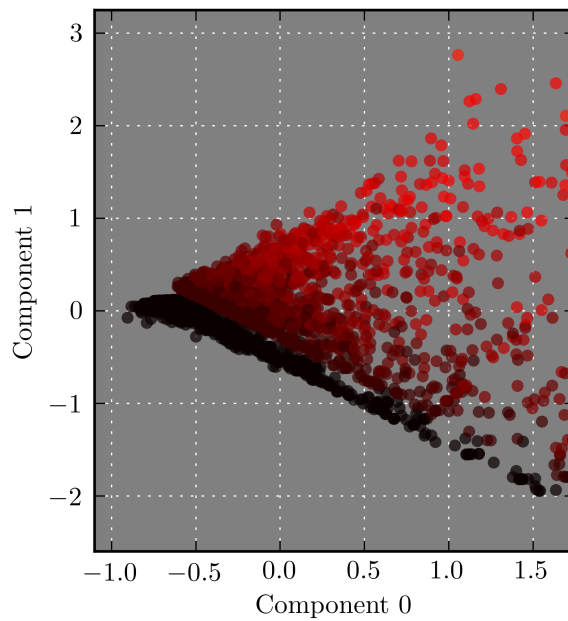
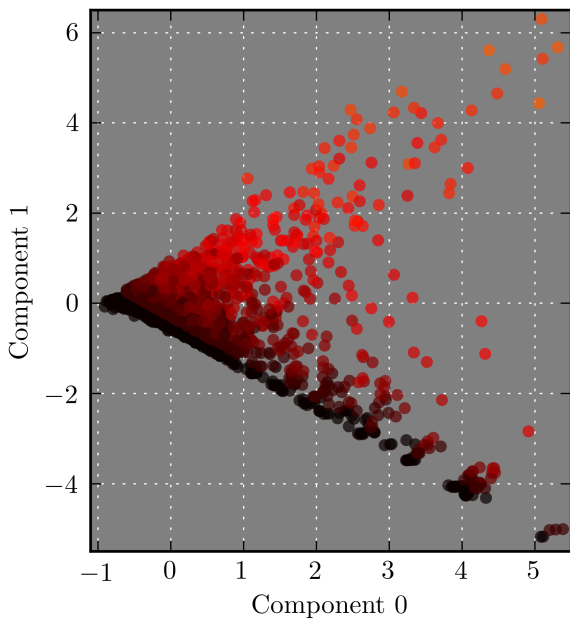
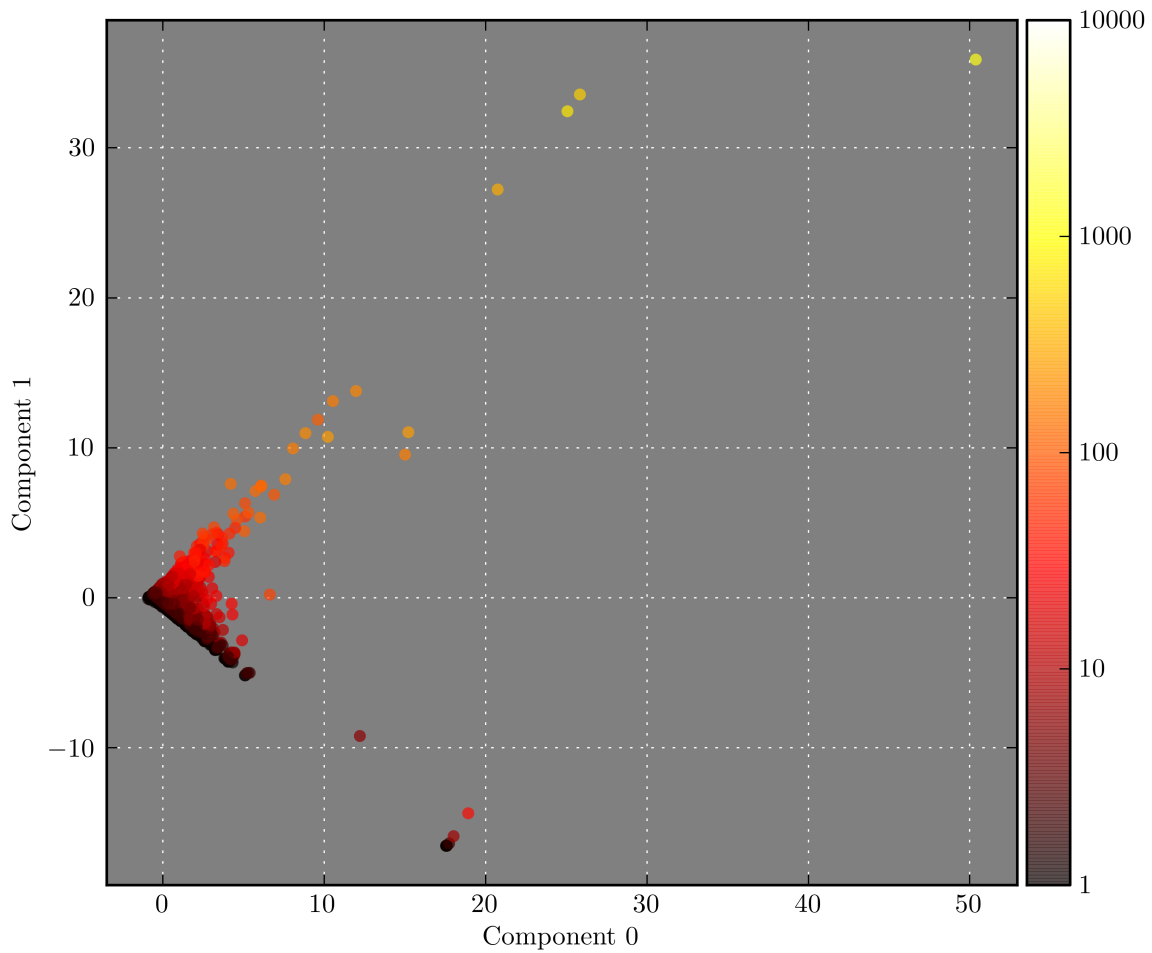


Figure D.20: **Mouse:** Projection of clusters into PCA component space. Clusters in this figure are colored according to the number of sequences in found in that cluster.

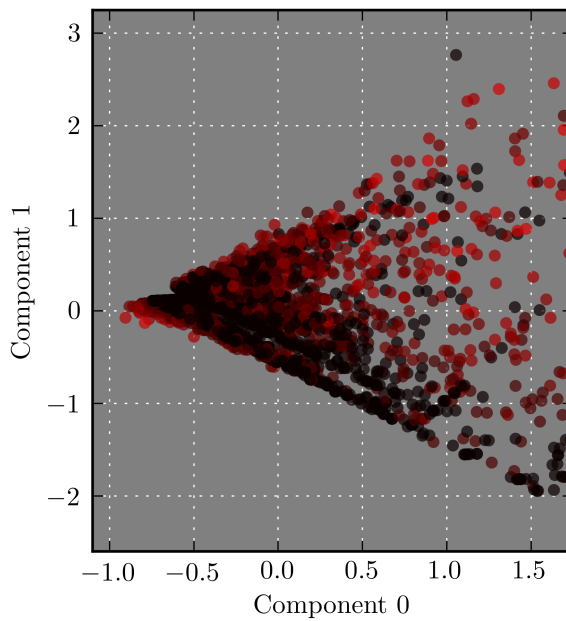
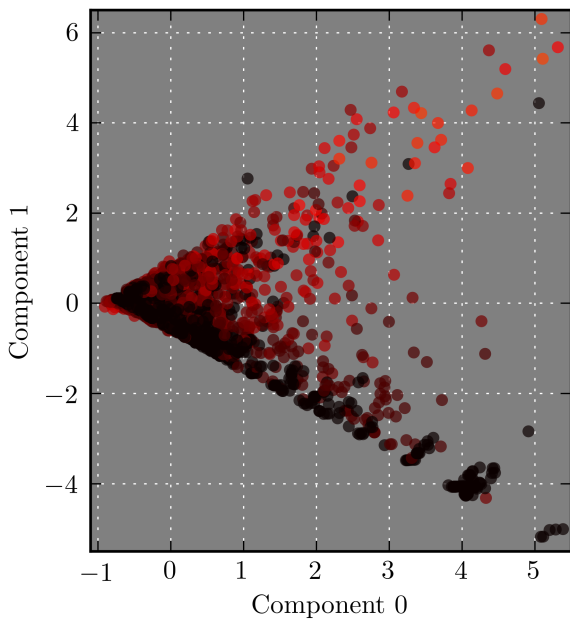
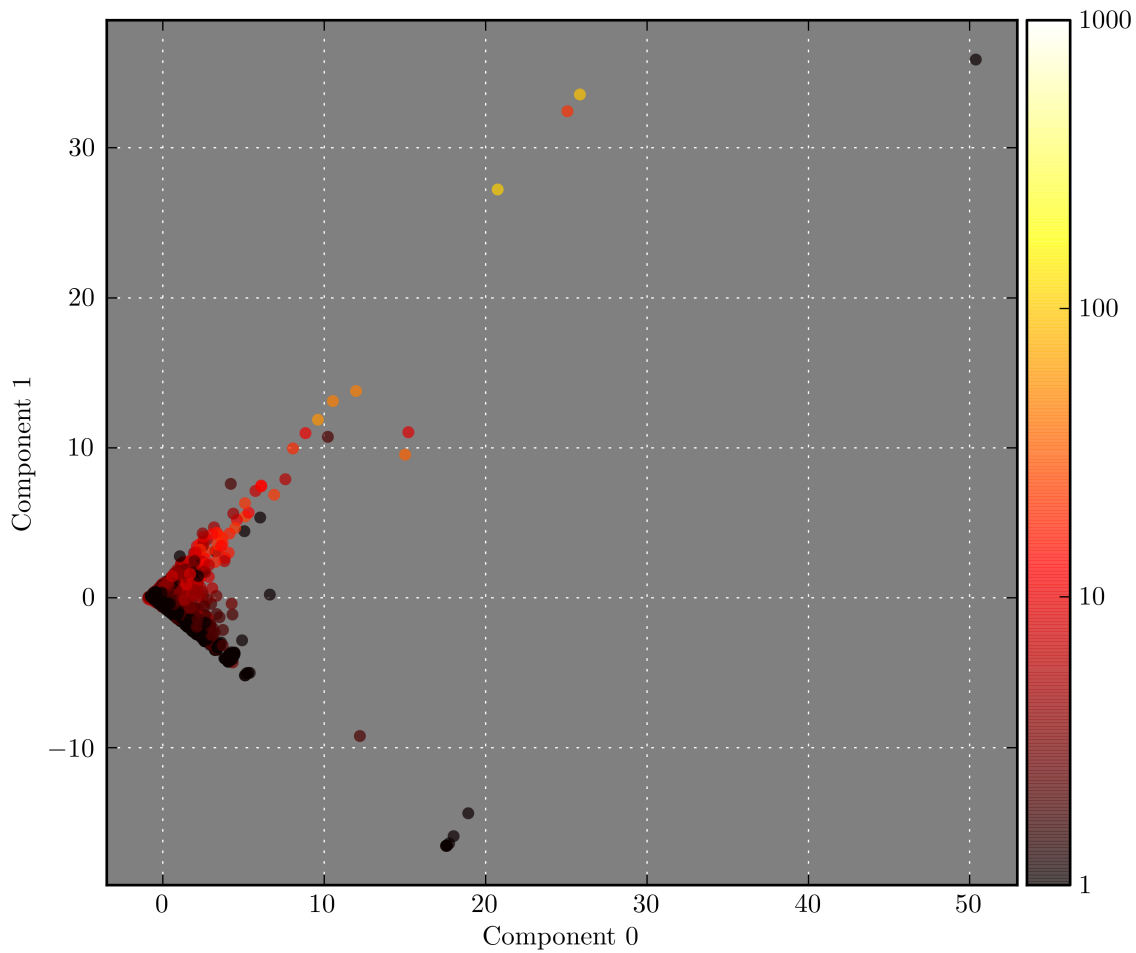


Figure D.21: **Mouse:** Projection of clusters into PCA component space. Clusters in this figure are colored according to the total number of distinct domains found in that cluster.

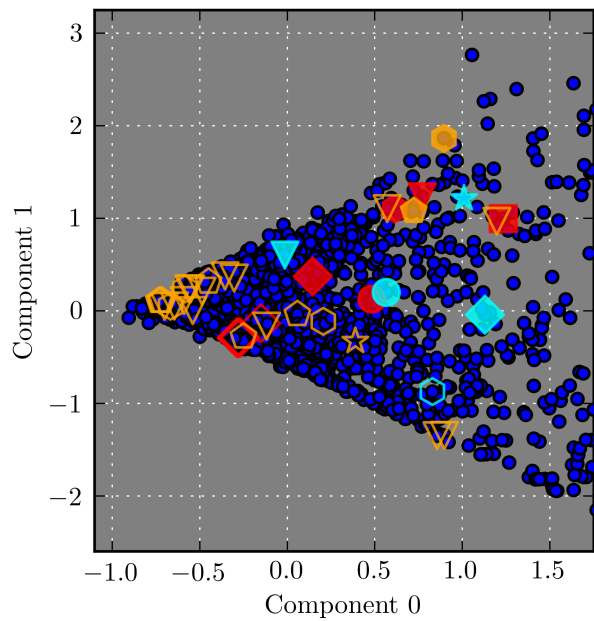
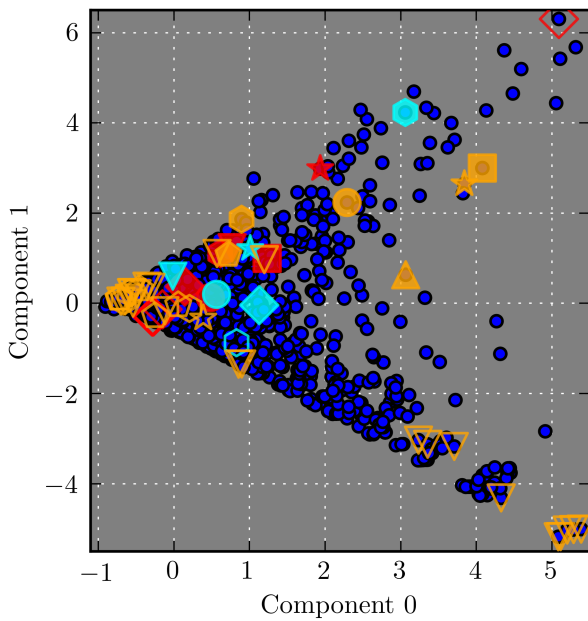
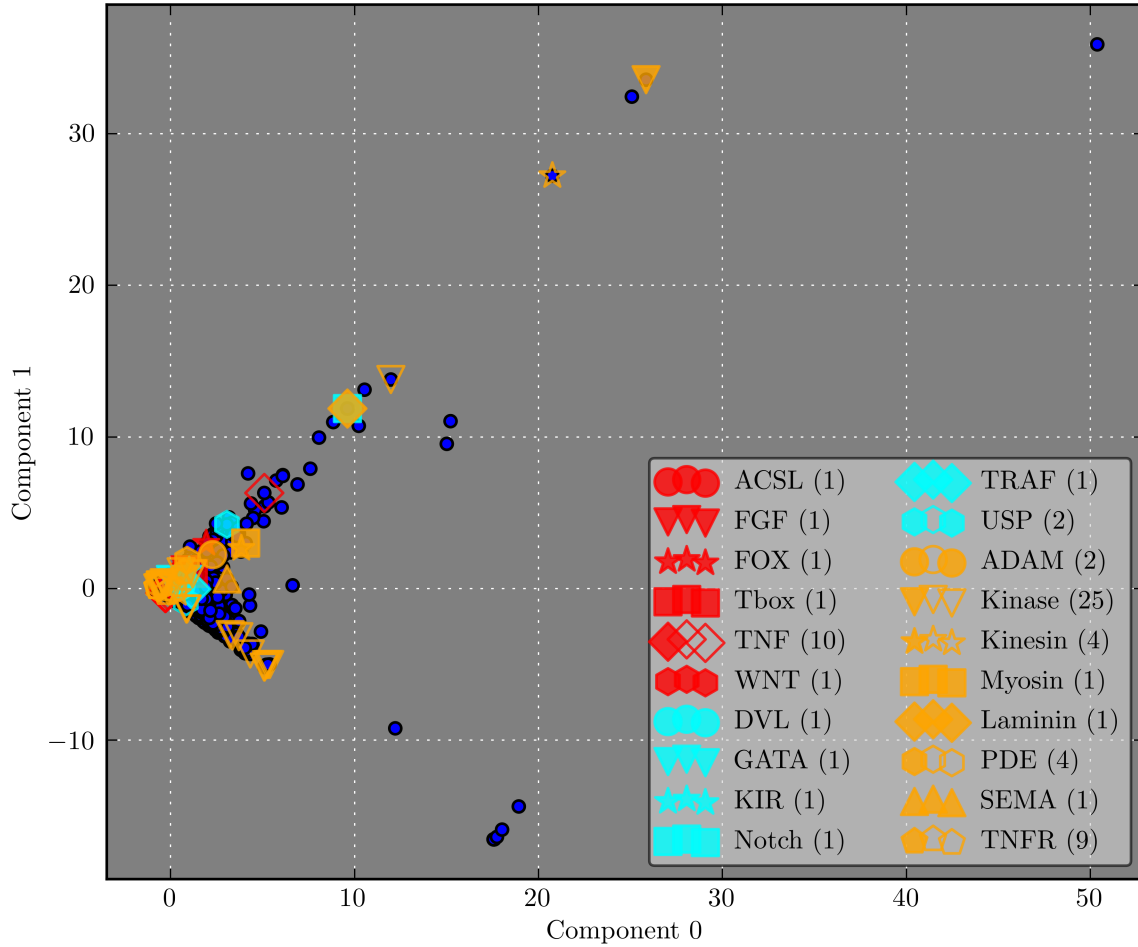


Figure D.22: **Mouse:** Projection of clusters into PCA component space. Clusters that contain at least one curated family member are outlined. The most representative cluster is shaded, and the number of clusters is in parentheses.

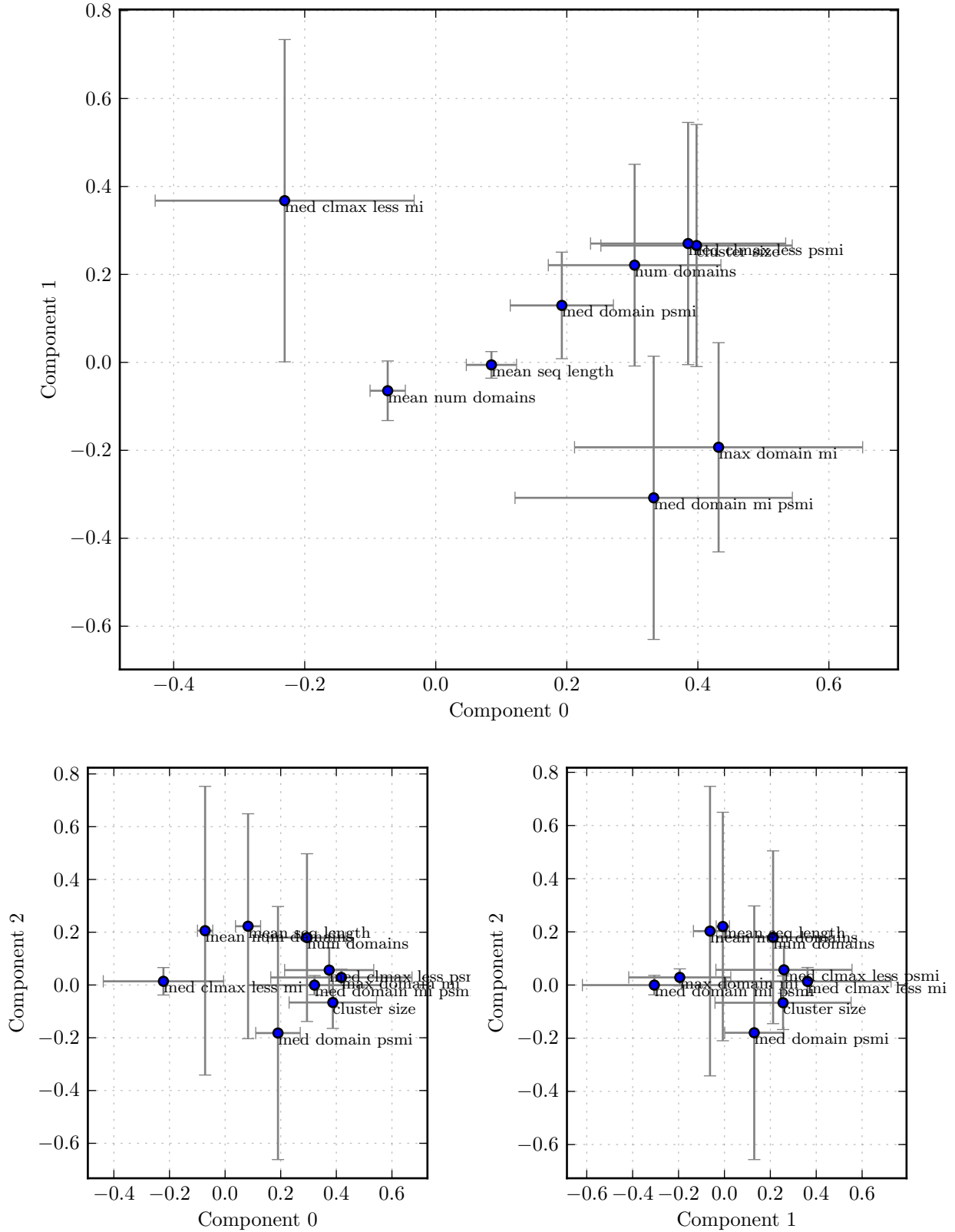


Figure D.23: **Human and mouse:** Coefficients of individual features in the PCA component space. Error bars indicate the standard deviation over 1000 samplings of 75% of the clusters in the clustering.

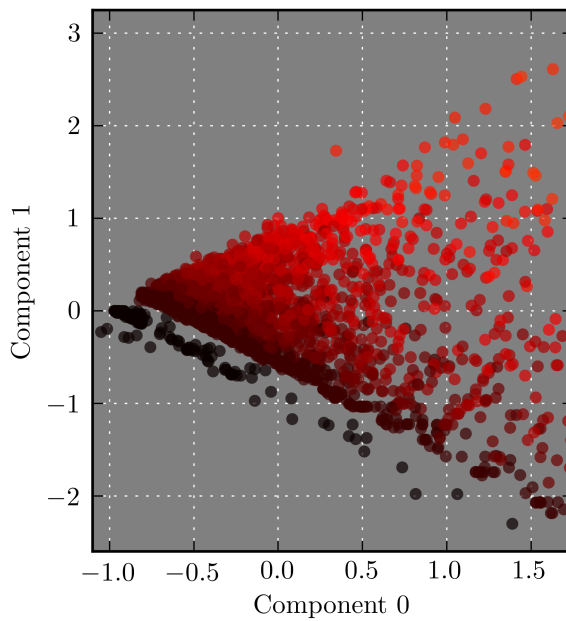
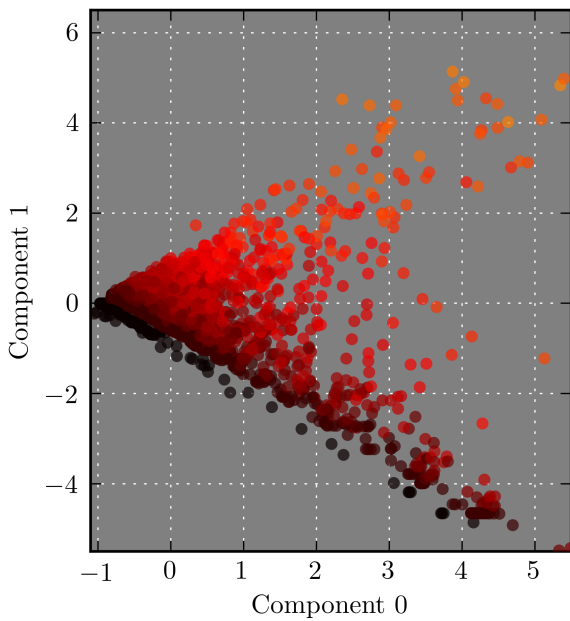
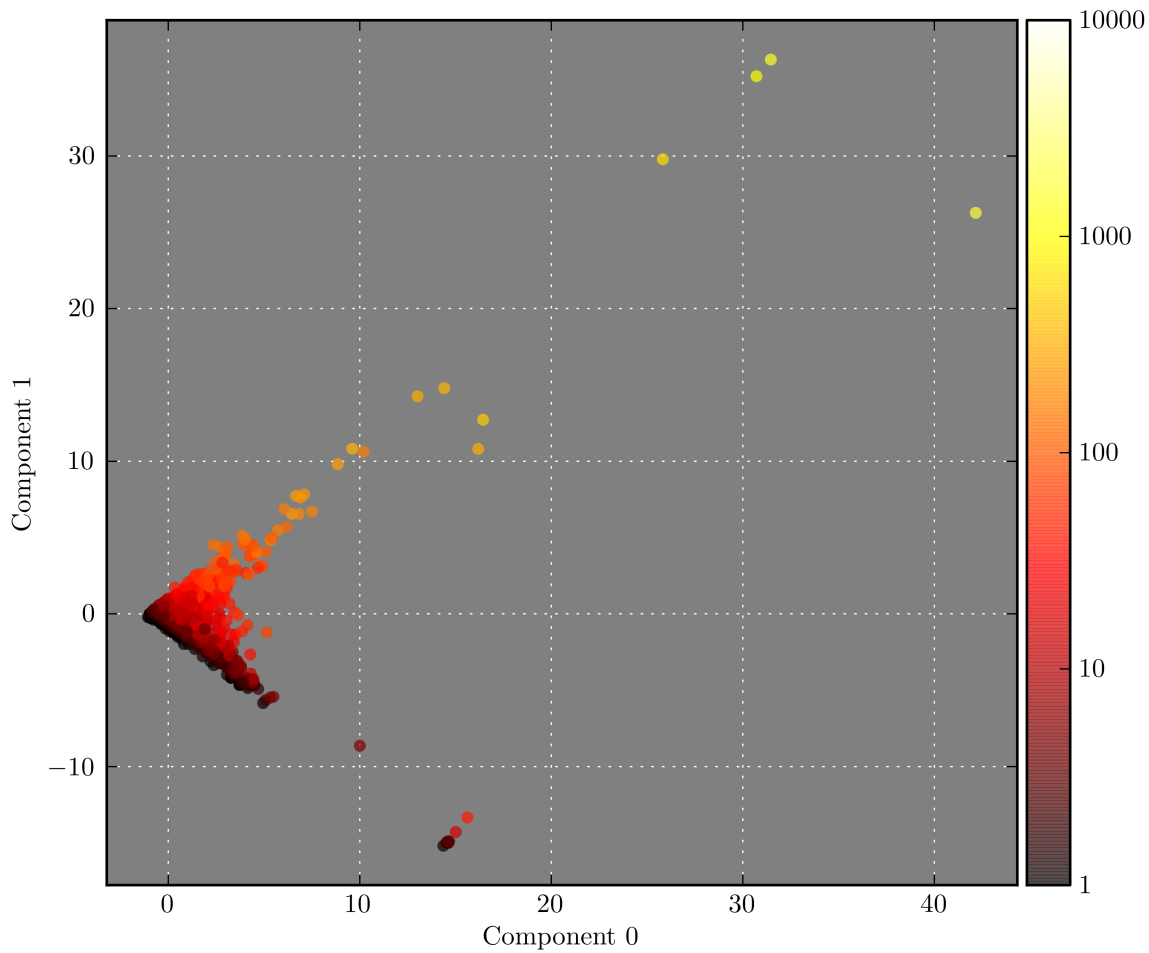


Figure D.24: **Human and mouse:** Projection of clusters into PCA component space. Clusters in this figure are colored according to the number of sequences in found in that cluster.

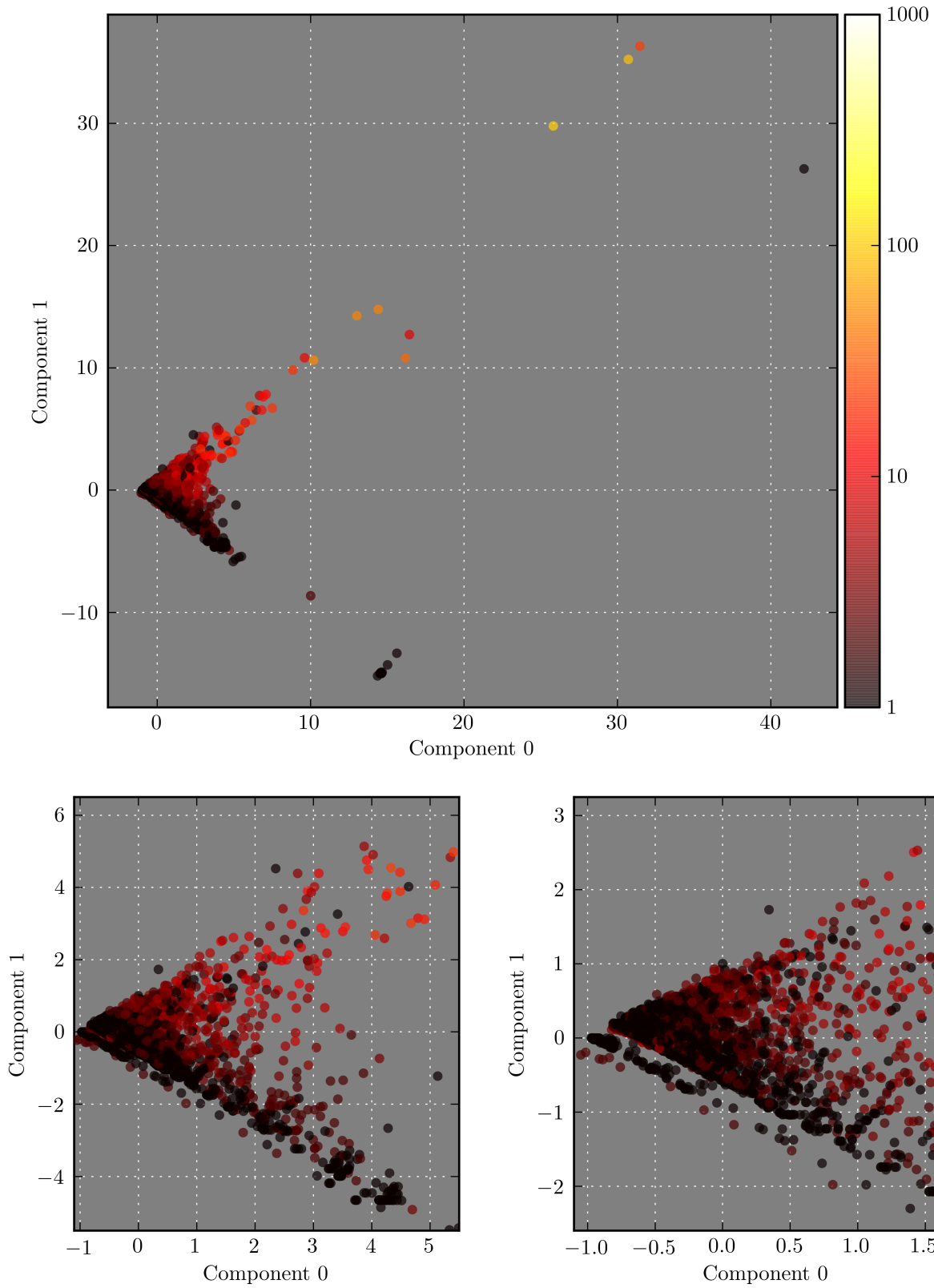


Figure D.25: **Human and mouse:** Projection of clusters into PCA component space. Clusters in this figure are colored according to the total number of distinct domains found in that cluster.

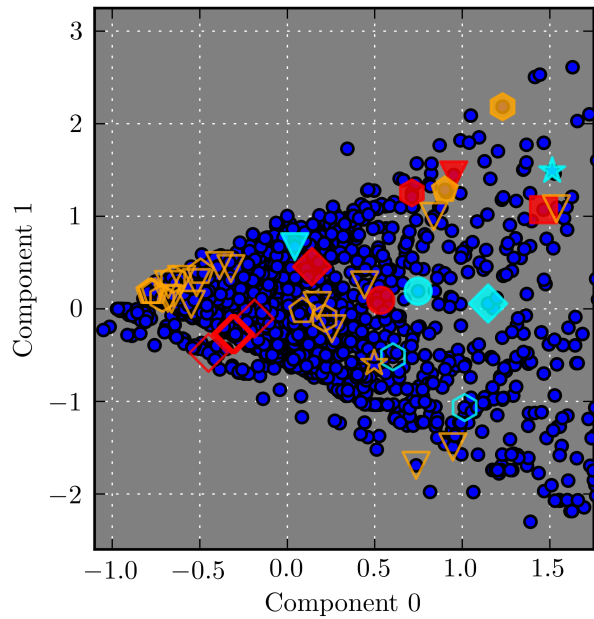
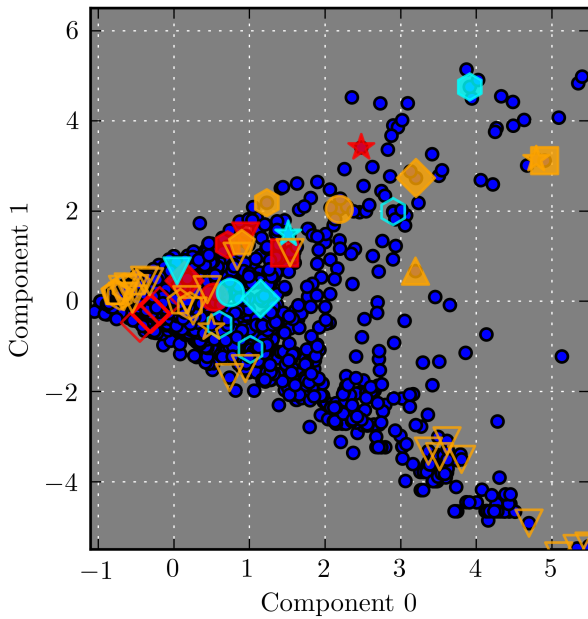
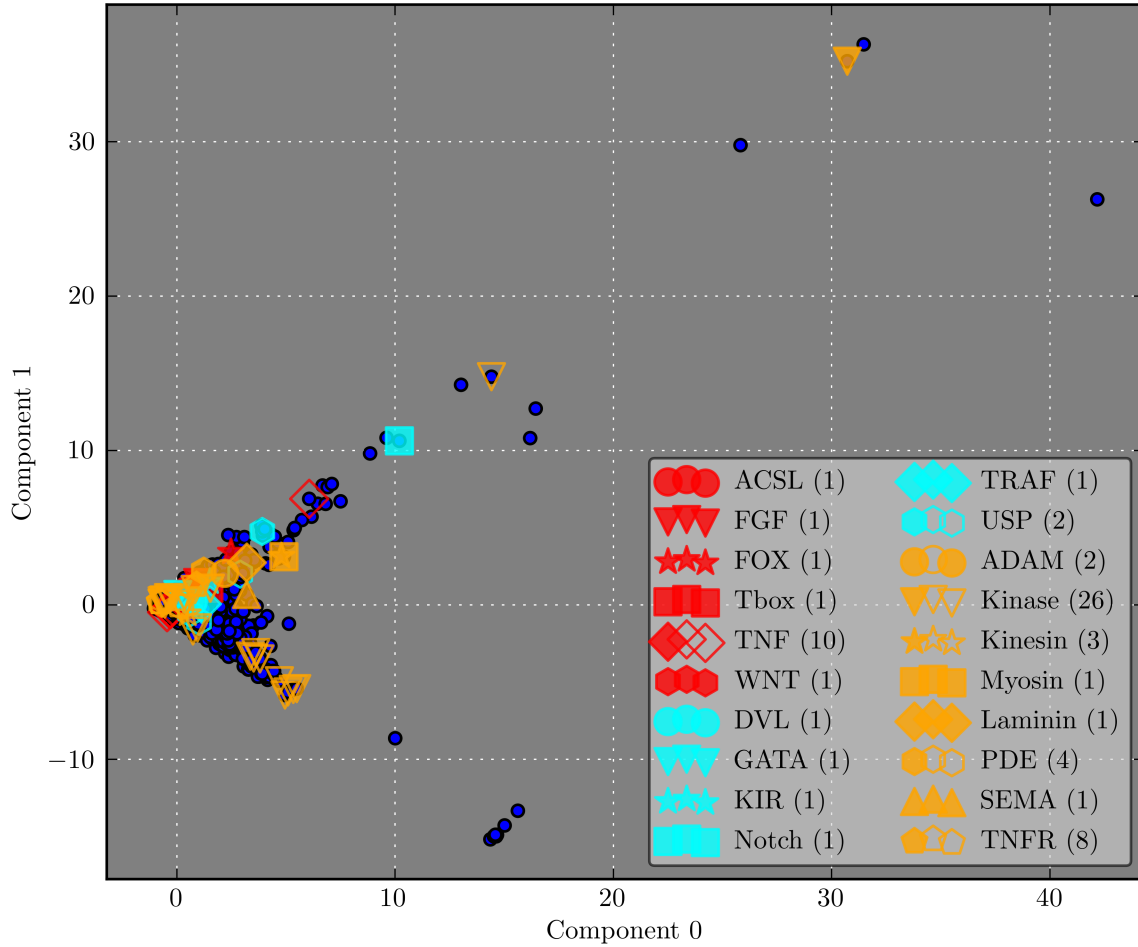


Figure D.26: **Human and mouse:** Projection of clusters into PCA component space. Clusters that contain at least one curated family member are outlined. The most representative cluster is shaded, and the number of clusters is in parentheses.

List of Figures

1.1	The evolutionary history of a hypothetical multidomain family, showing both gene duplications and domain insertions. Gene loci are depicted as lines, overlaid by polygons that represent domains. Genes x , y , and z share a common ancestor (a_0), but do not have identical domain composition; a domain (diamond, in green) was inserted into the ancestral gene a_1 after the divergence of gene z . Gene w shares a homologous domain with these genes, though there is no gene that is ancestral to both w and a member of the family.	7
1.2	Work-flow of data and methods used in this thesis. Arrows represent data flow. Double lines represent parallel data paths; e.g., clustering is performed on either a sequence similarity network or a Neighborhood Correlation network, and the resulting tree is specific to that data input. Brackets indicate the chapter number in which a method is discussed.	10
2.1	(a) A set of families grouped together due to a <i>chain</i> of domains, each found in one or more clusters. Note that no single domain is found throughout all families of the resulting cluster. (b) A set of families grouped together due to a single shared domain. Families are depicted as bounding circles, and domains as polygons on individual genes (lines).	20
3.1	Histogram of the sequences similarity scores that result between homologous pairs of sequences (in blue, wrt. right axis) of the 20-family benchmark, in the combined mouse and human genomes.	27
3.2	Histogram of the sequence similarity scores that result between homologous pairs of sequences (in blue, wrt. right axis), and between non-homologous pairs (in red, wrt. left axis), in the mouse and human genomes. These demonstrate that an effective bit-score threshold can be selected for some families, though no single threshold is suitable for all families.	28
3.3	Histogram of the sequence similarity scores that result between homologous pairs of sequences (in blue, wrt. right axis), and between non-homologous pairs (in red, wrt. left axis), in the human and mouse genomes. No bit-score threshold may be selected for any of these families to effectively separate homologous and non-homologous pairs of sequences.	29
3.4	Example graph components for intuition. In (a), x and y are members of families f_x and f_y , respectively, but joined by a single edge. (b) depicts a single family missing two edges, while (c) illustrates a case where edge weights must be used to distinguish between edge addition or deletion.	32
3.5	Histograms of the Neighborhood Correlation scores for curated families in mouse and human. Homologous pairs are in blue (wrt. right axis), and non-homologous pairs are in red (wrt. left axis). Additionally, in green (wrt. left axis) are scores of all pairs that include one member of a curated family, and one member from the 12-genome dataset, and not in mouse or human. . . .	33

3.6	Histograms of the Neighborhood Correlation scores for curated families in mouse and human. Homologous pairs are in blue (wrt. right axis), and non-homologous pairs are in red (wrt. left axis). Additionally, in green (wrt. left axis) are scores of all pairs that include one member of a curated family, and one member from the 12-genome dataset, and not in mouse or human. . . .	34
3.7	Histograms of the Neighborhood Correlation scores for the aggregate of all curated families in mouse and human. Homologous pairs are in blue (wrt. right axis), and non-homologous pairs are in red (wrt. left axis). Additionally, in green (wrt. left axis) are scores of all pairs that include one member of a curated family, and one member from the 12-genome dataset, and not in mouse or human. ALL is the set of all 20 families, while ALL-kinase is the set of 19 families, with the largest family, Kinase, excepted.	35
3.8	Heatmap of Neighborhood Correlation scores calculated using the complete network of sequence similarity (horizontal axis) and the first 500 hits returned by BLAST for each query sequence (vertical axis).	39
3.9	Heatmap of Neighborhood Correlation scores calculated using the complete network of sequence similarity (horizontal axis) 500-hits returned by BLAST for each query sequence (vertical axis). .	40
4.1	Demonstration of NESTED SET tree indexing. For each node (n) in the tree, two integers are stored. These are <i>left</i> (l_n) and <i>right</i> (r_n) indices are generated by sequential numbering of nodes during a depth first search. The value l_n is assigned when a node is first traversed, and r_n is assigned when the depth-first search revisits the node. All descendants of a node with indices (x, y) have values $(l_n > x, r_n < y)$, allowing them to be directly identified. All leaves have indices of the form $r_n = l_n + 1$	56
5.1	Scenarios involving various connected component motifs. The indicated change in magnitude of a network measure is reported as edges are removed from a component. Arrows indicate either an increase or decrease in the magnitude of a measure. These projections assume the component is embedded within a larger network of other components, and that larger network does not change.	71
5.2	Component and transitivity measures of simulated networks of cliques degraded by noise. A network of 1024 nodes is used, comprised of 16 cliques of size 4, and 8 each of sizes 8, 16, 32, and 64 nodes. The vertical bars represent the standard error over 1000 trials.	75
5.3	Component and transitivity measures of simulated networks of cliques degraded by noise. A network of 1024 nodes is used, comprised of 256 cliques of size 4. The vertical bars represent the standard error over 1000 trials.	76
5.4	Visualization of the <i>S. cerevisiae</i> genome after rescoring with Neighborhood Correlation. Edge color signifies the Neighborhood Correlation score, where gray indicates $NC \geq 0.3$, violet ≥ 0.4 , green ≥ 0.6 , orange ≥ 0.8 , and yellow ≥ 0.9 . The dense component at top-right contains all Kinases. Singleton nodes have been omitted for clarity.	78
5.5	Measures of the networks comprised of the 5616 genes in <i>S. cerevisiae</i> , with edges from sequence similarity, and Neighborhood Correlation calculated with one, four, and nine yeast genomes. NC score thresholds range from 0–1. The bit-score axis ranges from 31–1000 and is aligned with NC such that the density of the sequence similarity network is equivalent to the NC network of nine genomes.	80
5.6	Component and transitivity measures of the sequence similarity and Neighborhood Correlation networks. These networks are comprised of all Human and Mouse sequences in the 48-genome Panther dataset, and contain 45491 sequences.	82
6.1	Performance of hierarchical clustering of sequence similarity, using the single-linkage method. This figure depicts the Precision, Recall, and F that result from selection of the partitioning induced by cutting the tree at a range of cluster distance thresholds.	93

6.2	Performance of hierarchical clustering of sequence similarity, using the average-linkage method. This figure depicts the Precision, Recall, and F that result from selection of the partitioning induced by cutting the tree at a range of cluster distance thresholds.	94
6.3	Color legend for all heatmaps in this chapter	95
6.4	Heatmap of the F-statistic for all 20 benchmark families. Hierarchical clustering, by the single, average, and complete linkage methods has been performed for the set of mouse and human sequences using bit-score.	96
6.5	Heatmap of Precision for all 20 benchmark families. Hierarchical clustering, by the single, average, and complete linkage methods has been performed for the set of mouse and human sequences using bit-score.	97
6.6	Heatmap of Recall for all 20 benchmark families. Hierarchical clustering, by the single, average, and complete linkage methods has been performed for the set of mouse and human sequences using bit-score.	98
6.7	Clustering performance of single-linkage clustering of the Neighborhood Correlation network. . .	100
6.8	Clustering performance of average-linkage clustering of the Neighborhood Correlation network. .	101
6.9	Heatmap of the F-statistic for all 20 benchmark families. Hierarchical clustering, by the single, average, and complete linkage methods has been performed for the set of mouse and human sequences using Neighborhood Correlation scores computed with the 48-genome dataset.	102
6.10	Heatmap of Precision for all 20 benchmark families. Hierarchical clustering, by the single, average, and complete linkage methods has been performed for the set of mouse and human sequences using Neighborhood Correlation scores computed with the 48-genome dataset.	103
6.11	Heatmap of Recall for all 20 benchmark families. Hierarchical clustering, by the single, average, and complete linkage methods has been performed for the set of mouse and human sequences using Neighborhood Correlation scores computed with the 48-genome dataset.	104
6.12	Clustering performance of average-linkage clustering of the sequence similarity network of all 48 genomes in the Panther 7.0 dataset.	106
6.13	Clustering performance of average-linkage clustering of the Neighborhood Correlation network of all 48 genomes in the Panther 7.0 dataset.	107
6.14	Heatmap of the F-statistic for all 20 benchmark families when clustering is performed on the full 600k, 48-genome dataset. Results for hierarchical clustering using the average-linkage method is shown for both sequence similarity, and Neighborhood Correlation.	108
7.1	Mutual information of all domains in the working example, separated by cluster. For each domain found in a cluster, there are two points in the column of that cluster: MI, the mutual information of this domain with the entire clustering, and the pSMI of this domain and the particular cluster. Additionally, the MI cluster maximum is the maximal mutual information attainable for a hypothetical domain that occurs in every sequence of that cluster and no sequence outside of the cluster.	115
7.2	Clustering entropy as a function of the threshold used to partition the Neighborhood Correlation average-linkage clustering result. The entropy of each partitioning is in blue. The MI sum is the sum of the mutual information, over all domains. This latter value is a loose upper bound of the combined joint mutual information of all domains, calculated as in [84].	121
7.3	Mutual information of all domains in the 20-family benchmark set, separated by family. Families are ordered by descending size, indicated in parentheses. For each domain found in a family, there are two points in the column of that family: MI, the mutual information of that domain with the set of all families, and pSMI of this domain in the family. Additionally, the MI cluster maximum for each family is the maximal mutual information that would be obtained by a hypothetical domain that occurs exclusively in every sequence of that family.	123

7.4	Mutual information of all domains in clusters of human and mouse sequences. (a) contains the full set of 9000 clusters. (b) depicts only the largest clusters, of size indicated in parentheses. Clusters are ordered by descending size. The properties of each contained domain are represented by two points in the column of a cluster: MI, the mutual information of that domain with the entire clustering, and pSMI of this domain in the cluster. Additionally, the MI cluster maximum is depicted.	124
7.5	Mutual information as compared to the entropy of each domain in the family benchmark. The most numerous domains are labeled by Pfam identifier, with the number of sequence instances in parentheses.	126
7.6	Mutual information as compared to the entropy of each domain in the clustering of mouse and human sequences. The most numerous domains are labeled by Pfam identifier, with the number of sequence instances in parentheses.	127
7.7	Fraction of variance explained by successive PCA components. The first two components, alone, explain more 60% of the total data variance. The first three components explain approximately 75% of the variance.	129
7.8	Coefficients of individual features in the PCA component space. Error bars indicate the standard deviation over 1000 samplings of 75% of the clusters in the clustering.	130
7.9	Projection of clusters into PCA component space. This figure is a heatmap of 100x100 bins, where black points indicate a single cluster, and increasingly bright points represent as 3000 clusters at that coordinate.	132
7.10	Projection of clusters into PCA component space. Clusters in this figure are colored according to the number of sequences in found in that cluster.	134
7.11	Projection of clusters into PCA component space. Clusters in this figure are colored according to the total number of distinct domains found in that cluster.	135
7.12	Projection of clusters into PCA component space. Clusters that contain at least one curated family member are outlined. The most representative cluster is shaded, and the number of clusters is in parentheses.	137
7.13	Projection of clusters into PCA component space. Clusters in this figure are colored according to med-clmax-less-psmi.	139
7.14	Projection of clusters into PCA component space. Clusters in this figure are colored according to med-clmax-less-mi for that cluster.	140
C.1	Histogram of the sequence similarity scores that result between homologous pairs of sequences (in blue, wrt. right axis), and between non-homologous pairs (in red, wrt. left axis), in the mouse and human genomes. These demonstrate that an effective bit-score threshold can be selected for some families, though no single threshold is suitable for all families. See Figures 3.2, 3.3, and C.2.	156
C.2	Histogram of the sequence similarity scores that result between homologous pairs of sequences (in blue, wrt. right axis), and between non-homologous pairs (in red, wrt. left axis), in the mouse and human genomes. These demonstrate that an effective bit-score threshold can be selected for some families, though no single threshold is suitable for all families. See Figures 3.2, 3.3, and C.1.	157
C.3	Histograms of the Neighborhood Correlation scores for curated families in mouse and human. Homologous pairs are in blue (wrt. right axis), and non-homologous pairs are in red (wrt. left axis). Additionally, in green (wrt. left axis) are scores of all pairs that include one member of a curated family, and one member from the 12-genome dataset, and not in mouse or human. See Figures 3.5, 3.6, and C.4.	158
C.4	Histograms of the Neighborhood Correlation scores for curated families in mouse and human. Homologous pairs are in blue (wrt. right axis), and non-homologous pairs are in red (wrt. left axis). Additionally, in green (wrt. left axis) are scores of all pairs that include one member of a curated family, and one member from the 12-genome dataset, and not in mouse or human. See Figures 3.5, 3.6, and C.3.	159

D.1	<i>S. purpuratus</i> : Coefficients of individual features in the PCA component space. Error bars indicate the standard deviation over 1000 samplings of 75% of the clusters in the clustering.	162
D.2	<i>S. purpuratus</i> : Projection of clusters into PCA component space. Clusters in this figure are colored according to the number of sequences in found in that cluster.	163
D.3	<i>S. purpuratus</i> : Projection of clusters into PCA component space. Clusters in this figure are colored according to the total number of distinct domains found in that cluster.	164
D.4	<i>D. melanogaster</i> : Coefficients of individual features in the PCA component space. Error bars indicate the standard deviation over 1000 samplings of 75% of the clusters in the clustering.	165
D.5	<i>D. melanogaster</i> : Projection of clusters into PCA component space. Clusters in this figure are colored according to the number of sequences in found in that cluster.	166
D.6	<i>D. melanogaster</i> : Projection of clusters into PCA component space. Clusters in this figure are colored according to the total number of distinct domains found in that cluster.	167
D.7	<i>C. elegans</i> : Coefficients of individual features in the PCA component space. Error bars indicate the standard deviation over 1000 samplings of 75% of the clusters in the clustering.	168
D.8	<i>C. elegans</i> : Projection of clusters into PCA component space. Clusters in this figure are colored according to the number of sequences in found in that cluster.	169
D.9	<i>C. elegans</i> : Projection of clusters into PCA component space. Clusters in this figure are colored according to the total number of distinct domains found in that cluster.	170
D.10	<i>S. cerevisiae</i> : Coefficients of individual features in the PCA component space. Error bars indicate the standard deviation over 1000 samplings of 75% of the clusters in the clustering.	171
D.11	<i>S. cerevisiae</i> : Projection of clusters into PCA component space. Clusters in this figure are colored according to the number of sequences in found in that cluster.	172
D.12	<i>S. cerevisiae</i> : Projection of clusters into PCA component space. Clusters in this figure are colored according to the total number of distinct domains found in that cluster.	173
D.13	Ordering of clusters in mouse and human by their position within PCA components 0 (a) and 1 (b).	175
D.14	Ordering of clusters in mouse and human by their position within PCA component 2.	176
D.15	HUMAN : Coefficients of individual features in the PCA component space. Error bars indicate the standard deviation over 1000 samplings of 75% of the clusters in the clustering.	177
D.16	HUMAN : Projection of clusters into PCA component space. Clusters in this figure are colored according to the number of sequences in found in that cluster.	178
D.17	HUMAN : Projection of clusters into PCA component space. Clusters in this figure are colored according to the total number of distinct domains found in that cluster.	179
D.18	HUMAN : Projection of clusters into PCA component space. Clusters that contain at least one curated family member are outlined. The most representative cluster is shaded, and the number of clusters is in parentheses.	180
D.19	MOUSE : Coefficients of individual features in the PCA component space. Error bars indicate the standard deviation over 1000 samplings of 75% of the clusters in the clustering.	181
D.20	MOUSE : Projection of clusters into PCA component space. Clusters in this figure are colored according to the number of sequences in found in that cluster.	182
D.21	MOUSE : Projection of clusters into PCA component space. Clusters in this figure are colored according to the total number of distinct domains found in that cluster.	183
D.22	MOUSE : Projection of clusters into PCA component space. Clusters that contain at least one curated family member are outlined. The most representative cluster is shaded, and the number of clusters is in parentheses.	184
D.23	HUMAN AND MOUSE : Coefficients of individual features in the PCA component space. Error bars indicate the standard deviation over 1000 samplings of 75% of the clusters in the clustering.	185
D.24	HUMAN AND MOUSE : Projection of clusters into PCA component space. Clusters in this figure are colored according to the number of sequences in found in that cluster.	186
D.25	HUMAN AND MOUSE : Projection of clusters into PCA component space. Clusters in this figure are colored according to the total number of distinct domains found in that cluster.	187

D.26 **Human and mouse:** Projection of clusters into PCA component space. Clusters that contain at least one curated family member are outlined. The most representative cluster is shaded, and the number of clusters is in parentheses. 188

List of Tables

- 2.1 Curated benchmark of gene families, in the mouse and human genomes. 22
- 3.1 BLAST parameters used for all-against-all sequence similarity calculation. N is the number of sequences in the database, and Y is the number of residues. Parameters and values correspond to `blastall`, version 2.2.16. All parameters not specified are left at default values in this version of blast. Some of these parameters are the default, and are included for clarity. 37
- 7.1 Features of clusters. 128
- B.1 Table (part 1 of 2) of the 48-genomes included in the Panther 7.0 data set [141] used throughout this dissertation. 153
- B.2 Table (part 2 of 2) of the 48-genomes included in the Panther 7.0 data set [141] used throughout this dissertation. 154

List of Code

4.1	SQL table definitions for the storage of amino acid strings. <code>prot_seq_str</code> , short for “protein sequence string”, defines storage of an amino acid sequence. <code>crc</code> , a checksum, <code>length</code> , and <code>molecular weight</code> are stored to optimize look-up using an index when a sequence string is known, typically to check for an existing, duplicate sequence string. <code>prot_seq</code> establishes an internal sequence identifier (<code>seq_id</code>), and a pointer to a single protein sequence string. Many sequence identifiers may map to one sequence string, but every string stored in the table is unique.	46
4.2	SQL table definitions used to represent the data source and version from which a sequence was obtained. As is most appropriate in relational databases, separate tables are used to specifically define many-to-one relationships, such as many versions (<code>prot_seq_source_ver</code>) of the same data source (<code>prot_seq_source</code>).	47
4.3	SQL table definitions for sets of sequences. <code>prot_seq</code> defines a unique identifier for the set (<code>set_id</code>) and a human-readable name and description. <code>prot_seq_set_member</code> then associates sequence identifiers with that set instance.	48
4.4	SQL table definitions to represent complete “runs” of BLAST, and Neighborhood Correlation. The latter dependency upon BLAST is encoded by the table definitions. Each of these tables detail the complete set of parameters used during execution of the method.	49
4.5	A DICTARRAY: the network adjacency list representation. The nodes in a network are represented by a set of consecutive unsigned integers. Edge weights are represented as floating-point values. A hash data structure is used to map each Node to two arrays, one a list of nodes connected, and another a list of the corresponding edge weights. These lists are in order of the Node integer.	53
4.6	Definition of the table for storage of network edges as individual rows. Each row contains an integer identifying the network the edge corresponds to (<code>nc_id</code>), two node endpoints, and the edge weight. Beyond the size of the datatypes defined, each row in the database incurs an additional 20–40 bytes for internal representation, making this an inefficient means of network storage.	54
4.7	SQL table definition for storage of network edges, directly mirroring a DICTARRAY. In contrast to <code>blast_hit_nc</code> , the row overhead as compared to the size of the data is minimal, and access to all neighbors of a node may be achieved by querying a single row.	54
4.8	Definition of an SQL view to transparently emulate the <code>blast_hit_nc</code> table from the structure of a <code>blast_hit_nc_arr</code> table. The function <code>unzip()</code> transforms the array data type to a list of individual rows.	55
4.9	SQL table definition to represent hierarchical clustering trees. The indices created facilitate rapid arithmetic comparison of the <code>lft</code> and <code>rgt</code> NESTED SET indices.	57
4.10	SQL query to select all tree nodes immediately adjacent to a chosen threshold, specified by the parameter <code>%(distance)s</code> . The parameter <code>%(cr_id)s</code> references a specific tree stored in the table <code>jj_hcluster</code> .	58

4.11	SQL query to select all leaves under a given node, specified by parameter <code>%(cluster_id)s</code> in a given tree, which is specified by <code>%(cr_id)s</code>	58
4.12	Python C-extension function to resolve duplicate edges after lazy insertion of all edges from a non-symmetric BLAST sequence similarity network.	60
4.13	Calculation of the covariance between the adjacency lists of two sequence neighborhoods. Shown here in Python, this procedure is implemented as a C-extension of identical flow.	62

Bibliography

- [1] F. Abascal and A. Valencia. Clustering of proximal sequence space for the identification of protein families. *Bioinformatics*, 18(7):908–921, 2002. (Cited on p. 18.)
- [2] S. Altschul. Amino acid substitution matrices from an information theoretic perspective. *J Mol Biol*, 219:555–565, 1991. (Cited on p. 17.)
- [3] S. Altschul. A protein alignment scoring system sensitive at all evolutionary distances. *J Mol Evol*, 36(3):290–300, Mar 1993. (Cited on p. 17.)
- [4] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990. (Cited on p. 17, 36.)
- [5] S. Altschul, T. Madden, A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res*, 25(17):3389–3402, Sep 1997. (Cited on p. 17.)
- [6] S. Altschul, T. Madden, A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res*, 25(17):3389–3402, 1997. (Cited on p. 21, 30.)
- [7] S. Altschul, J. Wootton, E. Gertz, R. Agarwala, A. Morgulis, A. Schaffer, and Y. Yu. Protein database searches using compositionally adjusted substitution matrices. *FEBS J*, 272(20):5101–5109, Oct 2005. (Cited on p. 17.)
- [8] G. Apic, J. Gough, and S. Teichmann. Domain combinations in archaeal, eubacterial and eukaryotic proteomes. *J Mol Biol*, 310(2):311–325, Jul 2001. (Cited on p. 3.)
- [9] L. Aravind, V. Dixit, and E. Koonin. Apoptotic molecular machinery: vastly increased complexity in vertebrates revealed by genome comparisons. *Science*, 291(5507):1279–84, Feb 2001. (Cited on p. 1, 14.)
- [10] M. Ashburner, C. Ball, J. Blake, D. Botstein, H. Butler, et al. Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nat Genet*, 25:25–29, 2000. (Cited on p. 21.)
- [11] D. Babushok, E. Ostertag, and H. Kazazian. Current topics in genome evolution: molecular mechanisms of new gene formation. *Cell Mol Life Sci*, 64:542–554, Mar 2007. (Cited on p. 6, 14.)
- [12] A. Barabasi and Z. Oltvai. Network biology: understanding the cell’s functional organization. *Nat Rev Genet*, 5(2):101–113, Feb 2004. (Cited on p. 15, 67.)

- [13] P. Barrett, J. Hunter, J.T. Miller, J.C. Hsu, and P. Greenfield. matplotlib—a portable python plotting package. In *Astronomical Data Analysis Software and Systems XIV*, volume 347, page 91, 2005. (Cited on p. 143.)
- [14] M. Basu, L. Carmel, I. Rogozin, and E. Koonin. Evolution of protein domain promiscuity in eukaryotes. *Genome Res*, Jan 2008. (Cited on p. 15, 16.)
- [15] M. Basu, E. Poliakov, and I. Rogozin. Domain mobility in proteins: functional and evolutionary implications. *Brief Bioinform*, 10:205–216, Jan 2009. (Cited on p. 16.)
- [16] A. Bateman, L. Coin, R. Durbin, R. Finn, V. Hollich, S. Griffiths-Jones, A. Khanna, M. Marshall, S. Moxon, E. L. L. Sonnhammer, D. Studholme, C. Yeats, and S. Eddy. The Pfam protein families database. *Nucleic Acids Res*, 32(Database issue):D138–41, Jan 2004. (Cited on p. 83.)
- [17] I. Ben-Shlomo, S. Yu Hsu, R. Rauch, H. Kowalski, and A. Hsueh. Signaling receptome: a genomic and evolutionary perspective of plasma membrane receptors involved in signal transduction. *Sci STKE*, 2003(187):RE9, Jun 2003. (Cited on p. 1.)
- [18] A. Bhan, D.J. Galas, and T.G. Dewey. A duplication growth model of gene expression networks. *Bioinformatics*, 18(11):1486–1493, 2002. (Cited on p. 14.)
- [19] A. Bjorklund, D. Ekman, S. Light, J. Frey-Skott, and A. Elofsson. Domain rearrangements in protein evolution. *J Mol Biol*, 353(4):911–923, Nov 2005. (Cited on p. 20, 29.)
- [20] E. Bolten, A. Schliep, S. Schneckener, D. Schomburg, and R. Schrader. Clustering protein sequences—structure prediction by transitive homology. *Bioinformatics*, 17:935–941, 2001. (Cited on p. 18.)
- [21] E. Bornberg-Bauer, F. Beaussart, S. Kummerfeld, S. Teichmann, and J. Weiner. The evolution of domain arrangements in proteins and interaction networks. *Cell Mol Life Sci*, 62(4):435–445, Feb 2005. (Cited on p. 14, 15.)
- [22] B. Brejova, D. Brown, and T. Vinar. Optimal spaced seeds for homologous coding regions. In R. Baeza-Yates, E. Chávez, and M. Crochemore, editors, *Proceedings of Symposium on Combinatorial Pattern Matching (CPM’03)*, volume 2676 of *Lecture Notes in Computer Science*, pages 42–54, Morelia, Mexico, 2003. Springer. (Cited on p. 17.)
- [23] S. Brohée and J. van Helden. Evaluation of clustering algorithms for protein-protein interaction networks. *BMC Bioinformatics*, 7, Nov 2006. (Cited on p. 86.)
- [24] D. Brown and K. Sjolander. Functional classification using phylogenomic inference. *PLoS Comput Biol*, 2(6):479–483, Jun 2006. (Cited on p. 2.)
- [25] D.P. Brown, N. Krishnamurthy, and K. Sjölander. Automated protein subfamily identification and classification. *PLoS computational biology*, 3(8):e160, 2007. (Cited on p. 18.)
- [26] J. Buhler, U. Keich, and Y. Sun. Designing seeds for similarity search in genomic DNA. In Martin Vingron, Sorin Istrail, Pavel Pevzner, and Michael Waterman, editors, *RECOMB’03: Proceedings of the Seventh Annual International Conference on Research in Computational Molecular Biology*, pages 67–75. ACM Press, 2003. (Cited on p. 17.)
- [27] M. Buljan and A. Bateman. The evolution of protein domain families. *Biochem Soc Trans*, 37:751–755, Aug 2009. (Cited on p. 3, 14, 16.)
- [28] M. Buljan, A. Frankish, and A. Bateman. Quantifying the mechanisms of domain gain in animal proteins. *Genome Biol*, 11:R74, Jul 2010. (Cited on p. 16.)

- [29] K. P. Byrne and K. Wolfe. The Yeast Gene Order Browser: combining curated homology and syntenic context reveals gene fate in polyploid species. *Genome Res*, 15(10):1456–1461, Oct 2005. (Cited on p. 77.)
- [30] J. Celko. *Joe Celko's Trees and Hierarchies in SQL for Smarties, Second Edition*. Morgan Kaufmann, Feb 2012. (Cited on p. 55.)
- [31] F. Chen, A. Mackey, J. Vermunt, and D. Roos. Assessing performance of orthology detection strategies applied to eukaryotic genomes. *PLoS ONE*, 2(4):e383, 2007. (Cited on p. 22.)
- [32] C. Chothia and J. Gough. Genomic and structural aspects of protein evolution. *Biochem J*, 419:15–28, Apr 2009. (Cited on p. 14.)
- [33] A. Clauset, C. Shalizi R., and M. Newman. Power-law distributions in empirical data. *SIAM Rev*, 51:661–703, 2009. (Cited on p. 15.)
- [34] P.J.A. Cock, T. Antao, J.T. Chang, B.A. Chapman, C.J. Cox, A. Dalke, I. Friedberg, T. Hamelryck, F. Kauff, B. Wilczynski, et al. Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423, 2009. (Cited on p. 143.)
- [35] I. Cohen-Gihon, J. Fong, R. Sharan, R. Nussinov, T. Przytycka, and A. Panchenko. Evolution of domain promiscuity in eukaryotic genomes—a perspective from the inferred ancestral domain architectures. *Mol Biosyst*, 7:784–792, Mar 2011. (Cited on p. 16.)
- [36] I. Cohen-Gihon, R. Nussinov, and R. Sharan. Comprehensive analysis of co-occurring domain sets in yeast proteins. *BMC genomics*, 8(1):161, 2007. (Cited on p. 3.)
- [37] G. Conant and K. Wolfe. Increased glycolytic flux as an outcome of whole-genome duplication in yeast. *Mol Syst Biol*, 3:129, 2007. (Cited on p. 14.)
- [38] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms, 2nd Edition*. MIT Press/McGraw-Hill, 2001. (Cited on p. 51.)
- [39] J. Crabtree, S. Angiuoli, J. Wortman, and O. White. Sybil: methods and software for multiple genome comparison and visualization. *Methods Mol Biol*, 408:93–108, 2007. (Cited on p. 2.)
- [40] W. Day and H. Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of Classification*, 1(1):7–24, 1984. (Cited on p. 90, 91.)
- [41] A. de Mendoza, H. Suga, and I. Ruiz-Trillo. Evolution of the MaGuK protein gene family in premetazoan lineages. *BMC Evol Biol*, 10:93, Apr 2010. (Cited on p. 3.)
- [42] D. Defays. An efficient algorithm for a complete link method. *The Computer Journal*, 20(4):364–366, 1977. (Cited on p. 91.)
- [43] J. Demuth, T. Bie, J. Stajich, N. Cristianini, and M. Hahn. The evolution of mammalian gene families. *PLoS ONE*, 1:e85, 2006. (Cited on p. 2.)
- [44] C. Dessimoz, T. Gabaldón, D.S. Roos, E. Sonnhammer, J. Herrero, et al. Toward community standards in the quest for orthologs. *Bioinformatics*, 2012. (Cited on p. 22.)
- [45] A.K. Dunker, J.D. Lawson, C.J. Brown, R.M. Williams, P. Romero, J.S. Oh, C.J. Oldfield, A.M. Campen, C.M. Ratliff, K.W. Hipps, et al. Intrinsically disordered protein. *Journal of Molecular Graphics and Modelling*, 19(1):26–59, 2001. (Cited on p. 110.)
- [46] D. Ekman, A. Björklund, and A. Elofsson. Quantification of the elevated rate of domain rearrangements in metazoa. *J Mol Biol*, 372:1337–1348, Oct 2007. (Cited on p. 14.)

- [47] A. Enright, I. Iliopoulos, N. Kyrpides, and C. Ouzounis. Protein interaction maps for complete genomes based on gene fusion events. *Nature*, 402(6757):86–90, Nov 1999. (Cited on p. 14.)
- [48] A. Enright and C. Ouzounis. Functional associations of proteins in entire genomes by means of exhaustive detection of gene fusions. *Genome Biol*, 2(9):RESEARCH0034, 2001. (Cited on p. 14.)
- [49] A. Enright, S. Van Dongen, and C. Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Res.*, 30:1575–1584, 2002. (Cited on p. 18, 21.)
- [50] K. Evlampiev and H. Isambert. Modeling protein network evolution under genome duplication and domain shuffling. *BMC Syst Biol*, 1:49, Nov 2007. (Cited on p. 14, 15.)
- [51] W. Fitch. Distinguishing homologous from analogous proteins. *Syst Zool*, 19(2):99–113, Jun 1970. (Cited on p. 2, 5.)
- [52] W. Fitch. Homology: a personal view on some of the problems. *Trends Genet*, 16(5):227–231, May 2000. (Cited on p. 5, 6, 16.)
- [53] K. Florek, J. Lukaszewicz, J. Perkal, H. Steinhaus, and S. Zubrzycki. Sur la liaison et la division des points d’un ensemble fini. In *Colloquium Mathematicum*, volume 2, pages 282–285, 1951. (Cited on p. 90.)
- [54] J. Fong, L. Geer, A. Panchenko, and S. Bryant. Modeling the evolution of protein domain architectures using maximum parsimony. *J Mol Biol*, 366(1):307–315, Feb 2007. (Cited on p. 6, 14.)
- [55] A. Force, W.A. Cresko, F.B. Pickett, S.R. Proulx, C. Amemiya, and M. Lynch. The origin of subfunctions and modular gene regulation. *Genetics*, 170(1):433–446, 2005. (Cited on p. 14.)
- [56] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, 2010. (Cited on p. 18.)
- [57] Free Software Foundation. Gnu general public license, version 3, June 2007. (Cited on p. 143.)
- [58] D. Fulton, Y. Li, M. Laird, B. Horsman, F. Roche, and F. Brinkman. Improving the specificity of high-throughput ortholog prediction. *BMC bioinformatics*, 7(1):270, 2006. (Cited on p. 18.)
- [59] E.R. Gansner and S.C. North. An open graph visualization system and its applications. *Software - Practice and Experience*, 30:1203–1233, 1999. (Cited on p. 78.)
- [60] H. Gerstein and M. Gerstein. Annotation transfers for genomics: measuring functional divergence in multi-domain proteins. *Genome Res.*, 11:1632, 2001. (Cited on p. 2.)
- [61] M. Gerstein, C. Bruce, J. Rozowsky, D. Zheng, J. Du, J. Korbel, O. Emanuelsson, Z. Zhang, S. Weissman, and M. Snyder. What is a gene, post-ENCODE? History and updated definition. *Genome Res*, 17(6):669–681, Jun 2007. (Cited on p. 2.)
- [62] M.C. Good, J.G. Zalatan, and W.A. Lim. Scaffold proteins: Hubs for controlling the flow of cellular information. *Science*, 332(6030):680–686, 2011. (Cited on p. 1, 14.)
- [63] D. Graur and W. Li. *Fundamentals of Molecular Evolution*. Sinauer Associates Inc., Sunderland, MA., 1999. (Cited on p. 5.)
- [64] C. Greenman, P. Stephens, R. Smith, G. Dalgliesh, C. Hunter, et al. Patterns of somatic mutation in human cancer genomes. *Nature*, 446:153–158, Mar 2007. (Cited on p. 1.)
- [65] PostgreSQL Global Development Group. Postgresql, 2012. (Cited on p. 50, 143.)
- [66] M. Grunt, V. Zárský, and F. Cvrcková. Roots of angiosperm formins: the evolutionary history of plant FH2 domain-containing proteins. *BMC Evol Biol*, 8:115, Apr 2008. (Cited on p. 3.)

- [67] A.A. Hagberg, D.A. Schult, and P.J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15, Pasadena, CA USA, August 2008. (Cited on p. 143.)
- [68] J. Handl, J. Knowles, and D. Kell. Computational cluster validation in post-genomic data analysis. *Bioinformatics*, 21:3201–3212, Aug 2005. (Cited on p. 86.)
- [69] T. Harlow, J.P. Gogarten, and M. Ragan. A hybrid clustering approach to recognition of protein families in 114 microbial genomes. *BMC bioinformatics*, 5(1):45, 2004. (Cited on p. 18.)
- [70] A. Heger and L. Holm. Towards a covering set of protein family profiles. *Prog Biophys Mol Biol*, 73(5):321–337, 2000. (Cited on p. 19, 21, 26.)
- [71] H. Hegyi and M. Gerstein. Annotation transfer for genomics: measuring functional divergence in multi-domain proteins. *Genome Res*, 11:1632–1640, Oct 2001. (Cited on p. 2.)
- [72] S. Heinicke, M. Livstone, C. Lu, R. Oughtred, F. Kang, et al. The Princeton Protein Orthology Database (P-POD): a comparative genomics analysis tool for biologists. *PLoS ONE*, 2:e766, Aug 2007. (Cited on p. 2, 142.)
- [73] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933. (Cited on p. 129.)
- [74] A.L. Hughes and R. Friedman. Expression patterns of duplicate genes in the developing root in arabidopsis thaliana. *Journal of Molecular Evolution*, 60:247–256, 2005. (Cited on p. 5.)
- [75] M. Huynen and P. Bork. Measuring genome evolution. *PNAS*, 95(11):5849–5856, May 1998. (Cited on p. 20, 29.)
- [76] P. Jiang and M. Singh. Spici: a fast clustering algorithm for large biological networks. *Bioinformatics*, 26(8):1105–1111, 2010. (Cited on p. 18.)
- [77] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001. (Cited on p. 52, 143.)
- [78] G. Karev, Y. Wolf, F. Berezovskaya, and E. Koonin. Gene family evolution: an in-depth theoretical and simulation analysis of non-linear birth-death-innovation models. *BMC Evol Biol*, 4:32, Sep 2004. (Cited on p. 15.)
- [79] G. Karev, Y. Wolf, A. Rzhetsky, F. Berezovskaya, and E. Koonin. Birth and death of protein domains: a simple model of evolution explains power law behavior. *BMC Evol Biol*, 2:18–43, 2002. (Cited on p. 15.)
- [80] G. Karev, Y. Wolf, A. Rzhetsky, F. Berezovskaya, and E. Koonin. Birth and death of protein domains: a simple model of evolution explains power law behavior. *BMC Evol Biol*, 2(1):18, 2002. (Cited on p. 15.)
- [81] H. Kawaji, Y. Takenaka, and H. Matsuda. Graph-based clustering for finding distant relationships in a large set of protein sequences. *Bioinformatics*, 20(2):243–252, 2004. (Cited on p. 18.)
- [82] T. Kawashima, S. Kawashima, C. Tanaka, M. Murai, M. Yoneda, et al. Domain shuffling and the evolution of vertebrates. *Genome Res*, 19:1393–1403, Aug 2009. (Cited on p. 14.)
- [83] A. Kelil, S. Wang, R. Brzezinski, and A. Fleury. Cluss: Clustering of protein sequences based on a new similarity measure. *BMC bioinformatics*, 8(1):286, 2007. (Cited on p. 18, 141.)

- [84] J.P. Kern, M. Pattichis, and S.D. Stearns. Registration of image cubes using multivariate mutual information. In *Signals, Systems and Computers, 2003. Conference Record of the Thirty-Seventh Asilomar Conference on*, volume 2, pages 1645–1649. IEEE, 2003. (Cited on p. 119, 121, 191.)
- [85] S. Kim and J. Lee. Bag: A graph theoretic sequence clustering algorithm. *International Journal of Data Mining and Bioinformatics*, 1(2), 2006. (Cited on p. 18.)
- [86] E. Koonin, Y. Wolf, and G. Karev. The structure of protein universe and genome evolution. *Nature*, 420:218, 2002. (Cited on p. 15.)
- [87] E. Koonin, Y. Wolf, and G. Karev. *Power Laws, Scalefree Networks and Genome Biology*. Landes Bioscience, Georgetown, TX, 2005. (Cited on p. 15.)
- [88] A. Krause, J. Stoye, and M. Vingron. Large scale hierarchical clustering of protein sequences. *BMC Bioinformatics*, 6(1):15, Jan 2005. (Cited on p. 18.)
- [89] N. Krishnamurthy, D. Brown, and K. Sjölander. Flowerpower: clustering proteins into domain architecture classes for phylogenomic inference of protein function. *BMC evolutionary biology*, 7(Suppl 1):S12, 2007. (Cited on p. 18.)
- [90] A. Krishnan, M. Tomita, and A. Giuliani. Evolution of gene regulatory networks: Robustness as an emergent property of evolution. *Physica A: Statistical Mechanics and its Applications*, 387(8-9):2170–2186, 2008. (Cited on p. 14.)
- [91] S. Kummerfeld and S. Teichman. Relative rates of gene fusion and fission in mutli-domain proteins. *Trends in Genetics*, 21:25–30, 2005. (Cited on p. 14.)
- [92] M.C. Lagomarsino, A.L. Sellerio, P.D. Heijning, and B. Bassetti. Universal features in the genome-level evolution of protein domains. *Genome biology*, 10(1):R12, 2009. (Cited on p. 14, 15.)
- [93] J. Laherrere and D. Sornette. Stretched exponential distributions in nature and economy: “fat tails” with characteristic scales. *The European Physical Journal B-Condensed Matter and Complex Systems*, 2(4):525–539, 1998. (Cited on p. 15.)
- [94] L. Li, C. Stoeckert, and D. Roos. OrthoMCL: identification of ortholog groups for eukaryotic genomes. *Genome Res*, 13(9):2178–89, Sep 2003. (Cited on p. 18.)
- [95] J. Liu and B. Rost. Domains, motifs and clusters in the protein universe. *Curr. Opin. Chem. Biol.*, 7(1):5–11, Feb 2003. (Cited on p. 21.)
- [96] Y. Loewenstein, E. Portugaly, M. Fromer, and M. Linial. Efficient algorithms for accurate hierarchical clustering of huge datasets: tackling the entire protein space. *Bioinformatics*, 24:i41–i49, Jul 2008. (Cited on p. 91, 143.)
- [97] M. Long, E. Betran, K. Thornton, and W. Wang. The origin of new genes: glimpses from the young and old. *Nat Rev Genet*, 4(11):865–75, Nov 2003. (Cited on p. 6, 13.)
- [98] M. Lynch and J. Conery. The origins of genome complexity. *Science*, 302(5649):1401–1404, Nov 2003. (Cited on p. 14.)
- [99] M. Lynch and A. Force. The probability of duplicate gene preservation by subfunctionalizatio. *Genetics*, 154(1):459–473, 2000. (Cited on p. 14.)
- [100] Q. Ma, G.W. Chirn, R. Cai, J. Szustakowski, and NR Nirmala. Clustering protein sequences with a novel metric transformed from sequence similarity scores and sequence alignments with neural networks. *BMC bioinformatics*, 6(1):242, 2005. (Cited on p. 17, 18.)

- [101] E. Marcotte, M. Pellegrini, H. Ng, D. Rice, T. Yeates, and D. Eisenberg. Detecting protein function and protein-protein interactions from genome sequences. *Science*, 285(5428):751–753, Jul 1999. (Cited on p. 15.)
- [102] D. Medini, A. Covacci, and C. Donati. Protein homology network families reveal step-wise diversification of type III and type IV secretion systems. *PLoS Comput Biol*, 2(12):e173, Dec 2006. (Cited on p. 18.)
- [103] T. Meinel, A. Krause, H. Luz, M. Vingron, and E. Staub. The SYSTERS protein family database in 2005. *Nucleic Acids Research*, 33:D226–D229, 2005. (Cited on p. 142.)
- [104] H. Mi, Q. Dong, A. Muruganujan, P. Gaudet, S. Lewis, and P. Thomas. PANTHER version 7: improved phylogenetic trees, orthologs and collaboration with the Gene Ontology Consortium. *Nucleic Acids Res*, 38:D204–D210, Jan 2010. (Cited on p. 4, 23, 43.)
- [105] M. Middendorf, E. Ziv, C. Adams, J. Hom, R. Koytcheff, et al. Discriminative topological features reveal biological network mechanisms. *BMC Bioinformatics*, 5:181, Nov 2004. (Cited on p. 15.)
- [106] S. Mohseni-Zadeh, P. Brezellec, and J. Risler. Cluster-C, an algorithm for the large-scale clustering of protein sequences based on the extraction of maximal cliques. *Comput Biol Chem*, 28(3):211–218, Jul 2004. (Cited on p. 18.)
- [107] A. Moore, A. Björklund, D. Ekman, E. Bornberg-Bauer, and A. Elofsson. Arrangements in the modular evolution of proteins. *Trends Biochem Sci*, 33:444–451, Sep 2008. (Cited on p. 14.)
- [108] A. Murzin, S. Brenner, T. Hubbard, and C. Chothia. SCOP: a structural classification of proteins database for the investigation of sequences and structures. *J Mol Biol*, 247(4):536–40, Apr 1995. (Cited on p. 21.)
- [109] M. Nikolski and D.J. Sherman. Family relationships: should consensus reign? consensus clustering for protein families. *Bioinformatics*, 23(2):e71–e76, 2007. (Cited on p. 18.)
- [110] A. Novozhilov, G. Karev, and E. Koonin. Biological applications of the theory of birth-and-death processes. *Brief Bioinform*, 7(1):70–85, Mar 2006. (Cited on p. 14.)
- [111] S. Ohno. *Evolution by genome duplication*. Berlin: Springer Verlag, 1970. (Cited on p. 5, 13.)
- [112] T. E. Oliphant. Python for scientific computing. *Computing in Science & Engineering*, 9:10–20, May 2007. (Cited on p. 59, 143.)
- [113] A. Paccanaro, J. Casbon, and M. Saqi. Spectral clustering of protein sequences. *Nucleic Acids Res*, 34(5):1571–1580, 2006. (Cited on p. 18, 21.)
- [114] J. Park, S. Teichmann, T. Hubbard, and C. Chothia. Intermediate sequences increase the detection of homology between sequences. *J Mol Biol*, 273(1):349–354, Oct 1997. (Cited on p. 21.)
- [115] L. Patthy. Genome evolution and the evolution of exon-shuffling—a review. *Gene*, 238(1):103–114, Sep 1999. (Cited on p. 14.)
- [116] F. Pearl, C. Bennett, J. Bray, A. Harrison, N. Martin, A. Shepherd, I. Sillitoe, J. Thornton, and C. Orengo. The CATH database: an extended protein family resource for structural and functional genomics. *Nucleic Acids Res*, 31(1):452–455, Jan 2003. (Cited on p. 21.)
- [117] P. Pipenbacher, A. Schliep, S. Schneckener, A. Schonhuth, D. Schomburg, and R. Schrader. ProClust: improved clustering of protein sequences with an extended graph-based approach. *Bioinformatics*, 18 Suppl 2:182–191, 2002. (Cited on p. 18.)

- [118] A. Pires-daSilva, R.J. Sommer, et al. The evolution of signalling pathways in animal development. *Nature Reviews Genetics*, 4(1):39–49, 2003. (Cited on p. 1.)
- [119] S. Proost, M. Van Bel, L. Sterck, K. Billiau, T. Van Parys, et al. PLAZA: a comparative genomics resource to study gene and genome evolution in plants. *Plant Cell*, 21:3718–3731, Dec 2009. (Cited on p. 142.)
- [120] K.D. Pruitt, T. Tatusova, W. Klimke, and D.R. Maglott. Ncbi reference sequences: current status, policy and new initiatives. *Nucleic acids research*, 37(suppl 1):D32–D36, 2009. (Cited on p. 3.)
- [121] T. Przytycka, G. Davis, N. Song, and D. Durand. Graph theoretical insights into evolution of multidomain proteins. *Journal of Computational Biology*, 13(2):351–363, 2006. (Cited on p. 15.)
- [122] T. Przytycka and Y. Yu. Scale-free networks versus evolutionary drift. *Comput Biol Chem*, 28:257–264, Oct 2004. (Cited on p. 15.)
- [123] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011. ISBN 3-900051-07-0. (Cited on p. 143.)
- [124] S. Rahmann, T. Wittkop, J. Baumbach, M. Martin, A. Truss, and S. Böcker. Exact and heuristic algorithms for weighted cluster editing. *Comput Syst Bioinformatics Conf*, 6:391–401, 2007. (Cited on p. 18, 21.)
- [125] N. Rebscher, C. Deichmann, S. Sudhop, J.H. Fritzenwanker, S. Green, and M. Hassel. Conserved intron positions in fgfr genes reflect the modular structure of fgfr and reveal stepwise addition of domains to an already complex ancestral fgfr. *Development genes and evolution*, 219(9):455–468, 2009. (Cited on p. 3.)
- [126] P. Sarkar, D. Chakrabarti, and A.W. Moore. Theoretical justification of popular link prediction heuristics. In *International Conference on Learning Theory (COLT)*, pages 295–307, 2010. (Cited on p. 31.)
- [127] O. Sasson, A. Vaaknin, H. Fleischer, E. Portugaly, Y. Bilu, N. Linial, and M. Linial. ProtoNet: hierarchical classification of the protein space. *Nucleic Acids Res*, 31(1):348–352, Jan 2003. (Cited on p. 18.)
- [128] A. Schaffer, L. Aravind, T. Madden, S. Shavirin, J. Spouge, Y. Wolf, E. Koonin, and S. Altschul. Improving the accuracy of PSI-BLAST protein database searches with composition-based statistics and other refinements. *Nucleic Acids Res*, 29(14):2994–3005, Jul 2001. (Cited on p. 17.)
- [129] B.E. Shakhnovich, E. Deeds, C. Delisi, and E. Shakhnovich. Protein structure and evolutionary history determine sequence space topology. *Genome research*, 15(3):385–392, 2005. (Cited on p. 15.)
- [130] R. Sibson. SLINK: An optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16(1):30–34, 1973. (Cited on p. 91.)
- [131] B. Snel, G. Lehmann, P. Bork, and M. Huynen. STRING: a web-server to retrieve and display the repeatedly occurring neighbourhood of a gene. *Nucleic Acids Res*, 28(18):3442–4, 2000. (Cited on p. 18.)
- [132] R.R. Sokal and C.D. Michener. A statistical method for evaluating systematic relationships. *University of Kansas Science Bulletin*, 38:1409–1438, 1958. (Cited on p. 90.)
- [133] J. Song and M. Singh. How and when should interactome-derived clusters be used to predict functional modules and protein function? *Bioinformatics*, 25(23):3143–3150, 2009. (Cited on p. 86.)

- [134] N. Song, J. Joseph, G. Davis, and D. Durand. Sequence similarity network reveals common ancestry of multidomain proteins. *PLoS Comput Biol*, 4:e1000063, Apr 2008. (Cited on p. 4, 6, 8, 9, 17, 22, 23, 26, 29, 30, 36, 44, 127.)
- [135] N. Song, R. Sedgewick, and D. Durand. Domain architecture comparison for multidomain homology identification. *J Comput Biol*, 14(4):496–516, May 2007. (Cited on p. 20, 29, 30.)
- [136] Michael P. H. Stumpf and Mason A. Porter. Critical truths about power laws. *Science*, 335(6069):665–666, 2012. (Cited on p. 15.)
- [137] R. Tatusov, N. Fedorova, J. Jackson, A. Jacobs, B. Kiryutin, E. Koonin, D. Krylov, R. Mazumder, S. Mekhedov, A. Nikolskaya, B. Rao, S. Smirnov, A. Sverdlov, S. Vasudevan, Y. Wolf, J. Yin, and D. Natale. The COG database: an updated version includes eukaryotes. *BMC Bioinformatics*, 4(1):41, Sep 2003. (Cited on p. 2.)
- [138] J. Taylor and J. Raes. Duplication and divergence: the evolution of new genes and old ideas. *Annu Rev Genet*, 38:615–643, 2004. (Cited on p. 13, 14.)
- [139] S. Teichmann and M. Babu. Gene regulatory network growth by duplication. *Nat Genet*, 36(5):492–496, May 2004. (Cited on p. 14.)
- [140] A. Theodosiou, S. Arhondakis, M. Baumann, and S. Kossida. Evolutionary scenarios of notch proteins. *Molecular biology and evolution*, 26(7):1631–1640, 2009. (Cited on p. 3.)
- [141] P. Thomas, A. Kejariwal, M. Campbell, H. Mi, K. Diemer, et al. PANTHER: a browsable database of gene products organized by biological function, using curated protein family and subfamily classification. *Nucleic Acids Res*, 31(1):334–341, Jan 2003. (Cited on p. 153, 154, 195.)
- [142] H. Tordai, A. Nagy, K. Farkas, L. Banyai, and L. Patthy. Modules, multidomain proteins and organismic complexity. *FEBS J*, 272(19):5064–5078, Oct 2005. (Cited on p. 1, 14, 15.)
- [143] G. Van Rossum and F.L. Drake. *Python language reference manual*. Network Theory Limited, 2003. (Cited on p. 143.)
- [144] C. Vogel, C. Berzuini, M. Bashton, J. Gough, and S. Teichmann. Supra-domains: evolutionary units larger than single protein domains. *J Mol Biol*, 336(3):809–823, Feb 2004. (Cited on p. 3.)
- [145] C. Vogel, S. Teichmann, and J. Pereira-Leal. The relationship between domain duplication and recombination. *J Mol Biol*, 346(1):355–365, Feb 2005. (Cited on p. 14.)
- [146] R.M. Waterhouse, E.M. Zdobnov, F. Tegenfeldt, J. Li, and E.V. Kriventseva. Orthodb: the hierarchical catalog of eukaryotic orthologs in 2011. *Nucleic acids research*, 39(suppl 1):D283–D288, 2011. (Cited on p. 142.)
- [147] P.A. Watkins, D. Maiguel, Z. Jia, and J. Pevsner. Evidence for 26 distinct acyl-coenzyme a synthetase genes in the human genome. *Journal of lipid research*, 48(12):2736–2750, 2007. (Cited on p. 3.)
- [148] J. Weiner, A. Moore, and E. Bornberg-Bauer. Just how versatile are domains? *BMC Evol Biol*, 8:285, Oct 2008. (Cited on p. 15.)
- [149] J. Weston, A. Elisseff, D. Zhou, C. Leslie, and W. Noble. Protein ranking: from local to global structure in the protein similarity network. *PNAS*, 101:6559–6563, 2004. (Cited on p. 18.)
- [150] D. Wheeler, T. Barrett, D. Benson, S. Bryant, K. Canese, V. Chetvernin, D. Church, M. Dicuccio, R. Edgar, S. Federhen, M. Feolo, L. Geer, W. Helmberg, Y. Kapustin, O. Khovayko, D. Landsman, D. Lipman, T. Madden, D. Maglott, V. Miller, J. Ostell, K. Pruitt, G. Schuler, M. Shumway, E. Sequeira, S. Sherry, K. Sirotkin, A. Souvorov, G. Starchenko, R. Tatusov, T. Tatusova, L. Wagner, and E. Yaschenko. Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res*, 36:D13–D21, Jan 2008. (Cited on p. 2.)

- [151] T. Wittkop, J. Baumbach, F. Lobo, and S. Rahmann. Large scale clustering of protein sequences with FORCE -a layout based heuristic for weighted cluster editing. *BMC Bioinformatics*, 8:396, Oct 2007. (Cited on p. 18, 21.)
- [152] C. Wu, H. Huang, L. Yeh, and W. Barker. Protein family classification and functional annotation. *Comput Biol Chem*, 27(1):37–47, Feb 2003. (Cited on p. 2.)
- [153] S. Wuchty. Scale-free behavior in protein domain networks. *Mol. Biol. Evol*, 18:1694–1702, 2001. (Cited on p. 15.)
- [154] Y. Ye and A. Godzik. Comparative analysis of protein domain organization. *Genome Res*, 14(3):343–353, Mar 2004. (Cited on p. 3, 15, 117.)
- [155] Y.K. Yu, E.M. Gertz, R. Agarwala, A.A. Schäffer, and S.F. Altschul. Retrieval accuracy, statistical significance and compositional similarity in protein sequence database searches. *Nucleic Acids Research*, 34(20):5966–5973, 2006. (Cited on p. 17.)
- [156] J. Zhang. Evolution by gene duplication: an update. *Trends in Ecology and Evolution*, 18(6):292–298, June 2003. (Cited on p. 5.)
- [157] Z. Zhang, A. Schaffer, W. Miller, T. Madden, D. Lipman, E. Koonin, and S. Altschul. Protein sequence similarity searches using patterns as seeds. *Nucleic Acids Res*, 26(17):3986–3990, Sep 1998. (Cited on p. 17.)