Computational analysis of protein structures: data visualization, vector quantization, and spectral clustering

---

A Thesis
Presented to
The Division of Mathematics and Natural Sciences
Reed College

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Arts

Kevin J. Lynagh

May 2010

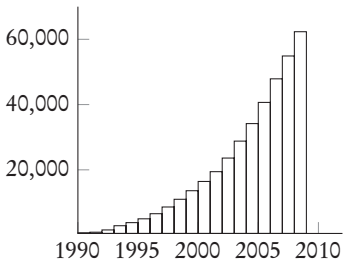
Approved for the Division
(Physics)

Darrell F. Schroeter

Preface

In this thesis I explore the topic of protein structure. The number of known protein structures is growing at a rapid clip (see right), and we must develop new approaches to characterize and glean biological insight from these structures. I develop computational methods for two structural bioinformatics problems. The first problem regards the domain structure (structural subunits) of a protein, which is interesting from both an evolutionary viewpoint (shared domains hint at a shared ancestor) and from a physical one (how do proteins fold?). The second problem I discuss in this thesis is a fast search of protein structures; given a new protein structure, which of the 60,000 currently known proteins does it most resemble? We cannot solve these two problems without the numerical assistance of computers, of course, but without computer-drawn visualizations we would not have any idea of what computations to carry out in the first place. Thus I also discuss principles for, and develop examples of, novel data visualizations using web technologies.

Protein structures deposited in the Protein Data Bank (PDB).



Over-the-table, this document is a thesis on structural biochemistry. Under-the-table, my agenda is much broader; I hope to highlight the role of computer technology in the early, hypothesis-generation stages of the scientific enterprise. The most captivating feature of mathematics is that one can follow any exposition at home with pen & paper. Likewise, the Internet allows for such participation in many scientific pursuits. All of the protein structures I examine are available to anyone online at the Protein Data Bank. All of the software I've used are also freely available online. Unfortunately, this is not the case for many of the scientific papers I've cited (I have reprinted their abstracts in the bibliography).

Computer technology has already led to many scientific advances over the past half century, but its full potential is yet unrealized. Computers can be wielded against far more difficult problems than linear regression, numerical integration, and instrument control; the Internet allows us to ask and answer previously unfathomable scientific questions. Like mathematics, computer technology provides us with new mechanisms for communication and explanation; two things without which scientific discovery would halt. I hope that this thesis in small part reflects this promise.

May 2010
Portland, Oregon

Acknowledgements

For making my education possible: my parents, Minhthu & Bill
For a year of consistently good advice: my advisor, Darrell Schroeter
For reviewing the manuscript: Babs Guerra-Torres, Erin Weber, Claire Payton, and Eric Franzosa
For kindly tolerating a half-time employee quartering in the late spring: Noah Pepper

Contents

Abstract

The biological function of a protein is determined by its structure, and extensive scientific and medical interest has led to an ever-growing corpus of solved protein structures. In this thesis I discuss computational methods for characterizing protein structure. I utilize these methods in two procedures that may help shed light on the functions of structured proteins. Specifically, I develop an algorithm to identify a protein's domains via spectral clustering C_α positions. I extend this procedure by using a neural network to incorporate residue volumes and backbone distances. The accuracy of the resulting partitioning is par with the three algorithms used by the CATH database. The second procedure I develop is a fast structural search algorithm. Protein structures are encoded as strings using backbone fragment prototypes derived from *k*-means clustering along structural features. This reduces the protein fold-level search problem to spell-check. This search procedure is competitive with full 3D structural alignment algorithms (as measured by ROC areas) and runs three orders of magnitude faster. Finally, I survey the structural characteristics of amino acid residues, in particular Voronoi-tessellation derived measures of residue and atom volume. I discuss computational methods of processing and visualization techniques for exploring copious quantitative data using web technologies.

To Rex E. Adelberger,

who taught me how to think like a physicist

Perhaps the most remarkable features of the molecule are its complexity and its lack of symmetry. The arrangement seems to be almost totally lacking in the kind of regularities which one instinctively anticipates, and it is more complicated than has been predicted by any theory of protein structure.

—Sir John Kendrew *et al.* on the structure of myoglobin in 1958

Structural characterization

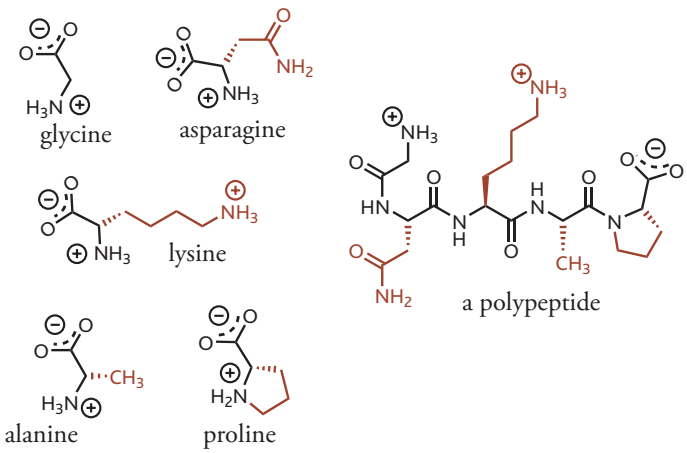
1

I tackle two problems of structural bioinformatics in this thesis: the automated detection of a protein's domains (subunits), and the fast search of the growing corpus of protein structures. Before we can approach these problems, we must first discuss protein structures themselves. Contrary to what Sir Kendrew *et al.*'s famous quotation [37] on the discovery of myoglobin's structure might lead you to believe, there do exist some commonalities among the variety of observed protein structures.

In this chapter, I discuss the most famous regularities of protein structure, the α-helix and β-pleated sheet motifs, and their common origin in the physical/energetic constraints of folded proteins. I then discuss the contact map representation of a protein's structure, which simplifies the 3D protein structure to a form more amendable to mathematical and computational analyses. Finally, I demonstrate a method for scoring the connectedness of the different components of a protein (its constituent amino acid residues) and give a novel correlation between that score and residue depth.

Levels of structure

Proteins are linear polymers of amino acid residues. There are 20 different amino acids, and each has different chemical properties: positive, negative, or neutral charge; a specific affinity for water; the ability to act as an acid and/or base, &c. These amino acids are covalently connected to each other to form the protein backbone (in the process they lose a water and are thereafter referred to as "residues"). For this thesis, this is effectively all of the protein chemistry you need to know.

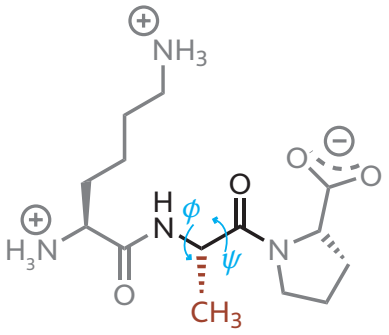


A polypeptide consisting of 5 amino acid residues (sidechains depicted in red). Proteins are just much longer polypeptides, typically 100–300 residues long. From the amino-terminus ($-NH_3^+$) to the carboxy-terminus ($-COO^-$): glycine, asparagine, lysine, alanine, proline. Life is vast, and some species incorporate special amino acids into their proteins–some enzymes contain selenocysteine and some Archaea proteins contain pyrrolysine. The vast majority of proteins, however, draw from the standard 20 amino acid residues (see appendix B for a listing).

The sequence of residue types along the backbone is known as a protein's *primary structure*. This ordering is (essentially) fixed by the genetic information stored in the DNA that codes for the protein. The sequence determines the final fold and function of a protein. The vast majority of observed proteins fold consistently; each amino

acid residue sequence condenses to the same structure each time it folds. The positioning of the residue sidechains on this scaffold then determines the protein function. Note that not every possible amino acid sequence consistently folds to a well-defined structure (the vast majority of sequences correspond to floppy chains). Rather, over the course of natural selection (nearly all observed proteins are drawn from living organisms), those genes that encoded proteins with degenerate minima are not propagated.*

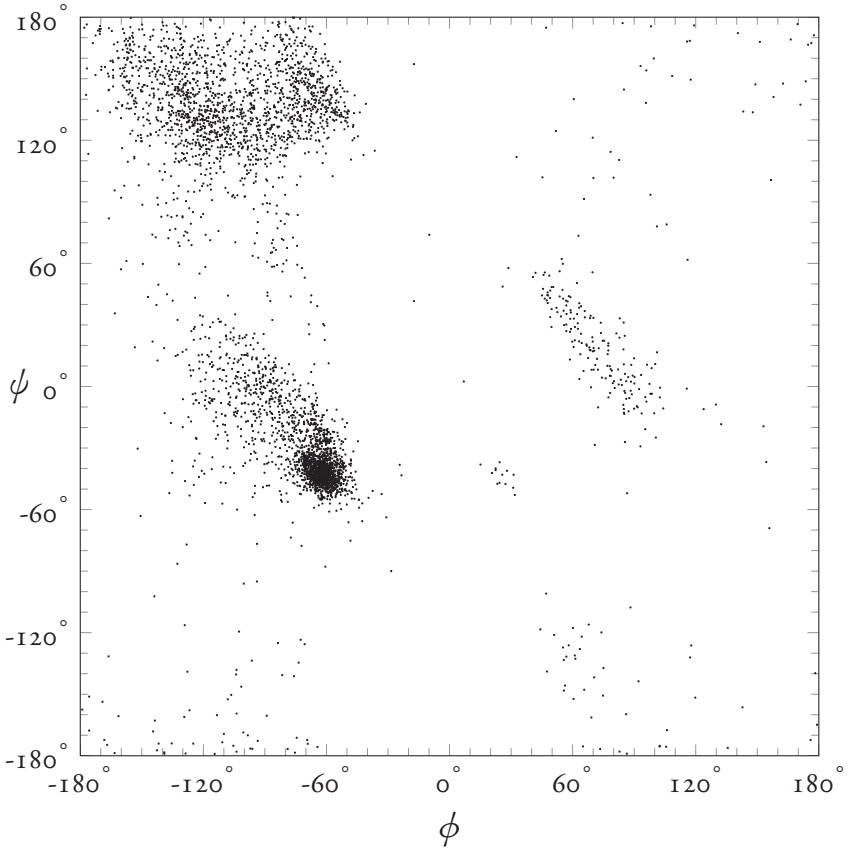
The *secondary structure* of a protein is the local conformation of the backbone chain. These are constrained by two different forces: the electronic nature of the backbone itself and the steric interactions of residue sidechains with parts of the backbone. The lone pair of electrons on the nitrogen of the peptide bond resonates onto the oxygen. This gives the bond a partial π-bond character; rotation is very slow at physiological temperatures and we can ignore it in our studies. The oxygen, nitrogen, and carbon atoms of a given residue then define a plane, leaving only the bonds connected to the C_α atom free to rotate. A more comprehensive introduction to protein chemistry can be found in Lehninger's [44] or Clayden's [13] texts. The dihedral angles of these bonds ϕ (C_α—N) & ψ (C_α—C),



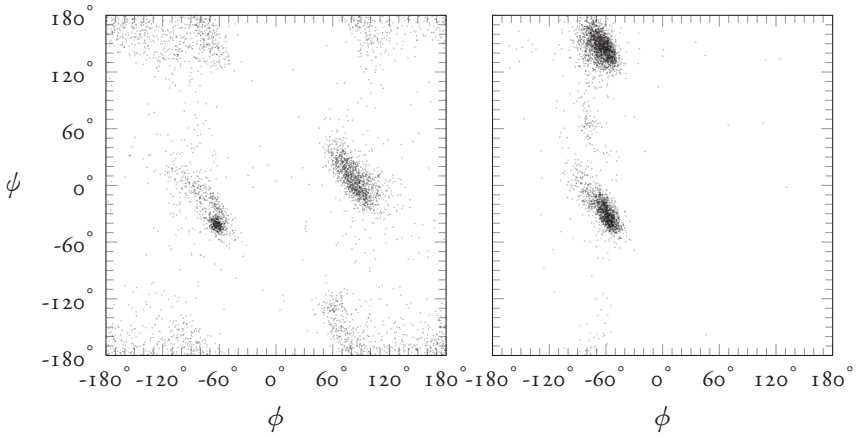
then define the backbone conformation of each residue. However, the ϕ and ψ angles are not independent of each other, as certain configurations would lead the sidechains to collide (sterically clash).

The plot of ϕ vs. ψ (known as a Ramachandran plot, on facing page) shows that the distribution of these angles fall into certain allowed regions. These regions correspond to residues within instances of the famous α-helix and β-pleated sheet structural motifs.

* Misfolded proteins can be more deleterious to the fitness of an organism than just in terms of wasted resources–they can sometimes have an outright toxic effect. The prion diseases–Creutzfeldt-Jakob in humans, scrapie in sheep, bovine spongiform encephalopathy (mad cow)–are believed to be caused by misfolded proteins that catalyze the misfolding of other proteins. However, there is no intrinsic biological reason that proteins must be ordered at all–evolution simply requires that something *work*. Dyson [17] reviews intrinsically disordered proteins, some of which become ordered only on binding some other protein, and some remain disordered and simply serve as linking chains in larger complexes.



Ramachandran plot of all residue types, sampled from the proteins listed in appendix H (4000 residues shown). Note that residues mostly fall into two regions on this plot–the cluster at the top left ($\phi \approx$ -120°, $\psi \approx$ 120°) corresponds to the extended backbones of loops and β-strands, and the cluster at mid-left ($\phi \approx$ -60°, $\psi \approx$ -30° through -50°) to helices.

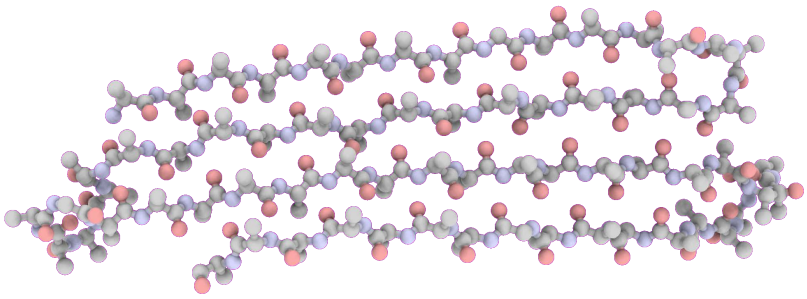


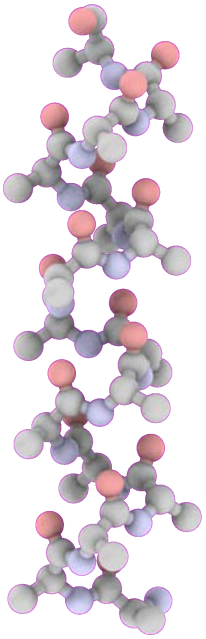
Ramachandran plots of two notable amino acid residues (4000 residues shown on each plot). Glycine (left plot) has no sidechain, which allows it to fall on a much wider range of (ϕ, ψ) backbone angle pairs than other residues. The sidechain of proline loops back and bonds with the nitrogen (technically making it an imino acid), which greatly restricts its ϕ angle compared to the other residues.

The two predominant backbone motifs are the α-helix and the β-pleated sheet, which commonly occur because they allow the oxygen and nitrogen atoms of the backbone to participate in internal hydrogen bonding interactions. Indeed, both were hypothesized to exist on these grounds by Linus Pauling and Robert Corey in 1951 [59, 60].

In the α-helix, the carbonyl oxygen of the n^{th} residue accepts a hydrogen bond from the nitrogen of the $(n+4)^{\text{th}}$ residue, about 2.8 Å away, and all of the sidechains point outward. On the Ramachandran plot, residues within an α-helix fall around $\phi \approx$ -60°, $\psi \approx$ -50°.

In contrast to the intra-segment bonding of the α-helix, hydrogen bonding occurs between the backbones of adjacent β-strands in the β-pleated sheet motifs. The most common sheets consist of anti-parallel backbone segments connected by hairpin turns, but a variety of topological configurations exist. These sheets can range from two to more than a dozen strands. The dihedral angles of residues along the sheet are the same as those of an unfolded chain: $\phi = \psi \approx \pm$ -60 ° through -180 °.





Ball-and-stick renderings of an α-helix (above) and β-sheet (left). Carbon atoms are gray, oxygen atoms red, and nitrogen atoms blue. For clarity, the sidechains have all been truncated to a single carbon atom (i.e. all residues shown are alanine residues), though of course in reality they vary.

Note that both of these secondary structures form to satisfy the hydrogen bonding requirements of the backbone, taking the place of the hydrogen bonds to water that exist when the protein is unfolded. The unique chemical properties of each sidechain dictates the arrangement of secondary structure elements with respect to each other: the *tertiary structure* of a protein–its *fold*. For instance, if a helix consists of alternating hydrophobic (greasy) and hydrophilic (charged) residues, the helix would likely be arranged with the greasy side up against more grease (another helix or a sheet) and the charged sidechains exposed to water on the surface of the protein. While the α-helix and β-sheet are essentially the only two elements of secondary structure, we have observed a huge variety of folds from their combinations.*

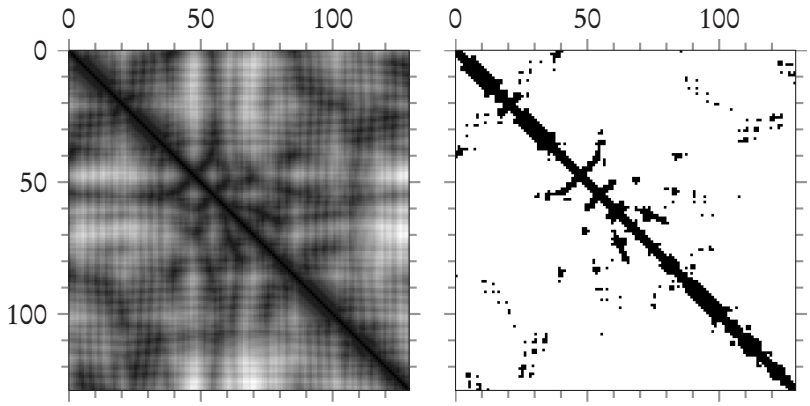
The final, broadest level of protein structure is the *quaternary structure*, which describes those proteins that consist of multiple chains. These subunits can take the form of independently functioning proteins that occasionally stick together and work as a single entity, or they may be disordered, unfolded chains that only settle into a well-defined structure in the presence of their partners. I do not discuss such protein complexes in this thesis.

Contact maps

A protein consisting of n amino acid residues can be described by an n by n symmetric *distance matrix* $\mathbf{D}$, with elements D_{ij} given by the Euclidean distance between residues i and j. The *contact map*, $\mathbf{C}$, is an even simpler matrix, with $C_{ij} = 1$ when residues i and j are "in contact" and zero otherwise. The contact map is a very coarse representation of a protein, giving $\frac{1}{2}n(n-1)$ binary values that describe the residue-residue contacts rather than

* See CATH [58] and SCOP [54], for two popular methodologies of defining protein-fold hierarchies. As of spring 2010, both of these databases list about 1200 different folds each, which is consistent with statistical and computational predictions [87, 93] of the total number of possible folds. The growing consensus is that, at least for small single domain proteins, we have likely observed all possible protein folds. Qian *et al.* [64] note that the number of proteins of a given fold family follows a power law distribution: popular folds occur far more frequently than the (many more) unpopular ones.

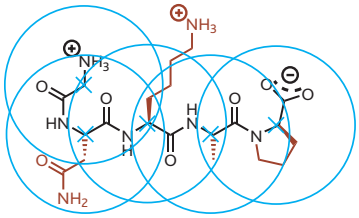
n(3 dimensions × 8ish atoms per residue) real number coördinates required to describe the positions of a protein's every atom.



Distance matrix (left) and 8 Å contact map of the 129-residue hen egg-white lysozyme protein (PDB ID: 2VB1) using radial C_α contact measures.

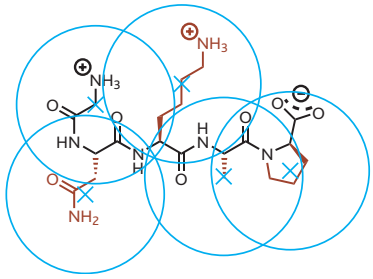
The hope is that the contact map contains the essential structural details of a protein's native conformation, and we can ignore the complexity of a full 3D atomic model. The contact map representation allows us to leverage mathematical results to solve various biological problems. In particular, we can interpret the contact map as a graph's adjacency matrix and thus examine protein structure using graph theoretic techniques.

Before we can use the contact map representation, we need to settle on what we mean by "in contact". The simplest way to define residue-residue contact is to use a radial cutoff; two residues are in contact if they are within some specified distance of one another. However, residues consist of multiple atoms, and to measure distances we need to pick a single atom or point to represent each residue. The C_α position is the most common choice (sidebar, first from top), which amounts to looking at the contact map of the backbone alone (ignoring the sidechains). This might not be as absurd as it first sounds, especially from a global structure perspective–some protein families have pairs of members with as little as 20% sequence similarity, and the space of possible sequences maps onto a much smaller space of possible structures (see Koonin *et al.* [41] for a discussion). The C_α contact map may implicitly contain all the sidechain information we need via the backbone conformation.



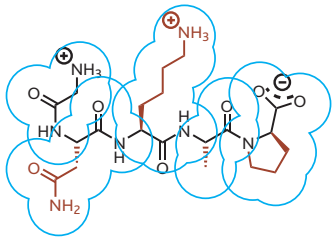
Radial cutoffs based on each residue's α-carbon.

If we want a contact map that explicitly takes the sidechains into account, we can use the first sidechain atom, C_β (sticking with C_α for glycine), or the geometric centre (sidebar, second) of the sidechain itself. Yet another common definition is to simply consider two residues to be neighbors if *any* of their heavy (non-hydrogen) atoms are within some cutoff distance (sidebar, third).



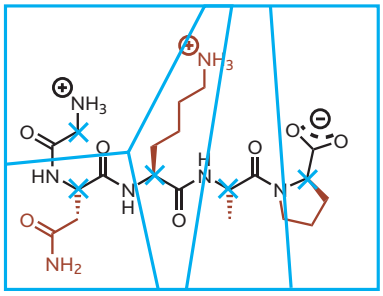
Radial cutoffs based on each residue's sidechain geometric centre (centroid).

All of these radial measures share one drawback: they do not give us a natural cutoff distance. We can appeal to the *van der Waals radii* or on other physical grounds, but it would certainly be elegant to have a parameter-free measure of contact.



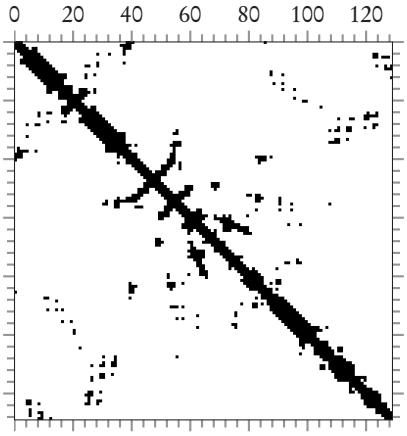
Radial cutoffs around any heavy atom.

One popular parameter-free contact measure is given by the Voronoi tessellation (at right), which is a mathematical process for dividing up a space into regions. Given a collection of points, the Voronoi tessellation assigns each one a unique convex cell consisting of all space closer to that point than to any other. A Voronoi cell gives us several data: volume (area, in 2D), the number of edges (and faces, in 3D), &c., and also an unambiguous metric of proximity: two points in the space are neighbors if their Voronoi cells share a face (edge, in 2D). One runs into similar conceptual problems as the radial methods when tessellating around a single point for each residue. Large and small residues are treated equally by the procedure, but again, there are similar solutions: we can include each heavy atom in the tessellation, and define two residues to be in contact if any of their atom cells share a face.



Voronoi diagram of a small polypeptide. Note that lysine is the only residue in this example with a bounded cell–the rest stretch out to infinity. Furthermore, if we define "in contact" as any cells sharing an edge, glycine and proline are actually neighbors, even though they are the two residues furthest from one another. The mathematics and algorithms of the Voronoi tessellation are discussed in appendix A and its application to protein structures in appendix B.

Regardless of the specific definition, contact maps tend to share some general patterns from which one can infer aspects of a protein's secondary structure. The α-helix motif, for instance, is closer to a compressed spring than a free string, and appears as a section of thick diagonal (see residues 30–40 in the protein with PDB ID: 2VB1 at right) on the contact map. Likewise, anti-parallel β-sheets appear as offshoots perpendicular to the diagonal (think of making a $\bigcup$ out of a string–the bits at the end will be close to the bits at the beginning, also see residues 50–60 at right).



8 Å C_α contact map of 129 residue protein with PDB ID: 2VB1. Note the strong diagonal, which indicates that residues close on the backbone chain are, of course, also close in space. Because of this, contact maps are sometimes referred to as "diagonal plots".

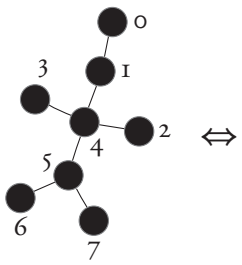
Distance matrices and contact maps are not just descriptive tools, but also serve as a milestone for protein structure prediction. Rather than directly tackle the sequence → structure problem, some researchers attempt to predict the contact map, from which the 3D structure can be recovered: sequence → contact map → structure. I review current work on this problem in appendix F.

This thesis is concerned with the fold-level, backbone structure of different proteins. Thus, we only employ the C_α contact map and distance matrices. Though we do not use the Voronoi tessellation to define residue-residue contact, the procedure does yield a natural measure of residue volume, which appears throughout this thesis. For a discussion of how consistently the contact map and Voronoi volume describes a protein structure, see appendix D.

Contact map eigenvektor

Given a contact map, it would be nice to have some sort of metric or score that gives the structural importance of each residue. Such a score might prove useful for a variety of studies. It might suggest residues to mutate for thermodynamic studies of protein folding. If a protein catalyzes a chemical reaction, this structural score might hint at the identities of the few active site residues.

To construct such a score, think of a contact map as the adjacency matrix of a graph, with each row/column corresponding to a residue (a node on the graph) and nonzero elements indicate neighbors (graph edges),

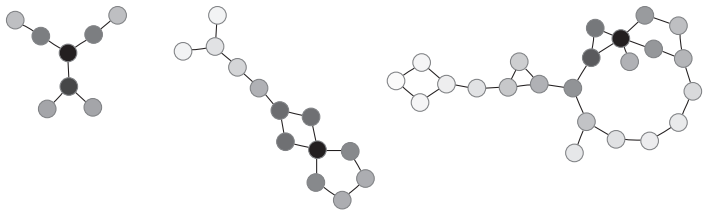


$$\Leftrightarrow \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}.$$

We can then use the principal eigenvektor of this adjacency matrix as a centrality measure.*

* Google's PageRank algorithm uses the principal eigenvektor of the Internet's graph–nodes are websites and edges the links between them–for scoring web pages in their search engine.

First, some examples: the nodes of the three simple graphs below have their vertexes colored by their principal eigenvektor components (darker is greater). Pleasingly, the central nodes score highly, and note that it is not just the degree of a node (the number of its edges) that determines the score, but something more subtle. The node of degree three at the top end of the centre graph, for instance, is much lighter than the node of degree three in the middle.



There are two quick technical points to note about the principal eigenvektor. First, note that a graph is uniquely defined by its adjacency matrix and vice versa, up to row/column permutations. The graph above at left could have an adjacency matrix  just as well as  (exchanging nodes 1 and 6), but these permutations have no effect on the spectrum of the matrix (proteins do, of course, have a natural enumeration for the nodes–the backbone order). Second, note that, because contact maps are non-negative and irreducible (there is a path between any pair of residues through their neighbors), the Perron-Frobenius theorem* guarantees that the components of the principal eigenvektor are all non-negative, so we can avoid trying to interpret "negative importance".

* The Perron-Frobenius theorem states that for a non-negative, irreducible matrix **A**;

- There is a real eigenvalue λ_0 that satisfies $|\lambda_0| \geq |\lambda|$ for any other eigenvalue λ.
- There is only one eigenvektor consisting entirely of non-negative entries, and it is the one associated with λ_0.

Another way to interpret the principal eigenvektor is as the limit of an iterative process. Multiplying a vector of ones by an adjacency matrix gives a vector whose components are the number of neighbors (the degree) of each node on the graph. If we multiply again, the components correspond to the number of a node's neighbors' neighbors. Repeating this process (taking care at each step to normalize the vector to have unit length) eventually yields

the principal eigenvektor. This is the *power method* for computing the principal eigenvektor.

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 1 \\ 1 \\ 4 \\ 3 \\ 1 \\ 1 \end{pmatrix} \xrightarrow{\text{normalize}} \frac{1}{\sqrt{34}} \begin{pmatrix} 1 \\ 2 \\ 1 \\ 1 \\ 4 \\ 3 \\ 1 \\ 1 \end{pmatrix}$$

$$\rightarrow \frac{1}{\sqrt{164}} \begin{pmatrix} 2 \\ 5 \\ 4 \\ 4 \\ 7 \\ 6 \\ 3 \\ 3 \end{pmatrix} \rightarrow \begin{pmatrix} 0.15 \\ 0.46 \\ 0.31 \\ 0.15 \\ 0.62 \\ 0.46 \\ 0.15 \\ 0.15 \end{pmatrix} \rightarrow \begin{pmatrix} 0.19 \\ 0.44 \\ 0.44 \\ 0.25 \\ 0.56 \\ 0.37 \\ 0.19 \\ 0.19 \end{pmatrix} \rightarrow \begin{pmatrix} 0.17 \\ 0.47 \\ 0.40 \\ 0.22 \\ 0.60 \\ 0.37 \\ 0.15 \\ 0.15 \end{pmatrix} \rightarrow \begin{pmatrix} 0.19 \\ 0.46 \\ 0.43 \\ 0.24 \\ 0.58 \\ 0.36 \\ 0.15 \\ 0.15 \end{pmatrix}$$

The principal eigenvektor component corresponding to a given residue is, in a sense, a protein-wide measure of that residue's connectedness to the others. This manifests itself in several ways. I discuss later in this chapter a correlation between a residues' principal eigenvektor component and its depth in the protein. Also, in chapter 2 I use the contact map eigenvektors to determine protein domain structure (subunits). Interestingly, in some cases the principal eigenvektor alone contains enough information to reconstruct the entire contact map [62].

Protein surfaces

Proteins do not exist in isolation; they function in the aqueous environment of a cell, where they interact with water, signal molecules, and other proteins.

The internal twisting of the backbone is interesting from the perspective of the folding problem, but it may not be pertinent to other biological questions. Most proteins do not completely unfold and refold as part of their function; arguably we can glean biological insight by just examining the protein from the outside. Are there pockets that fit certain molecules, charged sidechains for specific electrostatic interactions, or grooves to hold onto DNA?

For macroscopic objects, it is simple to just look at or touch something and establish a surface, but on the scale of proteins–typically a few nanometres (10^{-9} m) across–we have to come up with operational definitions. The most accurate definition (short of full quantum mechanical simulation) is the Lennard-Jones potential,

which describes the interaction energy between neutral molecules:

$$V(r) = \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^{6} \right],$$

where r is the separation between two atoms and σ the distance where the atoms do not interact. The r^{-12} term describes the repulsion between the atom's electrons due to Pauli exclusion (that spin $1/2$ particles (fermions) cannot occupy the same quantum state) and r^{-6} the London dispersion (induced dipole–dipole) force.

We can approximate this potential by treating neutral atoms as hard (non-intersecting) spheres;

$$V(r) = \begin{cases} 0 & \text{if } r > r_{\text{vdW}} \\ \infty & \text{if } r \leq r_{\text{vdW}}, \end{cases}$$

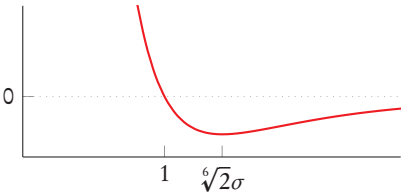
where r_{vdW} is the van der Waals radius of the atom. The most common values are those of Bondi [8]: hydrogen 1.20 Å, carbon 1.70 Å, nitrogen 1.55 Å, oxygen 1.52 Å.

One can derive the *van der Waals surface* of a molecule by taking union of the these hard spheres, but in a sense this is unphysical because it can lead to unreachable surfaces. Imagine trying to color a molecule with a sphere of paint: since the paint sphere is the same size as the spheres defining the surface, there will be places you can never paint! This surface is what one sees in space-filling molecular models, which are also known as CPK (Corey, Pauling, and Koltun) models.

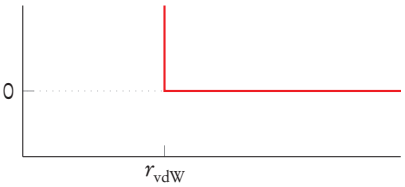
A more physically motivated definition of a molecular surface is the solvent-accessible area, which was first formulated by Lee and Richards [43]. This area is calculated as the sheet defined by rolling a ball with radius ≈ 1.4 Å (representing water) around the van der Waals surface of a molecule. Based on the sphere-to-sphere contact, one can use this measure to assign a numerical solvent-accessible area for each surface residue of a protein. With a full enclosing surface, one can also calculate the total volume of a protein.

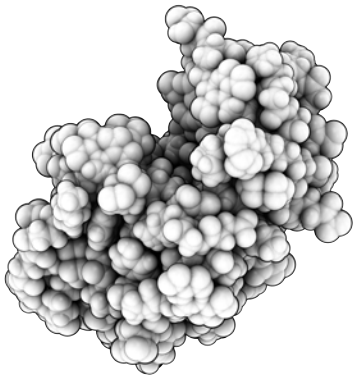
I use the program developed by Sanner *et al.* [68] to compute a triangulation of the solvent-accessible area of proteins. The set of vertexes making up the triangulation are effectively points on the surface of the protein, and I define the depth of each residue as the distance between the C_α and the closest surface vertex.

The Lennard-Jones potential, modeling the interaction of neutral molecules:

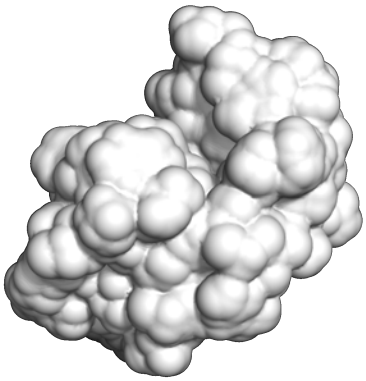


Hard sphere potential energy:





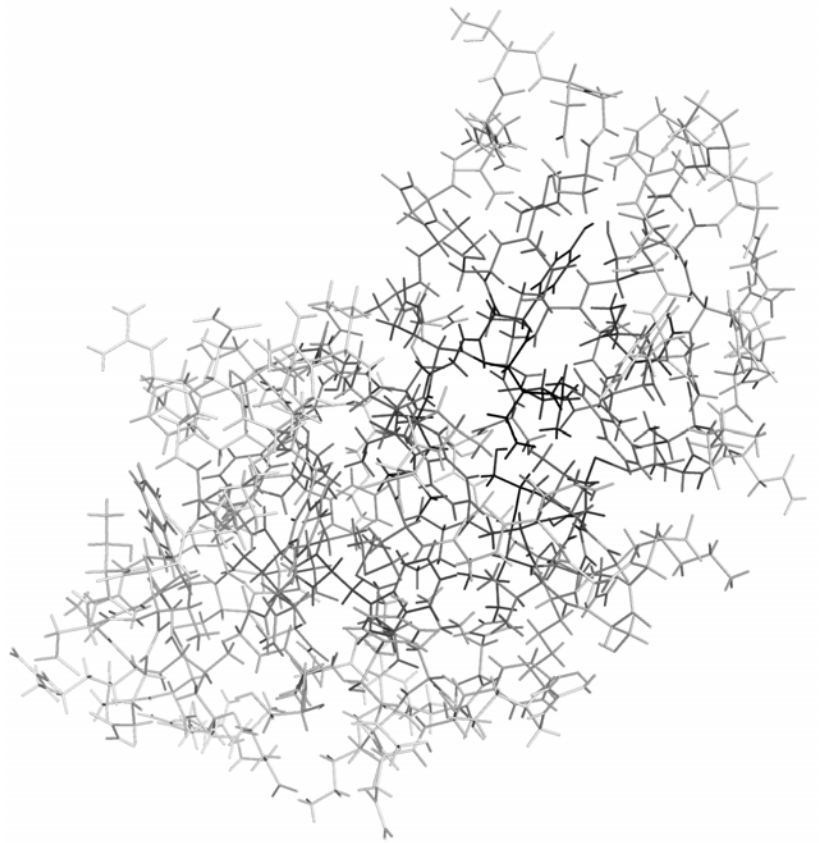
The van der Waals surface of the protein with PDB ID: 2VB1.

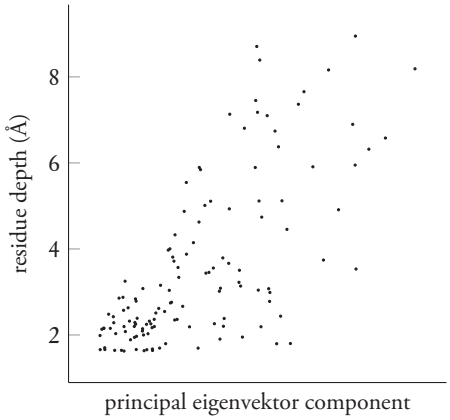


The solvent accessible surface of the protein with PDB ID: 2VB1.

Residue depth

An interesting and novel result from this thesis is that the depth of a residue correlates strongly with the principal eigenvektor component:





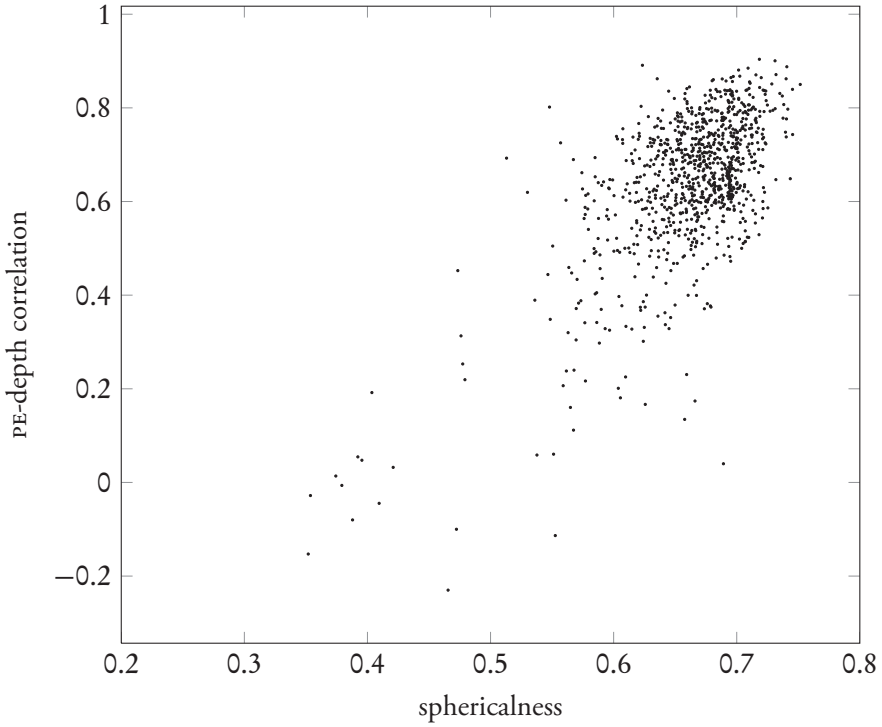
A line drawing of the protein with PDB ID: 2VB1 (left), colored by the residue principal eigenvektor component (darker is higher), and a plot of residue depth vs that component (below). The principal eigenvektor is normalized and the individual components are comparable only to each other (hence no units). The Pearson correlation coefficient (see page 102) is 0.713, and the sphericalness (see text) is 0.7.

The cores of proteins are known to be well-packed (see appendix 69), and so we expect more deeply buried residues to have more neighbors at a given cutoff distance than the residues on or near the surface. Furthermore, we should also expect this correlation to be stronger for proteins that are more spherical; the most central residues of an oblong protein might not always be deeper than those on the periphery.* I define the "sphericalness" of a protein as $r_{\text{vol}}/r_{\text{surf}}$ ratio, with the radii taken from the volume, V, and surface area, A, given by the MSMS program [68];

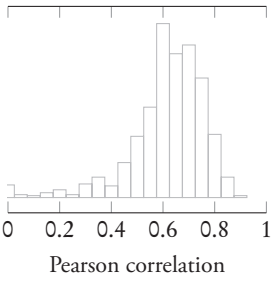
$$V = \tfrac{4}{3}\pi r_{\text{vol}}^3 \quad \rightarrow \quad r_{\text{vol}} = \sqrt[3]{\tfrac{3}{4\pi}V}$$
$$A = 4\pi r_{\text{area}} \quad \rightarrow \quad r_{\text{area}} = \sqrt{\tfrac{1}{4\pi}A},$$

and this correlation turns out to occur;

* Consider a cylindrical protein; the residue at the centre is at the same depth as all the resides that lie along the axis of the cylinder (namely, the radius of the cylinder).



For 988 proteins, I calculated the 8 Å radial sidechain centre contact map. Depths were calculated as the minimum distance from the residue sidechain centre (or α-carbon, for unresolved sidechains and glycine residues) to the closest vertex of the analytical surface area given by the MSMS program. The correlation coefficient of each protein is taken from the depth and PE components. The "sphericalness" is defined as the $r_{\text{vol}}/r_{\text{surf}}$ ratio, with the protein radii taken from the volume and surface area given by MSMS.



fin.

In this chapter we have briefly discussed protein structure and the standard methods of characterizing it. We saw how the protein backbone forms into α-helicies and β-sheets, and how those structural motifs can be seen on a protein's contact map. We discussed the principal eigenvektor of the contact map, which serves as a natural score for the structural importance and connectivity of a protein's constituent residues. In the next two chapters we will see how these measures can be used in two standing problems in structural biochemistry: the classification of a protein's residues into subunits (protein domains) and the construction of a fast protein fold-level search algorithm.

Notes

The study of protein structure is only a small part of biochemistry, a field having hundreds of fascinating sub-disciplines. Any biochemistry text will devote a few chapters to protein structure, but in particular I recommend Lehninger's Biochemistry;

> Albert L. Lehninger, David L. Nelson, and Michael M. Cox. *Lehninger Principles of Biochemistry*. W. H. Freeman. 2008. call no: QP514.2.L43

and for understanding the chemistry in more detail, Clayden*et al.*'s *Organic Chemistry* is an illuminating start;

> Jonathan Clayden et al. *Organic Chemistry*. Oxford University Press. 2000

Jane S. Richardson's *Anatomy & Taxonomy of Protein Structure*[*] gives a technical, yet broad overview of protein structure, and includes many awesome hand-drawn illustrations. David Goodsell,[*] however, is the rightly undisputed king of biomolecular illustrations– his two books,

> David Goodsell. *The machinery of life*. Springer-Verlag. 1993. call no: QH581.2.G66

> David Goodsell. *Bionanotechnology : lessons from nature*. Wiley-Liss. 2004. call no: QP514.2.G658

depict the workings of cells from what is visible under a microscope to the atomic level without sacrificing either scientific accuracy or lucid prose.

[*] `http://pibs.duke.edu/teaching/anatax/index.html`

[*] `http://mgl.scripps.edu/people/goodsell/illustration`

I shall not today attempt further to define the kinds of material I understand to be embraced within that shorthand description, and perhaps I could never succeed in intelligibly doing so. But I know it when I see it…

—Justice Potter Stewart on *Jacobellis V. Ohio*, 378 U.S. 184

Clustering

2

Now that we have many different measures for describing protein structures and their constituent residues–geometric data from the Voronoi tessellation (residue volume and area), depth, backbone angles, and pairwise intra-residue distances–we are ready to quantitatively approach current problems in the field of protein structure and bioinformatics.

The first problem I examine is the identification of protein domains (subunits of a large protein that fold/function independently) from unannotated structural data. If we can determine the domains of some new protein, then we might be able to uncover clues about its biological function by examining the domains individually and searching for other proteins with similar domains. This search is the second problem: given a query protein structure, what parts (if any) are similar to known proteins?

To solve these problems, we must consider all the potentially relevant data: the hundreds of residues of a given protein and the tens of thousands of solved protein structures. Thus, I begin this chapter with a discussion of *unsupervised learning* (*clustering*) techniques for partitioning data into "natural" groups.

Guyon *et al.* [26] point out that clustering techniques can be used for two distinct goals; exploratory data analysis and data preprocessing. The application of this chapter, automated identification of a protein's constituent domains, is an example of exploratory analysis. A clustering algorithm will inform us about a protein by saying "these residues seem to belong together, separate from this other group over here…". This is in contrast to the next chapter, where we use clustering as a data preprocessing tool to encode protein structures as strings, which we then use for fast search. In that case, the only thing we require from the clusters is that they lead to good search results.

In both cases, before we can cluster the data we must first choose the dimensions (the *features* of the data) along which the clusters are defined, the number of clusters in which to parcel the data, and the underlying mathematical particulars:

1. **feature selection**: what characteristics do we want to use to separate the data? To cluster residues into different domains, we will want to look at their relative spatial positions of course, but we might also want to consider their volumes, their relative positions along the protein backbone, and some other characteristics discussed in the previous chapter.

2. **proximity measure**: how should we mathematically define similarity between data? If we are clustering only on physical position, then we should simply use the distance between points in space. If we consider volume or the amino acid types, then we must also devise a formula that takes these

features into account and gives two residues a numerical proximity (a non-physical "distance").

3. **cluster criteria**: what kind of clusters are reasonable? How many data should each cluster have, and how dissimilar can points within the same cluster be?

4. **cluster algorithm**: what algorithm should we use? There are k^n ways to put n things into k groups and for $n \approx 300$ residues, $k \approx 3$ domains. This is a *huge* number of possible choices. We do not have the computational resources to score all of them and choose the best partitioning according to the proximity measure and cluster criteria, so we must choose a procedure that works quickly (enough) and well (enough) for our problem.

I begin with a discussion of k-means, a widely used clustering scheme. I then discuss some of its drawbacks and overview more recent spectral clustering methods. Finally, I develop and use a spectral clustering methodology for partitioning protein structures into domains (semi-independent subunits of biological interest).

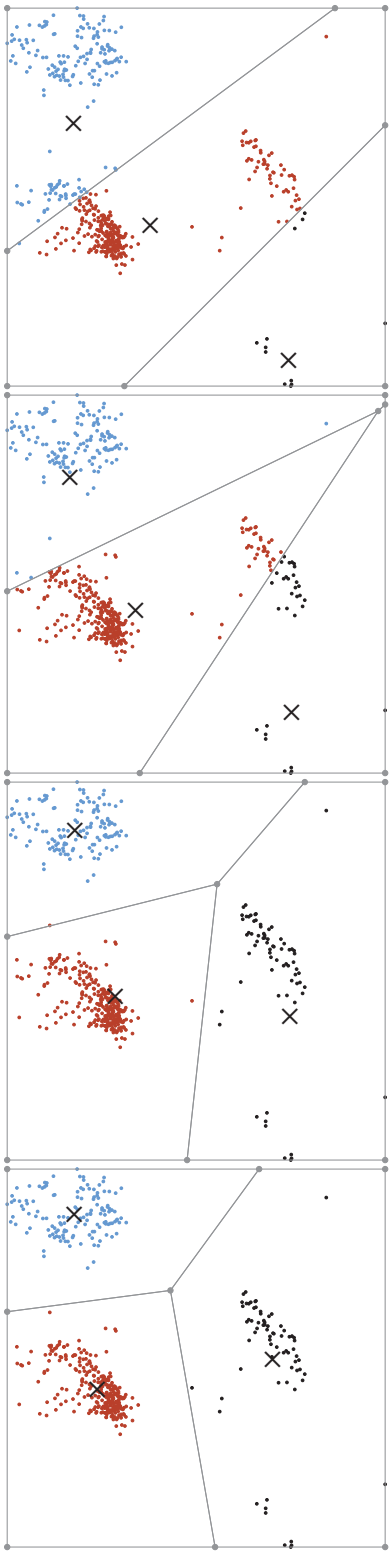
k-means

One of the most common clustering algorithms is k-means, which is a simple iterative procedure that partitions data into a specified number of clusters, k. The algorithm begins by randomly placing k *cluster means* and assigning each datum to the nearest mean (i.e. by the mean whose Voronoi cell the datum residues in). The means are then moved to the mean (*centroid*) of their corresponding data and the process iterated until the means reach fixed points (see figure at right). A score based on the deviation of each datum from its respective mean is then computed to assess the quality of the final partitioning.

The k-means algorithm has a great deal going for it in terms of simplicity: it is easy to understand, explain, and compute. However, the algorithm suffers from significant theoretical and practical drawbacks, foremost of which is the k of its namesake. The algorithm requires that the user specify in advance the number of clusters present in the data. Sometimes there is an obvious choice, but in the context of data exploration an ideal algorithm would determine some "meaningful" k from the data, presumably one that might illuminate characteristics of the underlying system. In practice, one can try different k and choose one according to the point deviations, but this is a tricky (read: *ad hoc*) business; the best possible score occurs when $k = n$, the number of data.

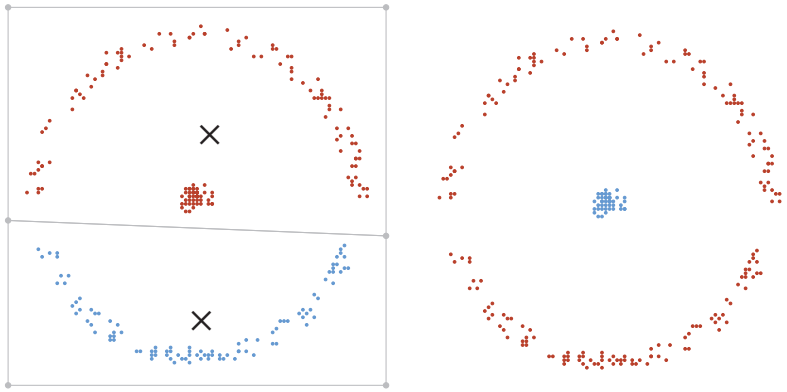
Another disadvantage is that the fitness landscape (the point deviation score) is non-convex, so the algorithm can become trapped in local minima and may not always find the best possible partition-

The first four iterations of a k-means clustering of a Ramachandran plot with $k = 3$. Lines indicate hard cluster boundaries (the Voronoi cells) of the three centroids (×).



ing. While the algorithm is guaranteed to converge,[†] there are no bounds on the time it takes to do so. In practice, one can choose the initial points from centroids of subsets of the data in the hope that the initial configuration will reflect the data's underlying structure and lead to faster convergence.

The possibility of becoming trapped in local minima highlights another undesirable attribute of the basic k-means algorithm; it draws "hard" boundaries between clusters, which gives a distinct assignment rather than "fuzzy" clustering algorithm that assigns points scores. Different configurations of the initial centroids will often lead to distinct clusterings of the data. Furthermore, k-means makes the very strong assumption that the data should fall into disjoint convex sets, though in some circumstance our intuition may disagree:



A toy data set un-intuitively partitioned by k-means (left) and more pleasingly by spectral clustering with k=2 and length scale σ = 2.5 mm (this long: —).

Even with these drawbacks, k-means is an excellent choice in certain situations. The algorithm is simple and, more importantly, *fast*; later in this thesis, I use k-means to cluster millions of protein fragments. The primary weakness of k-means clustering is non-convex data like the "rings" dataset above. In the next section, I describe a method to project non-convex datasets into a space where they can seperated by k-means.

Spectral clustering

Spectral clustering techniques are a family of algorithms with foundations in graph theory [48] that do not suffer from the same drawbacks as k-means. I first discuss an intuitive random walk

[†] To see that the k-means algorithm always converges, think of an analogous physical system where each point is fixed to the nearest mean by a spring. The positional energy of a spring attached to a point is cd^2, for some constant c and spring displacement d. The energy of the system takes the same form, $V = c\sum_i d_i^2$ (say all the points weigh the same and the springs have the same stiffness). Since a point changes clusters only when some other mean is closer and the means never move (on net) further from their points, each iteration can only decrease the system energy. Since the system energy is bounded from below (the data never move, so it will always take some energy to attach springs to them) and $\frac{\mathrm{d}}{\mathrm{d}t}V < 0$, the system must converge.

explanation of spectral clustering (and why it should work), then definine it in terms of matrix spectra.

Say you are taking a random walk through some data. You jump from one datum to another, preferring to hop over to nearby points rather than to ones very distant. In this framework, a cluster is just some subset of the data that you are likely to "hang out" in for some time. That is, if some data are bunched together, you will just hop around within them many times before you make a low-probability hop to a far away datum. This interpretation of clusters leads to intuitive partitioning, even for non-convex sets that stymie *k*-means.

To formalize the random walk* interpretation, we first have to define the probability of moving between points. The distance matrix, $\mathbf{D} = \left[d_{ij} \right] = |\mathbf{x}_i - \mathbf{x}_j|$, holds the pairwise distances between all the points. To specify our preference for nearby over far away points, we apply a *kernel function* to the elements of this matrix. A common choice for the kernel is a Gaussian (also known as the radial basis function),

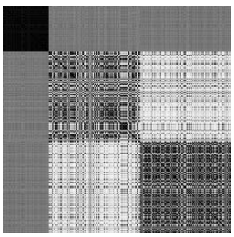
$$f(x) = \exp\left(-x^2/\sigma^2\right)$$

where the parameter σ is a characteristic length of our data.* This gives us the *affinity matrix*, $\mathbf{W} = \left[f\left(d_{ij}\right) \right]$.

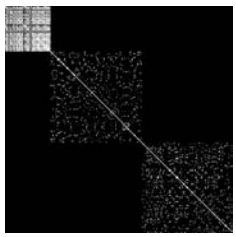
Note that if d_{ij} is large, then $\mathbf{x}_i$ and $\mathbf{x}_j$ are far away from one another. The reverse is true in the affinity matrix, and the magnitude of the elements depends on the kernel function. You can think of the kernel parameter σ as a contrast knob.

* In the business, these kinds of discrete random processes are known as Markov chains, which have property that the probability distribution over states (in our case, the data) in the next step depends only on the current state.

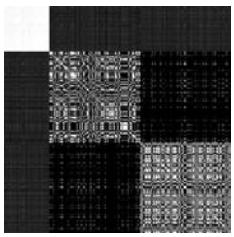
* Later in this chapter we cluster the atoms of a protein structure by their spatial positions. In that case, the characteristic length is on the scale of small molecules, 4 Å. However, the characteristic length σ refers to the distance between two data in any vector space, and will not always correspond to a physical distance.



distance matrix



affinity matrix, $\sigma = 0.05$



affinity matrix, $\sigma = 0.30$

Distance and affinity matrices for the ring data set. The first 50 rows correspond to the data at the centre, the next 100 to the top arc, and the final 100 to the bottom arc. Note that the matrices all appear to have block diagonal structure.

Finally, we normalize the rows of the affinity matrix so that we can interpret them as probability distributions. This forms the *transition matrix*:

$$\mathbf{P} = \left[\frac{w_{ij}}{\sum_j w_{ij}} \right],$$

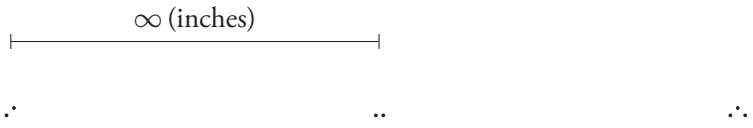
which we use to take the random walk. Each element P_{ij} gives the probability that we'll jump over to point j in a single step, given that we are at point i.

Now we have the transition matrix in hand, but how we actually use it to cluster the data remains unclear. First note that if two points

i, j are close to each other, then we expect two walkers (one at each point) to have similar odds of walking over to other data. That is, the rows i & j of $\mathbf{P}$ are similar; this suggests that we cluster the rows of $\mathbf{P}$. This is the right idea, but it is untenable in practice–the computational complexity of the k-means algorithm is $O(kdnl)$, with k clusters, n data in d dimensions, l iterations. If we if we want to cluster the rows of $\mathbf{P}$, then we would be projecting our data into n-dimensions, which gives a computational complexity that is quadratic in the largest term (the quantity of data), $O\left(kln^2\right)$. The crucial insight comes from matrix perturbation theory, which tells us that we should not look at the rows of $\mathbf{P}$, but rather at its largest eigenvektors [55]. To see why this is the case, we will take a quick diversion to discuss an ideal clustering scenario.

Ideal clustering

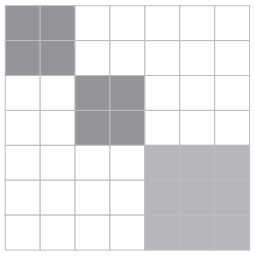
Consider the case of $k = 3$ clusters infinitely far apart from each other;



An ideal clustering scenario: three clusters of two, two, and three data. The clusters are infinitely far from each other (alternately, the data within each cluster are the same).

This is an ideal clustering problem: how could we *not* separate the data into groups, given that our margin of error is infinite? Since $e^{-\infty} = 0$, a random walker has no chance of ever leaving his initial cluster, and so the only nonzero elements of the transition matrix $\mathbf{P}$ occur in k sub-matrices along the diagonal;

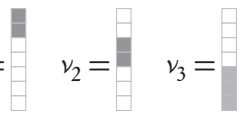
$$\mathbf{P} = \begin{pmatrix} 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 0 & 0 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 0 & 0 & 1/3 & 1/3 & 1/3 \end{pmatrix} =$$



.

In this special case, the eigenvalues and eigenvektors of $\mathbf{P}$ can be chosen as those of its sub-matrices (with the eigenvektors appropriately padded with zeros). Since the rows are normalized and the components of each sub-matrix are identical, the sub-matrices each have a single nonzero eigenvalue with unit magnitude, accompanied by the unit (eigen)vektor;

$$\lambda_1 = \lambda_2 = \lambda_3 = 1 \qquad v_1 = \quad v_2 = \quad v_3 = \quad .$$



Note that if we use these eigenvektors to define the columns of a matrix $\mathbf{X}$, then its rows cluster around k orthogonal points

The eigenvalue problem, $\mathbf{Mv} = \lambda\mathbf{v}$;

$$\begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \lambda \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

$$\tfrac{1}{2} \begin{pmatrix} \alpha + \beta \\ \alpha + \beta \end{pmatrix} = \begin{pmatrix} \lambda\alpha \\ \lambda\beta \end{pmatrix}$$

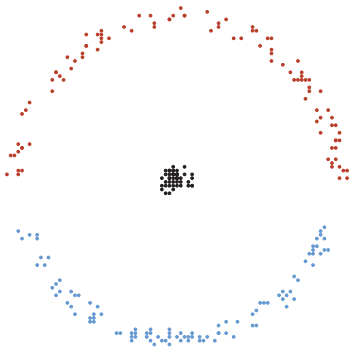
has nonzero solution $\alpha = \beta$, $\lambda = 1$.

on the unit k-sphere (with the rows of $\mathbf{X}$ normalized). It is *these* clusters, living in $\mathbb{R}^k$, that we separate using k-means, rather the rows of $\mathbf{P}$ (living in the much more expansive and desolate $\mathbb{R}^n$). Not only is the subspace defined by the first k eigenvektors much smaller, but also the data cluster on orthogonal parts of the k-sphere. This gives us some assurance that we do not have to worry about interlocked rings or other non-convex shapes that might trip up a centroid-based clustering scheme like k-means.

So we have the following algorithm;

1. build the distance matrix $\mathbf{D}$ from the data
2. apply some kernel function to $\mathbf{D}$ to form the affinity matrix $\mathbf{W}$
3. normalize the rows of $\mathbf{W}$ to form the transition matrix $\mathbf{P}$
4. find the first k eigenvektors $v_1, \ldots, v_k$ of $\mathbf{P}$ and define $\mathbf{X} = \left[v_1 v_2 ... v_k\right]$, normalized such that $\sum_{j=1}^{k} \mathbf{X}_{ij} = 1$
5. cluster the rows of $\mathbf{X}$ using k-means and assign the cluster labels back onto the original dataset (datum $\mathbf{s}_i$ is given the same label as row i of $\mathbf{X}$)

This procedure works quite well, as demonstrated at the beginning of the section; it neatly separates the ring from the centre. If we want $k = 3$, we throw the third eigenvektor of $\mathbf{P}$ onto $\mathbf{X}$, cluster the rows, and again get a nice looking partitioning (see right).



Data partitioned by spectral clustering with k=3 and length scale σ = 2.5 mm (this long: —).

Inferring k

At this point, we have a spectral clustering algorithm that, compared to k-means, is both more difficult to explain and far more expensive to compute. So far, the only advantage spectral clustering has over k-means is its ability to partition non-convex data in a more agreeable way than the seemingly arbitrary results of k-means. This is great for contrived toy examples, but you might be dubious about how often dense, interlocked rings or entwined helices occur in "real data", and if one should go to so much trouble to account for them. I would agree with you, and so it should come as no surprise that spectral clustering gives us something k-means does not: a theoretically justified way of finding the natural number of clusters, k in a given data set.

This is the real advantage of spectral clustering methods. The standard k-means approach requires not only several runs for each k (to see if the resultant assignments come from the data or are simply artifacts from the random initial centroid positions), but also several guesses at k itself. In spectral clustering, once you have constructed the transition matrix $\mathbf{P}$, it is simple to infer from its spectrum a natural value for k. In fact, one can extract several

"natural" values for k, roughly equivalent to looking at the data at different resolutions.

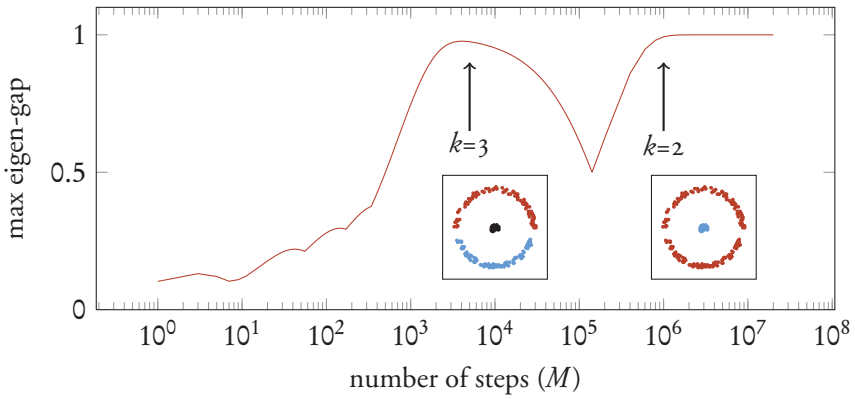
In the ideal case, we had three clusters infinitely far apart from each other, leading to three nonzero eigenvektors with unity eigenvalue. In general, a dataset consisting of k clusters will lead to $\mathbf{P}$ having k large eigenvalues (see Ng *et al.* [55]). To find k we find the position of the largest eigen-gap;

$$k = \underset{i}{\operatorname{argmax}}\left(\lambda_i - \lambda_{i+1}\right),$$

with $|\lambda_i| > |\lambda_{i+1}|$.

Azran and Ghahramani [2] extend this procedure to find several natural k for a set of data. Their method is most easily understood in the random walk interpretation, where $\mathbf{P}$ is the matrix that moves a random walker forward one step. Taking the transition matrix to the M^{th} power gives information about multi-step walks; $(P^M)_{ij}$ is the probability of moving from i to j in M steps over *all* possible paths. If two rows of $\mathbf{P}^M$ are similar, then the paths taken by random walkers starting at those points will be similar: after M steps, they end up in the same set of points. With more steps, a walker will explore more data, and the maximum eigen-gap of $\mathbf{P}^M$ will tend to occur at smaller k, indicating fewer, larger clusters. Eventually, as $M \to \infty$, the principal eigenvektor of $\mathbf{P}^M$ (with $\lambda_0 = 1$) dominates–with enough steps, the walker will visit all of the data.

Azran and Ghahramani give a heuristic for choosing k: natural clusters in the data manifest themselves as maxima on the plot of P^M's maximum eigen-gap vs. M, the number of steps in the walk. If we carry this procedure out for the rings data–looking at the eigen-gaps of $\mathbf{P}^M$ vs. M and choosing k from the maxima on that plot–we get results that agree with our intuition. There are two maxima with the eigen-gaps near unity, which indicates that there are two strong clusterings of the data. The first occurs for $M \approx 10^4$ steps, and represents k=3 groups in the data. The second, broader clustering occurs after $M \approx 10^6$ steps and corresponds to k=2 groups in the data;



Plot of the the maximum eigen-gap of $\mathbf{P}^M$ vs. steps of the random walk, M of the rings data with σ=0.05. Note that the largest two eigen-gaps occur for k=3 and k=2.

Theoretically, this is exactly what we want, but at first glance it is a computational nightmare. Spectral clustering seems to demand

that we find M different powers of a large matrix *and* find the spectrum of each one. Luckily, the calculation of eigen-gaps is quite tractable. First note that we can write the matrix as

$$\sum_i^n \lambda_i \nu_i \nu_i^{\mathrm{T}},$$

and since the eigenvektors of a symmetric matrix are orthogonal, we can see from this expression that all powers of the matrix share the same eigenvektors;

$$\begin{aligned}\mathbf{P}^M &= \left(\sum_i^n \lambda_i \nu_i \nu_i^{\mathrm{T}}\right)^M \\ &= \sum_i^n \lambda_i^M \nu_i \nu_i^{\mathrm{T}}.\end{aligned}$$

All we need to do is look at the powers of the eigenvalues, which is computationally very cheap. Also note that the transition matrix is dominated by those eigenvalues that survive M steps of the walk, i.e. $\lambda_i \approx 1$.

We now have two knobs to twiddle: σ, the characteristic length scale of the affinity matrix $\mathbf{W}$ (page 18) and M, the steps of a random walk to take among the data. Azran and Ghahramani [2] describe them as follows: "by setting the value of σ we explicitly define the structure in the data, and by scanning over M we reveal this structure." The interval of interest for σ is the range of distances between the data;

$$\sigma \in \left(\min_{i,j} d\left(\mathbf{x}_i, \mathbf{x}_j\right),\ \max_{i,j} d\left(\mathbf{x}_i, \mathbf{x}_j\right)\right),$$

and for M it is however many steps it takes for the eigenvalues $\lambda \neq 1$ to die off. We can then generalize our search for the number of clusters and find k by maximization of the eigen-gap

$$k(M, \sigma) = \operatorname*{argmax}_k \left(\lambda_k^M(\sigma) - \lambda_{k+1}^M(\sigma)\right).$$

In this section we have discussed the general problem of clustering data, and have seen two algorithms that can be used to partition a dataset. The k-means algorithm, though fast and simple, does not perform well for intertwined data like the example of one cluster surrounding another. Furthermore, one must specify k in advance, which is not ideal if we hope to use clustering for exploratory data analysis. In the upcoming case of protein domain decomposition, we know that something like k=20 domains is unreasonable, but k-means cannot help us decide for a given protein whether k=1 or k=15.

We then discussed an alternative, spectral clustering, which performs well on non-convex data sets and also highlights certain

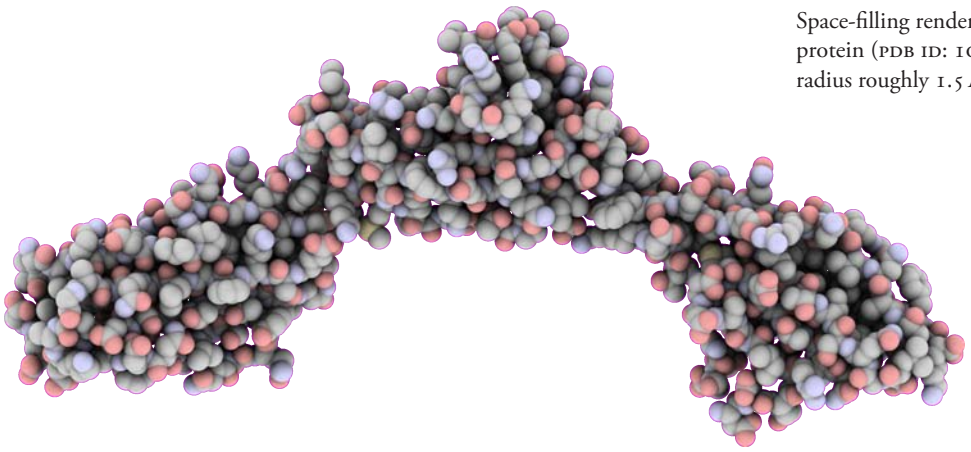
values of k in a theoretically motivated way. The maths are a bit more involved than those of k-means, but spectral clustering maintains an intuitive explanation in terms of a random walk. In the next section I consider the application of this spectral clustering algorithm to the problem of decomposing protein structures into domains.

Domain decomposition

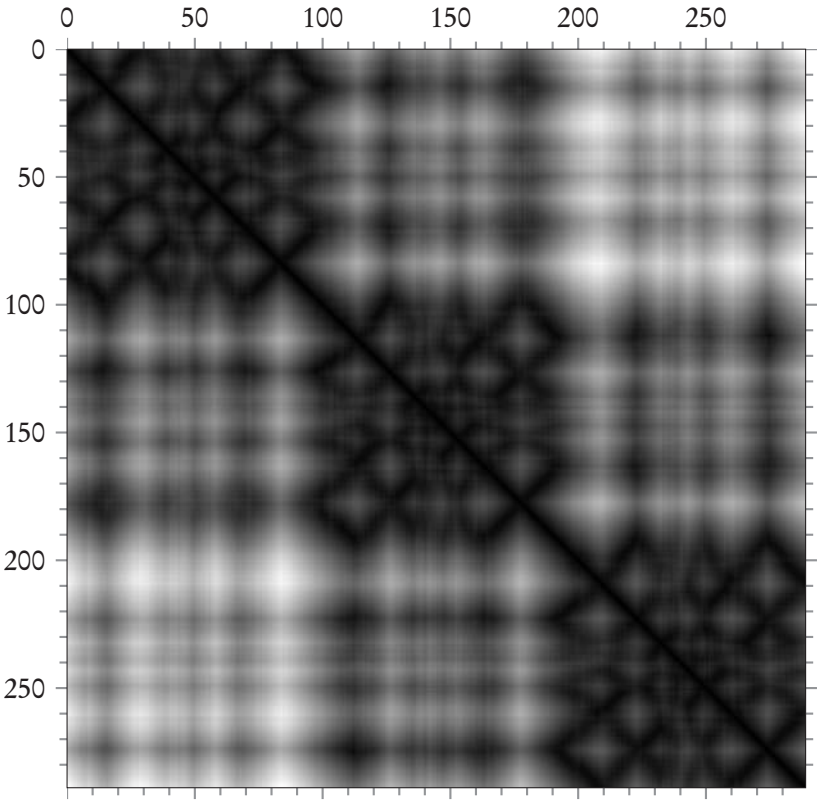
Some proteins consist of multiple domains: semi-independent segments of the protein backbone that have their own hydrophobic cores and/or specific biological functions. There are numerous algorithms based on this generally accepted, rough definition (see Veretnik *et al.* [85] for a comprehensive review). The majority of these algorithms rely on the fact that there are more intra-domain residue-residue contacts than inter-domain contacts. If a protein has multiple domains, this will be reflected in a block diagonal C_α distance matrix or contact map. Spectral clustering, then, can be used as yet another domain partitioning algorithm.

I use the CATH database [58] as the standard of truth for evaluating the performance of the spectral clustering algorithm. One should note though, that this is my choice, and there are several other well known taxonomies of protein structure. In particular, SCOP [54], which relies entirely on human annotation and has an evolutionary focus, and FSSP [31], which is a fully automated database built around DALI [33] pairwise structure alignment program. These databases agree on the domain assignments of 75% of their common chains [27]. For additional perspective on the automatic partitioning of protein domains, see Holland *et al.*'s aptly named paper: *Partitioning protein structures into domains: why is it so difficult?* [30].

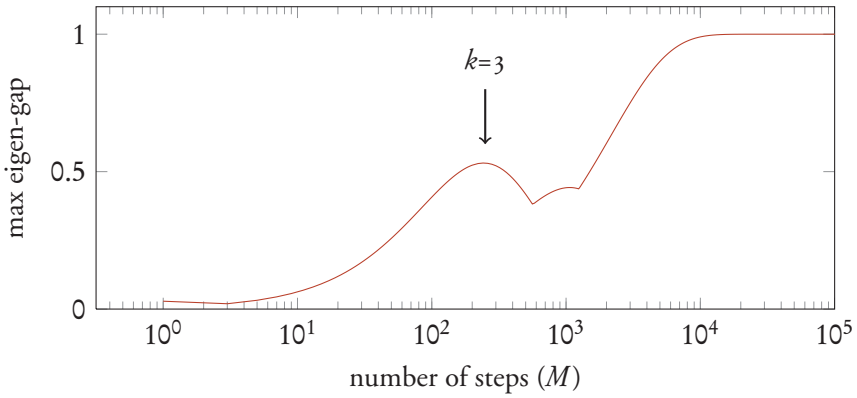
As a first example of a multi-domain protein, consider a fragment of the neural cell adhesion molecule NCAM. This looks like a very easy problem, as the chain appears to be clearly separated into three contiguous domains. Not surprisingly, the largest eigen-gap occurs at $k = 3$.



Space-filling rendering of a three domain protein (PDB ID: 1QZ1). Each sphere has radius roughly 1.5 Å.



C_α distance matrix of a three domain protein with PDB ID: 1QZ1. Darker squares indicate closer residues. Note that this matrix has block diagonal structure–the first hundred residues are close to each other, as are roughly the next and final hundred residues, consistent with the space-filling rendering.

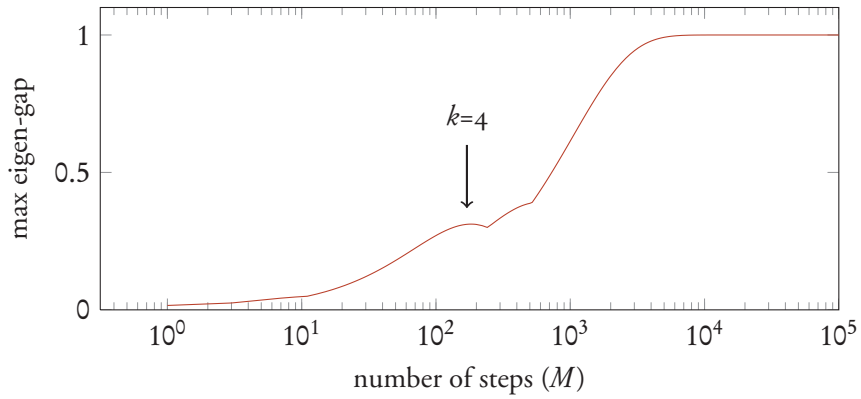


Plot of the maximum eigen-gap vs. steps of random walk of the backbone C_α of a three domain protein (PDB ID: 1QZ1). The kernel function's σ=4 Å. The largest local maximum occurs around M=250, corresponding to k=3 clusters.

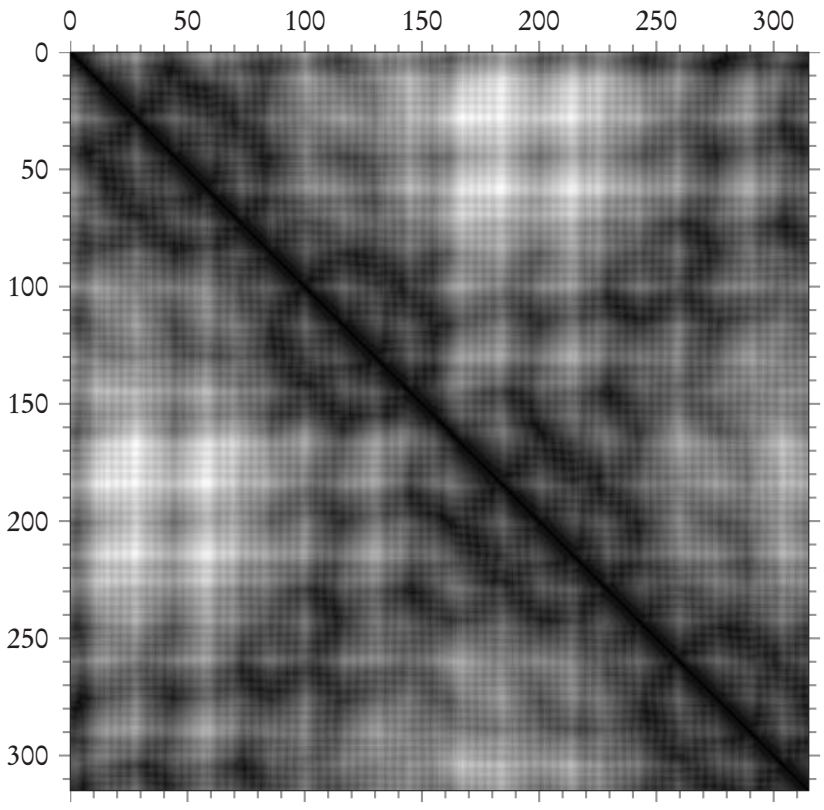
The assigned domains match up quite well with the assignments of the CATH database (proteins sometimes begin with a negative sequence index for various historic accidents, consensus alignments to other sequences, an initial mischaracterization of the amino acid residue sequence, &c.);

domain	CATH assignment (residue sequence #)	spectral assignment (residue sequence #)
1	-1–96	-1–96
2	97–192	97–189
3	193–289	190–289

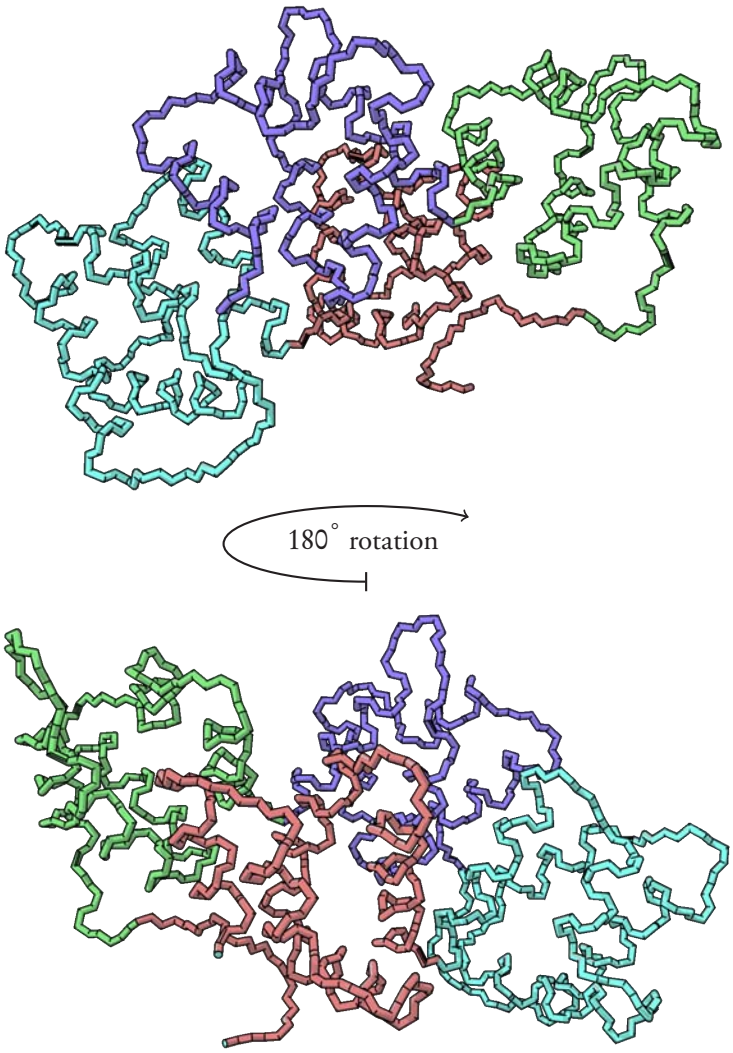
A slightly more difficult example is that of annexin A4, a sodium bound cellular protein (PDB ID: 2ZHJ) made up of 322 residues. In this case, the random walk has a single eigen-gap local maximum of 0.37, corresponding to four domains (on next page).



Plot of the maximum eigen-gap vs. steps of random walk for the backbone C_α of a four domain protein (PDB ID: 2ZHJ). The kernel function's σ=4 Å. The only local maximum occurs around 10^2 steps, corresponding to k=4. Note the small ridge (but not maximum) around 500, which corresponds to k=2.



C_α distance matrix of a four domain protein with PDB ID: 2ZHJ. Darker squares indicate closer residues.



Backbone of 315 residue protein with PDB ID: 2ZHJ, colored according to spectral clustering of residue C_α position with $\sigma = 4$ Å. Note that even though the algorithm does not take backbone connectivity into special account, the clusters are contiguous except for beginning of the chain (bottom right of top figure), which is ■ rather than ■.

These assignments match up quite well with the assignments of the CATH database;

domain	CATH assignment (residue sequence #)	spectral assignment (residue sequence #)
1	13–85	9–85
2	86–159	86–158, 317
3	160–245	159–241, 243–244
4	246–316	2–8, 242, 245–316

To more rigorously test this spectral clustering methodology, I applied it to the dataset* used by Jones *et al.* [36] to test the CATH consensus approach.

The dataset consists of the protein chains:

- 1 domain: 1AAK, 1ACE, 1BBHA, 1BBPA, 1BRD, 1FXIA, 1GKY, 1GMFA, 1GMPA, 1GOX, 1OFV, 1PYPA, 1RBP, 1RCB, 1RVEA,

* That protein domain identification is a hard (or at least, somewhat ill-defined) problem is reflected by the fact that the domains of three of the proteins (5% of the dataset) have since been reassigned since Jones *et al.*'s paper [36]:

  - 1PHH: 3 → 2 domains
  - 2HAD: 2 → 1 domains
  - 8ACN: 2 → 3 domains

1snc, 1tie, 1tlk, 1ula, 1wsyA, 2azaA, 2ccyA, 2rn2, 2stv, 2tmvP, 3chy, 3cla, 3dfrA, 4blmA, 5p21, 2had

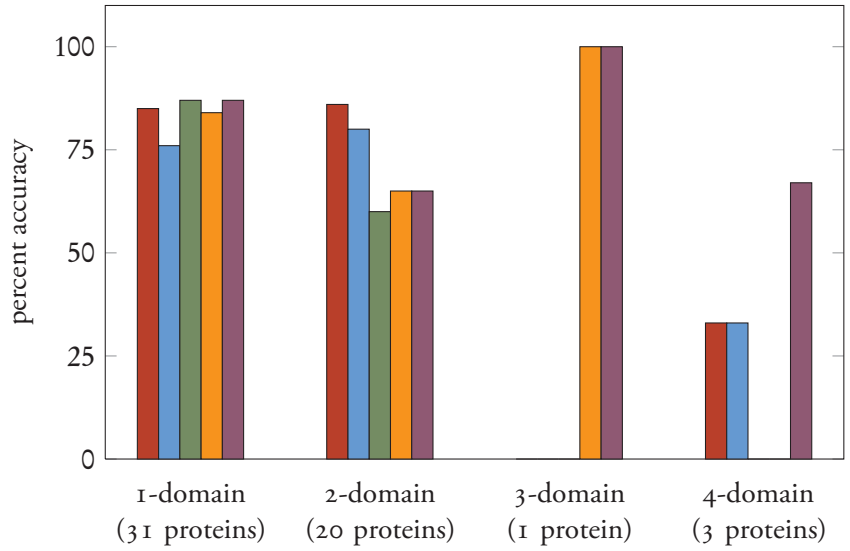
- 2 domains: 1ezm, 1fnr, 1gpb, 1lap, 1pfkA, 1ppn, 1rhd, 1sgt, 1vsgA, 1wsyB, 2cyp, 3cd4, 3gapA, 3pgk, 4gcr, 5fbpA, 8adh, 8atcA, 8atcB, 1phh
- 3 domains: 3grs
- 4 domains: 1atnA, 2pmgA, 8acn

I use the following algorithm to determine the number of domains of each protein chain:

1. Find the spectrum of the transition matrix **P** built from the distance matrix of C_α atoms according to the procedure described on page 20 with σ=4 Å.*
2. Choose the number of domains k as the largest local maximum of the max eigen-gap vs. random walk steps plot. If a local maximum does not exist but some eigen-gap is larger than a cutoff ϵ and corresponds to $k \neq 1$ choose that k. Otherwise, choose k equal to unity.

* Varying this cutoff between 3 Å and 7 Å has little effect (results not shown).

On the same test set [36] used by the CATH algorithms, this spectral clustering procedure correctly classifies 73% of the test set compared to 76%, 71%, and 67% for PUU, *Domak*, & *Detective*, respectively:



Relative performance of the CATH domain partitioning algorithms (data from Jones *et al.* [36]): PUU, Domak, DETECTIVE, the spectral clustering algorithm with cutoff ϵ = 0.2 and σ=4 Å, and the neural network spectral clustering algorithm with ϵ = 0.1 and σ= 0.1.

As we have seen, spectral clustering of the C_α distance matrix gives good domain partitionings; comprabable to the algorithms currently used by structural biologists. But if we have a protein's structure, we have a good deal more information than just the C_α pairwise distances. This additional information might more reliably indicate the number of domains within a protein, but it is not

immediately clear how to incorporate such data into our spectral decomposition algorithm. Presumably it would somehow fit into the distance metric, making the components of the distance matrix generalized distances in a vector space with dimensions of residue volume, residue type, solvent exposed area, the C_α position, &c. rather than just the physical inter-C_α distance. A block diagonal matrix is ideal for spectral clustering: hopefully, we can find a generalized distance that gives a blockier affinity matrix. That is, the components corresponding to residues in the same domain become higher and for those in different domains, lower.

The problem is now to find a way to use more structural data than just the C_α distances to determine if given pair of residues are in the same domain (according to the operational definition given by CATH). Note that

1. we have several dimensions, but only one (spatial distance) has an intuitive relationship to the problem at hand (residues that are physically far away from each other are probably not in the same domain), and

2. we have extensive training examples: a protein of length n has $\frac{1}{2}n(n-1)$ residue pairs each, and CATH has nearly 10,000 protein chains classified into domains.

These two facts suggest that this problem might be readily solved by a general statistical learning framework: a generalized regression procedure that can "learn" from examples (this pair of residues is in the same domain, but this pair is not) and then predict whether or not a novel pair of residues are likely to lay in the same domain.

Neural network

Consider a familiar linear regression model, $f(\mathbf{x})$, which takes the form of a weighted sum of fixed nonlinear basis functions:

$$f(\mathbf{x}) = \sum_{j=1}^{d} w_j g_j(\mathbf{x}),$$

with input vector $\mathbf{x}$, weights w_j, and basis functions $g_j(\mathbf{x})$. If you expect a second degree polynomial to fit your data, for instance, you take your model to be

$$f(x) = w_1 + w_2 x + w_3 x^2$$

and adjust the weights w_j until the polynomial fits your data.

Artificial neural networks can be thought of as a generalization of the regression procedure for the case where one has no idea of the basis functions. Instead, one allows the basis functions to vary according to some nonlinear function $h(\cdot)$ with weighted components

of the datum at hand:

$$g_j(\mathbf{x}) = h\left(\sum_{i=1}^{m} w_{ji}^{(1)} x_i\right),$$

where m is the dimension of the input vector $\mathbf{x}$. The nonlinear function $h(\cdot)$ is known as the *hidden layer* activation function, and is typically chosen to be a sigmoidal function giving output between 0–1. We then have (for d *hidden nodes*)

$$f(\mathbf{x}) = \sum_{j=1}^{d} w_j^{(2)} h\left(\sum_{i=1}^{m} w_{ji}^{(1)} x_i\right),$$

which we transform using the *output layer* activation function $\zeta(\cdot)$ to get the components of the output*;

$$y(\mathbf{x}) = \zeta\left(\sum_{j=1}^{d} w_{kj}^{(2)} h\left(\sum_{i=1}^{m} w_{ji}^{(1)} x_i\right)\right).$$

* I am using the neural network to come up with a scalar "distance", but in general the output can be a vector–just take $f(\mathbf{x})$ to give a component, and train a different set of weights for each output dimension.

To train the network, one chooses a set of T example pairs $(\mathbf{x}, \mathbf{y})$ and iteratively adjusts the weights of the hidden and output layers ($w_{ji}^{(1)}$ and $w_{kj}^{(2)}$, respectively) to minimize the error; $E = \sum_{t=1}^{T} \left(f(\mathbf{x}_t) - y_t\right)^2$. Conceptually, the simplest way to adjust these weights is via gradient descent: for a set of weights $\mathbf{w}$, $\mathbf{w} \to \mathbf{w} - \eta \nabla E(\mathbf{w})$ (with a learning rate η), but in practice one uses more efficient and numerically robust methods [7, 75].

I used as a training set about 250,000 residue pairs drawn from six proteins,* with the input vectors having components: Euclidean distance, backbone distance, volume of residue A, volume of residue B, and the output a scalar "distance"–zero if the pair was in the same domain, and one if they are in different domains. To reduce the training time of the network, I limited the number of pairs by random sampling, taking 80% of pairs within the same domain and 20% of pairs in different domains.

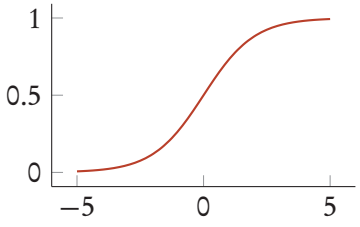
*

2-domain	3-domain	4-domain
2PGX	1FOO	2ZHJ
3CB6	1CUK	
	2HE7	

The six proteins used to train the neural network. These proteins are, of course, disjoint from the test set.

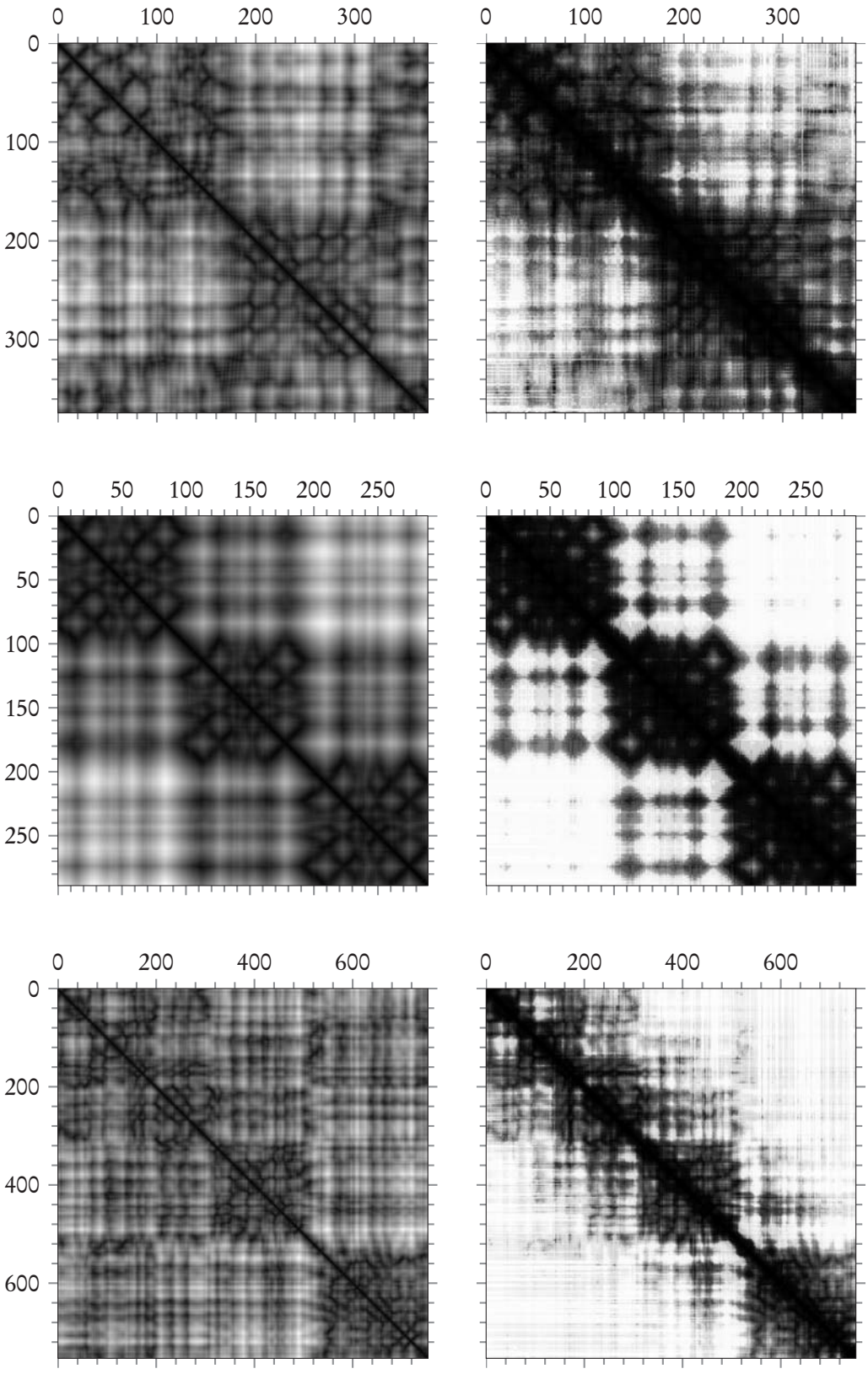
I used d=20 hidden nodes, and for both the hidden and output layers, I used the logistic function function:

$$h(x) = \zeta(x) = \frac{1}{1 + e^{-x}}$$



The neural network "distance" matrices do appear to be far blockier than standard Euclidean distance matrices (on next page). This results in slightly improved performance on the test proteins–correctly partitioning 78% of the proteins in the set compared to using just Euclidean distance, which correctly partitions 73%. Note that the neural network "distance" seems to especially help on the 4-domain proteins (see graph on page 27).

Comparison of inter-C_α Euclidean distance matrix (left) with a neural network "distance" matrix for PDB IDs 1QZ1, 8ADH, 8ACN, with two, three, and four domains, respectively (top to bottom):



fin.

In this chapter we have discussed the use of data clustering as an exploratory tool. We have seen two algorithms for clustering data: the fast and simple *k*-means approach, and a more fruitful spectral clustering technique. We applied the latter to the problem of protein domain decomposition, clustering amino acid residues based on their positions in space, which resulted in good agreement with expert classifications. We then extended the spectral clustering method by using the output of a neural network as the distance measure over which to cluster. This allowed us to incorporate additional residue data into the procedure and led to improved results.

In this chapter, the clusters themselves were the results of interest: how many domains are in a given protein, and which residues belong to which domain? In the next chapter we will see that data clustering can also be useful as part of a larger scheme–structure search. For structure search, we must answer all of the same questions to cluster the data: what features should we use, what is an appropriate distance measure, &c. The ultimate result we are interested in is the efficacy of the search procedure, not the clusters themselves.

Notes

I implemented the neural network system using the Fast Artificial Neural Network Library [56], which is very well documented, has bindings to many popular languages, and is LGPL open source to boot.

Bishop gives an excellent overview of the theory and practice of several machine learning methods (including neural networks);

> Christopher M. Bishop. *Pattern recognition and machine learning*. Springer. 2006. call no: Q327.B52

and for a more advanced mathematical discussion, see Theodoridis and Koutroumbas' *Pattern Recognition*;

> Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition, Third Edition*. Academic Press. 2009

which discusses in detail the often overlooked subject of feature generation and selection.

The subject of clustering and machine learning methods in general are not just academic curosities. The field tends to be quite pragmatic, with both practical & theoretical advances coming from commerical enterprise as often as from the academy. Internet retailers in particular use clustering as a data preprocessing scheme, grouping customers based on their purchases and recommending products purchased by similar customers. Recently some significant

new matrix factorization and ensemble learning methods [42] have have been developed in a $1 million prize competition sponsored by the movie rental company Netflix for predicting a viewer's movie ratings based on his or her past rentals.

A broad introduction to the conceptual problems in the field as well as practical implementation details from a non-academic perspective is given in the several hundred blog posts of Bradford Cross*, who currently runs a startup that predicts flight delays from historical weather and FAA data. Another excellent resource on both the basics & the state of the art of machine learning topics can be found in conference talks and tutorials available online at `www.videoLectures.net`.

* `measuringmeasures.com`

I also have two technical notes about spectral clustering: first, the transition matrix $\mathbf{P}$ consists entirely of non-negative elements, which tells us (by the Perron-Frobenius theorem, page 7) that the largest eigenvalue of $\mathbf{P}$ must correspond to an eigenvektor with strictly positive components.* Since the rows of the transition matrix sum to unity, then we know that the vector of all ones is the eigenvektor with eigenvalue unity, and since the Gershgorin circle theorem tells us that no eigenvalue of a square matrix can be larger than $\max \sum_{i \neq j} |P_{ij}|$, this vector must be the principal eigenvektor. But if all its elements are equal, the principal eigenvektor gives us no information about the points and so we can omit it from step 4 (page 20) to reduce computational time.

* In the ideal clustering example the unique largest eigenvalue is unity, corresponding to more than one eigenvektor–this is because in that case the matrix is reducible (if you think of it as an adjacency matrix, it corresponds to three separate graphs). In general, however, this will not be the case, and the largest eigenvalue will correspond to a single eigenvektor.

Second, there is a deep connection between this random walk clustering process and the problem of "graph cutting". An extensive and very readable discussion is given by Luxburg [48]. The setup should sound familiar: each datum is a node in a graph G with undirected edges between them having weights $w_{ij} \geq 0$ given by a similarity function $f(i,j)$. We can express this as a real-valued adjacency matrix, $\mathbf{W} = \left[w_{ij} \right]$, and define for each node a characteristic "volume" (degree): $V_i = \sum_j w_{ij}$. In this context, the problem of finding k is equivalent to choosing edges of G to sever such that we end up with k disconnected graphs.

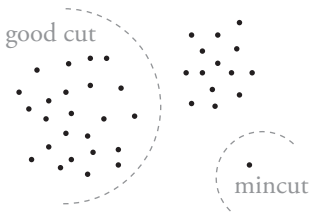
If we let $W(A,B)$ be a function of two subgraphs of G;

$$W(A,B) = \sum_{i \in A, j \in B} w_{ij},$$

we can use $W(A,B)$ to define the mincut;

$$\text{mincut}(A_1, \ldots, A_k) = \tfrac{1}{2} \sum_{i}^{k} W\left(A_i, \overline{A}_i\right),$$

where $\overline{A}$ the complement of A. To find k clusters of G, one finds the disjoint subgraphs $A_1 \ldots A_k$ such that this sum is minimized. In practice, this method tends to partition outliers into their own clusters because (by definition) outliers are dissimilar to every point and thus have very low volumes (see right).



Shi and Malik [69] defined normalized graph cut;

$$\mathrm{Ncut}(A_1,\ldots,A_k) = \tfrac{1}{2}\sum_i^k \frac{W\left(A_i,\overline{A_i}\right)}{\mathrm{Vol}\left(A_i\right)},$$

which compensates for this tendency, but finding a solution is an NP-hard combinatorial problem (see Shi and Malik [69], appendix A). As it turns out, spectral clustering approximately minimizes the normalized graph cut,* and both are special cases of a more general kernel k-means problem–see Dhillon *et al.* [16] for details.

* Though there is no guarantee regarding the error of spectral methods; in practice they seem to work very well, but there exist known pathological cases. See Luxburg [48] pg 13 for a discussion.

How complex a code can it be if these knuckleheads are usin' it?
Then again, what does it say about us if we can't break it?

—Detective Kima Greggs

Structure search

3

In the previous chapter we discussed the use of clustering for exploratory data analysis. Given some protein structure, we used clustering techniques to identify subunits of biological interest: its domains. In this chapter, we use clustering techniques for a different goal: fast search of protein structures. I discuss breaking proteins apart into short (≈5 residue) backbone fragments and clustering these fragments into "letters". We can then approximately represent a protein's 3D structure as a string drawn from this structural alphabet. Essentially, this recasts the problem of 3D protein structure search as spell-check, a problem we can solve with well-studied and implemented string algorithms.

If we want to roughly describe the secondary structure of some new protein, we can partition its residues by where they fall on a Ramachandran plot (into regions of "helix", "sheet", and "other") and then very succinctly describe the residues as going "other helix other sheet helix helix …" rather than "the first residue is at x, y, z, the next one is at x_2, y_2, z_2, …".

Pretend we encode two small proteins using this "helix, sheet, other" alphabet (`h`, `s`, `o`), with protein **a** going as `ohoshho` and protein **b** as `osssshhhoo`. To compare the string representations of these proteins, we need to use a string distance metric. The most common such metric is the Levenshtein [45] edit distance, which is based on the number of *insertions*, *deletions*,* and *substitutions* required to turn one string into the other. In this case, we can get from the shorter string to the longer by making the following edits:

* An insertion into a shorter string is the same as a deletion in the longer string, and often these two operations are counted together as *indels*.

string	operation	Levenshtein distance
`ohoshho`	(Protein A)	0
`ohoshhho`	insertion	1
`ososhhho`	substitution	2
`ossshhho`	substitution	3
`osssshhho`	insertion	4
`osssshhhoo`	insertion	5

Calculation of the Levenshtein edit distance from the string **a** = `ohoshho` to **b** = `osssshhhoo` in five steps.

The total edit distance is five, which gives a normalized similarity score between the two strings;

$$\begin{aligned}\text{Levenshtein similarity} &= 1 - \frac{\text{Levenshtein distance}}{\text{length of the longer string}} \\ &= 1 - \frac{5}{10} \\ &= 0.5.\end{aligned}$$

There exist weighted edit distances, where operations are counted differently: an `H` → `S` substitution might cost more than a `H` → `0` or gap, for instance. For both weighted and unweighted distances, the

length of the optimal path of edits from one string to the other is found using dynamic programming algorithms.* Calculating this similarity score between a query protein's string and all the strings in a corpus allows us to search: the best matches are the proteins with strings most similar to our query string. The general search procedure goes as follows:

1. Construct a structural alphabet by first collecting backbone fragments from the protein structures to be searched. Cluster these fragments using k-means along some combination of their backbone angles, the residue volumes or depths, and other structural features. The residue fragment closest to its cluster mean the *prototype*; it is the archetypical backbone fragment of that letter.

2. Encode the proteins to be searched using this structural alphabet to form a string corpus.

Then, to search this corpus with a query protein:

3. Encode the query protein as a string using the structural alphabet derived in step 1. Since we used k-means to determine the letters, we can encode a new protein by assigning each fragment to the nearest mean (letter) in the (backbone angles, depth, volume) vector space.

4. Compare the query string with all the strings in the corpus according to a string metric and rank the results.

This search relies on two parts: the structural alphabet and the string similarity metric. I briefly discuss some existing alphabets and string metrics, as well as their previous applications for searching protein structures. I then highlight some shortcomings of the existing alphabets and construct a set of new alphabets that perform better in the protein search problem using Levenshtein edit distance. Surprisingly, these alphabets, coupled with the simple, unweighted edit distance score, appear to be competitive with computationally expensive 3D-alignment methods that determine similarity by trying to overlay two protein structures.

Quantization of protein structure

The quantization of a protein's 3D structure into a string of fragment prototypes is not a particularly new idea: In 1976, Levitt and Chothia [46] noted that the general structure of proteins (the arrangements of their constituent α-helices & β-sheets) could be described by four classes. Since then, many researchers have attempted to find other such "structural building blocks" to describe protein conformations.

An intuitive method of generating structural building blocks is Kolodny and Levitt's [39, 40] k-means procedure, which uses

* See Gusfield's text [24] for a comprehensive introduction to string alignment algorithms and their applications in biological sequence analysis.

coördinate RMSD between backbone fragments as the distance metric. Essentially, the building blocks are constructed by breaking proteins into short, f-residue long pieces, clustering these pieces into k groups, and averaging the coöordinates of the fragments in each cluster to form a fragment prototype. Friedberg *et al.* [21] use Kolodny's k=20 letter alphabet of f=5 residue fragments with the Smith-Waterman* string alignment algorithm to search protein structures.

Guyon *et al.*'s SA-search [25] also uses the Smith-Waterman alignment algorithm, but with a 12-letter structural alphabet determined by the hidden Markov models of Camproux *et al.* [11]. These models are trained on inter-C_α distances drawn from a sliding, four-residue long window along the backbone chain (the fourth "distance" is essentially the signed volume of the segment) rather than the coördinates themselves.

Rather than look at the individual positions of atoms, the authors of *La Jolla* [4] classify residues by their ϕ & ψ angles and perform fast searches between the resulting strings using an n-gram method.* Likewise, Tung *et al.* [81] construct a 23-letter structural alphabet based on the κ and α angles of five-residue backbone fragments and search with the BLAST algorithm of the genomics community, which uses several heuristics to quickly find good (but not necessarily optimal) string alignments according to a substitution penalty matrix.*

* The Smith-Waterman alignment algorithm [71] is used to score string pairs using a weighted edit distance. For instance, we don't expect residues to dramatically change between related proteins; a substitution of tryptophan (which is big and greasy) to glutamate (which is negatively charged) should give a lower similarity score than a substitution of tryptophan to tyrosine. These specific substitution weights are stored in a "penalty matrix" that is generated from alignments of the strings that *should* be similar. In this case, we would choose the weights according to aligned sequences of proteins known to be evolutionarily related. The Smith-Waterman algorithm also allows for an affine gap penalty, where opening a gap is more costly than extending one. This allows for pairs like

```
hhsshhsshh
hhsshhsshhooooooooo
```

to be scored as similar.

* n-grams (k-mers, q-grams) are word fragments (`fortnight` has 2-grams `fo`, `or`, `rt`, …) commonly used to embed documents into a vector space, where similarity is given by the cosine distance (page 102).

* Tung *et al.*generated their matrix using alignments of pairs of protein structures from the same structural superfamily. One of the advantages of using BLAST emphasized by Tung *et al.* (who name their method 3D-BLAST) is that it gives an E-value score, which indicates the statistical significance of a search hit.

Shortcomings of previous approaches

The process of converting a 3D protein structure into a 1D string requires a structural alphabet and a string metric. Historically, these two pieces have been considered separately. The α-helix and β-sheet (the original "structural building blocks") were conceived of to explain and describe protein structure (not to search it) and algorithms like BLAST were developed to query very large amounts of genomic data. A successful structural search programme must consider both of these components together: developing a structural alphabet is a clustering problem, and as such it is ill-defined. Ultimately, the best structural alphabet for search can only be determined empirically.

Note that all of the structural alphabets I've mentioned are constructed from the geometric characteristics of fragments treated independently from their parent proteins. This is not surprising given that the idea of protein building blocks beyond "helix", "sheet", and "other" was first considered in the context of sequence/structure description and prediction [11, 18, 28, 40]–alphabets of building blocks are often described only by how well they can reconstruct a protein structure (i.e. the RMSD between a structure made from the alphabet prototypes and the original protein). But reconstruction is different than search, and an alphabet optimized to solve the former problem will not always be ideal for the latter.

To see why, suppose that there is some peculiar backbone fragment shape that sometimes occurs on the surface of a protein and sometimes deeply buried in the hydrophobic core: if one is only interested in reconstruction (finding literal building blocks), then the same shape is the same shape and there is no reason to distinguish between these two situations. But if the problem at hand is that of fold-level structure search, then these two scenarios are distinct. Clustering fragments on global measures (depth, volume, principal eigenvektor component, &c.) in addition to the intra-fragment angles and C_α distances might provide additional discriminative power in searches.

That we need to consider our final application–protein structure search–holds for the string metrics as well. We have far less protein structural data than we do genome sequence data,* and so we are not restricted to using approximate, heuristic algorithms like BLAST. The exact solution for pairwise alignments according to a substitution penalty matrix is given by the Smith-Waterman algorithm, but algorithms involving TF-IDF or other more elaborate metrics can still be used for protein structural string search–a "slow" string algorithm is still far faster than 3D structure superposition/alignment procedures.

* This is for biological as well as technological reasons: not every DNA nucleotide codes for an expressed protein, and it is far cheaper and faster to determine n letters of a DNA sequence than the native positions in space of an n-residue protein. Currently, NCBI's GenBank sequence database contains about 10^{11} nucleotide bases, while the protein data bank contains 10^4 proteins with 10^2–10^3 residues each. Roughly, we have ten thousand times as much genetic information as structural. If we come up with a structural alphabet with 255 characters or less, we could store all the protein strings, uncompressed, in about 10 megabytes, while the DNA sequence data would take almost a terabyte.

Test

Without some empirical test it would be tedious to argue that one structural alphabet is "better" than another or that one string similarity score "finds better matches" than another. To test the performance of different structural alphabets and string similarity metrics, I use the topology level of the CATH classification (page 97) as a gold standard of fold similarity (this is the same methodology followed by Kolodny *et al.* [38]): I selected m fold families of single domain proteins having n or more members to construct a set of mn proteins (randomly selecting n members from those families having more). For each structural alphabet and string metric pair, I ran an all vs. all search of protein structures.

I tested the following alphabets:

- Tung *et al.*'s [81] manual classification of fragments on a (κ, α) plot,* `http://3d-blast.life.nctu.edu.tw/download.php`
- Camproux *et al.*'s [11] structural alphabet used by SA-search [25], `http://bioserv.rpbs.jussieu.fr/cgi-bin/SA-Encoder`
- The protein's amino acid sequence
- A variety of alphabets constructed via normalized k-means of 5-residue fragments represented in ($\phi, \psi, \kappa, \alpha$, depth, volume, &c.)* vector spaces. I discuss the ϕ and ψ residue angles on page 2, the κ and α backbone angles are defined

* For any 5-residue segment, the $\kappa \in (0, \pi)$ angle of residue i is defined as the bond angle formed by the alpha carbons of residues $i-2, i, i+2$, and the $\alpha \in (-\pi, \pi)$ angle as the dihedral angle of residues $i-1, i, i+1, i+2$.

* In normalized k-means, one scales the range of the input variables so that one dimension doesn't dominate the others just because it is measured in small units.

above, depth is discussed on page 9, the volume is got from Voronoi tessellation (appendix A) around C_α (see appendix B for distributions). All of these measures are got from the 3rd residue of the 5-residue fragment.

In addition to the Levenshtein similarity scores using these alphabets, I computed the scores of full 3D structural alignments:

- The TM-score of Zhang *et al.* [91, 92] (see also page 94)
- The TM-score computed by the *Lovoalign* program of Martinez *et al.* [51]
- The STRUCTAL-score of Subbiah *et al.* [72], also computed by *Lovoalign*

I summarize the search results using ROC curves, a standard measure* of information retrieval systems. These are parametric plots of the rate of true positives against the rate of false positives found as a search cutoff is varied. Think of the search as ordering the entire corpus and drawing the ROC curve while scanning down from the best match to the worst: each correct match moves the curve up, and each false positive moves the curve to the right. If the search is good, you will see correct results before incorrect results, so the ROC curve will quickly jump up (a perfect test would have a curve that immediately shoots up to unity; ⌜), while a bad search (i.e. a random shuffle) scatters the correct results throughout the list, giving a 45° ROC line. Typically, ROC curves are summarized in a single scalar value by taking the area under the curve (AUC), which varies from $\frac{1}{2}$ to unity.* Also note that the accuracy,

$$\frac{\text{true positive} + \text{true negative}}{\text{all items}},$$

of a search is not a very useful measure, since (for any given query) most items in the corpus are irrelevant. If only one item in a corpus of a hundred was relevant to a given query, then the algorithm that never returned anything (i.e. marks everything as negative) will always have 99% accuracy.

For some cutoff score c, the true and false positive rates are given by

$$\begin{aligned}\text{true positive rate} &= \frac{\text{true positives above } c}{\text{all true positives in the corpus}} \\ &= \frac{T(c)}{nm(n-1)} \\ \text{false positive rate} &= \frac{\text{false positives above } c}{\text{all false positives in the corpus}} \\ &= \frac{F(c)}{n^2 m(m-1)},\end{aligned}$$

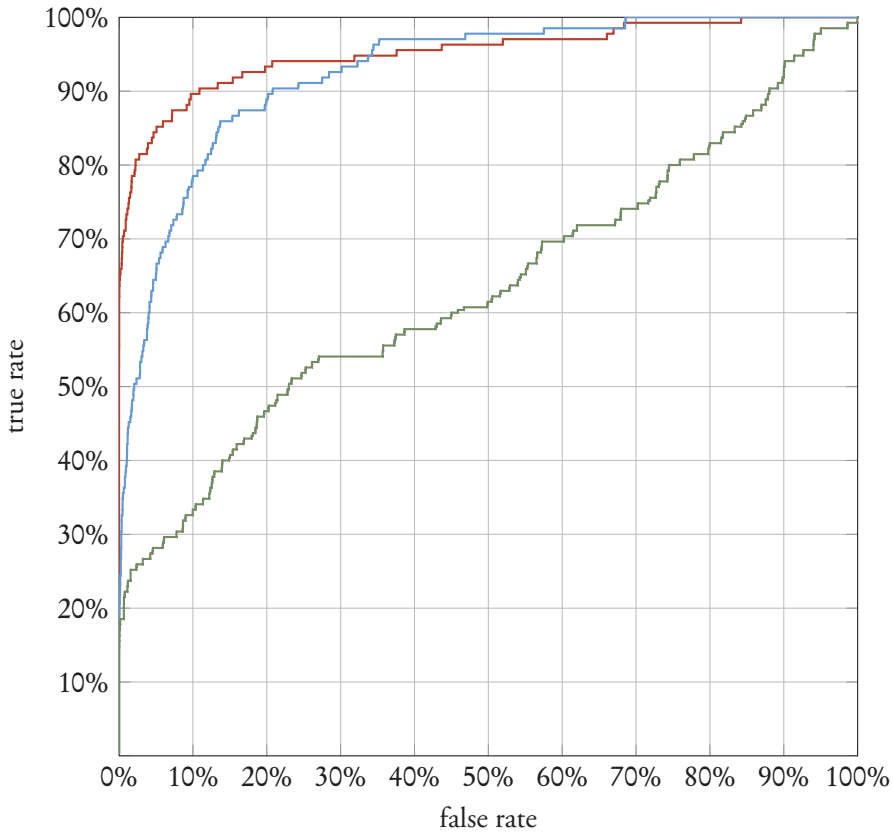
* ROC curves were developed with radar equipment in the mid-century–it is an acronym for "receiver operating characteristic".

* If you had a "terrible" procedure that had area less than $\frac{1}{2}$ under the curve (i.e. did *worse* than random) you could make a *good* search procedure out of it by simply turning its results upside down.

By construction there are n proteins in each of the m families, so each search corresponds to $(n-1)$ true positives, and there are nm searches. The number of false positives in the corpus are all the matches, $(nm-1)$ matches per search, minus the true positives: $nm(nm-1) - nm(n-1) = nm(nm-1-n+1) = n^2 m(m-1)$.

where $T(c)$ & $F(c)$ are the number of true and false matches above cutoff c, respectively.

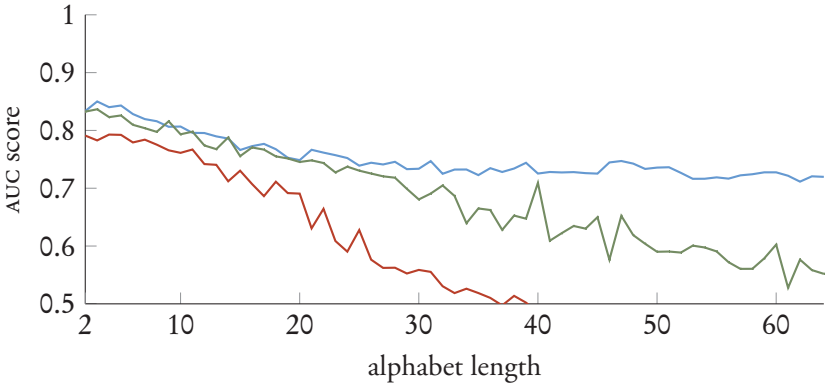
My search corpus consisted of m=45 CATH fold families having n=3 proteins each, for a total of 135 single domain proteins. Each search then has n-1=2 true positives (the other two proteins in that fold family) and 132 true negatives. This graph shows the results of three different searches; one using the TM-score (a full 3D alignment), one using a (ϕ, ψ, volume, depth) k-means generated structural alphabet (k=4), and one using the protein amino acid sequence (k=20 residues).



Three ROC curves based on TM-score, a (ϕ, ψ, volume, depth) k-means generated alphabet (k=4), and search using the protein amino acid sequence (k=20 amino acids). The AUCs are 0.95, 0.93, and 0.63, respectively. The TM-score is based on direct 3D structural alignment with the query protein structure, and scores for the two alphabets are given by Levenshtein string similarity (normalized edit distance) to the query protein's string. A higher curve is better, and a random shuffle of the data corresponds (on average) to the 45° line from (0%, 0%) to (100%, 100%).

The advantage of the general k-means formulation compared to hidden Markov models [11, 25] and backbone coördinate k-means [39, 40] is that one can cluster backbone fragments according to any combination of real-valued descriptors, not just the inter-C_α distances. Furthermore, k-means clustering and string similarity scores are easy to compute. The combination of structural descriptors and the number of letters k of an alphabet can be tuned empirically to give the best search performance, even on very large datasets. Each line in the plot below represents the 63 AUC scores of k=2..64 letter alphabets on an all-vs-all comparison of the proteins (63 × 135 × (135-1) = 1,139,670 protein-protein comparisons per line).

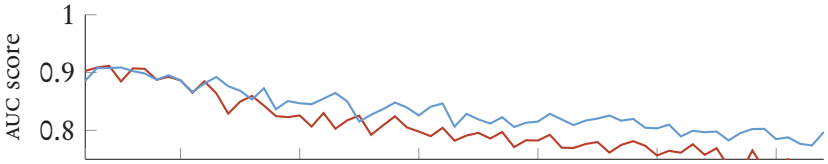
There are a variety of interesting trends in these scores. First, note that the Levenshtein edit distance AUC score decreases dramatically with the alphabet size:



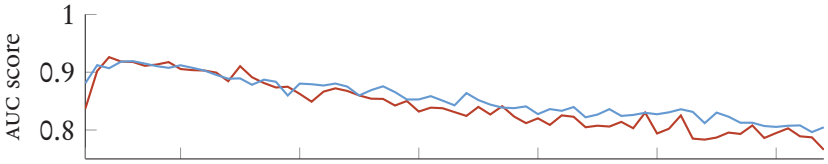
AUC scores of the depth, volume, and (depth, volume) alphabets and the Levenshtein score vs. alphabet size (2–64 letters). Note that the AUC score of the depth alphabet decreases faster with the length of the alphabet than the volume alphabet's score. This suggests that depthis more variable than residue Voronoi volume (consistent with the stability observations in appendix D, in particular the graph on page 95).

This is not surprising, given that this score does not preferentially weight certain substitutions–two fragments that would be assigned the same letter in a k=4 alphabet will likely have different letters in a k=32 alphabet, and a metric that treats all mismatches equally cannot account for the relative distances between clusters. Because this happens to all of the protein strings, though, one might expect the performance of rank-based retrieval to remain the same, but it appears that this is not the case.

Note that the five-residue κ and α backbone angles are more effective than the local ϕ, ψ angles, especially as the alphabet size k increases, and that this trend holds even after we include volume:

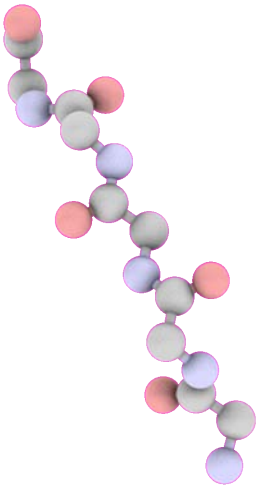


AUC scores of (ϕ, ψ) and (κ, α) alphabets vs. alphabet size k.



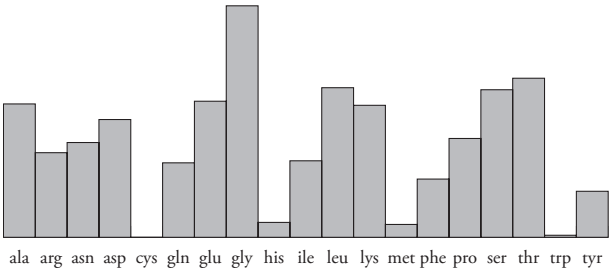
AUC scores of $(\phi, \psi,$ volume) and $(\kappa, \alpha,$ volume) alphabets vs. alphabet size k.

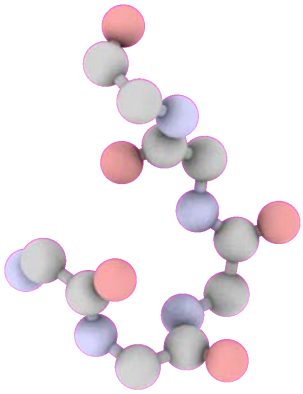
The k-means approach also allows us to see the prototypical representative of each structural letter–the closest fragment to the centroid. On the next page are the letters that make up the $(\kappa, \alpha,$ volume) k=4 alphabet, which has an AUC = 0.92. Note that the prototypes are simply two strands & two turns. The difference is that the first two prototypes occur near the surface of their respective proteins (unbounded Voronoi volumes are capped at 400 Å^3), while the third and fourth have volumes typical of deeply buried residues ($\approx$ 160 Å^3).



PDB: 2E01
seqnos: 237–241
κ = 130°
α = -128°
volume = 400 Å^3

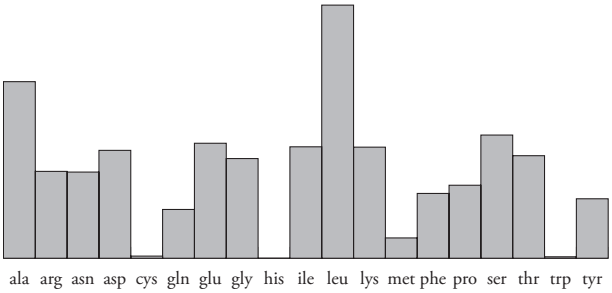
cluster 1:
3813 fragments

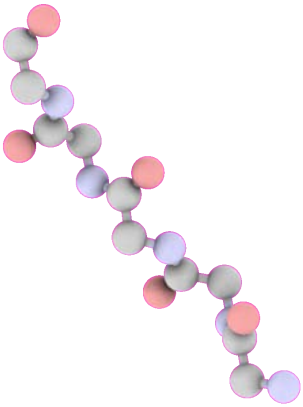




PDB: 1GBS
seqnos: 98–102
κ = 84°
α = 60°
volume = 397 Å^3

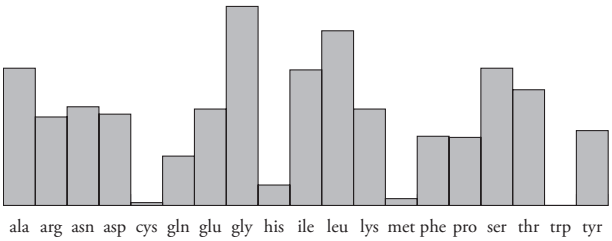
cluster 2:
7503 fragments

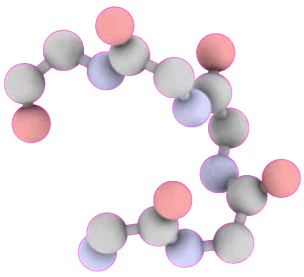




PDB: 1BHE
seqnos: 352–356
κ = 150°
α = -142°
volume = 165 Å^3

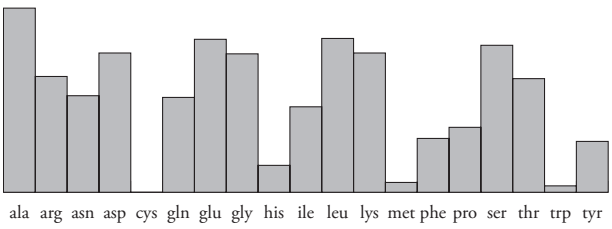
cluster 3:
4648 fragments





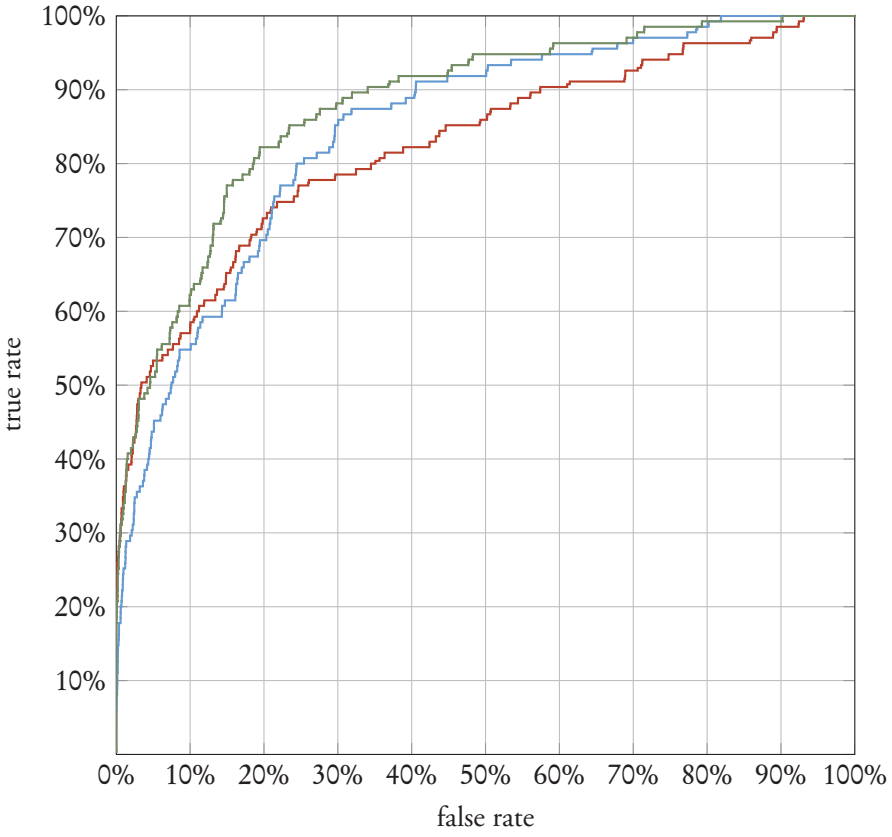
PDB: 1KOE
seqnos: 283–287
κ = 92°
α = 72°
volume = 172 Å^3

cluster 4:
6067 fragments



Comparison with existing alphabets

Comparing the performance of the (κ, α) alphabets with the manually derived alphabet of Tung *et al.* shows that for the same cardinality (k=20) the k-means generated alphabet performs slightly better (AUC = 0.85 vs. 0.83), and for k=4, much better: AUC = 0.92. Adding the dimension of residue volume for k=20 gives an AUC = 0.88 (not shown on plot).



Three ROC curves based on the 5-residue fragment κ, α angles: the manual classification of Tung *et al.* [81], the 20-letter (κ, α), and the 4-letter $(\kappa, \alpha$, volume) alphabets generated by k-means. Note that the areas under both k-means alphabet curves are greater than that of the manually classified (κ, α) classification, indicating greater retrieval performance (AUC = 0.85, 0.92, and 0.83).

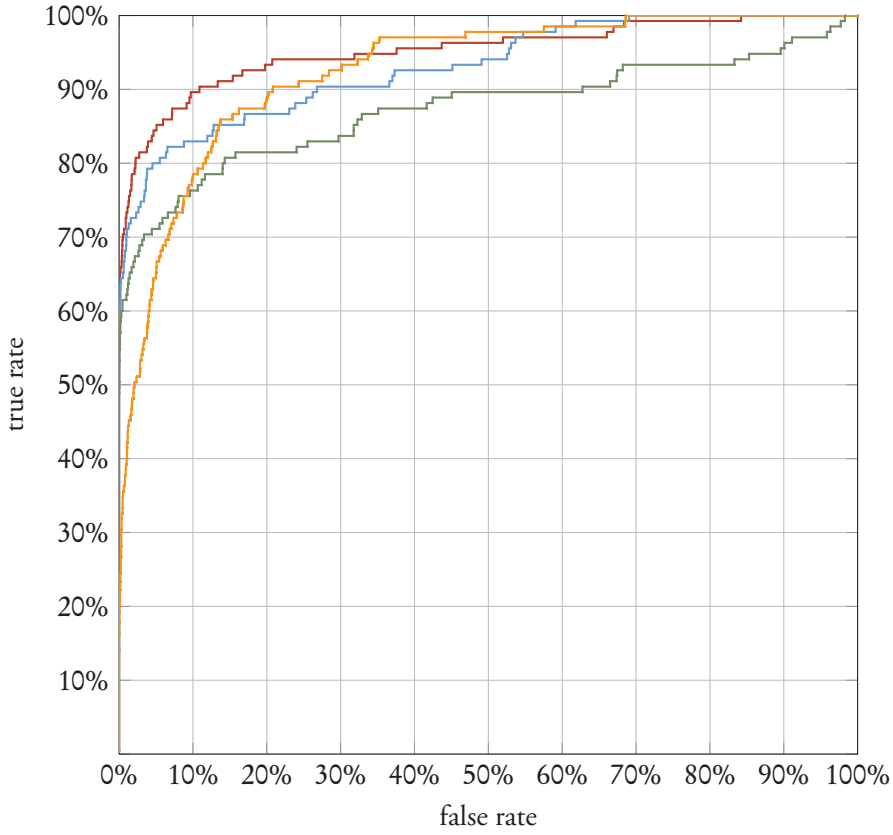
However, these comparisons may not be entirely fair, as the k-means alphabets are generated from the same proteins over which they search, while Tung's alphabet was chosen with regard to a larger set of proteins (including those with multiple-domains). But because Tung's alphabet is defined by a fixed partitioning of the (κ, α) plot, the individual letters of a given protein do not depend on other proteins in the corpus, so this performance *is* representative of that alphabet with respect to single domain proteins.

I also tested the hidden Markov model-generated alphabet of Camroux *et al.* [11], which gave an AUC score of 0.85 on a reduced test set.*

The improved performance of the k-means alphabets over the established κ, α [81] and hidden Markov model [11, 25] structural alphabets suggests that constructing a substitution penalty matrix to optimize the string similarity scores would yield large dividends. Already, several of these alphabets with the simple edit distance score have an AUC comparable to the TM and STRUCTAL scores, which are

* The group did not explicitly release their alphabet, either by encoding all of the protein structures in the protein data bank or providing a program for directly encoding `.pdb` files. I had to write a program (see page 84) to scrape their website, but only 35 fold families were fully represented (corresponding to 105 proteins, rather than the 145 proteins of 45 fold families as in all the other tests).

full 3D alignment-based searches (see graph below); for discussion of structural alignment, appendix D.1).



Three ROC curves based on TM-score, TM-score calculated by *Lovoalign*, the STRUCTAL score calculated by *Lovoalign*, and the k-means generated (ϕ, ψ, volume) k=4 alphabet, with AUC scores 0.95, 0.93, 0.87, and 0.93, respectively. Note that, as you would expect, all of these full 3D structure alignment programs perform quite well–only returning false positives after 60% of the true positives have been found, compared to 30% for the k-means generated alphabet.

fin.

In this chapter we created structural alphabets using k-means clustering of protein backbone fragments. Encoding proteins as strings via these fragments allowed us to use the Levenshtein string similarity metric to quickly perform fold-level searches. Many of these k-means alphabets have AUC scores that are competitive with far slower 3D structural alignment methods, even with an unoptimized string similarity metric. A clear future direction for this chapter of my thesis would be to further refine the search procedure by constructing substitution penalty matrices for the alphabets. Doing so would add negligible computational overhead, but very likely increase the search performance for larger alphabet sizes, allowing for more specific (and hence, more discriminating) letters.

As you have seen in this and the previous chapter, discussing protein structure can involve a substantial amount of data. Ideally, we can use this structure search procedure to explore the entire, ever-growing, set of solved protein structures (currently about 60,000) on the scale of residues and even individual atoms. In the next chapter, I discuss some of the challenges facing researchers in understanding these quantities of data. I offer some general principles for dealing with this influx of data and implement some proof-of-concept visualizations to display protein structural data.

Notes

Manning *et al.* have an excellent text on information retrieval;

> Christopher D. Manning, Prabhaka Raghavan, and Hinrich Schütze. *An Introduction to Information Retrieval*. Cambridge University Press. 2008. URL: `http://nlp.stanford.edu/IR-book/information-retrieval-book.html`

In particular, I found the discussion of search method evaluation (i.e. ROC curves) very helpful in writing this chapter. David MacKay's comprehensive textbook;

> David J.C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press. 2003. URL: `http://www.inference.phy.cam.ac.uk/mackay/itila/`

covers some data clustering schemes like k-means and classification machines like neural networks. The majority of his text discusses Bayesian probability theory and the design of codes for compression and transmission schemes. Both Manning and MacKay's books are available for free online.

I used Florian Frank's `amatch` rubygem* to compute the Levenshtein similarity metric between two strings. Florian also implemented several other common metrics, which I also used to calculate similarity scores:

* `http://flori.github.com/amatch/`

- Jaro [35]: edit distance based on letter matches, i, and transpositions, t, only

$$\text{Jaro}(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{3}\left(\frac{i}{|\mathbf{x}_1|} + \frac{i}{|\mathbf{x}_2|} + \frac{i-t}{i}\right),$$

  with the stipulation that characters can only match if they are not further than

$$\left\lfloor \frac{\max(|\mathbf{x}_1|, |\mathbf{x}_2|)}{2} \right\rfloor - 1$$

  from each other.

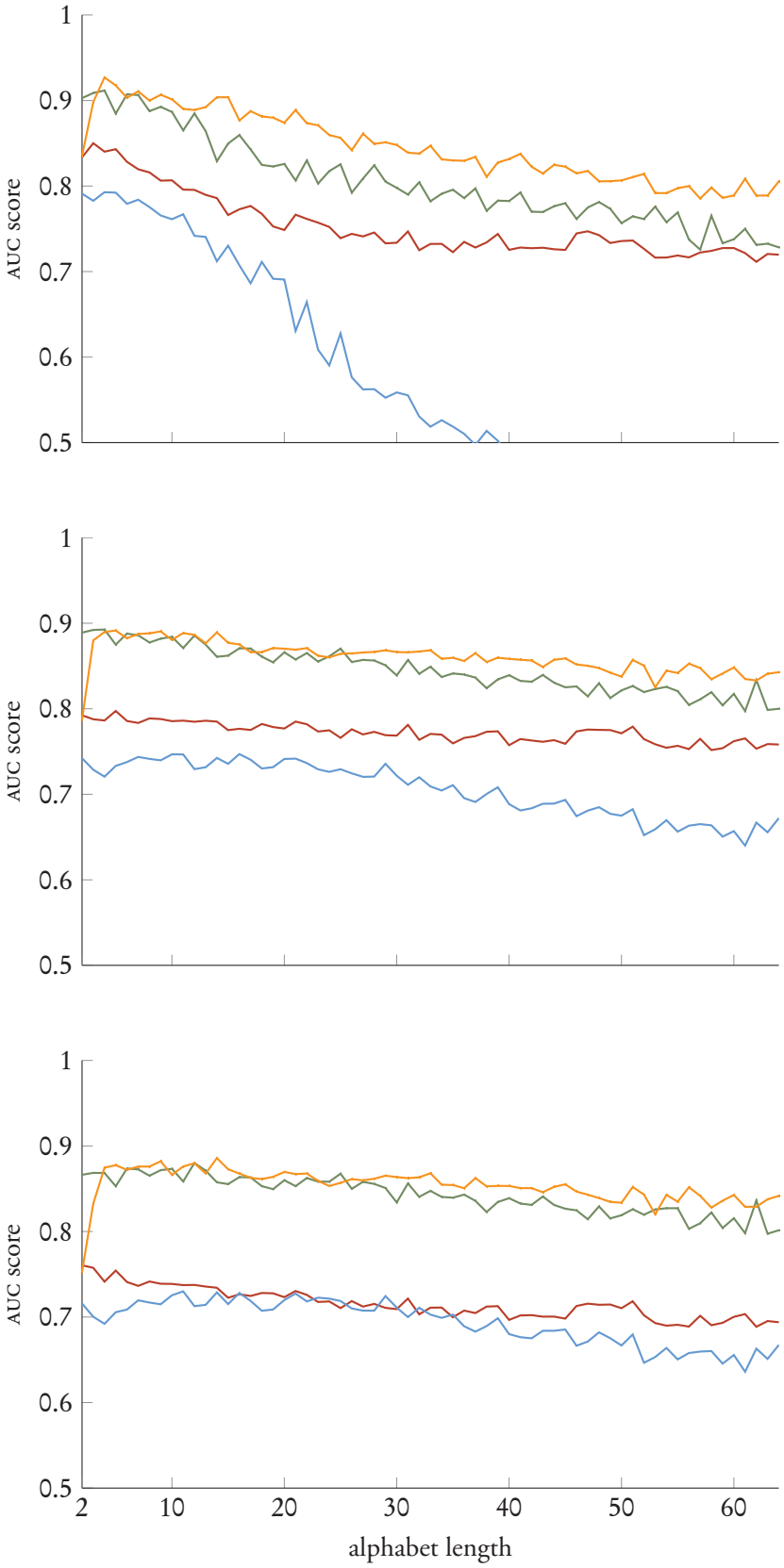
- Jaro-Winkler [88]: variant of the Jaro metric with additional weight p on common prefixes of length l,

$$\text{Jaro-Winkler}(\mathbf{x}_1, \mathbf{x}_2) = \text{Jaro}(\mathbf{x}_1, \mathbf{x}_2) + (l\, p\, (1 - \text{Jaro}(\mathbf{x}_1, \mathbf{x}_2)))$$

- longest common subsequence (normalized by the shorter string)

- longest common substring (normalized by the shorter string).

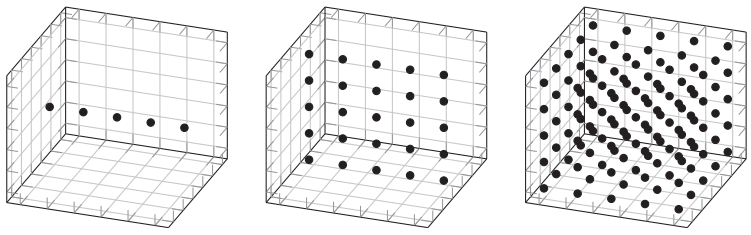
However, these metrics did not seem to provide improved retrieval performance over the Levenshtein metric. The longest common subsequence/substring metrics did not consistently perform better than random, but the Jaro & Jaro-Winkler metrics are notable for performing more reliably compared to the Levenshtein metric as the alphabet size increases (see figures on next page).



The AUC scores of volume, depth, (ϕ, ψ), and (ϕ, ψ, volume, depth) alphabets vs. alphabet size for Levenshtein (top), Jaro, and Jaro-Winkler string similarity metrics. Note that the Levenshtein metric performs better for smaller alphabets, but that the Jaro and Jaro-Winkler metrics do not degrade as quickly as the alphabet size increases.

In this chapter I've discussed global, fold-level structure comparison and search. Arguably though, looking at protein structure on a finer scale has more immediate consequences for human welfare; a universal physical understanding of protein structure would be of little use if we couldn't leverage it to develop specific drugs for specific proteins. The procedure I've outlined would be easily applicable to searching local structure if one had a robust, non-backbone-oriented structure-to-string encoding scheme.

On that scale, the details of specific amino acid residue types and their constituent atoms becomes pertinent The question is no longer "what other proteins have a helix here and a sheet there?" but instead, "what other histidine residues have two nearby serine hydroxyl groups?". These more specific questions require that we take into account more environmental features than just the backbone angles or residue depth. Mathematically, this means that we have to cluster structure fragments along additional dimensions. However, by increasing the dimension of a vector space we exponentially increase the number of data required to adequately sample it. In the business, this is affectionately known as "the curse of dimensionality". Roughly speaking, we can cover the unit interval in one dimension with 5 points but if we want the same coverage in two dimensions we need 25 points and in three dimensions we need 125:



Despite this hurdle, some groups are developing methodologies for searching protein structure along this fine-grained, residue local environment level: see Mooney *et al.* [53] for a nearest-neighbor based search and Yoon *et al.* [90] for a k-means based clustering search.

What was observed by us is the nature or matter of the Milky Way itself, which, with the aid of the spyglass, may be observed so well that all the disputes that for so many generations have vexed philosophers are destroyed by visible certainty, and we are liberated from wordy arguments.

—Galileo Galilei

Data exploration

4

The previous two chapters have introduced general techniques for two well-studied problems in structural bioinformatics: protein domain decomposition and protein structure search. We now discuss a different kind of problem that is not so often mentioned in the literature: data exploration. This distinct from *data analysis*–a term which suggests that we are simply plugging data into a theory or that we otherwise know what we are doing. Rather, it is the increasingly difficult problem of taking even a cursory, high-level view of the ever-expanding data available to researchers. The sheer quantity of these data must be overcome before one can formulate a hypothesis, much less rigorously test it. In this chapter I argue that the same technologies driving the proliferation of scientific data–inexpensive computing power and data storage–can also be leveraged to make knowledge out of that data. I discuss the construction of tools for data visualization and exploration, in particular those technologies that have proliferated with the Internet.

J.C.R. Licklider outlined two goals for the newly developed computer in his seminal 1960 paper, *Man-Computer Symbiosis* [47]:

1. to let computers facilitate formulative thinking as they now facilitate the solution of formulated problems, and
2. to enable men and computers to cooperate in making decisions and controlling complex situations without inflexible dependence on predetermined programs.

Licklider kept careful track of his research hours, and his description is unfortunately very familiar to the modern scientist, half a century later;

> About 85% of my thinking time was spent getting into a position to think, to make a decision, to learn something I needed to know. Much more time went into finding or obtaining information than into digesting it. Hours went into the plotting of graphs, and other hours into instructing an assistant how to plot. When the graphs were finished, the relations were obvious at once…No two experimenters had used the same definition or measure of speech-to-noise ratio. Several hours of calculating were required to get the data into comparable form. When they were in comparable form, it took only a few seconds to determine what I needed to know.

In this light, the Internet search engine Google is arguably the single most useful scientific tool of the past decade, despite never having been published in a scientific journal or its efficacy "proven" in some mathematical test* Google provides access to a wealth of general information that is more than enough for certain general topics, but for a growing number of scientific problems the search engine is of limited utility. Currently, there are attempts to provide general search engines for scientific data–see Stephen Wolfram's "computational knowledge engine", `http://wolframalpha.com/`–but I suspect that domain-specific corpora will nearly always require informed search procedures grounded in the respective disciplines.

* If the Google founders cared more about tenure, perhaps we would have a journal article instead of a website: "Conclusion: These results indicate that the `GOOGLE` algorithm clearly captures the essential essence ($p = 3.24 \times 10^{-6}$) of the inter-citations of distributed documents and represents a promising approach to efficient ranking of hypertextual information."

This is not to say that scientific problems are entirely disjoint from one another. Indeed, they share a number of characteristics, foremost of which is a deeply quantitative nature. This is both a blessing and a curse. Unlike Google, structural biochemists do not have to sift through a morass of uncurated, unannotated textual data. Scientific data derived from physical experiments–the conductivity of a metal, the crystal structure position of some atom, &c.–tend to be semantically well-defined. Instead, we have curated morasses of data in rigid schema; tables & tables & tables of data:

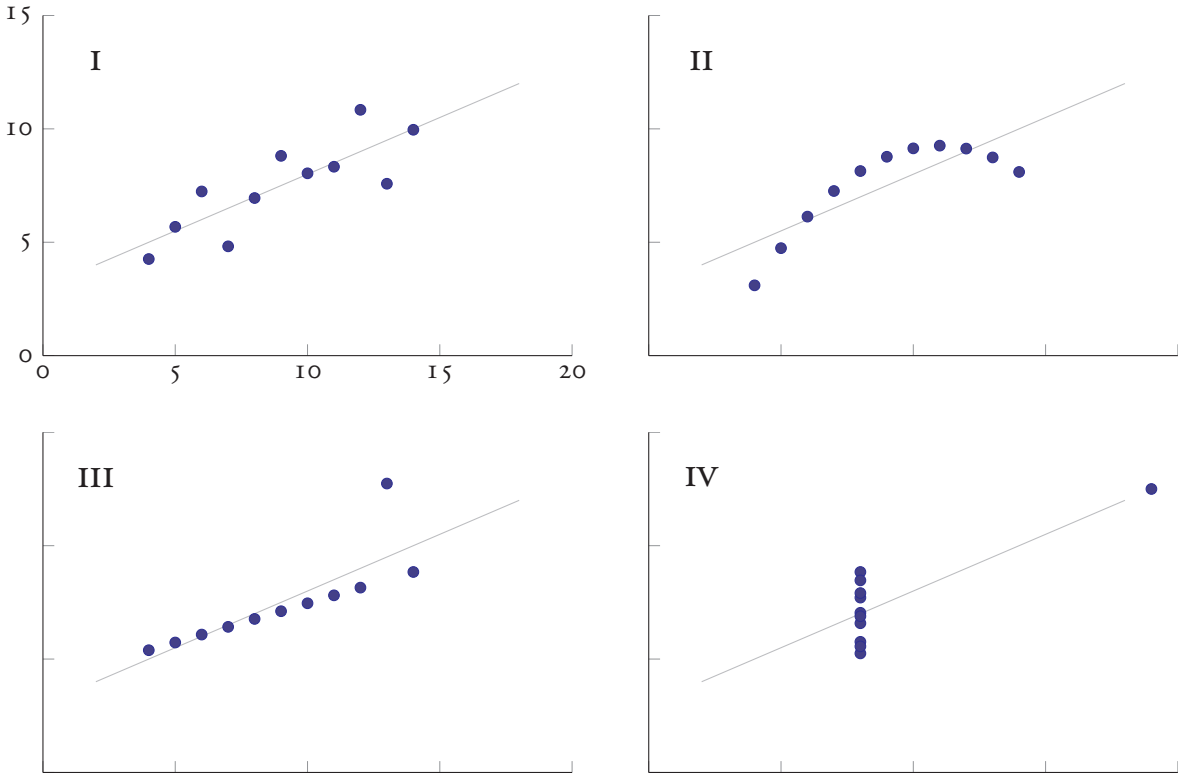
I		II		III		IV	
x	y	x	y	x	y	x	y
10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89

Anscombe's Quartet [1], four synthetic datasets having many of the same statistical properties:

- number of data: N = 11
- average X: $\overline{X}$ = 9.0
- average Y: $\overline{Y}$ = 7.5
- least squares regression line: $Y = 3 + 0.5X$
- standard error of regression slope: 0.118
- regression line fit: R^2 = 0.67
- regression sum of squares = 27.50
- student's t-test t = 4.24
- correlation coefficient = 0.82

Our computers and their programmers are as happy as clams: everything can be stored and indexed in relational databases, written to disk in identically-sized chunks, and queried using simple logical predicates: `SELECT x FROM data WHERE (x > 3.0) AND (y < -2.5)`. The problem is that it is hard to extract knowledge from data by looking at it directly; raw quantitative data cannot be *skimmed*. People are very good at skimming and understanding free-form text. Even if an author does not exactly specify her terminology and occasionally dangles participles, we can generally understand what she means. The flow of ideas that constitute written prose sustains enough context for the audience to fill in the gaps. In contrast, when the data speak we know exactly what they say: the sensor measured this and that, but it takes the entire scientific enterprise figure out what the hell they mean.

That we can easily extract certain subsets of the data–all `x > 3.0`–is a moot point if we never notice that `3.0` is somehow special. The challenge of data exploration is to show the data in a way that provides the same level of context as well-written prose. There are many different statistical tests and measures for summarizing data that give a kind of numerical context and indicate "special" values (see the summary of Anscombe's quartet, above), but they often pale in comparison to well crafted visual display:



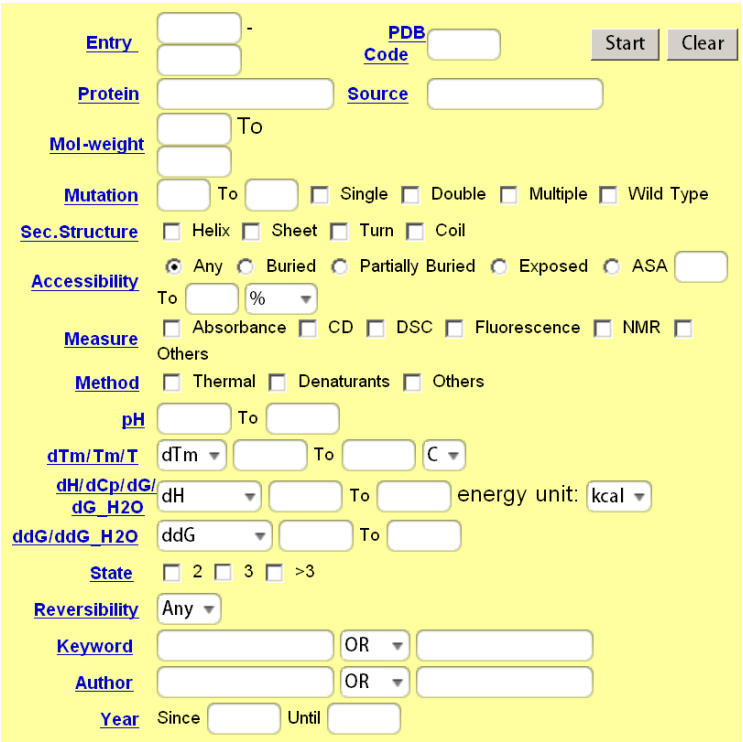
As Licklider said, "[w]hen the graphs were finished, the relations were obvious at once…". Though they are mathematically equivalent, we glean more knowledge from points on a plot than from rows on a table; we can see the data together. Although these four data sets share many of the same statistical properties (the common line of best fit is drawn in gray), upon plotting we can see that they are obviously very different. A linear regression with R^2=0.67 on data having a correlation coefficient of 0.82 might bring to mind plot I, but it could just as well describe a very strong linear correlation with a single outlier (plots III & IV) or a strong nonlinear correlation (plot II).

The effectiveness of scatter plots in conveying the relations between variables is not disputed. The utility of scatter plots is so well established within the scientific community that one never has to take the trouble of explaining the idea.* However, as we have seen in this thesis, it may take dozens of variables to describe a system of scientific interest (i.e. protein structure). This is especially

* "Each mark represents a sample from the experiment, and we move it to the *right* according to this first variable, then *up* by the value of this other variable…"

true when we do not have a theory that explains the phenomena; only then can we know which variables are relevant and which are extraneous.

The Internet provides an essentially boundless medium for the dissemination of quantitative information, but the current state of data within the scientific community is little beyond raw tabular display. The unlimited supply of pixels is typically employed (in black only) to display flat, lifeless numbers. Perhaps a bit of "interactivity" is mixed in by constructing a bijection between the data schema and HTML search boxes.



Screenshot of ProTherm [5], an online database of protein melting temperatures and typical example of a scientific database.

The underlying assumption of this design is that a scientific database will be used simply as a database, strictly for information retrieval. *This assumption is wrong.* It is akin to giving the new kid in town a phonebook in which to find new friends.* A researcher using a protein melting temperature database is more likely to ask, "what is the typical melting temperature of a protein of this size? (Is my protein different?)", "what range of temperatures can I reliably detect using this technique?", or "how does pH influence a protein's melting temperature?" long before they ask the only query this interface is designed to handle: "show me the melting temperatures of proteins with an $T \rightarrow A$ mutation at pH 7.5–8.0 as measured by NMR".*

This kind of interface provides no contextual clues about the data or valid queries to ask of it. What is the expected range of molecular weights for, say, a 200-residue protein? How many 200-residue proteins are in this database, anyway? Are melting temperatures typically recorded at physiological pH ≈ 7, or are most of the data

* The appropriate response is, of course, directions to the neighborhood bar and the next pickup softball game.

* Answer:

```
Mutation T-A
Measure :  NMR
pH : 7.5- 8.0
**end of results, 0 rows**
```

from studies about protein structure far outside biological ranges? To use this kind of interface, you must essentially *already know the answer*.

Bret Victor argues in *Magic Ink*[86] that software should be written to answer questions rather than to perform a strict set of functions;

> [T]he design of information software should be approached initially and primarily as a graphic design project.

Given the quantitative nature of scientific data and the power of modern computer systems for manipulating that data, it might seem puzzling that we should turn to graphic design. A graphic designer's product is, after all, static–once spilled, the ink on a page is inert and lifeless. And yet, there seem to be far more examples of intense, communicative, and informative pictures than scientific software or websites. We still capture and communicate our understanding of a topic via low-tech blackboards and textbook illustrations.

Perhaps the rigid constraints of print focus the designer's mind in a way that the potential interactivity and boundlessness of the virtual canvas does not. A cartographer must ask and answer:"who will be using my maps?" (hikers and drivers) and "what will they want to know?" (where are the trails? Can I park there? Are there hazards like snow or falling rocks? Bears?). Software designers, in contrast, seem to give in quite often to the temptation of their media and try to provide as much information for as many people as possible via countless dialog boxes, pop up windows, links, and buttons. Inevitably, this navigational and administrative clutter takes valuable real estate away from the data presentation itself, and the uninhibited desire to shove features onto an interface leads to a visual katzenjammer.

The primary challenge of information design is to provide an appropriate level of contextual detail. A hiking map is improved by the addition of height contours, but not by particulars of the water table. This challenge is acute for scientific data exploration because the audience and their goals are very broad. The appropriate level of contextual detail depends on the question at hand and the expertise of the user. Ideally, information applications should both fill users' imaginations with questions about the data and suggest routes for answering them. This is why graphical plots of data have proven so successful; it is impossible not to wonder about the two outliers in Anscombe's datasets III & IV.

Histogram explorer

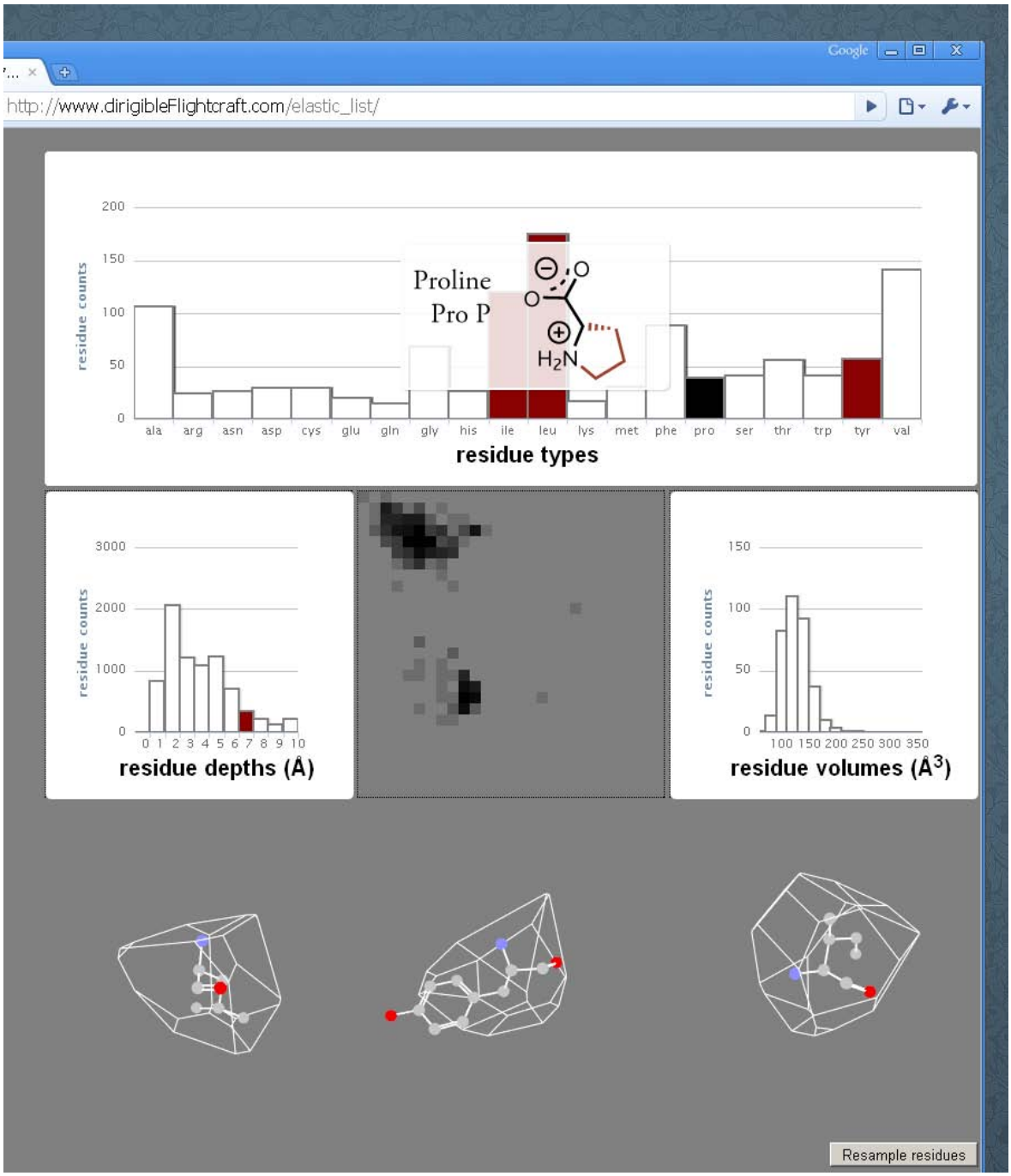
Scatter plots are one of the most commonly used information graphics in the scientific community. Typically, such plots are pierced with lines of best fit and confined to the "discussion of results" section of journal articles. Hence, scatter plots tend to suggest a causal relationship between two variables, with the *independent variable* on the abscissa and the *dependent variable* on the ordinate. If we are plotting two variables having different physical dimensions (length and time, say), it is unclear how to scale the respective axes of a scatter plot.* For the purpose of data exploration, we need a different kind of graphical display.

A simple histogram* fits the bill. A static histogram, though, is a terrible waste of space; each bar represents only a single scalar value. On a website this empty space can be employed as a navigational device. I use this idea to construct a general tool for exploring high-dimensional data. Each dimension of the data is displayed as a histogram. The example on the facing page depicts the amino acid type, depth, volume, and Ramachandran plots of residues drawn from proteins used in this thesis. When a bin is selected, the other histograms adjust to display that data. A selection on any of the histograms hides the data outside of that slice from the other histograms. Note that, unlike free-form query boxes, it is impossible to formulate a query that yields no results.

A histogram does not show all of the available data, just a broad statistical overview. To compensate, there are three windows under the display that show individual examples from the current query; in this case, amino acid residues and their Voronoi cells (see page 86 for some of the rendering code). Note that this website is interactive, but only minimally: mouse-overs provide contextual information (the chemical structure of an amino acid residue and an exact count of matches) and the query is refined or broadened by clicking on bins. This minimalism is deliberate; interactivity implies an underlying *state*, and, as Victor puts it, "stateful things can be *broken*." The only state maintained in this application is the current query, which is always visible in the form of colored histogram bars.

* A general rule of thumb is to chose an aspect ratio that banks the data to 45°; see Cleveland [14] for a discussion and mathematical procedure.

* A histogram is a series of adjacent rectangular bars that represent one dimension of a data set. The bars typically have the same width (though they needn't), with height given by the number of samples falling in that interval. One disadvantage of the histogram is that these bins are discontinuous. This problem becomes acute when selecting the interval length–picking it too small distorts the underlying distribution (i.e. more samples are needed for smoothness), but picking it too large may hide interesting features in the data. In the general histogram problem, kernel density estimation, one places small (typically Gaussian) bumps at each datum to form a smooth curve.



Screenshot of the histogram explorer showing isoleucine, leucine, and tyrosine residues at 6–7 Å depth. Note the mouse over pop up on proline ("information at the point of need") and the displayed examples (3D structures of the residues and their Voronoi cells) from the current query. As you would expect at this depth, the histogram of residue types shows a greater proportion of hydrophobic residues, and the volume histogram shows a tight spread of the residues' Voronoi volumes. I was inspired by Moritz Stefaner's *elastic lists* project: `http://moritz.stefaner.eu/projects/elastic-lists/`.

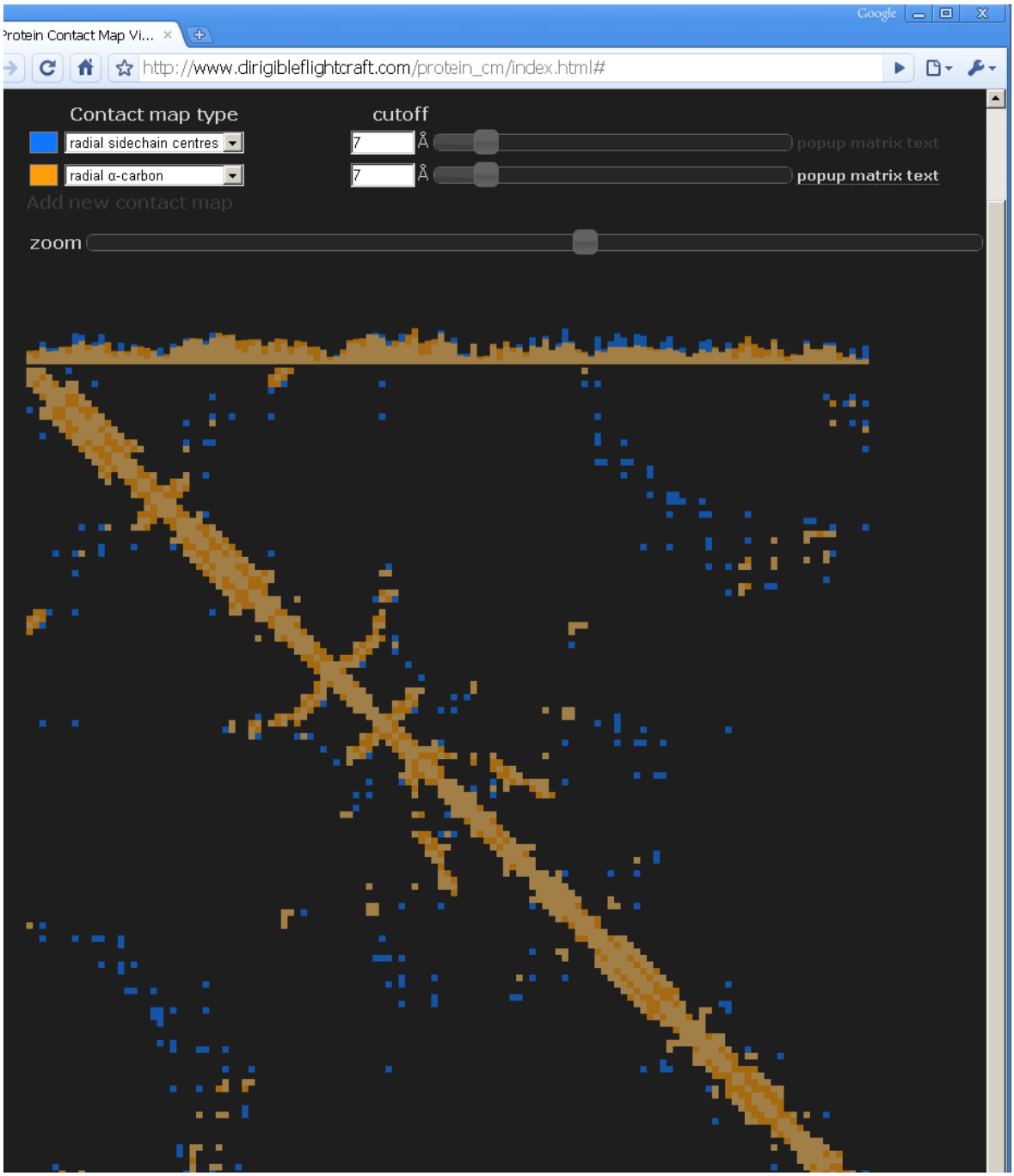
Contact map comparator

The histogram explorer is a general method for getting the lay of a large, high-dimensional dataset, but many of the same principles prove useful in an application for a much more specific problem: comparing the multiple definitions of residue–residue contact (discussed on page 4). It is not immediately clear how the cutoff value affects radial contact maps, the difference between the C_α and residue sidechain centroid contact maps, and if any one definition is superior to the others for a given application.

The contact map comparator shows transparent contact maps with their principal eigenvektors. One can vary the cutoff sliders and see instantaneously the effect on the individual contacts and the principal eigenvektors.*

Note that the users of this kind of specialized software tend to not only be knowledgeable in the field, but also have active research interests within it. Thus, scientific applications should provide a way to get at the data in question. In this example, contact maps can be exported directly from the browser as a text pop-up or download for additional analysis.

* It is a testament to the speed of modern computers that we can near-instantaneously compute the principal eigenvektor of a 300 × 300 matrix *in a web browser*.



Screenshot of the contact map comparator showing two different kinds of contact maps at the same cutoff–note that it is clear the radial sidechain centre map shows many more contacts than the radial C_α map. One can simultaneously display several contact maps (with user-defined colors) and interactively adjust their cutoff values, viewing both the map itself and its eigenvektor (along top). Each contact map has a link to download the matrix for analysis using *Mathematica* or *Matlab*. The only variables of state are the contact map types, their cutoffs, and colors, which (as in the histogram explorer) are always on display.

fin.

In this section we have discussed some general principles for constructing informative scientific visualizations. We have seen these principles implemented in two web-based programs; a general histogram-based data exploration tool, and the more specific contact map comparator. Perhaps it is a mere reflection of my mathematical naïevity, but over the course of my studies I have found data visualization to be far more aesthetically rewarding than even the most concise proofs or statistical regressions. Unfortunately, the subject is too often dismissed as artistic folly rather than embraced as scientific necessity. In some ways, the advent of computer technology has further displaced creative information design from working scientists, as there are far more practical difficulties in doodling across an electronic document than on physical correspondence. I discuss some solutions to these practical difficulties in appendix C.

Notes

The most prominent advocate of information design in the recent era is Edward Tufte, a statistician and political studies professor. His seminal work, *the visual display of quantitative information*, outlines general principles, though all four of his texts are well worth studying;

> Edward Tufte. *The visual display of quantitative information*. Graphic Press. 2001. call no: QA276.3.T83
>
> Edward Tufte. *Envisioning information*. Graphic Press. 1990. call no: P93.5.T54
>
> Edward Tufte. *Visual explanations: images and quantities, evidence and narrative*. Graphic Press. 1997. call no: QA276.3.T84
>
> Edward Tufte. *Beautiful evidence*. Graphic Press. 2006. call no: P93.5.T84

and his website* serves as a *de facto* hub for practical advice.

* `edwardtufte.com/bboard/`

For discussion of visual perception in the context of cartography, see

> Edward Imhof. *Cartographic relief presentation*. ERSI Press. 2007

and for more technical, statistical aspects of data analysis see John Tukey's* and William Cleveland's classic texts;

> John W. Tukey. *Exploratory data analysis*. Addison-Wesley. 1977. call no: HA29.T783

* Tukey was quite prolific, he coined "bit" as a contraction of "binary digit" and came up with the reduction underlying the Cooley-Tukey fast Fourier transform algorithm during a meeting President Kennedy's scientific advisory committee on the subject of Soviet nuclear test detection [65].

William S. Cleveland. *The elements of graphing data*. AT&T Bell Laboratories. 1994. call no: QA90.C54

The principles of good information design apply to any medium, but for specific advice on designing websites and applications, see Jakob Nielsen's writing on usability* and the Bret Victor's previously discussed work [86].

* `useit.com`

fin.

5

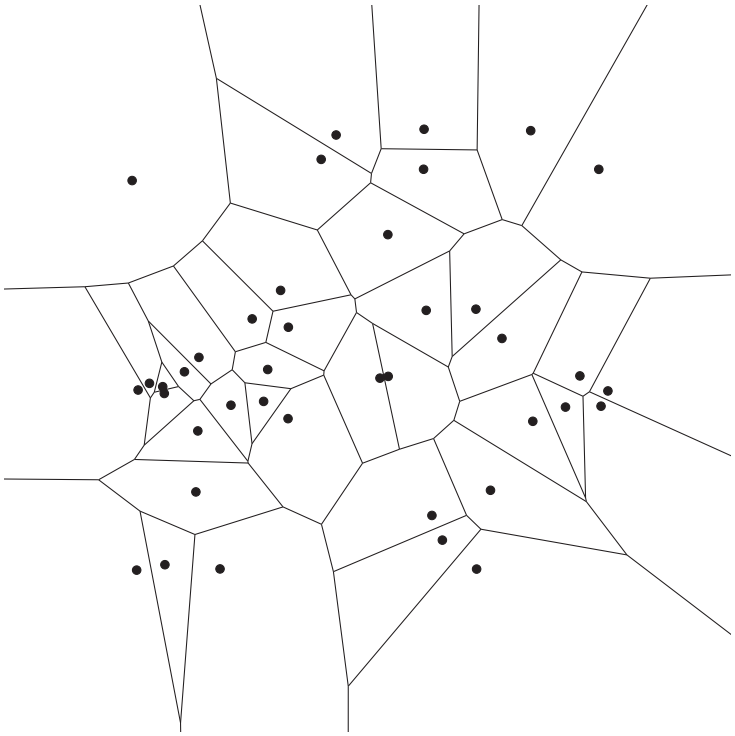
Over the past four chapters, we saw various mathematical measures derived from protein structures; the backbone angles, the contact map and distance matrix representations, their prinicpal eigenvektors, and the Voronoi tessellation. We then used k-means and spectral clustering with these measures to attack two structual biochemistry problems; the automated identification of domains, and the fast search of protein structures.

The recent surge of technical advances in molecular biology and biochemistry has led to a landslide of biological data. Practitioners have been forced to take crash courses in computer science and enigneering just to keep up with the quantity of data. This situation is hardly unique to biology; the rapidly falling costs of storing and sharing data has influnced nearly all scientific and social enterprises. The seemingly omnipresent Google search engine is barely 12 years old, and the Internet remains a fertile resource. Just as I have examined thousands of freely avaliable protein structures on my personal computer, so too can amateur astronomoers, economists, linguists, and engineers further their own studies. Although the particulars of this thesis relate to protein structure, the methods we have discussed are quite general. I hope that you will find some of them useful in your own research.

The Voronoi tessellation

A

For a set of n points, the Voronoi tessellation gives a collection of vertexes and edges that partitions space into n convex polyhedra (cells):



The Voronoi tessellation of a random set of points in 2D. Note that some of the cells have unbounded volumes–their edges are connected to the "infinite vertex" (not shown).

Since these cells are uniquely defined for a given set of points (known as Voronoi sites), in some sense they yield a natural set of measures for the sites; including the familiar measures volume and surface area (or, in 2 dimensions, area and perimeter). There are other data one can derive about the shape of the cell: the number of faces, the deviation of the cell centroid from the site position, &c, and also an unambiguous proximity metric between sites: two sites are neighbors if their Voronoi cells share a face.

The Voronoi tessellation is an old and intuitive idea. Its current name is due to a paper by Georgy Voronoi in 1908, though it can also be found in Descartes's treatment of cosmic fragmentation in Part III of *Principia Philosophi* (1644), in solid state physics ("Wigner–Seitz cells"), and in biology ("plant polygons"). See de Berg *et al.* [6], chapter 7 for a discussion.

Let us take a brief digression to satisfy the mathematicians. The Voronoi cell $V(x_i)$ is defined to be the set of all points at least as close to the site x_i as to any other site x;

$$V(x_i) \equiv \left\{ \mathbf{p} : |\mathbf{x}_i - \mathbf{p}| \leq |\mathbf{x}_j - \mathbf{p}| \ \forall j \neq i \right\}.$$

For two sites $\mathbf{x}_i$ and $\mathbf{x}_j$, let us call their perpendicular bisector B_{ij}. Note that every point $\mathbf{p} \in B_{ij}$ is equidistant from $\mathbf{x}_i$ & $\mathbf{x}_j$.* Let $H(\mathbf{x}_i, \mathbf{x}_j)$ be the closed halfspace with boundary B_{ij} containing

* You can prove this by constructing the triangle $\mathbf{x}_i, \mathbf{x}_j, \mathbf{p}$ and using Euclid's side-angle-side theorem.

$\mathbf{x}_i$. Then $H\left(\mathbf{x}_i, \mathbf{x}_j\right)$ can be considered all of those points closer to $\mathbf{x}_i$ than to $\mathbf{x}_j$. Since the Voronoi cell $V\left(\mathbf{x}_i\right)$ consists of those points at least as close to the site $\mathbf{x}_i$ as to any other, we can write it as

$$V\left(\mathbf{x}_i\right) = \bigcap_{j \neq i} H\left(\mathbf{x}_i, \mathbf{x}_j\right).$$

Mathematicians sometimes use the intersection of halfspaces to define the notion of convexity for a polytope. For our purposes, this result means that all Voronoi cells are convex–this means you can create it by snapping a rubber band (sheet in 3 dimensions) around the cell's vertexes, and that a line between any two points in the cell never leaves the cell (there are no mountains or crevasses). Finally, to make things friendlier, we place a single Voronoi vertex at infinity and attach all of the half-infinite edges to this vertex.

Computation

The Voronoi diagram has deep connections with other topics in computational geometry, and it can be calculated by many existing software packages. The Computational Geometry Algorithms Library, CGAL,* is a very complete C++ library that can do a lot of exact calculations in n-dimensional worlds that I cannot imagine. The 3D Voronoi diagrams are computed as the dual of the Delaunay triangulation. This library is very well documented and maintained, but I decided against using it because it is very complex and more general than I required.

* cgal.org

QHull calculates the convex hull, Voronoi diagram, and Delauny triangulation for points in up to seven dimensions. In three dimensions, it has a runtime of $O(n \log v)$, with n input points and v convex hull vertexes. This is certainly fast enough for the $n = 300$ residues or $n = 3000$ atoms I consider in this thesis, but from what I could gather the output is simply the Voronoi vertexes and edges–I need the volume, areas, centroids &c.

The library I ended up using for the Voronoi calculation was Chris Rycroft's Voro++ [67], a program he created in the course of his PhD thesis [66] on granular flow. In his words,

> Voro++ makes use of an alternative method of computation, and it aims to be most suited to research problems in materials science, physics, and engineering that frequently involve large systems of particles, often with non-standard boundary conditions.

The "alternative method" is a cell-by-cell calculation, rather than one on the entire set of points at once. The cells are typically what is of interest to non-geometers, and this architecture also allows for parallelization since Voronoi cells are defined only by their neighbors. Voro++ handles all calculations in 3D only.

There is no separate treatment for an "infinite vertex"–users must specify a bounding region (which can be an arbitrary shape). The algorithm for calculating the Voronoi cell is straightforward: for each particle, the cell is constructed by first initializing it to be a large rectangular box and then repeatedly cutting it with planes that correspond to the perpendicular bisectors between that particle and its neighbors. The complete algorithm* is given on in Rycroft's thesis [66], pg 82;

1. For some particle of diameter d at $\mathbf{x}_0$, set its Voronoi cell to be the same size as the global container.
2. Set $s_{\text{test}} = d/2$
3. Find all particles $\mathbf{x}'$ within the spherical shell $s_{\text{test}} < |\mathbf{x}_0 - \mathbf{x}'| < s_{\text{test}} + d$ around the particle at $\mathbf{x}_0$.
4. For each of these neighbor particles $\mathbf{x}'$, cut $\mathbf{x}_0$'s Voronoi cell with the plane that perpendicularly bisects the vector between $\mathbf{x}_0$ and $\mathbf{x}'$.
5. Let R be the distance of the Voronoi vertex furthest from $\mathbf{x}_0$; if $2R > s_{\text{test}}$, add d to s_{test} and repeat from 3.

* For Voronoi cells that are very far from spherical, this algorithm is not very efficient–it might take many iterations for the test spherical shell to extend far enough for a particularly spiky Voronoi cell. Rycroft has rewritten much of the program since his thesis, and the cell calculation now switches to a different plane-cutting algorithm after some number of iterations of the shell algorithm described at left (Rycroft, personal communication).

For efficiency, Rycroft's algorithm simply tests the particles nearest $\mathbf{x}_0$, because those are the ones most likely to create facets on the cell. Once all of the particles within a distance $2R$ from $\mathbf{x}_0$ have chopped away at our cell, we are done–the cutting planes of particles greater than $2R$ away must be more than R away from our cell, leaving it unscathed. Note that specifying the particle diameter allows one to reduce the search space for neighboring particles, since particles cannot overlap. I simply chose $d = 1$, since the van der Waals radius of the smallest atom in proteins, hydrogen, is greater than that: 1.2 Å.

The greatest feature of Voro++, though, is its interface–Rycroft obviously knows how to engineer a concise, eminently usable software library. In particular, Voro++ comes with a `sprintf` type function for printing the data of a given cell. Since I am studying proteins (rather than packed materials in some container), I don't need to worry about the boundary conditions* and this function gives all the data I need;

* I hard code the boundary as a simple cube with faces at ±1000 Å, which gives enough room for all the protein structures discussed in this thesis.

```
1  //compile with g++ -Wall -ansi -pedantic -O3 -Ivoro++/src -o
       voropp3d.o -c voropp3d.cpp && g++ voropp3d.o -o voropp3d
   #include "voro++.cc"
3
   // Set up constants for the container geometry
5  const fpoint x_min=-1000,x_max=1000;
   const fpoint y_min=-1000,y_max=1000;
7  const fpoint z_min=-1000,z_max=1000;

9  // Set up the number of blocks that the container is divided into.
   const int n_x=1,n_y=1,n_z=1;
11
```

```
int main() {
        // Create a container with the geometry given above, and make it
             non-periodic in each of the three coordinates. Allocate space
              for 25000 points (atoms).
        container con(x_min,x_max,y_min,y_max,z_min,z_max,
             n_x,n_y,n_z,
                     false,false,false,25000);

        con.import(); //with no argument, this reads from stdin

        //for each cell, print the serial# | voronoi vertexes | vertexes that
             make up each face | id of face neighbors | areas of faces |
             volume | centroid position | relative centroid position
        con.print_all_custom("%i|%P|%t|%n|%f|%v|%C|%c");

}
```

This program takes an indexed set of x, y, z coördinates;

```
1 1.34 2.97 -4.01
2 -0.34 9.53 -2.99
3 3.83 5.02 1.37
```

⋮

and returns the Voronoi vertexes, faces, areas, &c. in an easy to parse list.

```
1|(-1000,-1000,-1000) (1000,-1000,-1000) (-1000,-95.032,-1000) (1000,417.163,-1000)
    (-1000,-1000,845.265) (1000,-1000,-80.3857) (-1000,-343.024,594.93)
    (1000,354.42,-596.475)|(0,1,3,2) (0,4,5,1) (0,2,6,4) (1,5,7,3) (2,3,7,6) (4,6,7,5)|-5 -3
    -1 -2 2 3|2.32213e+06 908702 2.76488e+06 2.08622e+06 1.32783e+06 2.34515e+06|2.44583e
    +09|-57.121 -500.942 -404.122|-58.461 -503.912 -400.112
2|(-1000,-95.032,-1000) (1000,417.163,-1000) (-1000,1000,-1000) (1000,1000,-1000)
    (1000,354.42,-596.475) (1000,1000,71.3153) (-1000,1000,1000) (29.0012,1000,1000)
    (-1000,48.5732,1000) (-1000,-343.024,594.93)|(0,9,4,1) (0,2,6,8,9) (0,1,3,2) (1,4,5,3)
    (2,3,5,7,6) (4,9,8,7,5) (6,7,8)|1 -1 -5 -2 -4 3 -6|2.08622e+06 1.67787e+06 463405 2.40897
    e+06 3.54912e+06 2.58989e+06 489510|2.85591e+09|-239.961 535.965 -153.364|-239.621
    526.435 -150.374
3|(-1000,-1000,845.265) (1000,-1000,-80.3857) (-1000,48.5732,1000) (-1000,-343.024,594.93)
    (-1000,-1000,1000) (1000,-1000,1000) (29.0012,1000,1000) (1000,1000,1000)
    (1000,1000,71.3153) (1000,354.42,-596.475)|(0,1,9,3) (0,4,5,1) (0,3,2,4) (1,5,7,8,9)
    (2,3,9,8,6) (2,6,7,5,4) (6,8,7)|1 -3 -1 -2 2 -6 -4|2.34515e+06 2.62789e+06 1.23512e+06
    2.58989e+06 263201 3.51049e+06 450876|2.69825e+09|305.759 -113.202 528.641|301.929
    -118.222 527.271
```

⋮

I use this simple wrapper program to Voro++ to compute all of the volumes described in this thesis–I simply feed it the coördinates of C_α atoms or sidechain centroids.

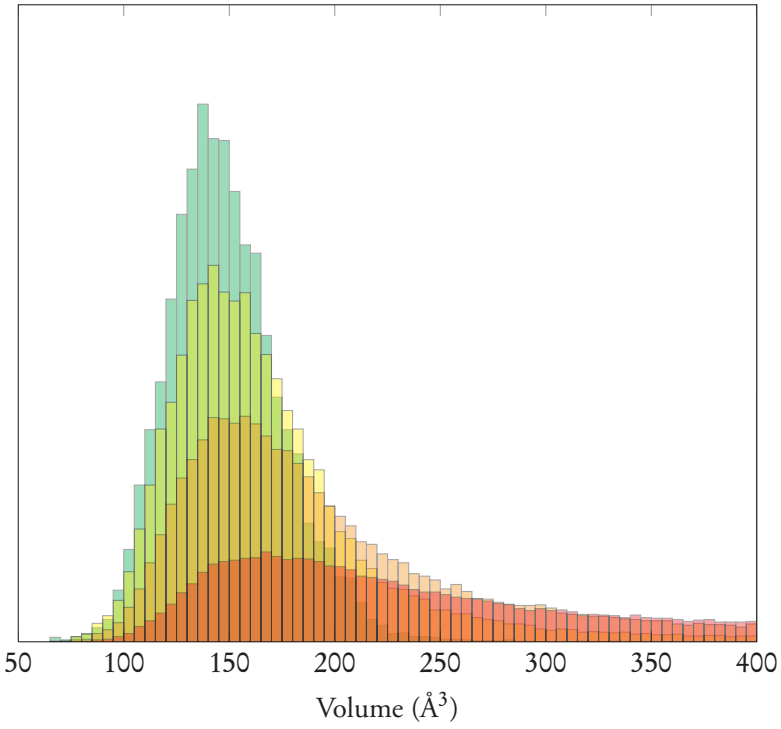
Voronoi cell statistics

B

The Voronoi procedure (described in appendix A uniquely assigns each residue (or atom) a convex polyhedron. We can derive several measures from this cell to describe a residue in terms of its local environment. The most physically immediate dimension is the cell volume.

It is well known that protein cores are tightly packed. Indeed, Pontius *et al.* [61] suggested using deviations from average residue volumes derived from the Voronoi tessellation of high resolution crystal structures as a structure quality measure. This packing efficiency is immediately apparent if we view the C_α tessellation cell volumes of residues at different depths;

Poupon [63] gives an excellent review of protenomic applications of the Voronoi tessellation and related geometric ideas.



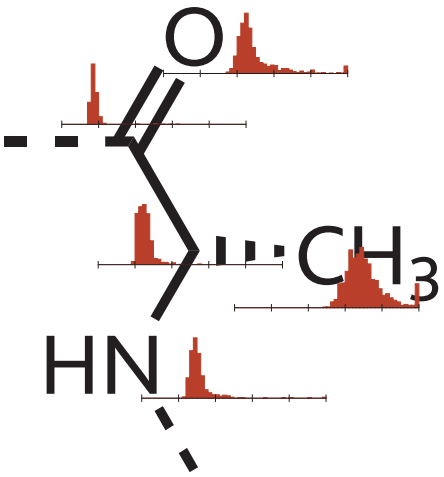
Histograms of C_α tessellation volumes for residues at depths 2–3, 3–4, 4–5, 7–8 (n = 205,887; 79,463; 49,221; and 11,872 residues, respectively). Unbounded volumes (i.e. greater than 400 Å) not shown. Note that both the average and the variance of the distribution of volumes decreases with residue depth.

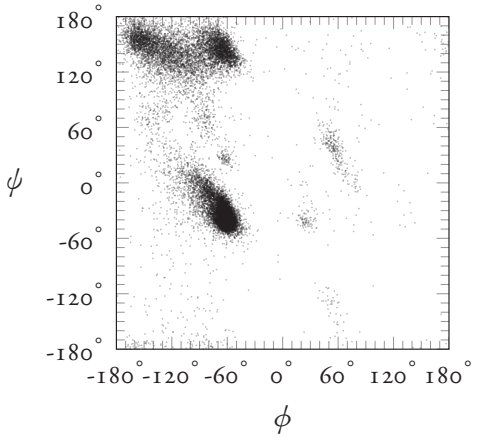
The following pages show the distribution of volumes for each atom of the 20 amino acids. These volumes are derived from atom-wise Voronoi tessellation of residues with C_α deeper than 4 Å. The distributions are shown from 0–50 $Å^3$ in 1 $Å^3$ bins.* Atoms having volumes greater than 50 $Å^3$ counted in the final bin. In total, 2,048,634 atoms are represented from the proteins listed in appendix H. Ramachandran plots are also shown for each amino acid.

* For reference, the *van der Waals volumes* (see page 9) for the organic elements are; hydrogen 6.03 $Å^3$, carbon 12.11 $Å^3$, nitrogen 10.06 $Å^3$, oxygen 9.68 $Å^3$.

Alanine

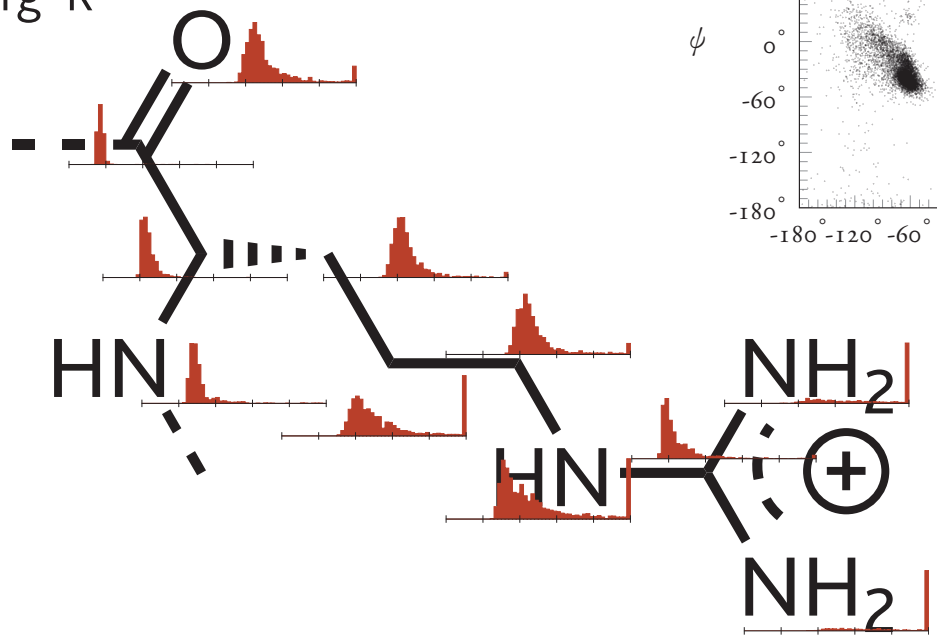
Ala A

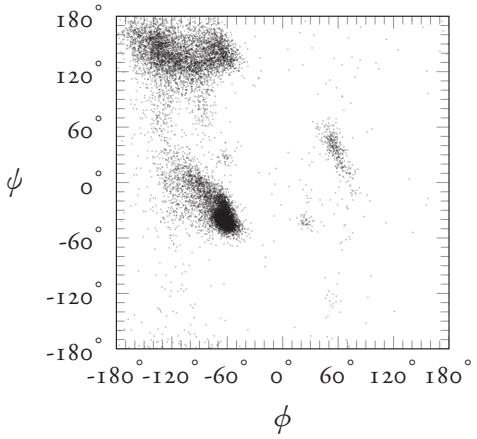




Arginine

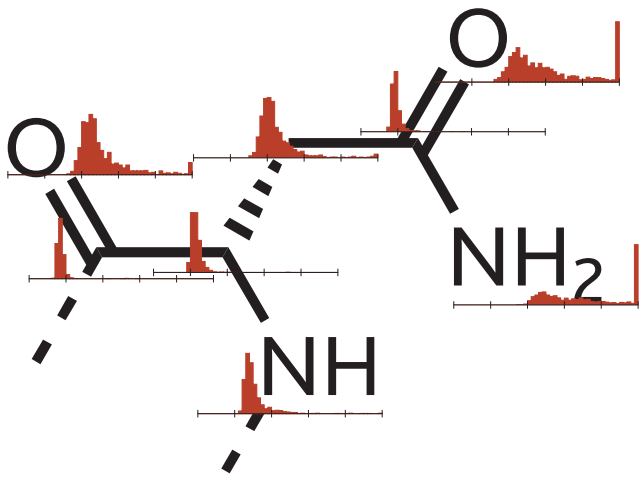
Arg R

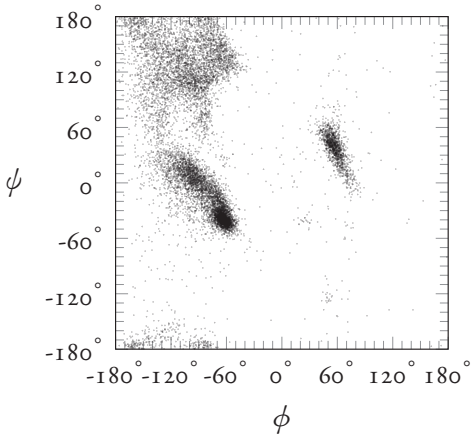




Asparagine

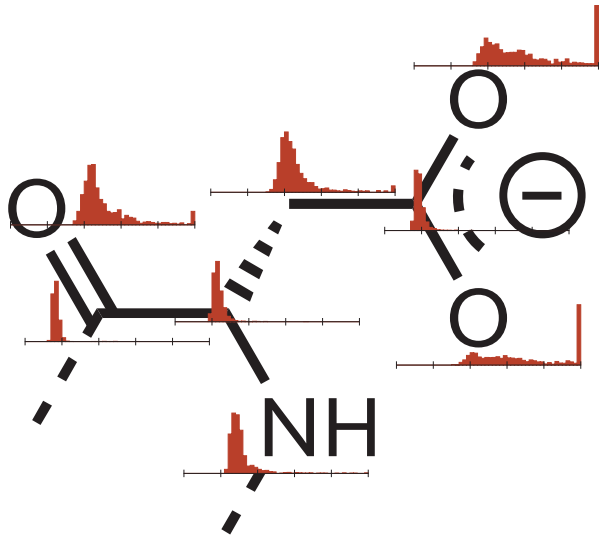
Asn N

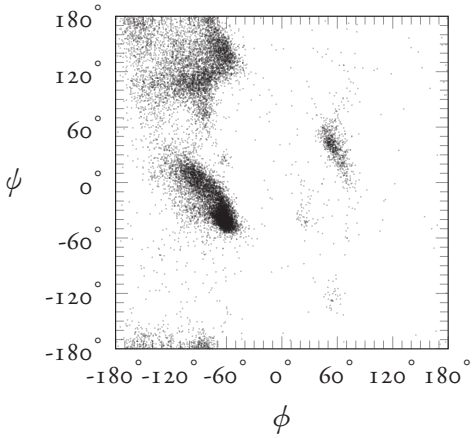




Aspartate

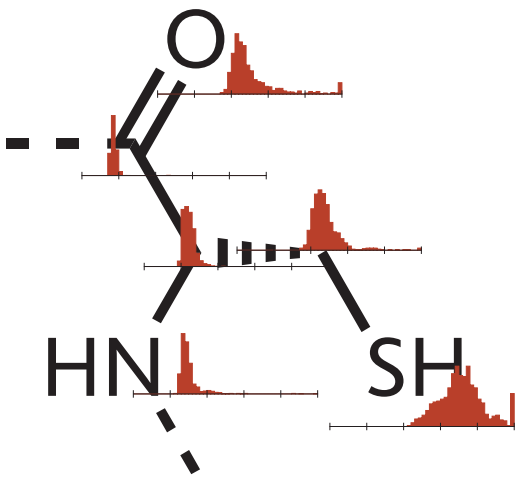
Asp D

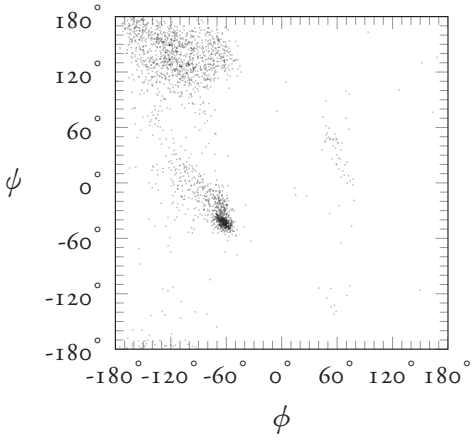




Cysteine

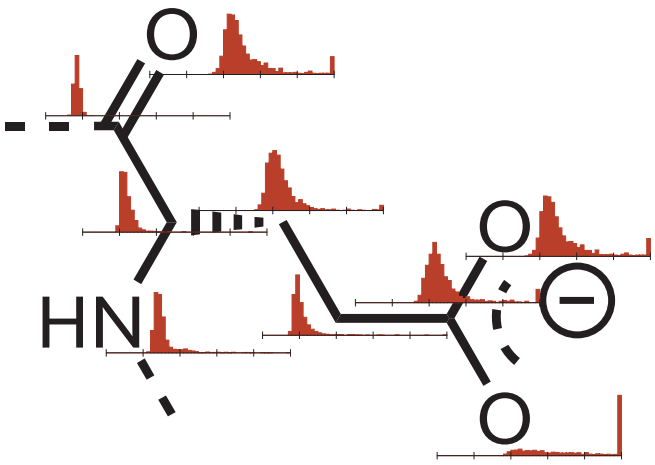
Cys C

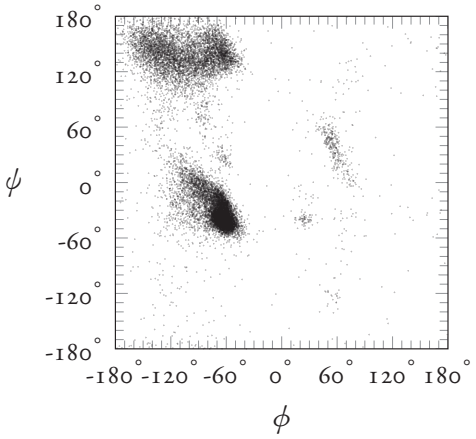




Glutamate

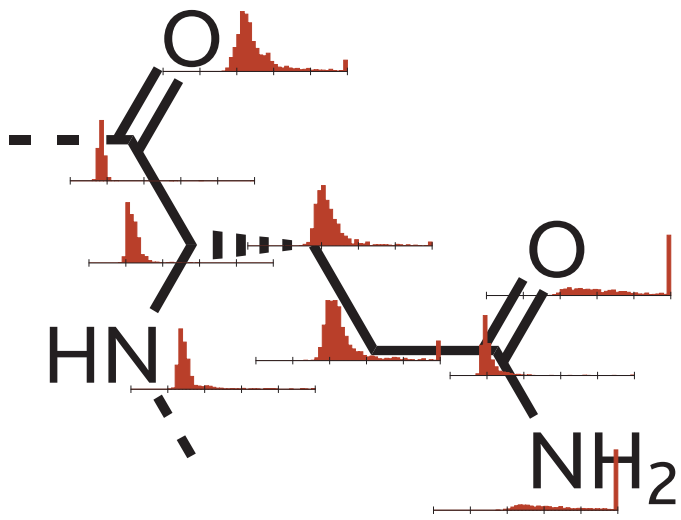
Glu E

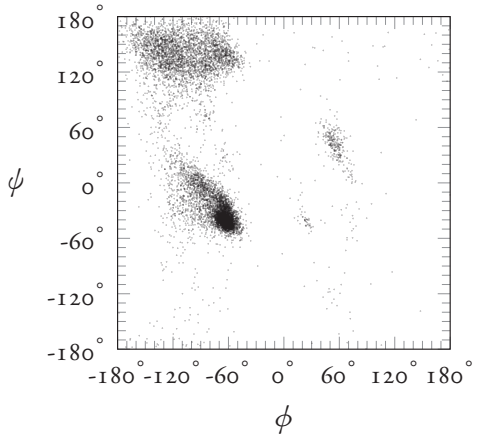




Glutamine

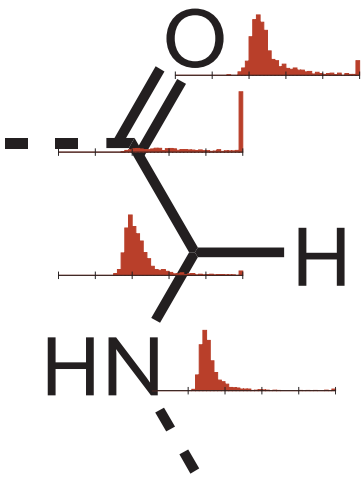
Gln Q

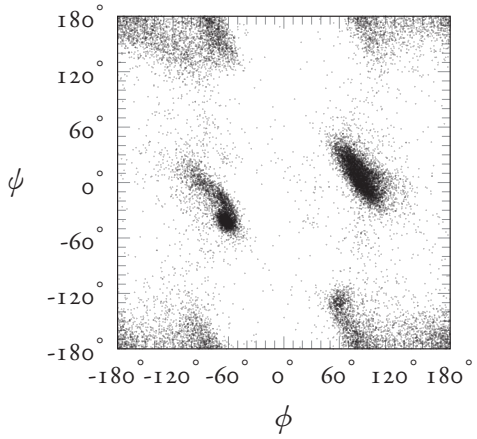




Glycine

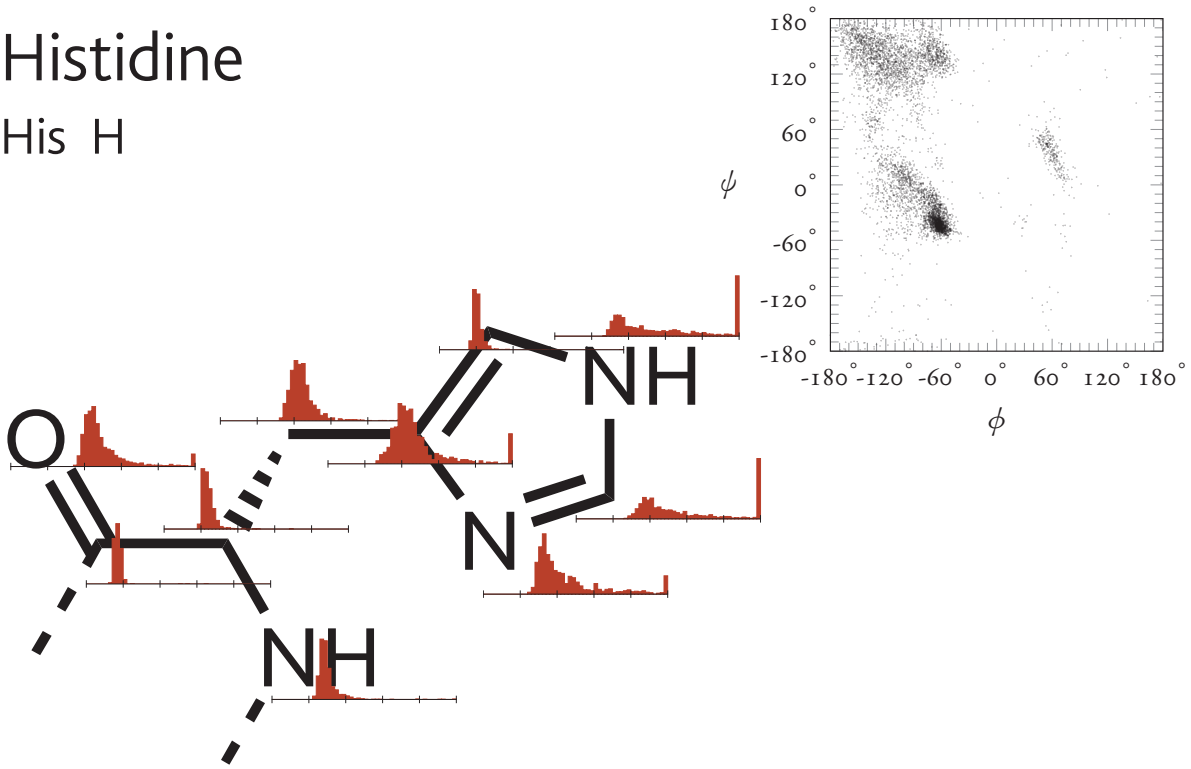
Gly G





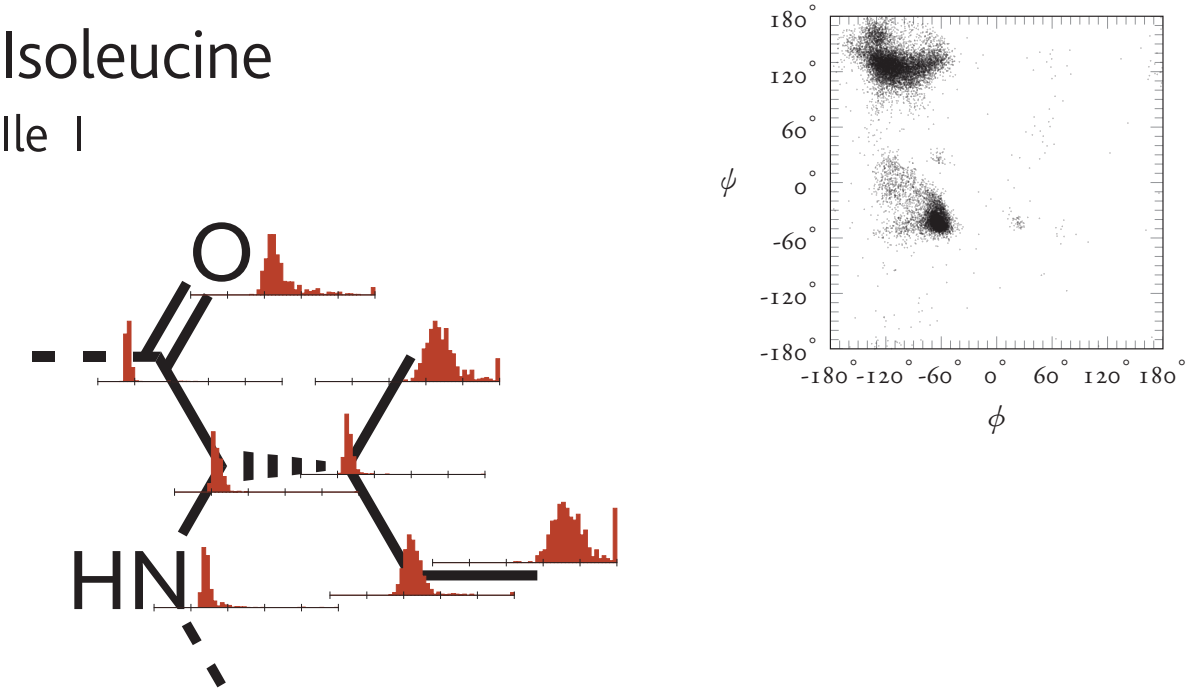
Histidine

His H



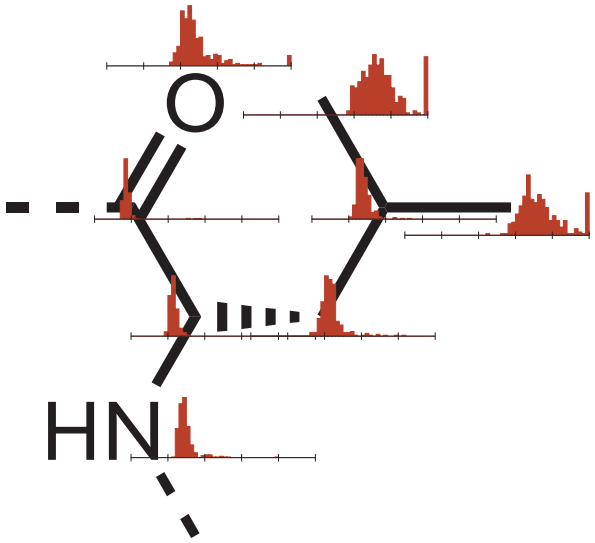
Isoleucine

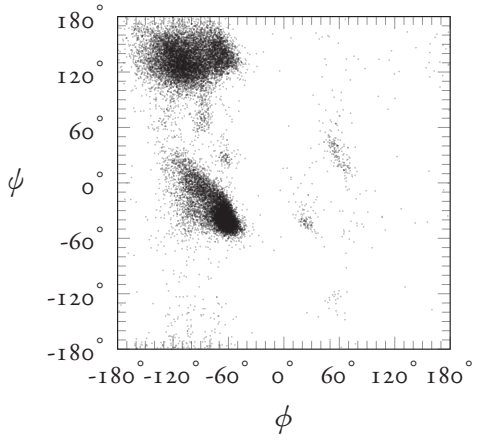
Ile I



Leucine

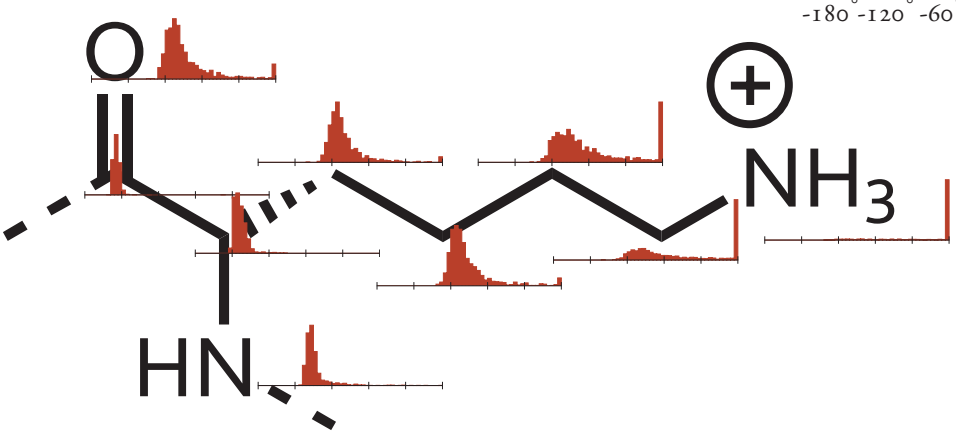
Leu L

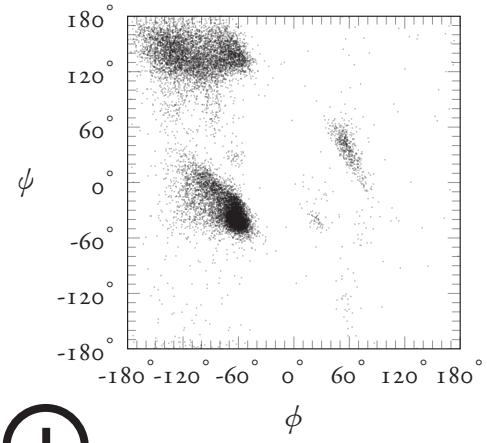




Lysine

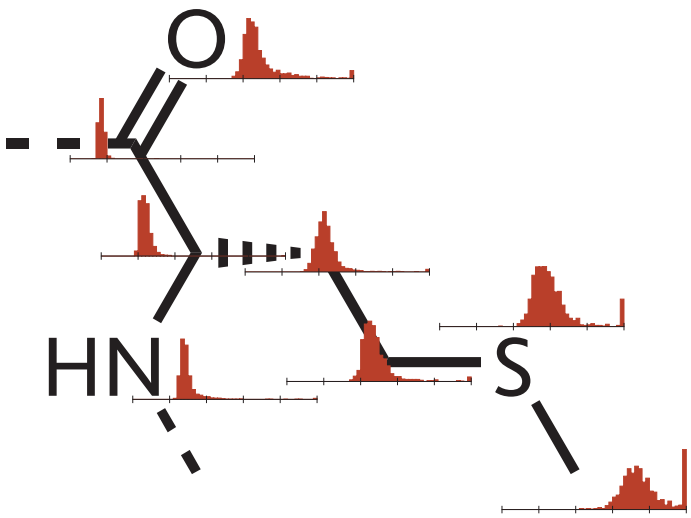
Lys K

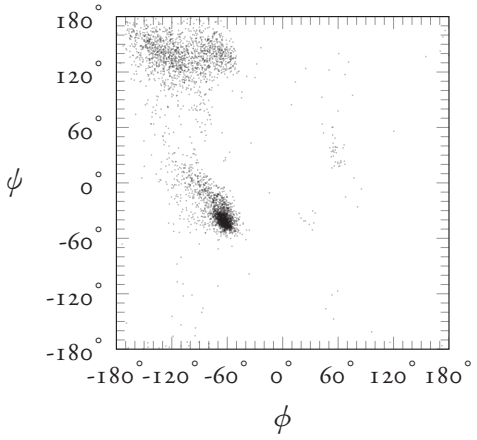




Methionine

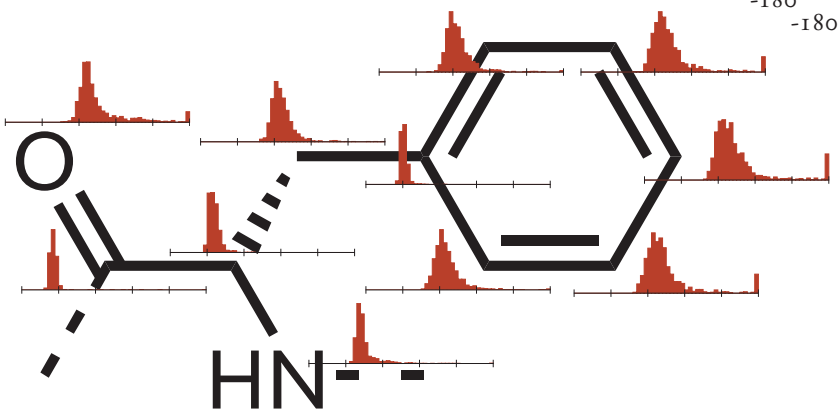
Met M

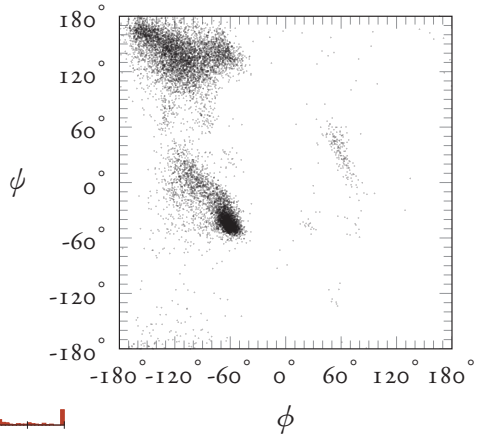




Phenylalanine

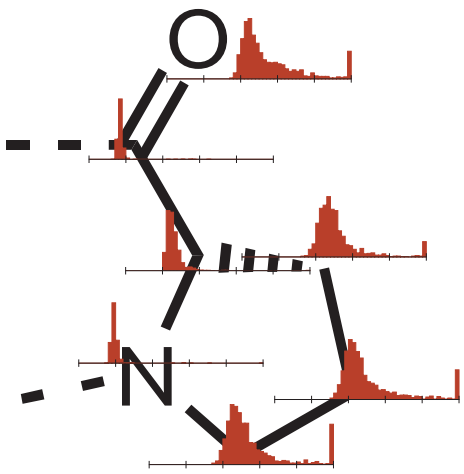
Phe F

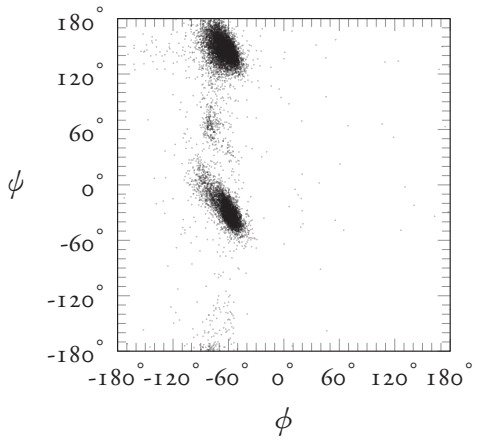




Proline

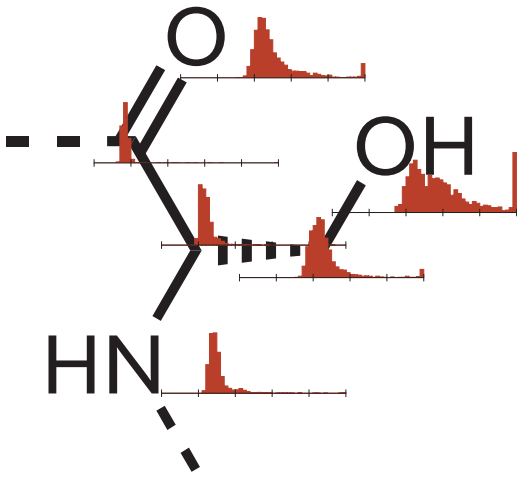
Pro P

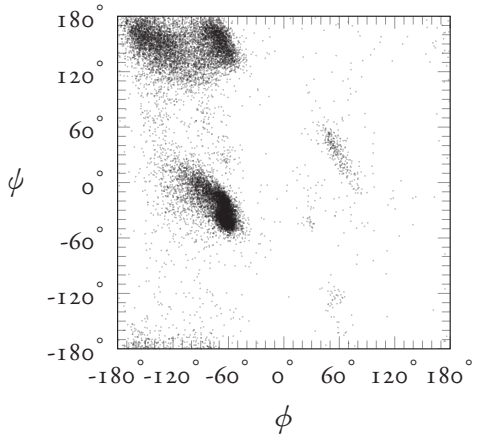




Serine

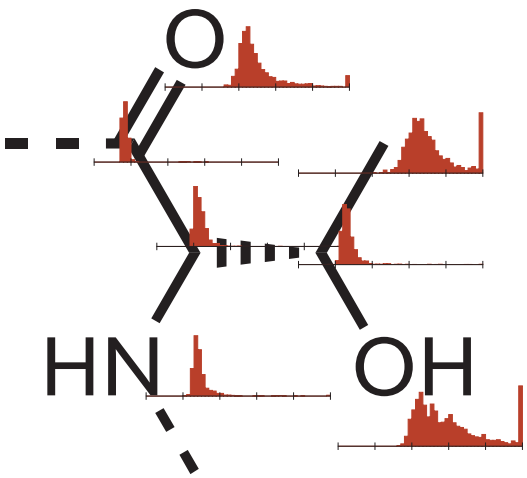
Ser S

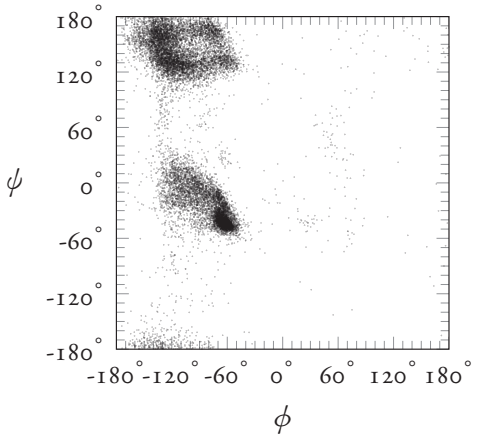




Threonine

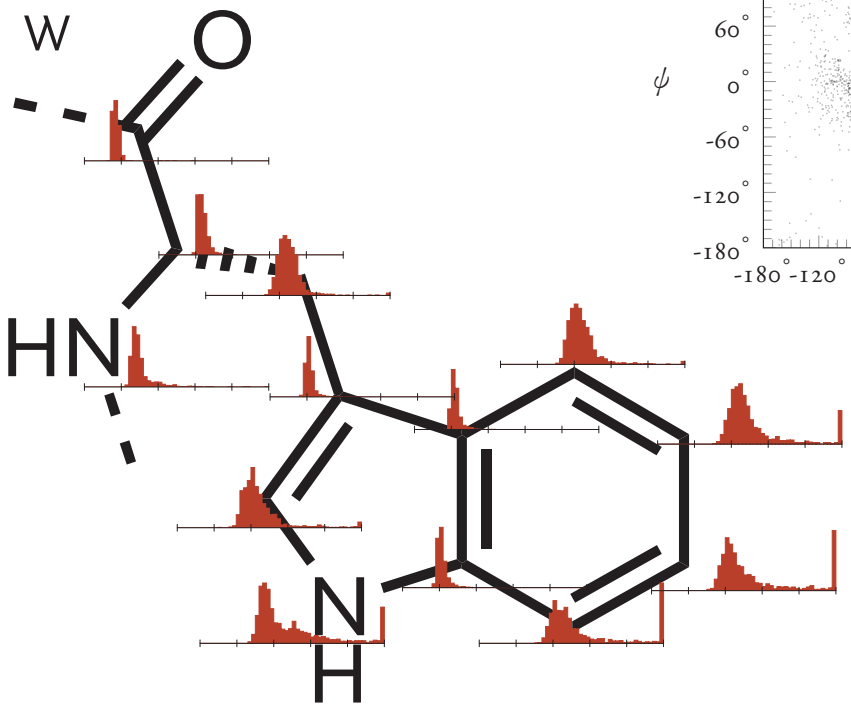
Thr T

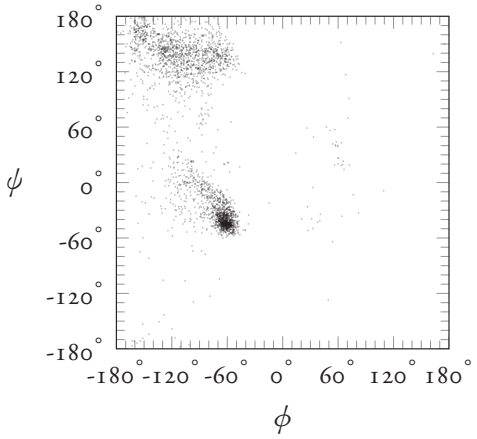




Tryptophan

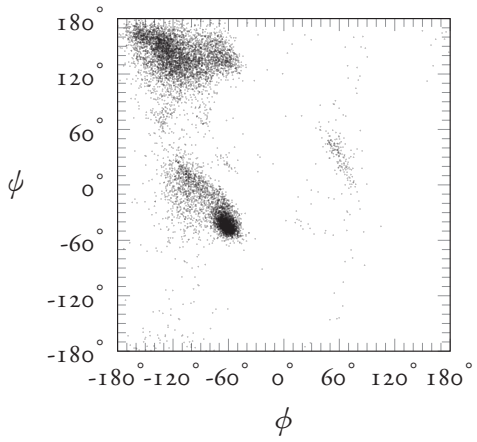
Trp W

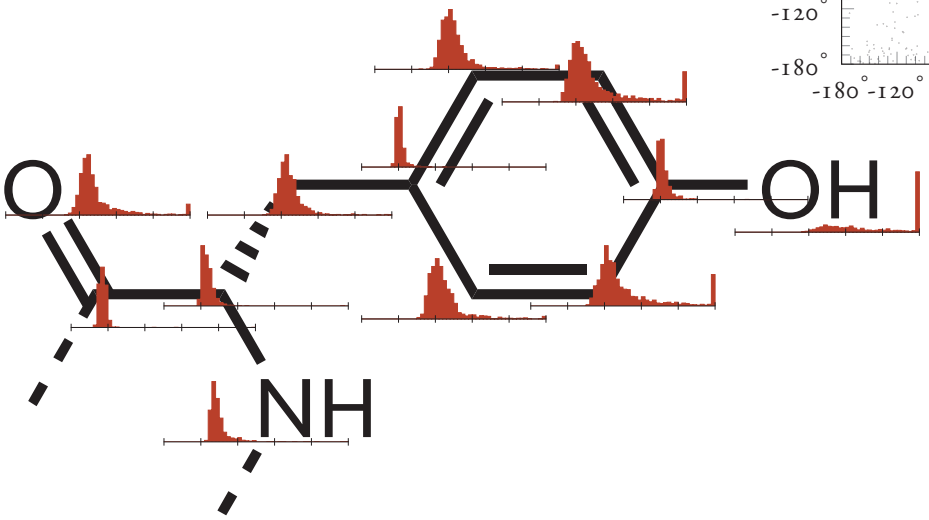




Tyrosine

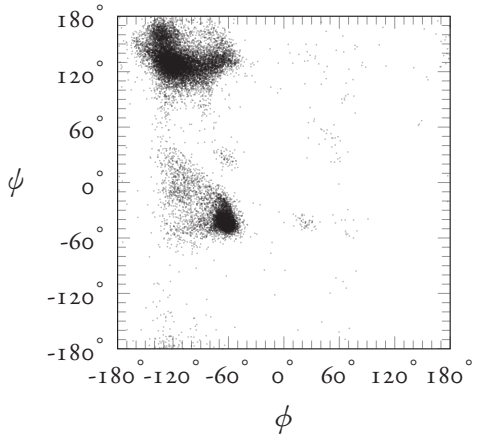
Tyr Y

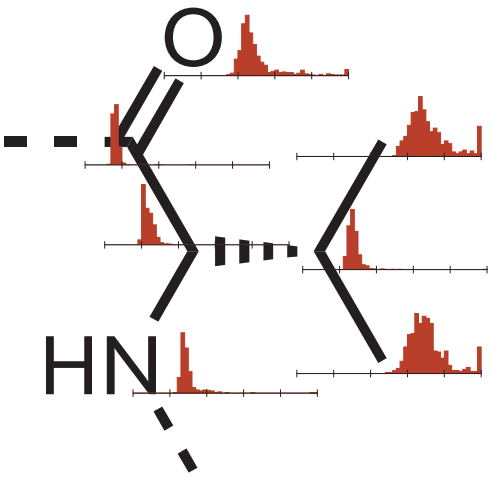




Valine

Val V





Software architecture

C

None of the results in this thesis would have been possible without the work of thousands of very clever computer programmers. In this appendix I discuss some of this work, and in particular the ideas that have developed with the Internet over the past decade but have not yet taken a strong hold in the scientific community. I discuss dynamic programming languages in contrast to the older and more established staticly-typed, procedural languages; the construction of the web-based visualization applications discussed in chapter 4; and the MapReduce programming style for large, Internet-sized datasets.

Ruby

The vast majority of the code written for this thesis was done in *Ruby*, a dynamic programming language developed by Yukihiro "Matz" Matsumoto in the mid-1990's. The focus of the language is on code readability and programmer productivity rather than on execution speed;

> Often people, especially computer engineers, focus on the machines. They think, "By doing this, the machine will run faster. By doing this, the machine will run more effectively. By doing this, the machine will something something something." They are focusing on machines. But in fact we need to focus on humans, on how humans care about doing programming or operating the application of the machines. We are the masters. They are the slaves.

Language creator Matz on the philosophy of Ruby in an interview at `http://www.artima.com/intv/ruby4.html`

For instance one iterates over lines of `my_file.txt` in a low level language like C:

```
my_file = fopen("my_file.txt", "r");
char current_line[300];
while(fgets(current_line, 300, my_file) != NULL){
  < code here >
}
fclose(my_file);
```

The programmer has to manually open the file, allocate memory, and at each line check to make sure the computer hasn't read past the end of the file. In contrast, Ruby performs memory allocation automatically, and files have a method that handles opening/closing and iterating over the lines via a *block* (an anonymous function). The equivalent code can be written:

```
open('my_file.txt').each_line{|current_line| <code here> }
```

These features stem from Ruby's object orientation–everything in the language is an *object*–a bundle of data and functions to operate on that data. Even primitives like numbers are considered objects, and come with several useful methods:

```
5.+(3) # => true.
# Note that '5 + 3' is just a shorthand for this method call
7.even? # => false
3.4.floor # => 3
49.gcd(14) # => 7
```

As in most object oriented languages, each Ruby object belongs to a certain *class*. These classes are arranged hierarchically, with children *inheriting* methods from their parents. At the root of this hierarchy is the `Object` class from which all other classes derive. The `Object` class has a `methods` method that returns an array listing all of the methods to which the object responds. Because every class in Ruby ultimately inherits from the `Object` class, everything responds to `methods`–we can list all of the methods specific to the number 4 by taking the difference between the methods of 4 and the methods of all objects:

```
Object.methods - 4.methods #=> [:—@, :+, :—, :*, :/, :div, :%, :modulo,
    :divmod, :fdiv, :**, :abs, :magnitude, :~, :&, :|, :^, :[], :<<, :>>, :to_f, :
    size, :zero?, :odd?, :even?, :succ, :integer?, :upto, :downto, :times, :next,
    :pred, :chr, :ord, :to_i, :to_int, :floor, :ceil, :truncate, :round, :gcd, :lcm, :
    gcdlcm, :numerator, :denominator, :to_r, :singleton_method_added, :
    coerce, :+@, :quo, :remainder, :real?, :nonzero?, :step, :to_c, :real, :
    imaginary, :imag, :abs2, :arg, :angle, :phase, :rectangular, :rect, :polar, :
    conjugate, :conj, :between?]

#note that we use the array difference method,
[1,2,4] - [1,2] #=> [4]
```

The language even allows programs to modify native objects at run time to "mix in" additional functionality.* For instance, I added a method to calculate the l^2 norm of an array of numbers

```
class Array
  def magnitude
    Math.sqrt(self.collect{|x| x**2}.sum)
  end
end
[2,3,6].magnitude #=> 7
```

* The practice of dynamically modifying a class at runtime is known as *monkey patching*. It can be quite useful but, as you might expect, changing the language itself at runtime can lead to mysterious errors.

The flexibility of the Ruby type system encourages *duck typing*: if an object walks like a duck and talks like a duck, then you can think of it as a duck.*

* Thus, dynamically modifying a class at runtime is sometimes known as *duck punching*; presumably, if something doesn't quack the way you want, you can punch it until it does.

Ruby is an *interpreted* language: code is executed line-by-line rather than read entirely into memory, analyzed, and converted into machine code by a compiler before it is executed. This is one of the major distinctions between dynamic/interpreted/scripting languages (Ruby, Perl, Python, JavaScript) and staticly-typed / compiled languages (C, Fortran, Java). The compiled languages are typically faster at executing code, because the compiler can optimize routines if it knows exactly the kind of data that will be fed in. The compilation process also provides a safety net by type-checking code.

For instance, if somewhere in the code you try to add a number to a string, the compiler will raise an error. The dynamic languages trade off the benefits of compilation for faster development time. In an interpreted language, it is very easy to test out parts of the code as you are writing it and hack together elaborate structures without having to explicitly specify the internal memory representation or data types.

This flexibility also provides far more opportunities to shoot oneself in the foot–type errors are discovered only at runtime, for instance, and if such bugs exist in infrequently run branches of the code then they will not be discovered until particular edge cases are executed. The question of whether to use a static or dynamically typed language perhaps reduces to having

- a statically-typed program logically refuse to deal with data that doesn't exactly fit the exact specifications written into the program and exit cleanly, versus
- a dynamic language program happily accept data according to loose specifications and occasionally blow up because of it.

I've found that my research problems are filled with edge cases and the flexibility of dynamic languages allows me to deal with them far more easily than I could in a staticly-typed language.

As an example, consider the the `.pdb` protein structure files. *Most* of the time, the atom lines look like this:

```
ATOM      1  N   GLU A   1      -7.323 -10.325   9.684  1.00 14.64           N
ATOM      2  CA  GLU A   1      -6.817  -9.013  10.069  1.00 11.95           C
ATOM      3  C   GLU A   1      -5.545  -8.630   9.322  1.00 11.43           C
ATOM      4  O   GLU A   1      -4.962  -9.471   8.636  1.00 12.86           O
ATOM      5  CB  GLU A   1      -6.523  -9.026  11.551  1.00  6.93           C
ATOM      6  CG  GLU A   1      -7.784  -9.325  12.328  1.00  9.45           C
ATOM      7  CD  GLU A   1      -7.414  -9.467  13.799  1.00 25.10           C
ATOM      8  OE1 GLU A   1      -7.466 -10.596  14.283  1.00 13.46           O
ATOM      9  OE2 GLU A   1      -6.996  -8.481  14.418  1.00 22.27           O
ATOM     10  N   ASP A   2      -5.189  -7.353   9.445  1.00 12.23           N
ATOM     11  CA  ASP A   2      -4.029  -6.789   8.755  1.00 18.14           C
ATOM     12  C   ASP A   2      -2.760  -7.036   9.574  1.00 15.38           C
ATOM     13  O   ASP A   2      -2.699  -6.646  10.741  1.00 16.61           O
ATOM     14  CB  ASP A   2      -4.303  -5.290   8.593  1.00  7.40           C
ATOM     15  CG  ASP A   2      -3.231  -4.568   7.767  1.00 19.67           C
ATOM     16  OD1 ASP A   2      -3.582  -3.568   7.135  1.00 35.89           O
ATOM     17  OD2 ASP A   2      -2.047  -4.901   7.861  1.00 21.70           O
```

Each atom has a serial number starting from 1, an atom role (atom 2 above is the C_α), belongs to a residue, &c. Parsing this data looks very easy–"okay, this column is the residue type, these columns are the x, y, z positions of the atoms, …"–and it is! Except until it's not. Sometimes atom positions cannot be resolved, so certain atoms are simply missing.

Or, what is far worse, sometimes atoms are *kind of* resolved and the crystallographers give you a choice:

```
ATOM    737  N   LYS A  97      56.240  28.959   7.316  1.00 24.23           N
ATOM    738  CA  LYS A  97      55.119  28.051   7.473  1.00 23.40           C
ATOM    739  C   LYS A  97      54.149  28.161   6.292  1.00 22.44           C
ATOM    740  O   LYS A  97      53.662  29.266   6.056  1.00 23.26           O
ATOM    741  CB  LYS A  97      54.275  28.331   8.732  1.00 22.39           C
ATOM    742  CG ALYS A  97      53.085  27.397   8.872  0.50 31.37           C
ATOM    743  CG BLYS A  97      55.094  28.243  10.020  0.50 24.65           C
ATOM    744  CD ALYS A  97      53.355  26.217   9.787  0.50 66.65           C
ATOM    745  CD BLYS A  97      56.295  27.331   9.835  0.50 37.70           C
ATOM    746  CE ALYS A  97      52.068  25.600  10.321  0.50 40.47           C
ATOM    747  CE BLYS A  97      56.891  26.917  11.174  0.50 60.11           C
ATOM    748  NZ ALYS A  97      51.207  26.620  10.986  0.50 29.30           N
ATOM    749  NZ BLYS A  97      58.379  26.846  11.113  0.50 89.54           N
```

Of course, the atom serial numbers cannot be repeated, so what looked like a nice index can now have holes. Speaking of pseudo-indexes, the residue sequence number *usually* starts at one, but for mysterious biological/historical reasons some `.pdb` files begin at 2, 234, or -1. For this 8 month research project it has sufficed to investigate and fix parsing errors from these outliers rather than read the 195 page official format specification.*

The Ruby language really took off in the United States with the *Ruby on Rails* web framework in 2005, and there are many libraries in Ruby for interacting with Internet technologies. In chapter 3 I compare several protein structural alphabets. One of the research groups I discuss did not release their structural alphabet or a program for encoding proteins; they only had a web site for querying the encodings of individual proteins. I needed the encodings of more than a hundred of these proteins. Rather than tediously use the website to manually copy down the results, I wrote a simple Ruby program to do it for me. For a collection of PDB IDs, I submit the form and collect the resulting alphabets in a hash table to be used later:

```
require 'mechanize'  #library for interacting with web pages

a = Mechanize.new
page = a.get('http://bioserv.rpbs.jussieu.fr/cgi-bin/SA-Encode')
f = page.forms.first
pdb_id_input = f.field_with(:name => 'PDBid')

h = Hash.new  #A hash table to keep the sequences in
pdb_ids.each{|pdb_id|
  #set and submit the form
  pdb_id_input.value = pdb_id
  results_page = f.submit

  #The resulting alphabet in given in <h6></h6> tags--remove fragment
      distinctions so all the pieces are just concatenated together
  seq = results_page.parser.css('h6').text.gsub(/unknwnA_\d
      /, '')
  h[pdb_id] = seq
}
```

* A useful seat-of-the-pants overview of the protein structure `.pdb` file format is given by Bob Rogers;

`http://bmerc-www.bu.edu/needle-doc/latest/atom-format.html`

JavaScript

The JavaScript programming language was written by Brendan Eich at Netscape in the mid 1990s as a way for novice programmers (i.e. early web designers) to make interactive web pages. Originally known as LiveScript, Netscape renamed the language JavaScript in a flurry of synergistic marketing excitement regarding Sun Microsystems' Java applets. This confusing name change (Java and JavaScript are *very* different languages), the incompatible browser object models of early web browsers, and the amateur-programmer base plagued the language in its early years, damning it in the minds of experienced programmers and seasoned web surfers as a foreboding sign of a soon-to-crash web browser.

Since its introduction, every major browser has included a JavaScript interpreter, making the language too ubiquitous to die. In 2004–5 Google released *Gmail* and *Google Maps*, which use JavaScript to make asynchronous server requests after the initial page has loaded. This *ajax*-style programming breathed a second life into JavaScript. The language was no longer used solely for shipping rate calculators and picture-change-on-mouse-over effects, but for useful dynamic interactions. Instead of clicking on a cardinal direction and waiting for a new page to load, one could just drag with the mouse and see new landscapes appear.

The popularity of ajax web programming and "Web 2.0" brought many improvements to the JavaScript ecosystem. Web framework projects (most notably John Resig's *jQuery*) extended the simple core language with functions for extensive manipulation of the DOM, ajax-style communications, animation, event handling, &c. As static web pages morphed into dynamic web applications, browser developers worked to speed up their JavaScript interpreters and provided consoles to debug and profile JavaScript code. This popularity spilled over into other domains, and programmers started using JavaScript for more than just manipulating web page elements on the client side.

For this thesis, I've written JavaScript programs to

- script Adobe's desktop vector graphics program, *Illustrator*, for the initial layouts of appendix B
- run MapReduce queries in MongoDB(later in this appendix)
- generate a 3D residue viewer (page 56)
- power the histogram explorer and contact map viewer depicted in chapter 4

The JavaScript language itself is very simple, having just eight objects.* The arrays are indexed by any value; effectively, they act as hash tables. As in Ruby, Perl, Python, &c. (and, ironically, very unlike Java), types are properties of values, not of variables. Like

* The eight types of JavaScript objects:

1. array
2. boolean
3. date
4. math
5. number
6. string
7. regular expression
8. global properties and functions

LISP, functions are first class objects–they can be created on-the-fly and passed around like any other data.

JavaScript is typically written with an *event / callback* style rather than the far more common procedural style found in most scripts or compiled programs. This is due to its upbringing in web browsers; server requests may take upwards of a few hundred milliseconds to return, and an event-based programming style allows the program to continue execution rather than idle. For example, in a procedural style of programming one would write

```
var x = 2;
var y = get_data_from_database();
var z = x + y;
alert("The result is " + z);
```

and in evented style,

```
var x = 2;
get_data_from_database(function(y){
  var z = x + y;
  alert("The result is " + z);
});
```

In the event-style program, an anonymous function is passed to `get_data_from_database()`. This function is called when the data is finally received, perhaps several seconds later.

For a more concrete example of these ideas, the next page shows an except of JavaScript code that displays residues as part of the histogram explorer described in chapter 4. This function gets the atomic coördinates and the Voronoi cell of a desired residue by making a call to a server-side* URL. The page at `/3d/json/2vb1_37/atoms`, for instance, returns a JSON object of the 37^{th} residue of the protein with PDB ID: 2VB1;

* I use the *Sinatra* Ruby web framework, which is a minimalist alternative to Rails that is perfect for the small, single-page applications I built over the course of this thesis.

```
{"atoms":
  {" N ":{"serial":733,"r":[6.79,14.10,18.92]},
   " C ":{"serial":734,"r":[6.06,13.54,16.65]},
   " O ":{"serial":735,"r":[6.37,12.97,15.61]},
   " CA ":{"serial":747,"r":[7.11,14.20,17.56]} ...}
 "seqno":"37",
 "residue_type":"asn",
 "cell_geometry":
   {"lines":[13,26,26,27,27,0,0,13,27,6,6,0,0,27,...],
    "vertexes":[13.57,22.43,8.77,46.85,16.68,19.64,...]
   }
}
```

```
function get_residue_atoms(pdb_seqno) {
  $.getJSON( '/3d/json/' + pdb_seqno + '/atoms', function(r){
    var atoms = r.atoms;
    var alpha_carbon = atoms[' CA '];

    //we want to translate everything so the alpha carbon is at the origin
    top_transform.identity();
    top_transform.translate(-alpha_carbon.r[0],-alpha_carbon.r[1],-alpha_carbon.r[2]);

    //draw the atoms
    jQuery.each(atoms, function(atom_role, a){
      var transform = g_pack.createObject('Transform');
      transform.parent = atoms_transform;
      //set the id to the atom serial
      transform.createParam("atom_role", "o3d.ParamString").value = atom_role;

      //sometimes hydrogens are like 'HH1', and sometimes more like ' HA '. Cannot match 'H' because that's how the
          pdb spells 'eta'.
      if (atom_role.match(/^(H|\sH)/)){ //add hydrogens as small spheres
        transform.addShape(g_pack.getObjects('hydrogen_sph', 'o3d.Shape')[0]);
      }else{
        transform.addShape(g_pack.getObjects('heavy_sph', 'o3d.Shape')[0]);
      }
      //move that sphere instance to the atom location
      transform.translate(a.r[0],a.r[1],a.r[2]);

      //add a color param (the slice makes a copy of the array, so we don't accidently modify the hardcoded color
          constants)
      transform.createParam('emissive', 'ParamFloat4').value = (element_colors[atom_role.substring
          (0,2).replace(/^\s*/, '')] || [0, 0, 0, 1]).slice(0);
    });

    //draw the bonds and the voronoi cell
    make_residue_bonds(r.residue_type, atoms);
    make_cell(r.cell_geometry);

    //look at the origin from the front (+Zhat)
    view_info.drawContext.view = g_math.matrix4.lookAt(
      [0, 0, 30], // eye
      [0, 0, 0], //target
      [0, 1, 0]); // up

    // Set our render callback for animation. This sets a function to be executed every time a frame is rendered.
    g_client.setRenderCallback(renderCallback);
   });
}
```

The server-side Ruby code simply fetches these pre-computed data from a database and returns it as a JSON string;

```
get '/3d/json/:pdb_seqno/atoms' do
  pdb_id, seqno = params[:pdb_seqno].split('_')
  p = PDB.from_database(pdb_id)
  t = Tessellation.from_database(pdb_id, :point_type => :alpha_carbon)
  r = p.residue_with_seqno(seqno)
  atom_hash = Hash.new
  r.atoms.each{|a| atom_hash[a.atom_role] = {:serial => a.serial, :r => a.r}}
  return JSON.generate {:atoms => atom_hash,
                        :seqno => r.seqno,
                        :residue_type => r.residue_type,
                        :cell_geometry => t.cells[r.seqno].geometry_hash}
end
```

MapReduce

The *MapReduce* paradigm for distributed processing was popularized by Google in 2004[15] (and later by Apache's *Hadoop* project) for fault tolerant calculation of certain kinds of problems. In particular, MapReduce is often used to do analysis on datasets stored across multiple computers: an unofficial tagline is "bring the computation to the data". A MapReduce operation is defined by two functions;

1. **Map** $(\text{key}, \text{value}) \rightarrow \left(\text{key}', \text{value}'\right)$
2. **Reduce** $(\text{key}, \text{values}) \rightarrow \text{values}'$.

The map function is applied in parallel to each value (datum) of the input set and emits zero or more pairs of transformed keys and values. These key/value pairs are collected by the reduce function, which transforms them into a single result. The reduce function may be applied iteratively as the map is taking place to save memory and leverage available computing power, and so the function must be;

- **associative**: $\text{reduce}\left(\text{key}_1, \text{value}_1\right) + \text{reduce}\left(\text{key}_1, \text{value}_2\right) = \text{reduce}\left(\text{key}_1, \text{value}_1 + \text{value}_2\right)$
- **communicative**: $\text{reduce}\left(\text{key}_1, (\text{value}_1 + \text{value}_2)\right) = \text{reduce}\left(\text{key}_1, (\text{value}_2 + \text{value}_1)\right)$
- **idempotent**: $\text{reduce}(\text{key}, \text{reduce}(\text{key}, \text{values})) = \text{reduce}(\text{key}, \text{values})$

The MapReduce operation is complete when there is nothing left for the reduce function to operate on.

MongoDB can execute arbitrary MapReduce JavaScript functions, which I use to calculate several statistics. Lets say, for instance, I want a histogram of the volumes of leucine residues that lay between 6–7 Å from the protein surface. The problem can be expressed in two parts; finding the volumes of those residues and binning and counting these volumes. This can be computed in the application layer, of course;

```
residues = pdb_ids.collect{|pdb_id| PDB.from_database(
    pdb_id).residues}.filter{|residue| (6..7).include?(
    residue.depth_alpha)}
volumes = residues.collect{|r| r.volume}
hist = volumes.to_histogram(:bin_size => 10, :max => 400)
```

but it can also be computed on the database using the MapReduce methodology;

```
//the map function
function(){
  var bin_size = 10; // binsize in cubic angstroms
  var hist = new Object;
  this.residues.forEach(function(r){
    if(r.residue_type == 'leu' && r.depth_alpha > 6 && r.
        depth_alpha < 7){
      var vol_bin = Math.min(Math.floor(r.volume / bin_size
          )*bin_size, 400) ;
```

```
      if(hist[vol_bin]){
        hist[vol_bin] += 1.0;
      }else{
        hist[vol_bin] = 1.0;
      }
    }
  });
  emit('result', hist); //we want all of the hists to be merged, so just
       use a constant key
}

//the reduce function
function(key, vals){
  total = new Object;
  vals.forEach(function(val){
    for(type in val){
      if(total[type]){
        total[type] += val[type];
      }else{
        total[type] = val[type];
      }
    }
  });
  return total;
}
```

The latter has several very significant advantages for computations involving large amounts of data. To compute in the application layer (i.e. within a Ruby program), we have to transfer *all* of the data about a given protein transfered from the database process to the application process. This can cost a bit in needless (but relatively fast) copies within RAM but *far* more if the data must be transfered over a network. Even if we specifically queried the database to only return the volumes of the appropriate residues, we might be getting 10^7 of numbers when in the end we only want 10^1 bins. Furthermore, since the map function is applied to all the objects in the database, it is more efficient to let the database choose the order in which it does this processing–the `pdb_ids` list in the application might not correlate with the order on disk of the protein data, for instance, and specifically querying them one by one would needlessly seek the disk.

The MongoDB implementation of MapReduce-style programming can simplify even very short queries. For instance, I store the CATH database classification of every protein within the protein objects themselves: each protein has fields `C`, `A`, `T`, `H`. In chapter 3 I develop a structure search procedure, and use the topology (`T`) level of CATH as a gold standard. I can find all of the CATH classifications having at least r proteins already in my database using this code:

```
topos = $p.map_reduce(
   "emit([this.c, this.a, this.t].join('.'), 1);",
   "function(k, vals){var sum = 0; for(var i in vals) sum += vals[i]; return sum
        }",
   :query => {:domains => 1}
).find(:value => {'$gt' => r-1}).to_a
```

The first argument is the map function (emits for each protein the CAT code as a key). The second argument is just an additive reduce function, summing up how many times it has seen the different keys. The `:query` line restricts the MapReduce operation to just single domain proteins. The `map_reduce` method returns an object containing all of the keys and the final values determined by the reduce function, and the `.find` method collects the keys that have been emitted r or more times.

These are only a few instances of the MapReduce programming I did in this thesis. The style allows one to formulate certain problems in a natural way to employ parallel-processing and distributed systems without requiring an understanding of the implementation. This abstraction has already taken off in certain corners of the Internet, and I suspect it will prove useful for some time.

Stability of measures

D

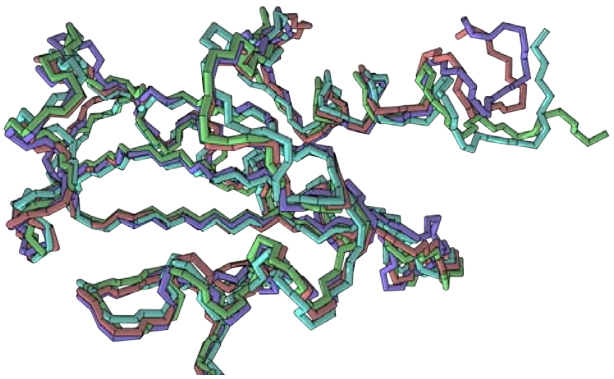
When we simplify the 3D structure of a protein into a binary matrix (the contact map), or that 2D matrix into a 1D vector (the contact map's principal eigenvektor), we need to ensure that our mappings are in some sense friendly. To use properties of the contact map as a tool for protein similarity detection, it must be "essentially unchanged" between close homologues, not to mention different experimental structures of a given protein. From the contact map's simple definition, intuitively it seems that the contact map would be pretty much the same for proteins of the same fold. The case is not so clear for its principal eigenvektor and other derived properties, however. The PE would not be a very useful protein representation if it radically changed when a few contacts are added or removed.

To test the stability of these representations, we need to come up with slightly different structures of the same protein and ensure that the contact map and its spectrum are not substantially different. One could make very simple changes; displacing random atoms by some small distance δ. Alternatively, one could build an ensemble of "realistic" protein structures by running explicit molecular dynamics simulations: adjust the protein backbone using classical mechanics (i.e. $F = ma$ where F is some terribly complicated function involving the Lennard-Jones potential, solvent interactions, thermal motion, &c.).

As it turns out, we can avoid the complexity of these methods* by using protein structures elucidated by NMR spectroscopy. Nuclear magnetic resonance spectroscopy refers to the measurement of the local environment of nuclei with nonzero spin. In the particular case of protein structure elucidation, spin polarization correlations (the nuclear Overhauser effect) between adjacent (not necessarily bonded) atoms give estimates of inter-atomic distances. There are often several models that fit the experimental constraints–experimentalists typically give 10–20 models for a given protein structure.

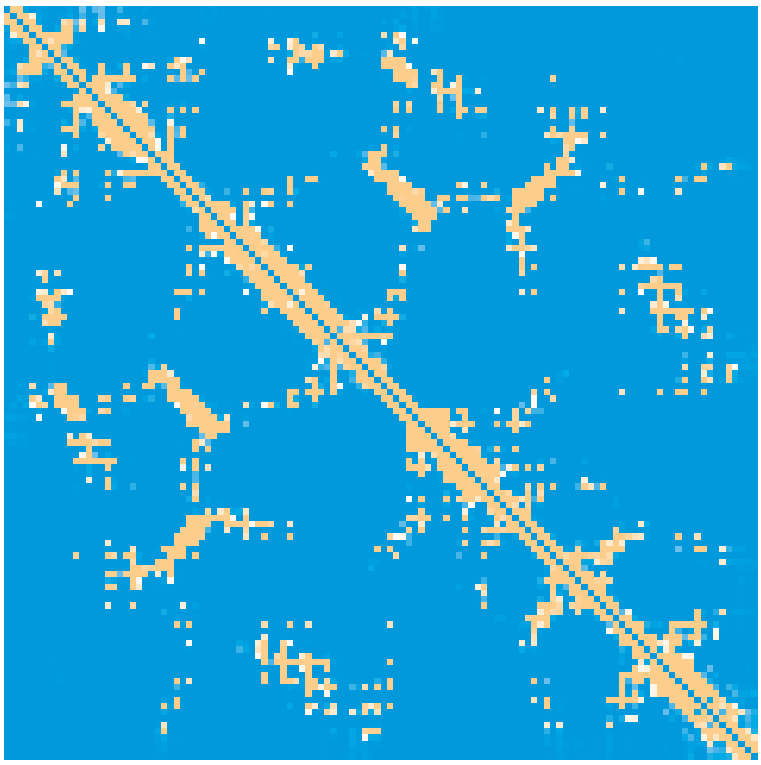
These models provide a natural set of structures for testing the stability of the contact map and its eigenvektor–each model represents a possible configuration of a given protein. Since the models are of the same protein, we can compare the positions of residues directly and we do not have to muck with gapped structural or sequence alignments (as we might have to if we tested on homologues).

* Not to mention drifting from the problem at hand–we are interested in the properties of contact maps in relation to experimentally determined protein structures, *not* their behavior under some perturbative methodology of our own construction.

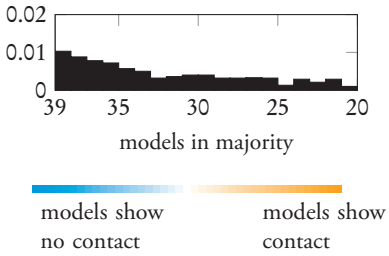


Backbone trace of the first 4 models of the NMR resolved protein 1MEK. Note that the core of the structure is well resolved (i.e. the models overlap) because there are many constraints, while different models show very different positions on some surfaces. The average RMSD of 39 models compared to the first is about 1.5 Å. This deviation reflects both experimental limits and also the dynamic nature of proteins–many surface residues *are* physically unconstrained.

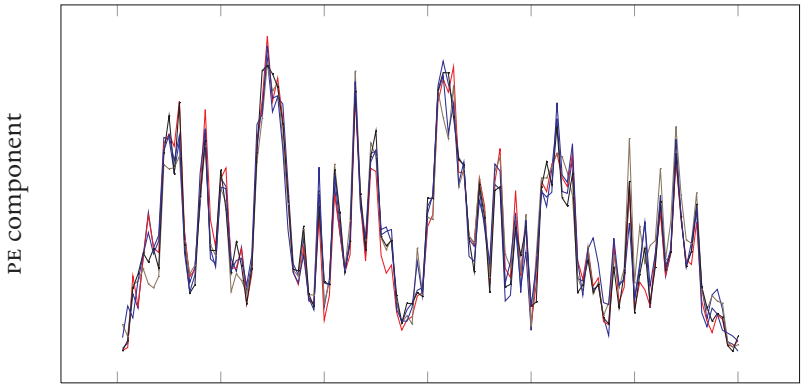
To demonstrate the stability of the contact map and its spectra, I focus on the 120 residue protein fragment 1MEK, which is reported with 40 NMR models. The 10 Å contact maps derived from these models are in good agreement, with all of the models agreed on 90% of the (non-)contacts.



Consensus 10 Å C_α contact map of 1MEK from an ensemble of 40 NMR models. Note that 90% of the (non-)contacts are identical between all 40 models, but the distribution of models in the majority for the remaining contacts is relatively flat (100% consensus contacts not shown);

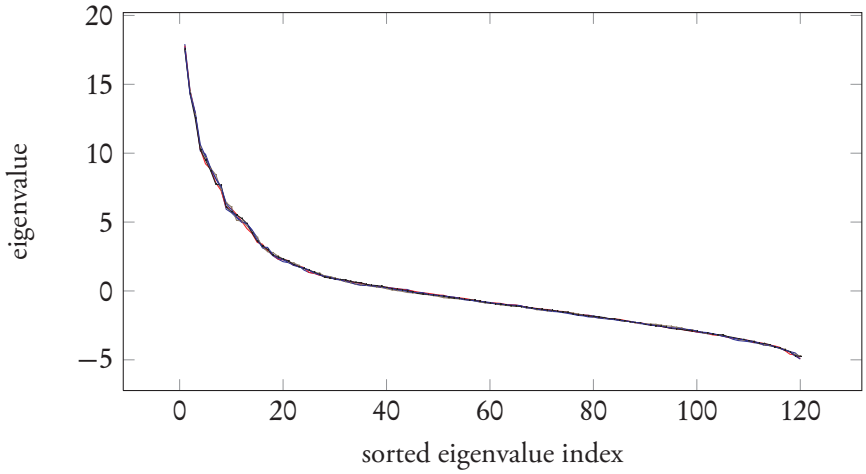


That there is little contact map variation is not surprising; NMR models are constructed from pairwise distance constraints–i.e. an experimentally measured real-valued contact map. Likewise, we find that the spectra of the contact maps are not different: The principal eigenvektors look very similar (first model vs. others dot products are all in the range 0.98–0.99);



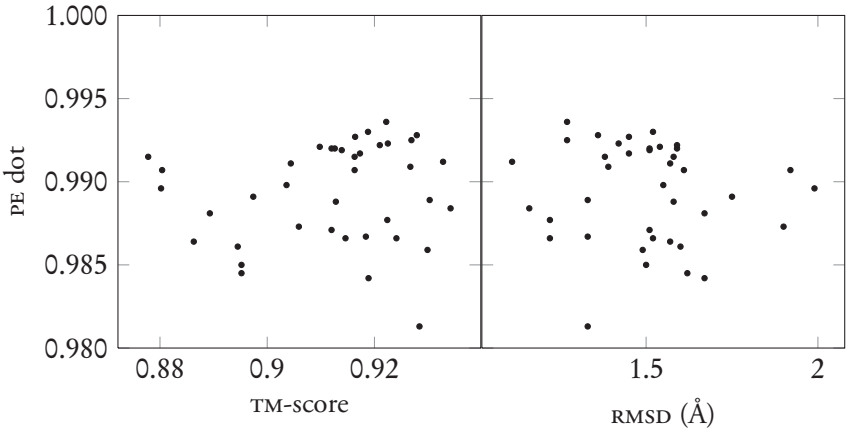
The principal eigenvektors derived from the 10 Å C_α contact maps of the first five NMR models, ordered by residue sequence number.

and the spectrum of eigenvalues seems even more similar;



The ordered eigenvalues derived from the 10 Å C_α contact maps of the first five NMR models.

Finally, note that variation of principal eigenvektors correlates with the structural variation as given by RMSD and TM-score measures;



Correlation of principal eigenvektor dot products and RMSD & TM alignments of first-to-all NMR models of 1MEK. Notice that the principal eigenvektor dot product trends as expected (albeit weakly) for both of these measures.

Structure comparison & alignment

The RMSD and TM-score are measures of *rigid body*[*] comparison–esentially trying to line up two protein structures so that they overlap In general, we require the following procedure (see Bourne and Shindyalov [9] for further discussion):

[*] In contrast to the general problem of protein structure comparison and alignment, which is a nuanced, biologically informed problem that subsumes a great deal of this thesis.

1. represent the proteins in some coördinate-independent space
2. compare the proteins according to some score
3. optimize the alignment (the correspondence[*] of residues in one protein to the residues of the other) according to this score. Note that this optimization is of the residue–residue correspondence, not necessarily the structures in space.

[*] If the proteins are of different length, then some residues will have to be omitted from the alignment (perhaps at the cost of a gap-penalty).

4. score the significance of the final alignment with regard to a baseline given by comparison against a random set of proteins

Once proteins are aligned by some method, the alignment must be scored. The simplest and by far the most reported metric is the root mean square deviation of the distances between aligned residues;

$$\text{RMSD} = \sqrt{\frac{1}{N}\sum_{i<j}\left|\mathbf{r}_{\alpha_i} - \bar{\mathbf{r}}_{\alpha_j}\right|^2}.$$

Despite its simplicity, the RMSD measure suffers from some deficiencies for comparing protein folds. In particular, it suffers from the same problem as all l^2-norm based measures in that outliers have a substantial effect on the final score. In *vivo*, proteins are *not* entirely rigid bodies–surface resides and dangling chains move freely (this is why they are sometimes difficult to resolve in both crystallographic and NMR structures), and thus it is possible for different models of *the same protein* to have large RMSDs from one another. For instance, if a protein consists of two domains attached by a hinge, the near diagonal elements of the distance matrices of the open and closed structures of the protein will tend to look very similar–the backbones of the individual domains are not changing very much at all–but the RMSD between the optimal rigid body alignment of the two structures will be quite high.[*]

[*] See the FATCAT [89] algorithm for alignment of flexible and hinged proteins.

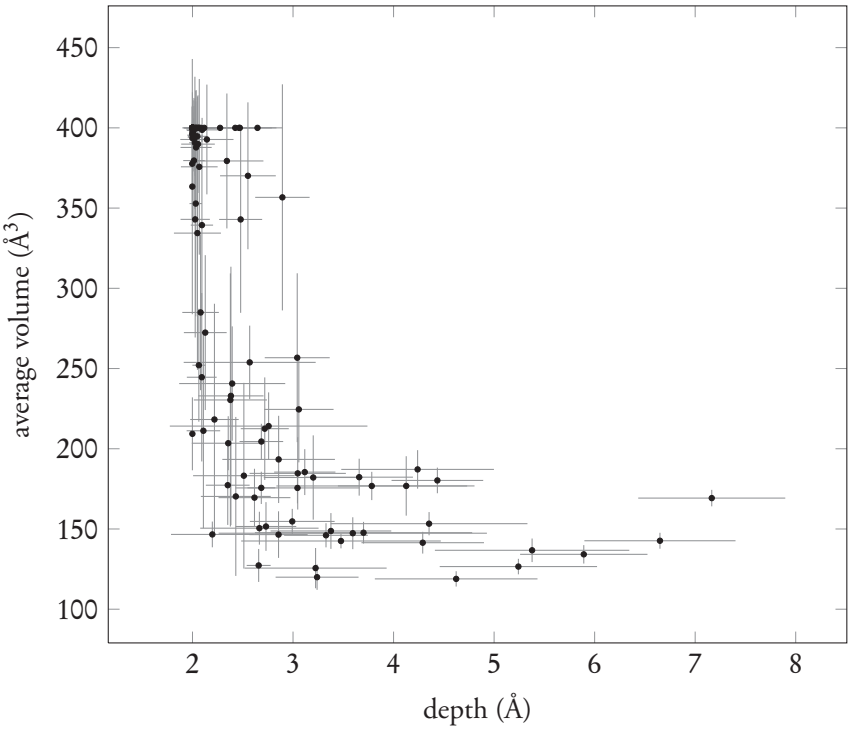
There exist many alternative metrics (see Hasegawa and Holm [29] for an overview of 26 flavors), but for brevity the structure comparison score I use in this thesis is the TM-score [91, 92],

$$\text{TM-score} = \max\left(\frac{1}{L_{\text{target}}}\sum_{i=1}^{L_{\text{aligned}}}\frac{1}{1+\left(\frac{d_i}{d_0\left(L_{\text{target}}\right)}\right)^2}\right),$$

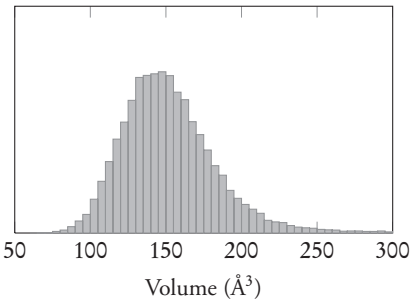
where L_{target} is the length of the target (native) structure, L_{aligned} the length of the aligned residues of the query, d_i the distance between

the i^{th} pair of aligned residues, and $d_0 = 1.24\sqrt[3]{L_{\text{target}} - 15} - 1.8$ a normalization length. This score has several nice properties–in particular it always lies in (0–1] and is normalized to be independent of protein length. A perfect structural match gives a score of unity, and random pairs of structures from the PDB give "an approximately constant value of ≈ 0.17" [91].

The other measure I investigate in this thesis is the Voronoi tessellation. Like the contact map principal eigenvektor, it is not immediately clear that small perturbations of the backbone will not lead to drastic changes in the Voronoi cell. Both the cell volume and its variation between models decreases with residue depth, suggesting that the Voronoi tessellation is stable within the ordered regions (i.e. core) of proteins. Also note that the variation in volume for core residues is far less than the variation of volumes within the core. That is, each residue maintains its own characteristic volume within an ensemble of NMR models.



Average α-carbon Voronoi tessellation cell volume vs. average residue depth from the 40 models of 1MEK. Unbounded volumes set to 400 Å^3. Note that the cell volumes of a single residue in the protein core between NMR models is more tightly distributed (see the volume error bars on plot at left) than for core residues in general (see appendix B); the plot below shows the volume of all residues at 5 Å depth.



The CATH *structural classification*

E

The CATH [58] domain structure database is one of several online databases that attempts to classify solved protein structures into families and "superfamilies" according to structural and evolutionary relationships. I use it as a gold standard of truth in two parts of this thesis. Domain assignments from spectral clustering algorithm of chapter 2 are compared to the assignments listed in CATH. The structural search procedure in chapter 3 uses the *topology* level of CATH to gauge whether a search hit is a true or false positive.

CATH was established in 1993 and currently contains about 1300 folds from 130,000 domains of 100,000 protein chains. The name is an acronym for a hierarchical classification scheme for protein domains: *Class*, *Architecture*, *Topology*, and *Homology*.* The class is determined by the domain's secondary structure content, currently in four categories: all α, α/β, all β, and those (odd) proteins with few secondary structures. The architecture level describes the orientation of these secondary structures in space, regardless of the chain connectivity.

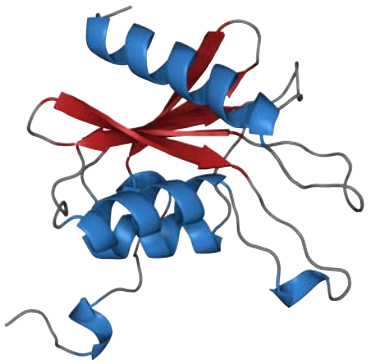
For instance, a protein would be classified under the "3-layer α-β-α sandwich" architecture level if it had α-helicies surrounding a (delicious) β-sheet core, no matter the order in which these secondary structures occur along the backbone. The topology level takes the backbone connectivity into account. After these three structural classification levels, CATH organizes protein domains into homologous "superfamilies" having 35, 60, 95, and 100% amino acid residue sequence identity.

New structures are classified in several stages: the first stage looks for close relatives using fast pairwise sequence methods or (if necessary) more distant relatives by slower sequence profiles and structure comparison tools. If a good match is found, then that domain of the new structure can take on the classification of its match. If not, a variety of automatic and manual methods are used to first infer the domain boundaries and then to either manually assign these new domains into existing architectures or establish new ones.

There are two bottlenecks in this procedure. The first is the structure comparison step, which takes far longer than sequence search and alignment but is indispensable–about 15% of distantly related proteins* have relationships that are currently detectable only by direct structure comparison [57]. This one vs. all pairwise structure comparison is a problem that, like all search problems, will only become harder as more protein structures and solved. I discuss this problem in chapter 3.

The second bottleneck is the detection of domain boundaries. Currently CATH uses a consensus methodology [36] to determine the

* *Homologous* proteins share a common ancestor. Over the course of evolution, genes coding for proteins are shuffled around and are occasionally joined together to form a multi-domain protein from two previously separate proteins. That some proteins seem to consist of independently-folding domains is not just a physical curiosity of larger proteins–in several ways a domain is the fundamental unit of biological function.



Cartoon rendering of α-β-α sandwich protein 2FR5A00.

* For instance, the globin family of proteins that carry the oxygen-binding heme group, occur across a wide variety of species but some of the individual members have as little as 10% sequence identity though their functions remain the same.

domains of a new protein structure. If three independent programs*: PUU [32], *Domak* [70], and *Detective* [73], agree within a tolerance of 10 residues, then a new protein's domains are assigned automatically, otherwise the domain assignments must be manually checked. Note that all of the programs used for CATH domain assignment, like the spectral clustering algorithm of chapter 2, allows for backbone-discontinuous domains (i.e. a domain might consist of two different parts of the protein chain).

* Roughly, the PUU algorithm segments proteins by looking at the correlated motions of subunits of the structure, *Domak* cuts the backbone chain to maximize intra-segment contacts and minimize inter-segment contacts on the protein's 5 Å heavy atom contact map, and *Detective* partitions a protein by looking for hydrophobic cores.

Structure prediction

F

The contact map simplification is useful in two open questions of structural biology. In the ultimate project of predicting full structure given only sequence information, the contact map serves as a closer milestone, turning

$$\text{sequence} \xrightarrow{\text{hard}} \text{structure}$$

into the more tractable

$$\text{sequence} \xrightarrow{\text{less hard}} \text{contact map} \xrightarrow{\text{less hard}} \text{structure.}$$

The first problem is to map a vector consisting of amino acid residues to a binary matrix. The second problem is to map that binary matrix to n points in $\mathbb{R}^3$ (to specify the protein backbone).

The most successful approaches to the first problem involve the use of "supervised learning" techniques from the field of machine learning and statistical inference. Given a set of training examples (protein sequences and their native contact maps.*), we iteratively adjust the weights (i.e. train) some mathematical model to be used for predicting future data (I use a neural network to predict whether a pair of residues are in the same domain, see page 28)

Fariselli *et al.* [19] achieved an average contact map accuracy of 21% using a neural network to predict contact between two residues of a sequence. Their input vector incorporated evolutionary sequence conservation over a multiple alignment of the residues in question, as well as that of their immediate sequence neighbors (all proteins of the test set had at least 15 aligned sequences).

The most difficult part of using a standard neural network is translating the problem at hand into a vectorial representation– Fariselli fixed the input vector dimension by only considering pairs of residues, but intuitively we should have more success with methods that train on contact maps in their entirety rather than on their individual components one at a time–we should look at each protein holistically, rather than just as a collection of residue pairs. More recent groups[3, 74] have employed 2D recursive neural networks [20] (which can train on square matrices) and have achieved accuracies upwards of 30%. Support vector machines have also been used to predict contact maps, and in at least one case a group has achieved similar accuracies as the neural networks, 30% [12].

There are two general approaches to the second problem, contact map → structure. The first approach is to use a pairwise contact potential;

$$E(\mathbf{C}, \mathbf{s}) = \sum_{i<j} \mathbf{C}_{ij} w(s_i, s_j),$$

where $\mathbf{C}$ is the contact map, $\mathbf{s}$ the protein sequence, and $w(s_i, s_j)$ the interaction energy of residues i & j. Since contact is symmetric, w contains $\frac{1}{2}n(n+1) = 210$ free parameters (n=20 amino acids).

* One tends to turn to statistical inference and machine learning when explicit analytical models fail, and so as a general rule of thumb it is helpful to include (within computational feasibility) as much prior data as possible and let the learning algorithm sort it out. Fariselli *et al.* [19], for instance, used correlated mutation data (the tendency of a residue mutation to be followed by another, thermodynamically compensatory mutation) derived from multiple sequence alignments to infer physical proximity in contact map prediction.

Vendruscolo *et al.* [84] used the convergence properties of the perceptron (neural network) algorithm to prove that there *cannot* be a set of parameters for the simple contact model that universally differentiates native and decoy contact maps. As one might have suspected, there are important energetic factors beyond simple pairwise residue contact. Mirny and Domany [52] describe the construction and testing of an energy function that includes solvent interaction as well as pairwise residue contact.

Mapping the vertexes of the graph described by the contact map to points in $\mathbb{R}^3$ is an NP hard problem [10], but there exist several heuristic algorithms for reconstruction in practice. Vendruscolo *et al.* [83] build structures to satisfy a given contact map one residue at a time. For a given run of their stochastic algorithm, they first generate a set of points in allowed regions of space that satisfy covalent bond lengths and $C_\alpha - C_\alpha$ angles and select one according to a Boltzmann-like probability distribution with "energy" given by the contacts (un)satisfied by choosing that point. After a full backbone has been grown, it is relaxed by successive crankshaft moves of random points. Notably, this algorithm does not require that the full contact map be satisfied.

Vassura *et al.* [82]'s FT-COMAR use a similar two part methodology. Their algorithm first generates a complete initial set of points using a metric matrix embedding algorithm and then iteratively applies contact map derived pseudoforces to each point until they satisfy both the desired contact map and the constraints of protein geometry. The Vassura group ran their algorithm on the same set of proteins as Vendruscolo *et al.*, and reported slightly better backbone RMSD values for 9 Å contact maps (about 2 Å). On 13 Å contact maps, Vassura's algorithm typically gave about 1 Å backbone RMSDs.

Note that these methods cannot converge on physically unrealizable conditions and will never satisfy an "unphysical" contact map. The most useful contact map prediction algorithms, then, are those that give just a few true contacts rather than more contacts at the expense of additional false positives.

A major advantage of the distance matrix representation is that it is coördinate independent–all values are measured between parts of the structure rather than in relation to some external frame. A glaring deficiency of distance matrices, however, is that they cannot specify structural chirality. The C_α of all non-glycine residues is a stereogenic centre of the molecule–all naturally occuring amino acids have the same handedness. One way this local chirality manifests itself in protein secondary structure is the right handedness of the α-helix (such that all the sidechains point outward). However, we know this *a priori* and it is not problematic that contact maps are agnostic in this regard.

Clustering maths

G

Proximity measures are the explicit mathematical functions that give the "distance" or "similarity" of two points in our feature space. I discuss pairwise proximity measures within a real vector space (i.e. $\mathbb{R}^f$ for a dataset consisting of f real valued features).* A dissimilarity measure d on some vector space X is a function

$$d : X \times X \to \mathbb{R}$$

such that

$$\begin{aligned} \forall d_0 \in \mathbb{R} : -\infty < d_0 \leq d(\mathbf{x},\mathbf{y}) < +\infty \qquad & \forall \mathbf{x},\mathbf{y} \in X \\ d(\mathbf{x},\mathbf{x}) = d_0, \qquad & \forall \mathbf{x} \in X \\ d(\mathbf{x},\mathbf{y}) = d(\mathbf{y},\mathbf{x}). \qquad & \forall \mathbf{x},\mathbf{y} \in X \end{aligned}$$

A metric dissimilarity measure is one for which also

$$d(\mathbf{x},\mathbf{y}) = d_0 \Leftrightarrow \mathbf{x} = \mathbf{y}$$

and the triangle inequality holds;

$$d(\mathbf{x},\mathbf{z}) \leq d(\mathbf{x},\mathbf{y}) + d(\mathbf{y},\mathbf{z}) \qquad \forall \mathbf{x},\mathbf{y},\mathbf{z} \in X,$$

i.e. it is not possible to draw a triangle such that going along two sides is shorter than going along the third.

Note that our friend Euclidean distance,

$$d_{\text{euc}}(\mathbf{x},\mathbf{y}) = \sqrt{\sum_i^n (x_i - y_i)^2},$$

is a metric dissimilarity measure. I will refer to all dissimilarity measures as "distances", although they may behave very differently than what we are accustomed to think of as "distance". So, for instance, even if points $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ are collinear, $d(\mathbf{a},\mathbf{b}) + d(\mathbf{b},\mathbf{c})$ will not always equal $d(\mathbf{a},\mathbf{c})$.

Other popular metric dissimilarity measures include the *Manhattan norm*,

$$d_1(\mathbf{x},\mathbf{y}) = \sum_i w_i |x_i - y_i|,$$

for some weight vector $\mathbf{w}$, and the l_∞ norm,

$$d_\infty(\mathbf{x},\mathbf{y}) = \max_i w_i |x_i - y_i|.$$

Similarity measures s are defined in the same fashion, except with an s_0 an upper bound rather than a d_0 lower bound;

$$s : X \times X \to \mathbb{R}$$

* There are two significant extensions of proximity that I do not discuss, but one should be aware of. The first regards categorical (nominal) features–those variables that cannot be mapped to a real number. The second generalization is from pairwise vector comparisons to comparisons between vectors and groups of vectors or between groups and groups, which is required for some hierarchical clustering algorithms. For a discussion of these topics, see Theodoridis and Koutroumbas [75] (whose notation I follow in this text).

such that

$$\begin{aligned} \forall s_0 \in \mathbb{R} : -\infty < s(\mathbf{x},\mathbf{y}) \leq s_0 < +\infty \qquad & \forall \mathbf{x},\mathbf{y} \in X \\ s(\mathbf{x},\mathbf{x}) = s_0, \qquad & \forall \mathbf{x} \in X \\ s(\mathbf{x},\mathbf{y}) = s(\mathbf{y},\mathbf{x}), \qquad & \forall \mathbf{x},\mathbf{y} \in X \end{aligned}$$

A metric similarity measure is one for which also

$$s(\mathbf{x},\mathbf{y}) = s_0 \Leftrightarrow \mathbf{x} = \mathbf{y}$$

and

$$s(\mathbf{x},\mathbf{y})\, s(\mathbf{y},\mathbf{z}) \leq [s(\mathbf{x},\mathbf{y}) + s(\mathbf{y},\mathbf{z})]\, s(\mathbf{x},\mathbf{z}) \qquad \forall \mathbf{x},\mathbf{y},\mathbf{z} \in X.$$

As one might expect from their names, dissimilarity measures and similarity measures can be thought of as opposites. For instance, if $d(\mathbf{x},\mathbf{y})$ is a (metric) dissimilarity measure then we can use it to make a (metric) similarity measure

$$s(\mathbf{x},\mathbf{y}) \equiv \frac{\alpha}{d^{\beta}(\mathbf{x},\mathbf{y})} \qquad \alpha,\beta > 0.$$

The most popular similarity measure is the *cosine similarity*,

$$s_{\text{cosine}}(\mathbf{x},\mathbf{y}) = \frac{\mathbf{x}^{\mathrm{T}}\mathbf{y}}{|\mathbf{x}|\,|\mathbf{y}|},$$

which is just the *inner product* of two normalized vectors.

A common similarity measure used in statistics is *Pearson's correlation coefficient*,

$$s_{\text{Pearson}}(\mathbf{x},\mathbf{y}) = \frac{\mathbf{x}_d^{\mathrm{T}}\mathbf{y}_d}{|\mathbf{x}_d|\,|\mathbf{y}_d|},$$

where $\mathbf{x}_d$ and $\mathbf{y}_d$ are difference vectors;

$$\mathbf{x}_d = \begin{bmatrix} x_1 - \overline{x} \\ \vdots \\ x_f - \overline{x} \end{bmatrix} \quad \overline{x} = \tfrac{1}{f}\sum_i^f x_i, \qquad \mathbf{y}_d = \begin{bmatrix} y_1 - \overline{y} \\ \vdots \\ y_f - \overline{y} \end{bmatrix} \quad \overline{y} = \tfrac{1}{f}\sum_i^f y_i.$$

Note that this can also be written as the covariance $\text{cov}(X,Y)$ over the product of the standard deviations $\sigma_x\sigma_y$ for random variables X,Y;

$$\begin{aligned} s_{\text{Pearson}} &= \frac{\text{cov}(X,Y)}{\sigma_x\sigma_y} \\ &= \frac{\text{Expectation}[(x-\overline{x})(y-\overline{y})]}{\sigma_x\sigma_y}, \end{aligned}$$

where the expectation of n data is given by $\frac{1}{n-1}\sum_i^n \left(x_i - \overline{x}\right)\left(y_i - \overline{y}\right)$.

Pearson's correlation coefficient is independent of both scale and origin and is always in the range $[-1, 1]$. It can be roughly thought of as the noisiness of linear relationship between two variables, independent of the strength (i.e. slope) of that relationship. However, a high correlation coefficient does *not* imply there is a linear dependence between two variables (see page 52).

Datasets

H

Unless otherwise noted, the data for all figures were drawn from a nonhomologous set of single chain proteins retrieved 06 March 2010 from the PDB with the query:

```
Number of Chains Search : Min Number of Chains=1 Max Number of Chains=1
Experimental Method Search : Experimental Method=X-RAY
Ligand Search : Has Ligands=no
Chain Type Search : Contains Protein=Y, Contains DNA=N, Contains RNA=N, Contains DNA/RNA Hybrid=N
Sequence Length Search : Min Sequence Length=80 Max Sequence Length=1500
Homologue Removal - 50% Identity Cutoff
=> 1269 matches
```

```
1a1x 1a2j 1a3h 1a62 1a8q 1aaj 1acf 1acx 1agi 1ahc 1ak1
1ako 1ald 1aly 1amx 1arb 1arl 1at0 1b0x 1b8e 1b8p 1b9w
1bam 1bd8 1bea 1bec 1bed 1bfg 1bg6 1bgc 1bgf 1bhe 1bi5
1bj7 1bk7 1bkb 1bkr 1bm8 1bn6 1bol 1bqc 1bue 1buo 1by7
1byi 1byr 1bz4 1c1l 1c3m 1c3p 1c44 1cdy 1cei 1ceo 1cew
1cex 1chd 1cm3 1cnv 1cqy 1crz 1cs8 1cuk 1cv2 1cwy 1czt
1dbg 1ddw 1dfa 1dg3 1dhn 1din 1dix 1dja 1dq0 1dsl 1dst
1dto 1dua 1dun 1dup 1dus 1dv7 1dvo 1dzf 1e4f 1e5m 1ea2
1ecl 1edg 1edq 1edt 1eg3 1ei5 1ekg 1en2 1ep0 1erv 1es5
1es6 1es9 1et9 1eur 1ew4 1ey4 1eyb 1eyh 1f00 1f2j 1f32
1faa 1fc6 1fcq 1fhg 1fhu 1fkm 1fl0 1fmk 1fna 1fnf 1fnj
1fo9 1fqi 1fsf 1fus 1fwu 1g8a 1gak 1gbs 1gce 1gcu 1gnd
1gnv 1gp3 1gpp 1gpr 1gqe 1gqn 1gqz 1gs9 1gsh 1gsm 1gso
1gvl 1gvp 1gxn 1gxq 1gyu 1gyv 1gz2 1h13 1h4a 1h6t 1h6u
1h75 1h99 1hbq 1hcl 1hcv 1hf8 1hka 1hlw 1hnk 1hq8 1huf
1huw 1hyp 1hzt 1i04 1i1x 1i2h 1i39 1i60 1i9y 1iap 1icx
1ifc 1ifg 1ift 1ihc 1iiw 1ijb 1ik6 1iko 1ilk 1io1 1io2
1iqq 1is5 1iuk 1iv8 1ixk 1ixl 1iz4 1izm 1j1i 1j23 1j27
1j2a 1j2v 1j33 1j3a 1j5x 1j74 1j7g 1j7x 1j85 1j8b 1j8m
1j8r 1jb3 1jhc 1jhs 1jks 1jl1 1jln 1jmw 1jne 1jos 1jpc
1jpe 1jq0 1jvw 1jy1 1jyh 1jyk 1k1b 1k30 1k4n 1k6k 1k7k
1k8u 1k95 1kcm 1kex 1kf5 1khi 1khx 1kid 1klx 1kmz 1kn3
1knb 1kng 1koe 1kqx 1kr4 1ks9 1kt9 1kw4 1kwb 1kwf 1kxo
1kyh 1kzf 1l2h 1l2l 1l3k 1l6p 1l7r 1lci 1lcl 1lcy 1lfp
1lg7 1lib 1lit 1lkf 1lki 1lln 1lmi 1ln4 1lpl 1lsy 1ltu
1lu4 1lwb 1lzl 1m1h 1m1s 1m5i 1m5t 1md6 1mf7 1mh8 1mi8
1mix 1mla 1mml 1msc 1mtz 1mw7 1mwp 1mzl 1n7e 1n7o 1n93
1n9p 1nar 1nc5 1neu 1nfn 1ng2 1ng6 1nh9 1nig 1nij 1nj4
1nko 1nkr 1nm8 1noa 1nog 1npu 1nri 1nty 1nwa 1nxo 1o0x
1o13 1o20 1o22 1o3u 1o4w 1o50 1o6d 1o8x 1o9g 1oa4 1oem
1ogo 1oi7 1ois 1ojq 1olr 1opc 1otm 1ouv 1ow1 1ox3 1oxj
1oz9 1p14 1p1l 1p2f 1p3c 1p4p 1p9a 1p9h 1pbn 1pbv 1pdo
1pev 1pgs 1pgv 1pgx 1pht 1pko 1pne 1pqe 1prz 1psw 1ptf
1pv5 1pzc 1pze 1q2y 1q42 1q5z 1q8b 1q8i 1qad 1qau 1qcx
1qcz 1qgv 1qkk 1qnt 1qqf 1qto 1qts 1qw2 1qwk 1qx3 1qyu
1qz1 1qzm 1qzn 1r1w 1r26 1r29 1r2d 1r3d 1r3f 1r5q 1r62
1r7j 1r8n 1r9h 1r9w 1rc9 1rfe 1rgp 1rh9 1rhs 1ri6 1ris
```

1riy 1rj1 1rl0 1rl6 1roa 1rss 1rw7 1rwr 1rwz 1ryb 1rz2
1s21 1s29 1s7i 1s7k 1sau 1sbx 1sdo 1se8 1sfp 1sjy 1sll
1smb 1snt 1sqg 1sqh 1sqw 1sr8 1srv 1sur 1suu 1t07 1t1d
1t1u 1t2i 1t3y 1t45 1t5i 1t6l 1t95 1tca 1td4 1ten 1tfe
1tfu 1tgn 1thx 1tib 1tig 1tje 1tk1 1tm2 1tmy 1tol 1tp6
1tq3 1tqg 1tr9 1ts9 1tua 1tuo 1tvq 1twu 1txj 1tyv 1tzv
1u09 1u2h 1u2k 1u3y 1u53 1u5x 1u77 1u9a 1u9c 1u9p 1uai
1uaj 1uch 1uek 1ugn 1uh9 1ukf 1uln 1ulr 1unp 1uoh 1uok
1upi 1us2 1usg 1usm 1uxo 1uyl 1v05 1v0l 1v0s 1v2z 1v4a
1v6t 1v77 1v7q 1v8e 1v8i 1vaj 1vf8 1vgj 1vhf 1vi4 1vin
1vjk 1vjs 1vjv 1vk4 1vku 1vlo 1vpk 1vpr 1w41 1w7b 1w7l
1w8u 1wba 1wd3 1wde 1wer 1wgb 1whi 1who 1wj9 1wka 1wly
1wna 1wnh 1wos 1wou 1wp5 1wpa 1wr2 1wrj 1wvh 1wwc 1wwi
1wyc 1x1e 1x3o 1x42 1x56 1x6j 1x6z 1x91 1xak 1xaw 1xba
1xdw 1xdz 1xfk 1xgw 1xix 1xkr 1xqo 1xte 1xw8 1y0k 1y2q
1y4c 1y6i 1y6x 1y9u 1yal 1yck 1yh2 1yhf 1yhv 1yks 1yoy
1yqb 1yrw 1yt4 1ytq 1yvr 1yw5 1yzf 1z0c 1z0p 1z21 1z3x
1z9l 1zce 1zcj 1zcu 1zd8 1zeq 1zgk 1zhv 1zlb 1zon 1zpw
1zuh 1zzk 2a4d 2a4v 2a6b 2a6z 2ad1 2ah5 2ahe 2ahn 2aif
2ap3 2aq5 2axo 2b02 2b0a 2b0j 2b1e 2b1k 2b29 2b2h 2b3m
2b49 2b4w 2b61 2b78 2baa 2bce 2bjq 2bjv 2bk8 2bog 2bqx
2byg 2bzv 2c6u 2c83 2cav 2cb1 2cbe 2cg7 2cgq 2ci2 2ci3
2cit 2ciu 2cjj 2ckx 2cl2 2cl3 2cvb 2cw4 2cwc 2cwr 2cwy
2cxa 2cxc 2cxh 2cxy 2cy7 2cyg 2czt 2d4l 2d4p 2d4x 2d59
2d7j 2d80 2d8e 2dfa 2dhq 2dp9 2dvk 2dvp 2dwk 2dyi 2e01
2e0t 2e2c 2e3h 2e3i 2e3s 2e7v 2e8b 2e8f 2ebb 2ece 2eey
2egu 2ehg 2ejx 2end 2eng 2erf 2es9 2esk 2eyi 2f15 2f1n
2f1s 2f1y 2f4q 2f68 2f6e 2f8h 2f9f 2fb6 2fb9 2fbo 2fbq
2fc3 2fcb 2fcf 2fd4 2fd5 2fem 2fez 2fg1 2fh7 2fi9 2fj8
2fk9 2fl4 2fl7 2fm9 2fo3 2fph 2fq3 2fq4 2fr2 2frg 2fu0
2fuj 2fuk 2fwg 2fwm 2fxq 2fyo 2fzp 2g2d 2g3a 2g40 2g5d
2g5x 2g69 2g7i 2g7s 2gau 2gbn 2gdn 2gen 2ggo 2gkg 2grc
2gs5 2gxg 2gyz 2gzq 2gzv 2h14 2h2c 2h2z 2h30 2h6d 2h7o
2h8e 2h8o 2hc8 2hdz 2he7 2hl9 2hlr 2hly 2hng 2hoq 2hp7
2hpw 2hq1 2hqv 2hrz 2huj 2hvm 2hwx 2hy7 2hzr 2i1u 2i3u
2i49 2i5d 2i5h 2i6v 2i9i 2iai 2ibl 2ict 2id7 2ie8 2igc
2iia 2ilr 2im9 2in0 2ios 2isb 2iu1 2iug 2ivy 2iwn 2iwq
2iwr 2ixm 2iy9 2j6b 2j70 2j71 2j9v 2jay 2jcp 2jeq 2jhy
2jic 2jjf 2jjw 2jli 2jlj 2lao 2lis 2nml 2nmu 2nq5 2nr7
2nrk 2nrr 2nsn 2nwd 2nwv 2nx2 2nxc 2nyc 2o0q 2o37 2o4d
2o6x 2o71 2o8p 2oa2 2ob5 2obi 2obt 2oc3 2odl 2oeb 2of3
2og3 2og4 2ogq 2oix 2okt 2oo2 2op6 2opj 2opw 2osa 2ouf
2ouj 2ova 2oy7 2oyz 2ozf 2p25 2p4d 2p4h 2p52 2p5d 2p65
2p84 2p8t 2p9b 2pb7 2pbo 2pbp 2pcy 2pe8 2pef 2pet 2pge
2pgx 2pko 2plc 2pln 2plq 2pmr 2pnd 2pne 2ppn 2ptd 2pth
2pts 2ptv 2pwq 2pz4 2q0z 2q12 2q34 2q43 2q4n 2q5x 2q82
2q99 2q9v 2qah 2qbv 2qdj 2qev 2qff 2qgm 2qht 2qjx 2qk1
2qni 2qol 2qpw 2qq5 2qqy 2qr3 2qr6 2qt4 2qug 2qvg 2qvk
2qvo 2qy9 2qyw 2qzq 2qzu 2r0s 2r2y 2r4q 2r60 2r6q 2ra8

2rb8 2rci 2rer 2rey 2rfa 2rfr 2rh3 2rhe 2rik 2ris 2rjd
2rk3 2rk5 2rkh 2rkq 2rkt 2sga 2sil 2tgi 2uvj 2uyo 2v4x
2v5b 2v75 2vc8 2vfb 2vfy 2vga 2vh7 2vim 2vjw 2vo8 2vpk
2vq4 2vsd 2vua 2vwr 2vy6 2vy8 2w0g 2w0i 2w1r 2w2r 2w61
2w8t 2wb3 2wbn 2wbz 2wh7 2wj5 2wmf 2wn4 2wnk 2woz 2wpg
2wso 2wux 2wwe 2x35 2x5y 2ygs 2yv4 2yv8 2yvn 2yvq 2yvt
2ywj 2ywk 2ywn 2yxb 2yxf 2yxm 2yxp 2yyo 2yz8 2z0m 2z10
2z14 2z1e 2z2i 2z5l 2z62 2z63 2z6p 2z84 2za7 2zbs 2zco
2zdb 2zfy 2zgr 2zhv 2zj8 2zov 2zpd 2zpm 2zq5 2zqe 2zrr
2zty 2zyo 3a0x 3a4c 3a7f 3a7l 3aap 3ado 3app 3b33 3b5o
3b79 3ba1 3bb7 3bci 3bcy 3bhs 3bn6 3bok 3bor 3bpv 3bqe
3bqx 3bu9 3but 3bzg 3bzq 3bzt 3c0f 3c2e 3c4b 3c5v 3c65
3c8u 3c8x 3c97 3caf 3can 3cb6 3cbh 3cbn 3ce7 3civ 3cj1
3cjw 3ckf 3cm0 3cml 3cnu 3co1 3cou 3cq1 3crm 3cs1 3csg
3ct5 3ctg 3ctk 3cx2 3cxr 3d2a 3d79 3dfg 3dgf 3dhn 3dj9
3dkm 3dlm 3dml 3dms 3dqp 3dsh 3du1 3dz1 3e0e 3e0h 3e5n
3e5t 3e7p 3e9o 3eaz 3eds 3eg4 3eiz 3ejc 3ejf 3ejg 3emi
3enu 3eo5 3eod 3erb 3ers 3esu 3etp 3etv 3eur 3evn 3evp
3exv 3ey6 3eye 3ezm 3f1x 3f2z 3f40 3f6f 3f6j 3f7m 3f8t
3f9q 3fcn 3fdr 3fh2 3fhf 3fkc 3flg 3fn7 3frr 3fs5 3ftd
3ftj 3ftt 3fwu 3fxh 3fy3 3fyq 3fze 3g06 3g39 3g40 3g4p
3g6l 3g9b 3g9t 3gbw 3gd0 3gfp 3ghj 3gms 3gre 3grh 3gs9
3gt0 3gvj 3gw3 3h04 3h2g 3h47 3h6j 3h6q 3h8z 3h9w 3ha9
3hak 3hc7 3hd4 3hdc 3hgl 3hkx 3hn7 3hnx 3hny 3hol 3hp4
3hqx 3htv 3hut 3hvv 3hvw 3hwu 3hxl 3hz8 3i0v 3i2n 3i2v
3i47 3i7m 3id1 3ilv 3im1 3isu 3iu5 3iv4 3jrz 3js6 3jsn
3jte 3ju0 3jve 3jz9 3jzz 3k0m 3k29 3k6u 3k8u 3kb5 3kbg
3kci 3kg4 3kjt 3kp8 3kq5 3kr9 3kt2 3kt9 3kv0 3l83 3lax
3leq 3lji 3lmd 3lmo 3psg 3pte 3seb 3tgl 3tss 3wrp 4eug
4pep 4tsv 6xia 8abp

Bibliography

[1] Anscombe

F. J. Anscombe. "Graphs in Statistical Analysis". In: *The American Statistician* 27.1 (1973), pp. 17–21. ISSN: 00031305. URL: http://www.jstor.org/stable/2682899.

Graphs are essential to good statistical analysis. Ordinary scatterplots and "triple" scatterplots are discussed in relation to regression analysis.

[2] Azran and Ghahramani

Arik Azran and Zoubin Ghahramani. "Spectral Methods for Automatic Multiscale Data Clustering". In: *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2006 (CVPR06)* (2006), pp. 190–197. DOI: 10.1109/CVPR.2006.289. URL: http://portal.acm.org/citation.cfm?id=1273503.

We consider the problem of multiclass classification where both labeled and unlabeled data points are given. We introduce and demonstrate a new approach for estimating a distribution over the missing labels where data points are viewed as nodes of a graph, and pairwise similarities are used to derive a transition probability matrix **P** for a Markov random walk between them. The algorithm associates each point with a particle which moves between points according to **P**. Labeled points are set to be absorbing states of the Markov random walk, and the probability of each particle to be absorbed by the different labeled points, as the number of steps increases, is then used to derive a distribution over the associated missing label. A computationally efficient algorithm to implement this is derived and demonstrated on both real and artificial data sets, including a numerical comparison with other methods.

[3] Baldi and Pollastri

P Baldi and G Pollastri. "The principled design of large-scale recursive neural network architectures–DAG-RNNs and the protein structure prediction problem". In: *J. Machine Learning Res* 4 (2003), pp. 575–602.

We describe a general methodology for the design of large-scale recursive neural network architectures (DAG-RNNs) which comprises three fundamental steps: (1) representation of a given domain using suitable directed acyclic graphs (DAG) to connect visible and hidden node variables; (2) parameterization of the relationship between each variable and its parent variables by feedforward neural networks; and (3) application of weight-sharing within appropriate subsets of DAG connections to capture stationarity and control model complexity. Here we use these principles to derive several *specific* classes of DAG-RNN architectures based on lattices, trees, and other structured graphs. These architectures can process a wide range of data structures with variable sizes and dimensions. While the overall resulting models remain probabilistic, the internal deterministic dynamics allows efficient propagation of information, as well as training by gradient descent, in order to tackle large-scale problems. These methods are used here to derive state-of-the-art predictors for protein structural features such as secondary structure (1D) and both fine- and coarse-grained contact maps (2D). Extensions, relationships to graphical models, and implications for the design of neural architectures are briefly discussed. The protein prediction servers are available over the Web at: www.igb.uci.edu/tools.htm.

[4] Bauer et al.

Raphael André Bauer et al. "Fast Structural Alignment of Biomolecules Using a Hash Table, N-Grams and String Descriptors". In: *Algorithms* 2.2 (2009), pp. 692–709. ISSN: 1999-4893. DOI: 10.3390/a2020692. URL: http://www.mdpi.com/1999-4893/2/2/692/.

This work presents a generalized approach for the fast structural alignment of thousands of macromolecular structures. The method uses string representations of a macromolecular structure and a hash table that stores n-grams of a certain size for searching. To this end, macromolecular structure-to-string translators were implemented for protein and RNA structures. A query against the index is performed in two hierarchical steps to unite speed and precision. In the first step the query structure is translated into n-grams, and all target structures containing these n-grams are retrieved from the hash table. In the second step all corresponding n-grams of the query and each target structure are subsequently aligned, and after each alignment a score is calculated based on the matching n-grams of query and target. The extendable framework enables the user to query and structurally align thousands of protein and RNA structures on a commodity machine and is available as open source from http://lajolla.sf.net.

[5] Bava, Gromiha, Uedaira, Kitajima, and Sarai

K. A. Bava, M. M. Gromiha, H. Uedaira, K. Kitajima, and A. Sarai. "ProTherm, version 4.0: thermodynamic database for proteins and mutants". In: *Nucleic Acids Research* 32 (2004), pp. D120–121.

Release 4.0 of ProTherm, thermodynamic database for proteins and mutants, contains approximately 14,500 numerical data (approximately 450version) of several thermodynamic parameters along with experimental methods and conditions, and structural, functional and literature information. The sequence and structural information of proteins is connected with thermodynamic data through links between entries in Protein Data Bank, Protein Information Resource and SWISS-PROT and the data in ProTherm. We have separated the Gibbs free energy change obtained at extrapolated temperature from the data on denaturation temperature measured by the thermal denaturation method. We have added the statistics of amino acid replacements and links to homologous structures to each protein. Further, we have improved the search and display options to enhance search capability through the web interface. ProTherm is freely available at http://gibk26. bse.kyutech.ac.jp/jouhou/Protherm/protherm.html.

[6] Berg, Kreveld, Overmars, and Schwarzkopf

M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry*. Springer. 2000. call no: QA448.D38.C65

[7] Bishop

Christopher M. Bishop. *Pattern recognition and machine learning*. Springer. 2006. call no: Q327.B52

[8] Bondi

A. Bondi. "van der Waals Volumes and Radii". In: *Journal of Physical Chemistry* 68.3 (1964), pp. 441–451. DOI: 10.1021/j100785a001. URL: http://dx.doi.org/10.1021/j100785a001.

Intermolecular van der Waals radii of the nonmetallic elements have been assembled into a list of . recommended. values for volume calculations. These values have been arrived at by selecting from the most reliable X-ray diffraction data those which could be reconciled with crystal density at 0° K. (to give reasonable packing density), gas kinetic collision cross section, critical density, and liquid state properties. A qualitative understanding of the nature of van der Waals radii is provided by correlation with the de Broglie wave length of the outermost valence electron. Tentative values for the van der Waals radii of metallic elements-in metal organic compounds-are proposed. The paper concludes with a list of increments for the volume of molecules impenetrable to thermal collision, the so-called van der Waals volume, and of the corresponding increments in area per molecule.

[9] Bourne and Shindyalov

Philip E. Bourne and Ilya N. Shindyalov. "Structure comparison and alignment". In: *Structural Bioinformatics*. Ed. by Philip E. Bourne and Helge Weissig. Wiley-Liss, Inc., 2003

[10] Breu and Kirkpatrick

Heinz Breu and David G. Kirkpatrick. "Unit disk graph recognition is NP-hard". In: *Comput. Geom. Theory Appl.* 9.1-2 (1998), pp. 3–24. ISSN: 0925-7721. DOI: http://dx.doi.org/10.1016/S0925-7721(97)00014-X.

Unit disk graphs are the intersection graphs of unit diameter closed disks in the plane. This paper gives a polynomial-time reduction from SATISFIABILITY to the problem of recognizing unit disk graphs. Equivalently, it shows that determining if a graph has sphericity 2 or less, even if the graph is planar or is known to have sphericity at most 3, is NP-hard. We show how this reduction can be extended to 3 dimensions, thereby showing that unit sphere graph recognition, or determining if a graph has sphericity 3 or less, is also NP-hard. We conjecture that K-sphericity is NP-hard for all fixed K greater than 1.

[11] Camproux, Tuffery, Chevrolat, Boisvieux, and Hazout

A.C. Camproux, P. Tuffery, J.P. Chevrolat, J.F. Boisvieux, and S. Hazout. "Hidden Markov model approach for identifying the modular framework of the protein backbone". In: *Protein Engineering* 12.12 (1999), pp. 1063–1073. DOI: 10.1093/protein/12.12.1063. URL: http://peds.oxfordjournals.org/cgi/content/abstract/12/12/1063.

The hidden Markov model (HMM) was used to identify recurrent short 3D structural building blocks (SBBs) describing protein backbones, independently of any a priori knowledge. Polypeptide chains are decomposed into a series of short segments defined by their inter-alpha-carbon distances. Basically, the model takes into account the sequentiality of the observed segments and assumes that each one corresponds to one of several possible SBBs. Fitting the model to a database of non-redundant proteins allowed us to decode proteins in terms of 12 distinct SBBs with different roles in protein structure. Some SBBs correspond to classical regular secondary structures. Others correspond to a significant subdivision of their bounding regions previously considered to be a single pattern. The major contribution of the HMM is that this model implicitly takes into account the sequential connections between SBBs and thus describes the most probable pathways by which the blocks are connected to form the framework of the protein structures. Validation of the SBBs code was performed by extracting SBB series repeated in recoding proteins and examining their structural similarities. Preliminary results on the sequence specificity of SBBs suggest promising perspectives for the prediction of SBBs or series of SBBs from the protein sequences.

[12] Cheng and Baldi

Jianlin Cheng and Pierre Baldi. "Improved residue contact prediction using support vector machines and a large feature set". In: *BMC bioinformatics* 8.1 (2007), p. 113. ISSN: 1471-2105. DOI: 10.1186/1471-2105-8-113. URL: http://www.ncbi.nlm.nih.gov/pubmed/17407573.

Background: Predicting protein residue-residue contacts is an important 2D prediction task. It is useful for ab initio structure prediction and understanding protein folding. In spite of steady progress over the past decade, contact prediction remains still largely unsolved.
Results: Here we develop a new contact map predictor (SVMcon) that uses support vector machines to predict medium- and long-range contacts. SVMcon integrates profiles, secondary structure, relative solvent accessibility, contact potentials, and other useful features. On the same test data set, SVMcon's accuracy is 4% higher than the latest version of the CMAPpro contact map predictor. SVMcon recently participated in the seventh edition of the Critical Assessment of Techniques for Protein Structure Prediction (CASP7) experiment and was evaluated along with seven other contact map predictors. SVMcon was ranked as one of the top predictors, yielding the second best coverage and accuracy for contacts with sequence separation > or = 12 on 13 de novo domains.
Conclusion: We describe SVMcon, a new contact map predictor that uses SVMs and a large set of informative features. SVMcon yields good performance on medium- to long-range contact predictions and can be modularly incorporated into a structure prediction pipeline.

[13] Clayden, Greeves, Warren, and Wothers

Jonathan Clayden, Nick Greeves, Stuart Warren, and Peter Wothers. *Organic Chemistry*. Oxford University Press. 2000

[14] Cleveland

William S. Cleveland. *The elements of graphing data*. AT&T Bell Laboratories. 1994. call no: QA90.C54

[15] Dean and Ghemawat

Jeffrey Dean and Sanjay Ghemawat. *MapReduce: Simplified Data Processing on Large Clusters*. Tech. rep. Google, 2004. URL: http://labs.google.com/papers/mapreduce-osdi04.pdf.

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shown in the paper.
Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling machine failures, and managing the required inter-machine communication. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system.
Our implementation of MapReduce runs on a large cluster of commodity machines and is highly scalable: a typical MapReduce computation processes many terabytes of data on thousands of machines. Programmers find the system easy to use: hundreds of MapReduce programs have been implemented and upwards of one thousand MapReduce jobs are executed on Google's clusters every day.

[16] Dhillon, Guan, and Kulis

Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. "Kernel k-means: spectral clustering and normalized cuts". In: *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. Seattle, WA, USA: ACM, 2004, pp. 551–556. ISBN: 1-58113-888-1. DOI: 10.1145/1014052.1014118. URL: http://dx.doi.org/10.1145/1014052.1014118.

Kernel k-means and spectral clustering have both been used to identify clusters that are non-linearly separable in input space. Despite significant research, these methods have remained only loosely related. In this paper, we give an explicit theoretical connection between them. We show the generality of the weighted kernel k-means objective function, and derive the spectral clustering objective of normalized cut as a special case. Given a positive definite similarity matrix, our results lead to a novel weighted kernel k-means algorithm that monotonically decreases the normalized cut. This has important implications: a) eigenvector-based algorithms, which can be computationally prohibitive, are not essential for minimizing normalized cuts, b) various techniques, such as local search and acceleration schemes, may be used to improve the quality as well as speed of kernel k-means. Finally, we present results on several interesting data sets, including diametrical clustering of large gene-expression matrices and a handwriting recognition data set.

[17] Dyson and Wright

H. Jane Dyson and Peter E. Wright. "Intrinsically unstructured proteins and their functions". In: *Nature Reviews Molecular Cell Biology* 6.3 (2005), pp. 197–208. ISSN: 1471-0072. DOI: 10.1038/nrm1589. URL: http://dx.doi.org/10.1038/nrm1589.

Many gene sequences in eukaryotic genomes encode entire proteins or large segments of proteins that lack a well-structured three-dimensional fold. Disordered regions can be highly conserved between species in both composition and sequence and, contrary to the traditional view that protein function equates with a stable three-dimensional structure, disordered regions are often functional, in ways that we are only beginning to discover. Many disordered segments fold on binding to their biological targets (coupled folding and binding), whereas others constitute flexible linkers that have a role in the assembly of macromolecular arrays.

[18] Etchebest, Benros, Hazout, and Brevern

C. Etchebest, C. Benros, S. Hazout, and A. G. de Brevern. "A structural alphabet for local protein structures: improved prediction methods". In: *Proteins* 59 (2005), pp. 810–827.

Three-dimensional protein structures can be described with a library of 3D fragments that define a structural alphabet. We have previously proposed such an alphabet, composed of 16 patterns of five consecutive amino acids, called Protein Blocks (PBs). These PBs have been used to describe protein backbones and to predict local structures from protein sequences. The Q16 prediction rate reaches 40.7% with an optimization procedure. This article examines two aspects of PBs. First, we determine the effect of the enlargement of databanks on their definition. The results show that the geometrical features of the different PBs are preserved (local RMSD value equal to 0.41 Å on average) and sequence-structure specificities reinforced when databanks are enlarged. Second, we improve the methods for optimizing PB predictions from sequences, revisiting the optimization procedure and exploring different local prediction strategies. Use of a statistical optimization procedure for the sequence-local structure relation improves prediction accuracy by 8% (Q16 = 48.7%). Better recognition of repetitive structures occurs without losing the prediction efficiency of the other local folds. Adding secondary structure prediction improved the accuracy of Q16 by only 1%. An entropy index (Neq), strongly related to the RMSD value of the difference between predicted PBs and true local structures, is proposed to estimate prediction quality. The Neq is linearly correlated with the Q16 prediction rate distributions, computed for a large set of proteins. An "expected" prediction rate QE16 is deduced with a mean error of 5%.

[19] Fariselli, Olmea, Valencia, and Casadio

P Fariselli, O Olmea, a Valencia, and R Casadio. "Prediction of contact maps with neural networks and correlated mutations." In: *Protein engineering* 14.11 (2001), pp. 835–43. ISSN: 0269-2139. URL: http://www.ncbi.nlm.nih.gov/pubmed/11742102.

Contact maps of proteins are predicted with neural network-based methods, using as input codings of increasing complexity including evolutionary information, sequence conservation, correlated mutations and predicted secondary structures. Neural networks are trained on a data set comprising the contact maps of 173 non-homologous proteins as computed from their well resolved three-dimensional structures. Proteins are selected from the Protein Data Bank database provided that they align with at least 15 similar sequences in the corresponding families. The predictors are trained to learn the association rules between the covalent structure of each protein and its contact map with a standard back propagation algorithm and tested on the same protein set with a cross-validation procedure. Our results indicate that the method can assign protein contacts with an average accuracy of 0.21 and with an improvement over a random predictor of a factor >6, which is higher than that previously obtained with methods only based either on neural networks or on correlated mutations. Furthermore, filtering the network outputs with a procedure based on the residue coordination numbers, the accuracy of predictions increases up to 0.25 for all the proteins, with an 8-fold deviation from a random predictor. These scores are the highest reported so far for predicting protein contact maps.

[20] Frasconi, Gori, and Sperduti

P. Frasconi, M. Gori, and A. Sperduti. "A general framework for adaptive processing of data structures." In: *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council* 9.5 (1998), pp. 768–86. ISSN: 1045-9227. DOI: 10.1109/72.712151. URL: http://www.ncbi.nlm.nih.gov/pubmed/18255765.

A structured organization of information is typically required by symbolic processing. On the other hand, most connectionist models assume that data are organized according to relatively poor structures, like arrays or sequences. The framework described in this paper is an attempt to unify adaptive models like artificial neural nets and belief nets for the problem of processing structured information. In particular, relations between data variables are expressed by directed acyclic graphs, where both numerical and categorical values coexist. The general framework proposed in this paper can be regarded as an extension of both recurrent neural networks and hidden Markov models to the case of acyclic graphs. In particular we study the supervised learning problem as the problem of learning transductions from an input structured space to an output structured space, where transductions are assumed to admit a recursive hidden state-space representation. We introduce a graphical formalism for representing this class of adaptive transductions by means of recursive networks, i.e., cyclic graphs where nodes are labeled by variables and edges are labeled by generalized delay elements. This representation makes it possible to incorporate the symbolic and subsymbolic nature of data. Structures are processed by unfolding the recursive network into an acyclic graph called encoding network. In so doing, inference and learning algorithms can be easily inherited from the corresponding algorithms for artificial neural networks or probabilistic graphical model.

[21] Friedberg et al.

Iddo Friedberg et al. "Using an alignment of fragment strings for comparing protein structures". In: *Bioinformatics* 23 (2006), pp. 219–224. DOI: 10.1093/bioinformatics/btl310.

Motivation: Most methods that are used to compare protein structures use three-dimensional (3D) structural information. At the same time, it hasbeenshownthata 1D string representationof local protein structure retains a degree of structural information. This type of representation can be a powerful tool for protein structure comparison and classification, given the arsenal of sequence comparison tools developed by computational biology. However, in order to do so, there is a need to first understand how much information is contained in various possible 1D representations of protein structure. Results: Here we describe the use of a particular structure fragment library, denoted here as KL-strings, for the 1D representation of protein structure. Using KL-strings,wedevelop an infrastructure for comparing protein structures with a 1D representation. This study focuses on the added value gained from such a description. We show the new local structure language adds resolution to the traditional three-state (helix, strand and coil) secondary structure description, and provides a high degree of accuracy in recognizing structural similarities when used with a pairwise alignment benchmark. The results of this study have immediate applications towards fast structure recognition, and for fold prediction and classification.

[22] Goodsell

David Goodsell. *Bionanotechnology : lessons from nature*. Wiley-Liss. 2004. call no: QP514.2.G658

David Goodsell. *The machinery of life*. Springer-Verlag. 1993. call no: QH581.2.G66

[24] Gusfield

Dan Gusfield. *Algorithms on strings, trees, and sequences : computer science and computational biology*. Cambridge Univ. Press. 1997. URL: http://www.amazon.com/exec/obidos/ASIN/0521585198

[25] Guyon, Camproux, Hochez, and Tuffery

Frederic Guyon, Anne-Claude Camproux, Joelle Hochez, and Pierre Tuffery. "SA-Search: a web tool for protein structure mining based on a Structural Alphabet". In: *Nucleic Acids Research* 32 (2004), W545–548. DOI: 10.1093/nar/gkh467. URL: http://nar.oxfordjournals.org/cgi/content/abstract/32/suppl_2/W545.

SA-Search is a web tool that can be used to mine for protein structures and extract structural similarities. It is based on a hidden Markov model derived Structural Alphabet (SA) that allows the compression of three-dimensional (3D) protein conformations into a one-dimensional (1D) representation using a limited number of prototype conformations. Using such a representation, classical methods developed for amino acid sequences can be employed. Currently, SA-Search permits the performance of fast 3D similarity searches such as the extraction of exact words using a suffix tree approach, and the search for fuzzy words viewed as a simple 1D sequence alignment problem. SA-Search is available at http://bioserv.rpbs.jussieu.fr/cgi-bin/SA-Search.

[26] Guyon, Luxburg, and Williamson

Isabelle Guyon, Ulrike von Luxburg, and Robert C. Williamson. "Clustering: Science or Art? Towards Principled Approaches". In: NIPS *2009 Workshop* (2009). URL: http://stanford.edu/~rezab/nips2009workshop/.

This paper deals with the question whether the quality of different clustering algorithms can be compared by a general, scientifically sound procedure which is independent of particular clustering algorithms. In our opinion, the major obstacle is the diffculty to evaluate a clustering algorithm without taking into account the context: why does the user cluster his data in the first place, and what does he want to do with the clustering afterwards? We suggest that clustering should not be treated as an application-independent mathematical problem, but should always be studied in the context of its end-use. Different techniques to evaluate clustering algorithms have to be developed for different uses of clustering. To simplify this procedure it will be useful to build a "taxonomy of clustering problems" to identify clustering applications which can be treated in a unified way.

[27] Hadley and Jones

C. Hadley and D. T. Jones. "A systematic comparison of protein structure classifications: SCOP, CATH and FSSP". In: *Structure* 7 (1999), pp. 1099–1112. URL: http://www.ncbi.nlm.nih.gov/pubmed/10508779.

Background: Several methods of structural classification have been developed to introduce some order to the large amount of data present in the Protein Data Bank. Such methods facilitate structural comparisons and provide a greater understanding of structure and function. The most widely used and comprehensive databases are SCOP, CATH and FSSP, which represent three unique methods of classifying protein structures: purely manual, a combination of manual and automated, and purely automated, respectively. In order to develop reliable template libraries and benchmarks for protein-fold recognition, a systematic comparison of these databases has been carried out to determine their overall agreement in classifying protein structures.
Results: Approximately two-thirds of the protein chains in each database are common to all three databases. Despite employing different methods, and basing their systems on different rules of protein structure and taxonomy, SCOP, CATH and FSSP agree on the majority of their classifications. Discrepancies and inconsistencies are accounted for by a small number of explanations. Other interesting features have been identified, and various differences between manual and automatic classification methods are presented.
Conclusion: Using these databases requires an understanding of the rules upon which they are based; each method offers certain advantages depending on the biological requirements and knowledge of the user. The degree of discrepancy between the systems also has an impact on reliability of prediction methods that employ these schemes as benchmarks. To generate accurate fold templates for threading, we extract information from a consensus database, encompassing agreements between SCOP, CATH and FSSP.

[28] Han and Baker

K. F. Han and D. Baker. "Recurring local sequence motifs in proteins". In: *Journal of Molecular Biology* 251 (1995), pp. 176–187. URL: www.ncbi.nlm.nih.gov/pubmed/7643386.

We describe a completely automated approach to identifying local sequence motifs that transcend protein family boundaries. Cluster analysis is used to identify recurring patterns of variation at single positions and in short segments of contiguous positions in multiple sequence alignments for a non-redundant set of protein families. Parallel experiments on simulated data sets constructed with the overall residue frequencies of proteins but not the inter-residue correlations show that naturally occurring protein sequences are significantly more clustered than the corresponding random sequences for window lengths ranging from one to 13 contiguous positions. The patterns of variation at single positions are not in general surprising: chemically similar amino acids tend to be grouped together. More interesting patterns emerge as the window length increases. The patterns of variation for longer window lengths are in part recognizable patterns of hydrophobic and hydrophilic residues, and in part less obvious combinations. A particularly interesting class of patterns features highly conserved glycine residues. The patterns provide a means to abstract the information contained in multiple sequence alignments and may be useful for comparison of distantly related sequences or sequence families and for protein structure prediction.

[29] Hasegawa and Holm

Hitomi Hasegawa and Liisa Holm. "Advances and pitfalls of protein structural alignment." In: *Current opinion in structural biology* 19.3 (2009), pp. 341–8. ISSN: 1879-033X. DOI: 10.1016/j.sbi.2009.04.003. URL: http://www.ncbi.nlm.nih.gov/pubmed/19481444.

Structure comparison opens a window into the distant past of protein evolution, which has been unreachable by sequence comparison alone. With 55,000 entries in the Protein Data Bank and about 500 new structures added each week, automated processing, comparison, and classification are necessary. A variety of methods use different representations, scoring functions, and optimization algorithms, and they generate contradictory results even for moderately distant structures. Sequence mutations, insertions, and deletions are accommodated by plastic deformations of the common core, retaining the precise geometry of the active site, and peripheral regions may refold completely. Therefore structure comparison methods that allow for flexibility and plasticity generate the most biologically meaningful alignments. Active research directions include both the search for fold invariant features and the modeling of structural transitions in evolution. Advances have been made in algorithmic robustness, multiple alignment, and speeding up database searches.

[30] Holland, Veretnik, Shindyalov, and Bourne

T. A. Holland, S. Veretnik, I. N. Shindyalov, and P. E. Bourne. "Partitioning protein structures into domains: why is it so difficult?" In: *Journal of Molecular Biology* 361 (2006), pp. 562–590. URL: www.ncbi.nlm.nih.gov/pubmed/16863650.

This analysis takes an in-depth look into the difficulties encountered by automatic methods for domain decomposition from three-dimensional structure. The analysis involves a multi-faceted set of criteria including the integrity of secondary structure elements, the tendency toward fragmentation of domains, domain boundary consistency and topology. The strength of the analysis comes from the use of a new comprehensive benchmark dataset, which is based on consensus among experts (CATH, SCOP and AUTHORS of the 3D structures) and covers 30 distinct architectures and 211 distinct topologies as defined by CATH. Furthermore, over 66structures are multi-domain proteins; each domain combination occurring once per dataset. The performance of four automatic domain assignment methods, DomainParser, NCBI, PDP and PUU, is carefully analyzed using this broad spectrum of topology combinations and knowledge of rules and assumptions built into each algorithm. We conclude that it is practically impossible for an automatic method to achieve the level of performance of human experts. However, we propose specific improvements to automatic methods as well as broadening the concept of a structural domain. Such work is prerequisite for establishing improved approaches to domain recognition. (The benchmark dataset is available from http://pdomains.sdsc.edu).

[31] Holm and Sander

L. Holm and C. Sander. "The FSSP database of structurally aligned protein fold families". In: *Nucleic Acids Research* 22 (1994), pp. 3600–3609. URL: http://www.ncbi.nlm.nih.gov/pubmed/7937067.

FSSP (families of structurally similar proteins) is a database of structural alignments of proteins in the Protein Data Bank (PDB). The database currently contains an extended structural family for each of 330 representative protein chains. Each data set contains structural alignments of one search structure with all other structurally significantly similar proteins in the representative set (remote homologs, < 30identity), as well as all structures in the Protein Data Bank with 70-30sequence identity relative to the search structure (medium homologs). Very close homologs (above 70marked structural differences. The alignments of remote homologs are the result of pairwise all-against-all structural comparisons in the set of 330 representative protein chains. All such comparisons are based purely on the 3D co-ordinates of the proteins and are derived by automatic (objective) structure comparison programs. The significance of structural similarity is estimated based on statistical criteria. The FSSP database is available electronically from the EMBL file server and by anonymous ftp (file transfer protocol).

[32] Holm and Sander

Liisa Holm and Chris Sander. "Parser for Protein Folding Units". In: *Proteins: Structure, Function, and Genetics* 268 (1994), pp. 256–268.

General patterns of protein structural organization have emerged from studies of hundreds of structures elucidated by X-ray crystallography and nuclear magnetic resonance. Structural units are commonly identified by visual inspection of molecular models using qualitative criteria. Here, we propose an algorithm for identification of structural units by objective, quantitative criteria based on atomic interactions. The underlying physical concept is maximal interactions within each unit and minimal interaction between units (domains). In a simple harmonic approximation, interdomain dynamics is determined by the strength of the interface and the distribution of masses. The most likely domain decomposition involves units with the most correlated motion, or largest interdomain fluctuation time. The decomposition of a convoluted 3-D structure is complicated by the possibility that the chain can cross over several times between units. Grouping the residues by solving an eigenvalue problem for the contact matrix reduces the problem to a one-dimensional search for all reasonable trial bisections. Recursive bisection yields a tree of putative folding units. Simple physical criteria are used to identify units that could exist by themselves. The units so defined closely correspond to crystallographers' notion of structural domains. The results are useful for the analysis of folding principles, for modular protein design and for protein engineering.

Liisa Holm and Chris Sander. "Protein Structure Comparison by Alignment of Distance Matrices". In: *Journal of Molecular Biology* 233.1 (1993), pp. 123–138. ISSN: 00222836. DOI: 10.1006/jmbi.1993.1489. URL: http://dx.doi.org/10.1006/jmbi.1993.1489.

With a rapidly growing pool of known tertiary structures, the importance of protein structure comparison parallels that of sequence alignment. We have developed a novel algorithm (DALI) for optimal pairwise alignment of protein structures. The three-dimensional co-ordinates of each protein are used to calculate residue–residue (C_α C_α) distance matrices. The distance matrices are first decomposed into elementary contact patterns, e.g. hexapeptide–hexapeptide submatrices. Then, similar contact patterns in the two matrices are paired and combined into larger consistent sets of pairs. A Monte Carlo procedure is used to optimize a similarity score defined in terms of equivalent intramolecular distances. Several alignments are optimized in parallel, leading to simultaneous detection of the best, second-best and so on solutions. The method allows sequence gaps of any length, reversal of chain direction and free topological connectivity of aligned segments. Sequential connectivity can be imposed as an option. The method is fully automatic and identifies structural resemblances and common structural cores accurately and sensitively, even in the presence of geometrical distortions. An all-against-all alignment of over 200 representative protein structures results in an objective classification of known three-dimensional folds in agreement with visual classifications. Unexpected topological similarities of biological interest have been detected, e.g. between the bacterial toxin colicin A and globins, and between the eukaryotic POU-specific DNA-binding domain and the bacterial λ repressor.

[34] Imhof

Edward Imhof. *Cartographic relief presentation*. ERSI Press. 2007

[35] Jaro

Matthew A. Jaro. "Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida". In: *Journal of the American Statistical Association* 84.406 (1989), pp. 414–420. ISSN: 01621459. URL: http://www.jstor.org/stable/2289924.

A test census of Tampa, Florida and an independent postenumeration survey (PES) were conducted by the U.S. Census Bureau in 1985. The PES was a stratified block sample with heavy emphasis placed on hard-to-count population groups. Matching the individuals in the census to the individuals in the PES is an important aspect of census coverage evaluation and consequently a very important process for any census adjustment operations that might be planned. For such an adjustment to be feasible, record-linkage software had to be developed that could perform matches with a high degree of accuracy and that was based on an underlying mathematical theory. A principal purpose of the PES was to provide an opportunity to evaluate the newly implemented record-linkage system and associated methodology. This article discusses the theoretical and practical issues encountered in conducting the matching operation and presents the results of that operation. A review of the theoretical background of the record-linkage problem provides a framework for discussions of the decision procedure, file blocking, and the independence assumption. The estimation of the parameters required by the decision procedure is an important aspect of the methodology, and the techniques presented provide a practical system that is easily implemented. The matching algorithm (discussed in detail) uses the linear sum assignment model to "pair" the records. The Tampa, Florida, matching methodology is described in the final sections of the article. Included in the discussion are the results of the matching itself, an independent clerical review of the matches and nonmatches, conclusions, problem areas, and future work required.

[36] Jones et al.

S. Jones et al. "Domain assignment for protein structures using a consensus approach: characterization and analysis". In: *Protein Science* 7 (1998), pp. 233–242

[37] Kendrew et al.

J. C. Kendrew et al. "A three-dimensional model of the myoglobin molecule obtained by x-ray analysis". In: *Nature* 181 (1958), pp. 662–666

[38] Kolodny, Koehl, and Levitt

R. Kolodny, P. Koehl, and M. Levitt. "Comprehensive evaluation of protein structure alignment methods: scoring by geometric measures". In: *Journal of Molecular Biology* 346 (2005), pp. 1173–1188. URL: http://www.ncbi.nlm.nih.gov/pubmed/15701525.

We report the largest and most comprehensive comparison of protein structural alignment methods. Specifically, we evaluate six publicly available structure alignment programs: SSAP, STRUCTAL, DALI, LSQMAN, CE and SSM by aligning all 8,581,970 protein structure pairs in a test set of 2930 protein domains specially selected from CATH v.2.4 to ensure sequence diversity. We consider an alignment good if it matches many residues, and the two substructures are geometrically similar. Even with this definition, evaluating structural alignment methods is not straightforward. At first, we compared the rates of true and false positives using receiver operating characteristic (ROC) curves with the CATH classification taken as a gold standard. This proved unsatisfactory in that the quality of the alignments is not taken into account: sometimes a method that finds less good alignments scores better than a method that finds better alignments. We correct this intrinsic limitation by using four different geometric match measures (SI, MI, SAS, and GSAS) to evaluate the quality of each structural alignment. With this improved analysis we show that there is a wide variation in the performance of different methods; the main reason for this is that it can be difficult to find a good structural alignment between two proteins even when such an alignment exists. We find that STRUCTAL and SSM perform best, followed by LSQMAN and CE. Our focus on the intrinsic quality of each alignment allows us to propose a new method, called "Best-of-All" that combines the best results of all methods. Many commonly used methods miss 10-50putting existing structural alignments into proper perspective, our study allows better comparison of protein structures. By highlighting limitations of existing methods, it will spur the further development of better structural alignment methods. This will have significant biological implications now that structural comparison has come to play a central role in the analysis of experimental work on protein structure, protein function and protein evolution.

[39] Kolodny and Levitt

R. Kolodny and M. Levitt. "Protein decoy assembly using short fragments under geometric constraints". In: *Biopolymers* 68 (2003), pp. 278–285. DOI: 10.1002/bip.10262. URL: http://www.ncbi.nlm.nih.gov/pubmed/12601789.

A small set of protein fragments can represent adequately all known local protein structure. This set of fragments, along with a construction scheme that assembles these fragments into structures, defines a discrete (relatively small) conformation space, which approximates protein structures accurately. We generate protein decoys by sampling geometrically valid structures from this conformation space, biased by the secondary structure prediction for the protein. Unlike other methods, secondary structure prediction is the only protein-specific information used for generating the decoys. Nevertheless, these decoys are qualitatively similar to those found by others. The method works well for all-alpha proteins, and shows promising results for alpha and beta proteins.

[40] Kolodny, Koehl, Guibas, and Levitt

R. Kolodny, P. Koehl, L. Guibas, and M. Levitt. "Small libraries of protein fragments model native protein structures accurately". In: *Journal of Molecular Biology* 323 (2002), pp. 297–307. URL: http://www.ncbi.nlm.nih.gov/pubmed/12381322.

Prediction of protein structure depends on the accuracy and complexity of the models used. Here, we represent the polypeptide chain by a sequence of rigid fragments that are concatenated without any degrees of freedom. Fragments chosen from a library of representative fragments are fit to the native structure using a greedy build-up method. This gives a one-dimensional representation of native protein three-dimensional structure whose quality depends on the nature of the library. We use a novel clustering method to construct libraries that differ in the fragment length (four to seven residues) and number of representative fragments they contain (25-300). Each library is characterized by the quality of fit (accuracy) and the number of allowed states per residue (complexity). We find that the accuracy depends on the complexity and varies from 2.9 Å for a 2.7-state model on the basis of fragments of length 7-0.76 Å for a 15-state model on the basis of fragments of length 5. Our goal is to find representations that are both accurate and economical (low complexity). The models defined here are substantially better in this regard: with ten states per residue we approximate native protein structure to 1 Å compared to over 20 states per residue needed previously. For the same complexity, we find that longer fragments provide better fits. Unfortunately, libraries of longer fragments must be much larger (for ten states per residue, a seven-residue library is 100 times larger than a five-residue library). As the number of known protein native structures increases, it will be possible to construct larger libraries to better exploit this correlation between neighboring residues. Our fragment libraries, which offer a wide range of optimal fragments suited to different accuracies of fit, may prove to be useful for generating better decoy sets for ab initio protein folding and for generating accurate loop conformations in homology modeling.

[41] Koonin, Wolf, and Karev

Eugene V. Koonin, Yuri I. Wolf, and Georgy P. Karev. "The structure of the protein universe and genome evolution". In: *Nature* 420.218–223 (2002). URL: http://dx.doi.org/10.1038/nature01256.

Despite the practically unlimited number of possible protein sequences, the number of basic shapes in which proteins fold seems not only to be finite, but also to be relatively small, with probably no more than 10,000 folds in existence. Moreover, the distribution of proteins among these folds is highly non-homogeneous . some folds and superfamilies are extremely abundant, but most are rare. Protein folds and families encoded in diverse genomes show similar size distributions with notable mathematical properties, which also extend to the number of connections between domains in multidomain proteins. All these distributions follow asymptotic power laws, such as have been identified in a wide variety of biological and physical systems, and which are typically associated with scale-free networks. These findings suggest that genome evolution is driven by extremely general mechanisms based on the preferential attachment principle.

[42] Koren

Yehuda Koren. *The BellKor Solution to the Netflix Grand Prize*. Tech. rep. BellKor, 2009. URL: http://www.netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pdf.

This article describes part of our contribution to the "Bell-Kor's Pragmatic Chaos" Final solution, which won the Netflix Grand Prize. The other portion of the contribution was created while working at AT&T with Robert Bell and Chris Volinsky, as reported in our 2008 Progress Prize report [3]. The final solution includes all the predictors described there. In this article we describe only the newer predictors. So what is new over last year's solution? First we further improved the baseline predictors (Sec. III). This in turn improves our other models, which incorporate those predictors, like the matrix factorization model (Sec. IV). In addition, an extension of the neighborhood model that addresses temporal dynamics was introduced (Sec. V). On the Restricted Boltzmann Machines (RBM) front, we use a new RBM model with superior accuracy by conditioning the visible units (Sec. VI). The final addition is the introduction of a new blending algorithm, which is based on gradient boosted decision trees (GBDT) (Sec. VII).

[43] Lee and Richards

B. Lee and F.M. Richards. "The interpretation of protein structures: Estimation of static accessibility". In: *Journal of Molecular Biology* 55.3 (1971), pp. 379–400. ISSN: 0022-2836. DOI: 10.1016/0022-2836(71)90324-X. URL: http://dx.doi.org/10.1016/0022-2836(71)90324-X.

A program is described for drawing the van der Waal's surface of a protein molecule. An extension of the program permits the accessibility of atoms, or groups of atoms, to solvent or solute molecules of specified size to be quantitatively assessed. As defined in this study, the accessibility is proportional to surface area. The accessibility of all atoms in the twenty common amino acids in model tripeptides of the type Ala-X-Ala are given for defined conformation. The accessibilities are also given for all atoms in ribonuclease-S, lysozyme and myogoblin. Internal cavities are defined and discussed. Various summaries of these data are provided. Forty to fifty per cent of the surface area of each protein is occupied by non-polar atoms. The actual numerical results are sensitive to the values chosen for the van der Waal's radii of the various groups. Since there is uncertainty over the correct values for these radii, the derived numbers should only be used as a qualitative guide at this stage.
The average change in accessibility for the atoms of all three proteins in going from a hypothetical extended chain to the folded conformation of the native protein is about a factor of 3. This number applies to both polar (nitrogen and oxygen) and non-polar (carbon and sulfur) atoms considered separately. The larger non-polar amino acids tend to be more .buried. in the native form of all three proteins. However, for all classes and for residues within a given class the accessibility changes on folding tend to be highly variable.

[44] Lehninger, Nelson, and Cox

Albert L. Lehninger, David L. Nelson, and Michael M. Cox. *Lehninger Principles of Biochemistry*. W. H. Freeman. 2008. call no: QP514.2.L43

[45] Levenshtein

Vladimir I. Levenshtein. *Binary codes capable of correcting deletions, insertions, and reversals*. Tech. rep. 8. 1966, pp. 707–710

[46] Levitt and Chothia

M. Levitt and C. Chothia. "Structural patterns in globular proteins". In: *Nature* 261 (1976), pp. 552–558. URL: http://www.ncbi.nlm.nih.gov/pubmed/934293.

A simple diagrammatic representation has been used to show the arrangement of alpha helices and beta sheets in 31 globular proteins, which are classified into four clearly separated classes. The observed arrangements are significantly non-random in that pieces of secondary structure adjacent in sequence along the polypeptide chain are also often in contact in three dimensions.

[47] Licklider

J.C.R. Licklider. "Man-computer symbiosis". In: IRE *Transactions on Human Factors in Electronics* HFE-1 (1960), pp. 4–11.

Man-computer symbiosis is an expected development in cooperative interaction between men and electronic computers. It will involve very close coupling between the human and the electronic members of the partnership. The main aims are 1) to let computers facilitate formulative thinking as they now facilitate the solution of formulated problems, and 2) to enable men and computers to cooperate in making decisions and controlling complex situations without inflexible dependence on predetermined programs. In the anticipated symbiotic partnership, men will set the goals, formulate the hypotheses, determine the criteria, and perform the evaluations. Computing machines will do the routinizable work that must be done to prepare the way for insights and decisions in technical and scientific thinking. Preliminary analyses indicate that the symbiotic partnership will perform intellectual operations much more effectively than man alone can perform them. Prerequisites for the achievement of the effective, cooperative association include developments in computer time sharing, in memory components, in memory organization, in programming languages, and in input and output equipment.

[48] Luxburg

Ulrike Luxburg. "A tutorial on spectral clustering". In: *Statistics and Computing* 17.4 (2007), pp. 395–416. ISSN: 0960-3174. DOI: 10.1007/s11222-007-9033-z. URL: http://www.springerlink.com/index/10.1007/s11222-007-9033-z.

In recent years, spectral clustering has become one of the most popular modern clustering algorithms. It is simple to implement, can be solved efficiently by standard linear algebra software, and very often outperforms traditional clustering algorithms such as the k-means algorithm. On the first glance spectral clustering appears slightly mysterious, and it is not obvious to see why it works at all and what it really does. The goal of this tutorial is to give some intuition on those questions. We describe different graph Laplacians and their basic properties, present the most common spectral clustering algorithms, and derive those algorithms from scratch by several different approaches. Advantages and disadvantages of the different spectral clustering algorithms are discussed.

[49] MacKay

David J.C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press. 2003. URL: http://www.inference.phy.cam.ac.uk/mackay/itila/

[50] Manning, Raghavan, and Schütze

Christopher D. Manning, Prabhaka Raghavan, and Hinrich Schütze. *An Introduction to Information Retrieval*. Cambridge University Press. 2008. URL: http://nlp.stanford.edu/IR-book/information-retrieval-book.html

[51] Martinez, Andreani, and Martinez

L. Martinez, R. Andreani, and J. M. Martinez. "Convergent algorithms for protein structural alignment". In: *BMC Bioinformatics* 8 (2007), p. 306.

Background: Many algorithms exist for protein structural alignment, based on internal protein coordinates or on explicit superposition of the structures. These methods are usually successful for detecting structural similarities. However, current practical methods are seldom supported by convergence theories. In particular, although the goal of each algorithm is to maximize some scoring function, there is no practical method that theoretically guarantees score maximization. A practical algorithm with solid convergence properties would be useful for the refinement of protein folding maps, and for the development of new scores designed to be correlated with functional similarity.
Results: In this work, the maximization of scoring functions in protein alignment is interpreted as a Low Order Value Optimization (LOVO) problem. The new interpretation provides a framework for the development of algorithms based on well established methods of continuous optimization. The resulting algorithms are convergent and increase the scoring functions at every iteration. The solutions obtained are critical points of the scoring functions. Two algorithms are introduced: One is based on the maximization of the scoring function with Dynamic Programming followed by the continuous maximization of the same score, with respect to the protein position, using a smooth Newtonian method. The second algorithm replaces the Dynamic Programming step by a fast procedure for computing the correspondence between C alpha atoms. The algorithms are shown to be very effective for the maximization of the STRUCTAL score.
Conclusion: The interpretation of protein alignment as a LOVO problem provides a new theoretical framework for the development of convergent protein alignment algorithms. These algorithms are shown to be very reliable for the maximization of the STRUCTAL score, and other distance-dependent scores may be optimized with same strategy. The improved score optimization provided by these algorithms provide means for the refinement of protein fold maps and also for the development of scores designed to match biological function. The LOVO strategy may be also used for more general structural superposition problems such as flexible or non-sequential alignments. The package is available on-line at http://www.ime.unicamp.br/ martinez/lovoalign.

[52] Mirny and Domany

L Mirny and E Domany. "Protein fold recognition and dynamics in the space of contact maps". In: *Proteins: Struct. Funct. Genet.* 26 (1996), pp. 391–410.

We introduce an energy function for contact maps of proteins. In addition to the standard term, that takes into account pairwise interactions between amino acids, our potential contains a new hydrophobic energy term. Parameters of the energy function were obtained from a statistical analysis of the contact maps of known structures. The quality of our energy function was tested extensively in a variety of ways. In particular, fold recognition experiments revealed that for a fixed sequence the native map is identified correctly in an overwhelming majority of the cases tested. We succeeded in identifying the structure of some proteins that are known to pose difficulties for such tests (BPTI, spectrin, and cro-protein). In addition, many known pairs of homologous structures were correctly identified, even when the two sequences had relatively low sequence homology. We also introduced a dynamic Monte Carlo procedure in the space of contact maps, taking topological and polymeric constraints into account by restrictive dynamic rules. Various aspects of protein dynamics, including high-temperature melting and refolding, were simulated. Perspectives of application of the energy function and the method for structure checking and fold prediction are discussed.

[53] Mooney, Liang, DeConde, and Altman

Sean D. Mooney, Mike Hsin-Ping Liang, Rob DeConde, and Russ B. Altman. "Structural characterization of proteins using residue environments". In: *Proteins* 61.4 (2005), pp. 741–747. DOI: 10.1002/prot.20661.

A primary challenge for structural genomics is the automated functional characterization of protein structures. We have developed a sequence-independent method called S-BLEST (Structure-Based Local Environment Search Tool) for the annotation of previously uncharacterized protein structures. S-BLEST encodes the local environment of an amino acid as a vector of structural property values. It has been applied to all amino acids in a nonredundant database of protein structures to generate a searchable structural resource. Given a query amino acid from an experimentally determined or modeled structure, S-BLEST quickly identifies similar amino acid environments using a K-nearest neighbor search. In addition, the method gives an estimation of the statistical significance of each result. We validated S-BLEST on X-ray crystal structures from the ASTRAL 40 nonredundant dataset. We then applied it to 86 crystallographically determined proteins in the protein data bank (PDB) with unknown function and with no significant sequence neighbors in the PDB. S-BLEST was able to associate 20 proteins with at least one local structural neighbor and identify the amino acid environments that are most similar between those neighbors.

[54] Murzin, Brenner, Hubbard, and Chothia

A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. "SCOP: a structural classification of proteins database for the investigation of sequences and structures". In: *Journal of Molecular Biology* 247 (1995), pp. 536–540. URL: www.ncbi.nlm.nih.gov/pubmed/7723011.

To facilitate understanding of, and access to, the information available for protein structures, we have constructed the Structural Classification of Proteins (SCOP) database. This database provides a detailed and comprehensive description of the structural and evolutionary relationships of the proteins of known structure. It also provides for each entry links to co-ordinates, images of the structure, interactive viewers, sequence data and literature references. Two search facilities are available. The homology search permits users to enter a sequence and obtain a list of any structures to which it has significant levels of sequence similarity. The key word search finds, for a word entered by the user, matches from both the text of the scop database and the headers of Brookhaven Protein Databank structure files. The database is freely accessible on World Wide Web (WWW) with an entry point to URL http://scop.mrc-lmb.cam.ac.uk/scop/.

[55] Ng, Jordan, and Weiss

Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. "On Spectral Clustering: Analysis and an algorithm". In: *Advances in Neural Information Processing Systems 14*. MIT Press, 2001, pp. 849–856.

Despite many empirical successes of *spectral clustering* methods algorithms that cluster points using eigenvectors of matrices derived from the data–there are several unresolved issues. First, there is a wide variety of algorithms that use the eigenvectors in slightly dierent ways. Second, many of these algorithms have no proof that they will actually compute a reasonable clustering. In this paper, we present a simple spectral clustering algorithm that can be implemented using a few lines of Matlab. Using tools from matrix perturbation theory, we analyze the algorithm, and give conditions under which it can be expected to do well. We also show surprisingly good experimental results on a number of challenging clustering problems.

[56] Nissen

Steffen Nissen. *Implementation Of A Fast Artificial Neural Network Library (FANN)*. Tech. rep. Department of Computer Science University of Copenhagen (DIKU), 2003. URL: http://leenissen.dk/fann/report/report.html.

This report describes the implementation of a fast artificial neural network library in ANSI C called fann. The library implements multilayer feedforward networks with support for both fully connected and sparse connected networks. Fann offers support for execution in fixed point arithmetic to allow for fast execution on systems with no floating point processor. To overcome the problems of integer overflow, the library calculates a position of the decimal point after training and guarantees that integer overflow can not occur with this decimal point.
The library is designed to be fast, versatile and easy to use. Several benchmarks have been executed to test the performance of the library. The results show that the fann library is significantly faster than other libraries on systems without a floating point processor, while the performance was comparable to other highly optimized libraries on systems with a floating point processor.

[57] Orengo, Pearl, and Thornton

C. A. Orengo, F. M. G. Pearl, and J. M. Thornton. "The CATH domain structure database". In: *Structural Bioinformatics*. Ed. by Philip E. Bourne and Helge Weissig. Wiley-Liss, Inc., 2003

[58] Orengo et al.

C. A. Orengo et al. "CATH–a hierarchic classification of protein domain structures". In: *Structure* 5 (1997), pp. 1093–1108. URL: www.ncbi.nlm.nih.gov/pubmed/9309224.

Background: Protein evolution gives rise to families of structurally related proteins, within which sequence identities can be extremely low. As a result, structure-based classifications can be effective at identifying unanticipated relationships in known structures and in optimal cases function can also be assigned. The ever increasing number of known protein structures is too large to classify all proteins manually, therefore, automatic methods are needed for fast evaluation of protein structures.
Results: We present a semi-automatic procedure for deriving a novel hierarchical classification of protein domain structures (CATH). The four main levels of our classification are protein class (C), architecture (A), topology (T) and homologous superfamily (H). Class is the simplest level, and it essentially describes the secondary structure composition of each domain. In contrast, architecture summarises the shape revealed by the orientations of the secondary structure units, such as barrels and sandwiches. At the topology level, sequential connectivity is considered, such that members of the same architecture might have quite different topologies. When structures belonging to the same T-level have suitably high similarities combined with similar functions, the proteins are assumed to be evolutionarily related and put into the same homologous superfamily.
Conclusions: Analysis of the structural families generated by CATH reveals the prominent features of protein structure space. We find that nearly a third of the homologous superfamilies (H-levels) belong to ten major T-levels, which we call superfolds, and furthermore that nearly two-thirds of these H-levels cluster into nine simple architectures. A database of well-characterised protein structure families, such as CATH, will facilitate the assignment of structure-function/evolution relationships to both known and newly determined protein structures.

[59] Pauling and Corey

Linus Pauling and Robert B. Corey. "Atomic coordinates and structural factors for two helical configurations of polypeptide chains". In: *Proceedings of the National Academy of Sciences* 37 (1951), pp. 235–240

Linus Pauling and Robert B. Corey. "The pleated sheet, a new layer configuration of polypeptide chains". In: *Proceedings of the National Academy of Sciences* 37 (1951), pp. 251–256

[61] Pontius, Richelle, and Wodak

Joan Pontius, Jean Richelle, and Shoshana J. Wodak. "Deviations from Standard Atomic Volumes as a Quality Measure for Protein Crystal Structures". In: *Journal of Molecular Biology* 264.1 (1996), pp. 121–136. ISSN: 0022-2836. DOI: DOI:10.1006/jmbi.1996.0628. URL: http://www.sciencedirect.com/science/article/B6WK7-45MG28S-W/2/45de53180a5dced370ec16cfb3863df9.

Standard ranges of atomic and residue volumes are computed in 64 highly resolved and well-refined protein crystal structures using the classical Voronoi procedure. Deviations of the atomic volumes from the standard values, evaluated as the volume Z-scores, are used to assess the quality of protein crystal structures. To score a structure globally, we compute the volume Z-score root mean square deviation (Z-score rms), which measures the average magnitude of the volume irregularities in the structure. We find that the Z-score rms decreases as the resolution and R-factor improve, consistent with the fact that these improvements generally reflect more accurate models. From the Z-score rms distribution in structures with a given resolution or R-factor, we determine the normal limits in Z-score rms values for structures solved at that resolution or R-factor. Structures whose Z-score rms exceeds these limits are considered as outliers. Such structures also exhibit unusual stereochemistry, as revealed by other analyses. Absolute Z-scores of individual atoms are used to identify problems in specific regions within a protein model. These Z-scores correlate fairly well with the atomicB-factors, and atoms having absolute Z-scores >3, occur at or near regions in the model where programs such as PROCHECK identify unusual stereochemistry. Atomic volumes, themselves not directly restrained in crystallographic refinement, can thus provide an indepen dent, rather sensitive, measure of the quality of a protein structure. The volume-based structure validation procedures are implemented in the program PROVE (PROtein Volume Evaluation), which is accessible through the World Wide Web.

[62] Porto, Bastolla, Roman, and Vendruscolo

M Porto, U Bastolla, H E Roman, and M Vendruscolo. "Reconstruction of protein structures from a vectorial representation". In: *Phys. Rev. Lett* 92 (2004), p. 218101.

We show that the contact map of the native structure of globular proteins can be reconstructed starting from the sole knowledge of the contact map.s principal eigenvector, and present an exact algorithm for this purpose. Our algorithm yields a unique contact map for all 221 globular structures of PDBselect25 of length $N \leq 123$ .We also show that the reconstructed contact maps allow in turn for the accurate reconstruction of the three-dimensional structure. These results indicate that the reduced vectorial representation provided by the principal eigenvector of the contact map is equivalent to the protein structure itself. This representation is expected to provide a useful tool in bioinformatics algorithms for protein structure comparison and alignment, as well as a promising intermediate step towards protein structure prediction.

[63] Poupon

Anne Poupon. "Voronoi and Voronoi-related tessellations in studies of protein structure and interaction". In: *Current Opinion in Structural Biology* 14.2 (2004), pp. 233 –241. ISSN: 0959-440X. DOI: 10.1016/j.sbi.2004.03.010. URL: http://dx.doi.org/10.1016/j.sbi.2004.03.010.

The three-dimensional structure of a protein can be modeled by a set of polyhedra drawn around its atoms or residues. The tessellation invented by Voronoi in 1908, and other tessellations of space derived from it, provide versatile representations of three-dimensional structures. In recent years, they have been used to investigate a series of issues relating to proteins: atom and residue volumes, packing, folding, interactions and binding.

[64] Qian, Luscombe, and Gerstein

J Qian, NM Luscombe, and M Gerstein. "Protein family and fold occurrence in genomes: Power-law behaviour and evolutionary model". In: *Journal of Molecular Biology* 313.4 (2001), 673–681.

Global surveys of genomes. measure the usage of essential molecular parts, defined here as protein families, superfamilies or folds, in different organisms. Based on surveys of the first 20 completely sequenced genomes, we observe that the occurrence of these parts follows a power-law distribution. That is, the number of distinct parts (F) with a given genomic occurrence (V) decays as $F = aV^{-b}$, with a few parts occurring many times and most occurring infrequently. For a given organism, the distributions of families, superfamilies and folds are nearly identical, and this is reflected in the size of the decay exponent b. Moreover, the exponent varies between different organisms, with those of smaller genomes displaying a steeper decay (i.e. larger b). Clearly, the power law indicates a preference to duplicate genes that encode for molecular parts which are already common. Here, we present a minimal, but biologically meaningful model that accurately describes the observed power law. Although the model performs equally well for all three protein classes, we focus on the occurrence of folds in preference to families and superfamilies. This is because folds are comparatively insensitive to the effects of point mutations that can cause a family member to diverge beyond detectable similarity. In the model, genomes evolve through two basic operations: (i) duplication of existing genes; (ii) net flow of new genes. The flow term is closely related to the exponent b and can accommodate considerable gene loss; however, we demonstrate that the observed data is reproduced best with a net inflow, i.e. with more gene gain than loss. Moreover, we show that prokaryotes have much higher rates of gene acquisition than eukaryotes, probably reflecting lateral transfer. A further natural outcome from our model is an estimation of the fold composition of the initial genome, which potentially relates to the common ancestor for modem organisms. Supplementary material pertaining to this work is available from www.partslist.org/powerlaw.

[65] Rockmore

Daniel N. Rockmore. "The FFT: An Algorithm the Whole Family Can Use". In: *Computing in Science and Engineering* 2.1 (2000), pp. 60–64. ISSN: 1521-9615. DOI: http://dx.doi.org/10.1109/5992.814659

[66] Rycroft

Chris H. Rycroft. "Multiscale modeling in granular flow". PhD thesis. MIT, 2007. URL: http://math.berkeley.edu/~chr/publish/phd.html.

Granular materials are common in everyday experience, but have long-resisted a complete theoretical description. Here, we consider the regime of slow, dense granular flow, for which there is no general model, representing a considerable hurdle to industry, where grains and powders must frequently be manipulated. Much of the complexity of modeling granular materials stems from the discreteness of the constituent particles, and a key theme of this work has been the connection of the microscopic particle motion to a bulk continuum description. This led to development of the .spot model., which provides a microscopic mechanism for particle rearrangement in dense granular flow, by breaking down the motion into correlated group displacements on a mesoscopic length scale. The spot model can be used as the basis of a multiscale simulation technique which can accurately reproduce the flow in a large-scale discrete element simulation of granular drainage, at a fraction of the computational cost. In addition, the simulation can also successfully track microscopic packing signatures, making it one of the first models of a flowing random packing. To extend to situations other than drainage ultimately requires a treatment of material properties, such as stress and strain-rate, but these quantities are difficult to define in a granular packing, due to strong heterogeneities at the level of a single particle. However, they can be successfully interpreted at the mesoscopic spot scale, and this information can be used to directly test some commonly-used hypotheses in modeling granular materials, providing insight into formulating a general theory.

Chris H. Rycroft. "VORO++: A three-dimensional Voronoi cell library in C++". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 19.4, 041111 (2009), p. 041111. DOI: 10.1063/1.3215722. URL: http://link.aip.org/link/?CHA/19/041111/1

[68] Sanner, Olson, and Spehner

Michel F. Sanner, Arthur J. Olson, and Jean-Claude Spehner. "Fast and robust computation of molecular surfaces". In: *SCG '95: Proceedings of the eleventh annual symposium on Computational geometry*. Vancouver, British Columbia, Canada: ACM, 1995, pp. 406–407. ISBN: 0-89791-724-3. DOI: http://doi.acm.org/10.1145/220279.220324.

In this paper we define the r-reduced surface of a set of n spheres representing a molecule in relation to the r-accessible and r-excluded surfaces. Algorithms are given to compute the outer component of the r-reduced surface in $O(r \log n)$ operations and the analytical description of the corresponding r-excluded surface in time $O(n)$. An algorithm to handle the self-intersecting parts of that surface is described. These algorithms have been implemented in a program called MSMS, which was tested on a set of 709 proteins. The CPU time spent in the different algorithms composing MSMS are given for this set of molecules.

[69] Shi and Malik

Jianbo Shi and Jitendra Malik. "Normalized cuts and image segmentation". In: *IEEE transactions on pattern analysis and machine intelligence* 22.8 (2000), pp. 888–905. URL: http://doi.ieeecomputersociety.org/10.1109/34.868688.

We propose a novel approach for solving the perceptual grouping problem in vision. Rather than focusing on local features and their consistencies in the image data, our approach aims at extracting the global impression of an image. We treat image segmentation as a graph partitioning problem and propose a novel global criterion, the normalized cut, for segmenting the graph. The normalized cut criterion measures both the total dissimilarity between the different groups as well as the total similarity within the groups. We show that an efficient computational technique based on a generalized eigenvalue problem can be used to optimize this criterion. We have applied this approach to segmenting static images, as well as motion sequences, and found the results to be very encouraging.

[70] Siddiqui and Barton

A. S. Siddiqui and G. J. Barton. "Continuous and discontinuous domains: an algorithm for the automatic generation of reliable protein domain definitions". In: *Protein Science* 4 (1995), pp. 872–884.

An algorithm is presented for the fast and accurate definition of protein structural domains from coordinate data without prior knowledge of the number or type of domains. The algorithm explicitly locates domains that comprise one or two continuous segments of protein chain. Domains that include more than two segments are also located. The algorithm was applied to a nonredundant database of 230 protein structures and the results compared to domain definitions obtained from the literature, or by inspection of the coordinates on molecular graphics. For 70% of the proteins, the derived domains agree with the reference definitions, 18% show minor differences and only 12% (28 proteins) show very different definitions. Three screens were applied to identify the derived domains least likely to agree with the subjective definition set. These screens revealed a set of 173 proteins, 97% of which agree well with the subjective definitions. The algorithm represents a practical domain identification tool that can be run routinely on the entire structural database. Adjustment of parameters also allows smaller compact units to be identified in proteins.

[71] Smith and Waterman

T. F. Smith and M. S. Waterman. "Identification of common molecular subsequences". In: *Journal of Molecular Biology* 147.1 (1981), pp. 195–197. ISSN: 0022-2836. DOI: DOI:10.1016/0022-2836(81)90087-5. URL: http://www.sciencedirect.com/science/article/B6WK7-4DN3Y5S-24/2/b00036bf942b543981e4b5b7943b3f9a.

The identification of maximally homologous subsequences among sets of long sequences is an important problem in molecular sequence analysis. The problem is straightforward only if one restricts consideration to contiguous subsequences (segments) containing no internal deletions or insertions. The more general problem has its solution in an extension of sequence metrics (Sellers 1974; Waterman et al., 1976) developed to measure the minimum number of "events" required to convert one sequence into another. These developments in the modern sequence analysis began with the heuristic homology algorithm of Needleman & Wunsch (1970) which first introduced an iterative matrix method of calculation. Numerous other heuristic algorithms have been suggested including those of Fitch (1966) and Dayhoff (1969). More mathematically rigorous algorithms were suggested by Sankoff (1972), Reichert et al. (1973) and Beyer et al. (1979), but these were generally not biologically satisfying or interpretable. Success came with Sellers (1974) development of a true metric mewure of the distance between sequences. This metric was later generalized by Waterman et al. (1976) to include deletions/insertions of arbitrary length. This metric represents the minimum number of "mutational events" required to convert one sequence into another. It is of interest to note that Smith et al. (1980) have recently shown that under some conditions the generalized Sellers metric is equivalent to the original homology algorithm of Needleman & Wunsch (1970). In this letter we extend the above ideas to find a pair of segments, one from each of two long sequences, such that there is no other pair of segments with greater similarity (homology). The similarity measure used here allows for arbitrary length deletions and insertions.

[72] Subbiah, Laurents, and Levitt

S. Subbiah, D. V. Laurents, and M. Levitt. "Structural similarity of DNA-binding domains of bacteriophage repressors and the globin core". In: *Current Biology* 3 (1993), pp. 141–148.

Background: In recent years, the determination of large numbers of protein structures has created a need for automatic and objective methods for the comparison of structures or conformations. Many protein structures show similarities of conformation that are undetectable by comparing their sequences. Comparison of structures can reveal similarities between proteins thought to be unrelated, providing new insight into the interrelationships of sequence, structure and function.
Results: Using a new tool that we have developed to perform rapid structural alignment, we present the highlights of an exhaustive comparison of all pairs of protein structures in the Brookhaven protein database. Notably, we find that the DNA-binding domain of the bacteriophage repressor family is almost completely embedded in the larger eight-helix fold of the globin family of proteins. The significant match of specific residues is correlated with functional, structural and evolutionary information.
Conclusion: Our method can help to identify structurally similar folds rapidly and with high-sensitivity, providing a powerful tool for analyzing the ever-increasing number of protein structures being elucidated.

[73] Swindells

M. B. Swindells. "A procedure for the automatic determination of hydrophobic cores in protein structures". In: *Protein Science* 4 (1995), pp. 93–102.

An algorithm is described for automatically detecting hydrophobic cores in proteins of known structure. Three pieces of information are considered in order to achieve this goal. These are: secondary structure, side-chain accessibility, and side-chain-side-chain contacts. Residues are considered to contribute to a core when they occur in regular secondary structure and have buried side chains that form predominantly nonpolar contacts with one another. This paper describes the algorithm's application to families of proteins with conserved topologies but low sequence similarities. The aim of this investigation is to determine the efficacy of the algorithm as well as to study the extent to which similar cores are identified within a common topology.

[74] Tegge, Wang, Eickholt, and Cheng

A.N. Tegge, Z. Wang, J. Eickholt, and J. Cheng. "NNcon: improved protein contact map prediction using 2D-recursive neural networks". In: *Nucleic Acids Research* 37.Web Server (2009), W515–W518. ISSN: 0305-1048. DOI: 10.1093/nar/gkp305. URL: http://www.nar.oxfordjournals.org/cgi/doi/10.1093/nar/gkp305.

Protein contact map prediction is useful for protein folding rate prediction, model selection and 3D structure prediction. Here we describe NNcon, a fast and reliable contact map prediction server and software. NNcon was ranked among the most accurate residue contact predictors in the Eighth Critical Assessment of Techniques for Protein Structure Prediction (CASP8), 2008. Both NNcon server and software are available at http://casp.rnet.missouri.edu/nncon.html.

[75] Theodoridis and Koutroumbas

Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition, Third Edition*. Academic Press. 2009

[76] Tufte

Edward Tufte. *Beautiful evidence*. Graphic Press. 2006. call no: P93.5.T84

Edward Tufte. *Envisioning information*. Graphic Press. 1990. call no: P93.5.T54

Edward Tufte. *The visual display of quantitative information*. Graphic Press. 2001. call no: QA276.3.T83

Edward Tufte. *Visual explanations: images and quantities, evidence and narrative*. Graphic Press. 1997. call no: QA276.3.T84

[80] Tukey

John W. Tukey. *Exploratory data analysis*. Addison-Wesley. 1977. call no: HA29.T783

[81] Tung, Huang, and Yang

Chi-Hua Tung, Jhang-Wei Huang, and Jinn-Moon Yang. "Kappa-alpha plot derived structural alphabet and BLOSUM-like substitution matrix for rapid search of protein structure database". In: *Genome Biology* 8.3 (2007), R31. ISSN: 1465-6906. DOI: 10.1186/gb-2007-8-3-r31. URL: http://genomebiology.com/2007/8/3/R31.

We present a novel protein structure database search tool, 3D-BLAST, that is useful for analyzing novel structures and can return a ranked list of alignments. This tool has the features of BLAST (for example, robust statistical basis, and effective and reliable search capabilities) and employs a kappa-alpha (κ, α) plot derived structural alphabet and a new substitution matrix. 3D-BLAST searches more than 12,000 protein structures in 1.2 s and yields good results in zones with low sequence similarity.

[82] Vassura et al.

Marco Vassura et al. "FT-COMAR: fault tolerant three-dimensional structure reconstruction from protein contact maps." In: *Bioinformatics (Oxford, England)* 24.10 (2008), pp. 1313–1315. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btn115. URL: http://www.ncbi.nlm.nih.gov/pubmed/18381401.

Fault Tolerant Contact Map Reconstruction (FT-COMAR) is a heuristic algorithm for the reconstruction of the protein three-dimensional structure from (possibly) incomplete (i.e. containing unknown entries) and noisy contact maps. FT-COMAR runs within minutes, allowing its application to a large-scale number of predictions.

[83] Vendruscolo, Kussell, and Domany

M Vendruscolo, E Kussell, and E Domany. "Recovery of Protein Structure from Contact Maps". In: *Folding and Design* 2 (1997), pp. 295–306.

Background: Prediction of a protein.s structure from its amino acid sequence is a key issue in molecular biology. While dynamics, performed in the space of two-dimensional contact maps, eases the necessary conformational search, it may also lead to maps that do not correspond to any real three-dimensional structure. To remedy this, an efficient procedure is needed to reconstruct three-dimensional conformations from their contact maps.
Results: We present an efficient algorithm to recover the three-dimensional structure of a protein from its contact map representation. We show that when a physically realizable map is used as target, our method generates a structure whose contact map is essentially similar to the target. Furthermore, the reconstructed and original structures are similar up to the resolution of the contact map representation. Next, we use nonphysical target maps, obtained by corrupting a physical one; in this case, our method essentially recovers the underlying physical map and structure. Hence, our algorithm will help to fold proteins, using dynamics in the space of contact maps. Finally, we investigate the manner in which the quality of the recovered structure degrades when the number of contacts is reduced.
Conclusions: The procedure is capable of assigning quickly and reliably a three-dimensional structure to a given contact map. It is well suited for use in parallel with dynamics in contact map space to project a contact map onto its closest physically allowed structural counterpart.

[84] Vendruscolo, Najmanovich, and Domany

M Vendruscolo, R Najmanovich, and E Domany. "Protein Folding in Contact Map Space". In: *Physical Review Letters* (1999), pp. 3–6.

Changing a few contacts in a contact map corresponds to a large scale move in confrontation space; hence, one gains a lot by using the contact map representation for protein folding. We developed an efficient search procedure in the space of physical contact maps, which could identify the native fold as of the lowest free energy, provided on had a free energy function whose ground state is the native map. We prove rigorously that the widely used pairwise contact approximation to the free energy cannot stabilize even a single protein.s native map. Testing the native map against a set of decoys obtained by gapless threading, one may be misled to the opposite conclusion.

[85] Veretnik and Shindyalov

Stella Veretnik and Ilya Shindyalov. "Computational methods for domain partitioning of protein structures". In: ed. by Ying Xu, Dong Xu, and Jie Liang. Springer, 2006. Chap. 4, pp. 125–145

[86] Victor

Bret Victor. *Magic Ink*. Tech. rep. worrydream.com, 2006. URL: http://worrydream.com/MagicInk/

[87] Wang

ZX Wang. "A re-estimation for the total numbers of protein folds and superfamilies". In: *Protein Eng.* 11.8 (1998), pp. 621–626. DOI: 10.1093/protein/11.8.621. URL: http://peds.oxfordjournals.org/cgi/content/abstract/11/8/621.

The issue of the number of protein folds is steeped in controversy despite its significance for understanding evolution and predicting protein structure from amino acid sequence. Using various assumptions, several research groups have tackled this problem with very different results. In the present study, a more rigorous statistical approach is used to address this question. From three different data sets, the total number of protein folds is estimated to be about 650. A detailed theoretical analysis suggests that (i) a random sample of non-transmembrane protein families has been selected for crystallization and structural determination, (ii) except for about 40 folds, most protein folds occurring in nature contain about the same number of different protein families. With the estimation of the total number of protein folds, the number of naturally occurring superfamilies can then be estimated as 1150.

[88] Winkler

W. E. Winkler. *The state of record linkage and current research problems*. Tech. rep. United States Internal Revenue Service, Statistics of Income Division, 1999. URL: www.census.gov/srd/papers/pdf/rr99-04.pdf

[89] Ye and Godzik

Yuzhen Ye and Adam Godzik. "Flexible structure alignment by chaining aligned fragment pairs allowing twists". In: *Bioinformatics* 19 (2003), pp. 246–255. DOI: 10.1093/bioinformatics/btg1086. URL: http://bioinformatics.oxfordjournals.org/cgi/content/abstract/19/suppl_2/ii246.

Motivation: Protein structures are flexible and undergo structural rearrangements as part of their function, and yet most existing protein structure comparison methods treat them as rigid bodies, which may lead to incorrect alignment. Results: We have developed the Flexible structure AlignmenT by Chaining AFPs (Aligned Fragment Pairs) with Twists (FATCAT), a new method for structural alignment of proteins. The FATCAT approach simultaneously addresses the two major goals of flexible structure alignment; optimizing the alignment and minimizing the number of rigid-body movements (twists) around pivot points (hinges) introduced in the reference protein. In contrast, currently existing flexible structure alignment programs treat the hinge detection as a post-process of a standard rigid body alignment. We illustrate the advantages of the FATCAT approach by several examples of comparison between proteins known to adopt different conformations, where the FATCAT algorithm achieves more accurate structure alignments than current methods, while at the same time introducing fewer hinges. Contacts: adam@burnham.org

[90] Yoon, Ebert, Chung, De Micheli, and Altman

Sungroh Yoon, Jessica Ebert, Eui-Young Chung, Giovanni De Micheli, and Russ Altman. "Clustering protein environments for function prediction: finding PROSITE motifs in 3D". In: BMC *Bioinformatics* 8.Suppl 4 (2007), S10. ISSN: 1471-2105. DOI: 10.1186/1471-2105-8-S4-S10. URL: http://www.biomedcentral.com/1471-2105/8/S4/S10.

Background: Structural genomics initiatives are producing increasing numbers of three-dimensional (3D) structures for which there is little functional information. Structure-based annotation of molecular function is therefore becoming critical. We previously presented FEATURE, a method for describing microenvironments around functional sites in proteins. However, FEATURE uses supervised machine learning and so is limited to building models for sites of known importance and location. We hypothesized that there are a large number of sites in proteins that are associated with function that have not yet been recognized. Toward that end, we have developed a method for clustering protein microenvironments in order to evaluate the potential for discovering novel sites that have not been previously identified.
Results: We have prototyped a computational method for rapid clustering of millions of microenvironments in order to discover residues whose surrounding environments are similar and which may therefore share a functional or structural role. We clustered nearly 2,000,000 environments from 9,600 protein chains and defined 4,550 clusters. As a preliminary validation, we asked whether known 3D environments associated with PROSITE motifs were "rediscovered". We found examples of clusters highly enriched for residues that share PROSITE sequence motifs.
Conclusion: Our results demonstrate that we can cluster protein environments successfully using a simplified representation and K-means clustering algorithm. The rediscovery of known 3D motifs allows us to calibrate the size and intercluster distances that characterize useful clusters. This information will then allow us to find new clusters with similar characteristics that represent novel structural or functional sites.

[91] Zhang and Skolnick

Y Zhang and J Skolnick. "Scoring function for automated assessment of protein structure template quality". In: *Proteins* 57 (2004), pp. 702–710.

We have developed a new scoring function, the template modeling score (TM-score), to assess the quality of protein structure templates and predicted full-length models by extending the approaches used in Global Distance Test (GDT) 1 MaxSub.2 and First, a protein size-dependent scale is exploited to eliminate the inherent protein size dependence of the previous scores and appropritely account for random protein structure pairs. Second, rather than setting specific distance cutoffs and calculating only the fractions with errors below the cutoff, all residue pairs in alignment/modeling are evaluated in the proposed score. For comparison of various scoring functions, we have constructed a large-scale benchmark set of structure templates for 1489 small to medium size proteins using the threading program PROSPECTOR$_3$ and built the full-length models using MODELLER and TASSER. The TM-score of the initial threading alignments, compared to the GDT and MaxSub scoring functions, shows a much stronger correlation to the quality of the final full-length models. The TM-score is further exploited as an assessment of all "new fold" targets in the recent CASP5 experiment and shows a close coincidence with the results of human-expert visual assessment. These data suggest that the TM-score is a useful complement to the fully automated assessment of protein structure predictions. The executable program of TM-score is freely downloadable at http://bioinformatics.buffalo.edu/tm-score.

Y Zhang and J Skolnick. "TM-align: a protein structure alignment algorithm based on the TM-score". In: *Nucleic Acids Research* 33 (2005), pp. 2302–2309.

We have developed TM-align, a new algorithm to identify the best structural alignment between protein pairs that combines the TM-score rotation matrix and Dynamic Programming (DP). The algorithm is ∼4 times faster than CE and 20 times faster than DALI and SAL. On average, the resulting structure alignments have higher accuracy and coverage than those provided by these most often-used methods. TM-align is applied to an all-against-all structure comparison of 10 515 representative protein chains from the Protein Data Bank (PDB) with a sequence identity cutoff, 95threshold of 0.5 is used. We also use TM-align to match the models predicted by TASSER for solved non-homologous proteins in PDB. For both folded and misfolded models, TM-align can almost always find close structural analogs, with an average root mean square deviation, RMSD, of 3 Å and 87% alignment coverage. Nevertheless, there exists a significant correlation between the correctness of the predicted structureandthe structural similarity of the model to the other proteins in the PDB. This correlation could be used to assist in model selection in blind protein structure predictions. The TM-align program is freely downloadable at http://bioinformatics.buffalo.edu/tm-align.

[93] Zhang, Hubner, Arakaki, Shakhnovich, and Skolnick

Yang Zhang, Isaac A. Hubner, Adrian K. Arakaki, Eugene Shakhnovich, and Jeffrey Skolnick. "On the origin and highly likely completeness of single-domain protein structures". In: *Proceedings of the National Academy of Sciences of the United States of America* 103.8 (2006), pp. 2605–2610. DOI: 10.1073/pnas.0509379103. URL: http://www.pnas.org/content/103/8/2605.abstract.

The size and origin of the protein fold universe is of fundamental and practical importance. Analyzing randomly generated, compact sticky homopolypeptide conformations constructed in generic simplified and all-atom protein models, all have similar folds in the library of solved structures, the Protein Data Bank, and conversely, all compact, single-domain protein structures in the Protein Data Bank have structural analogues in the compact model set. Thus, both sets are highly likely complete, with the protein fold universe arising from compact conformations of hydrogen-bonded, secondary structures. Because side chains are represented by their C atoms, these results also suggest that the observed protein folds are insensitive to the details of side-chain packing. Sequence specificity enters both in fine-tuning the structure and thermodynamically stabilizing a given fold with respect to the set of alternatives. Scanning the models against a three-dimensional active-site library, close geometric matches are frequently found. Thus, the presence of active-site-like geometries also seems to be a consequence of the packing of compact, secondary structural elements. These results have significant implications for the evolution of protein structure and function.

This thesis was designed by the author,
and set using LaTeX.

The body type is 11/13.6 Adobe Garamond by Robert Slimbach,
the sans font is Bliss by Jeremy Tankard,
and the monospace font is TXTT by Young Ryu.

The paper is Strathmore writing 24 lb. soft white wove.

This is version 1401:e9276c652136.