A FEATURES ANALYSIS TOOL FOR ASSESSING AND IMPROVING COMPUTATIONAL MODELS IN STRUCTURAL BIOLOGY

Matthew James O'Meara

A dissertation submitted to the faculty at the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the department of Computer Science in the Collage of Arts and Sciences.

Chapel Hill
2013

Approved by:

Jack Snoeyink

Brian Kuhlman

Jan Prins

Steve Marron

Diane Poszefsky

ABSTRACT

Matthew James O'Meara: A Features Analysis Tool For Assessing And Improving Computational Models In Structural Biology
(Under the direction of Jack Snoeyink and Brian Kuhlman)

The protein-folding problem is to predict, from a protein's amino acid sequence, its folded 3D conformation. State of the art computational models are complex collaboratively maintained prediction software. Like other complex software, they become brittle without support for testing and refactoring. Features analysis, a language of 'scientific unit testing', is the visual and quantitative comparison of distributions of features (local geometric measures) sampled from ensembles of native and predicted conformations. To support features analysis I develop a features analysis tool—a modular database framework for extracting and managing sampled feature instance and an exploratory data analysis framework for rapidly comparing feature distributions. In supporting features analysis, the tool supports the creation, tuning, and assessment of computational models, improving protein prediction and design.

I demonstrate the features analysis tool through 6 case studies with the Rosetta molecular modeling suite. The first three demonstrate the tool usage mechanics through constructing and checking models. The first evaluates bond angle restraint models when used with the Backrub local sampling heuristic. The second identifies and resolves energy function derivative discontinuities that frustrate gradient-based minimization. The third constructs a model for disulfide bonds.

The second three demonstrate using the tool to evaluate and improve how models represent molecular structure. I focus on modeling H-bonds because of their geometric specificity and environmental

dependence lead to complex feature distributions. The fourth case study develops a novel functional form for Sp^2 acceptor H-bonds. The fifth fits parameters for a refined H-bond model. The sixth combines the refined model with an electrostatics model and harmonizes them with the rest of the energy function.

Next, to facilitate assessing model improvements, I develop *recovery tests* that measure predictive accuracy by asking models to recover native conformations that have been partially randomized.

Finally, to demonstrate that the features analysis and recovery test tools support improving protein prediction and design, I evaluated the refined H-bond model and electrostatics model with additional corrections from the Rosetta community. Based on positive results, I recommend a new standard energy function, which has been accepted by the Rosetta community as the largest systematic improvement in nearly a decade.

ACKNOWLEGEMENTS

I would like to give thanks for the support and guidance I have received in the course of writing this dissertation. Coming from a background in mathematics and being plunged to the worlds of computational geometry and structural biology I owe a great deal to many people. My primary advisor in computer science, Jack Snoeyink, has shown me the beauty in geometric problems and taught me above all else that careful writing can lead to clear thinking. My primary advisor in biochemistry, Brian Kuhlman, has kindled my excitement for untangling the intricate details of molecular structure and how through studying these details simple and powerful ideas can emerge. I want to thank the institutional support I have received from the UNC Chapel Hill, the Department of Computer Science, NSF, NIH and DARPA for funding my research, and my committee members for their insightful guidance.

Participating in the Rosetta Community thoughout my graduate carrier has allowed me to thrive as a researcher. It was a privilege working shoulder to shoulder with Andrew Leaver-Fay on a range of fruitful projects. I will fondly remember the camaraderie of hacking at the Extreme Rosetta Workshops—thanks to Sam Deluca, Brian Weitzner, Tim Jacobs, Doug Renfrew, Steven Lewis, Will Sheffler, James Thompson and Florian Richter. The principal investigators Jeff Gray, John Karanicolas, Jens Meiler, Phil Bradley, Rhiju Das, Tanja Kortemme, and David Baker have been pillars of intellect, welcoming, and encouraging. I am grateful for the opportunity to have collaborated on scientific benchmarking with Amelie Stein, Roland Pache, Sergey Lyskov, Mike Tyka, Frank Dimaio, and Kevin Houlihan, on database development with Sam Deluca and Tim Jacobs, and on features analysis with Joe Harrison, Steven Combs, Doo Nam Kim, and Patrick Conway.

I feel incredibly lucky to be able to share this journey with my wife Teresa. She has been steadfast in her support and encouragement. Together we have built a family on trust and love. Raising Robin with Teresa has been an absolute joy. Robin inspires me to learn and explore the world.

TABLE OF CONTENTS

LIST OF FIGURES

> If we view statistics as a discipline in the service of science, and science as being an attempt to understand (i.e., model) the world around us, then the ability to reveal sensitivity of conclusions from fixed data to various model specifications, all of which are scientifically acceptable, is equivalent to the ability to reveal boundaries of scientific uncertainty.
>
> Rubin, 1984

1 Introduction

The protein folding problem, one of the best puzzles in all of science, is to computationally predict, from a sequence of a protein's amino acids, the 3D geometry of its stable, biologically active, folded conformation (Anfinsen 1973). State of the art prediction software, such as the Rosetta molecular modeling suite (Rohl et al. 2004, Leaver-Fay 2011), define energy functions over atomic coordinates and stochastically search for low energy conformations. The space of all conformations, whether parameterized in atom coordinates or the angles formed by chemical bonds between atoms, typically has thousands of degrees of freedom. These energy functions are often assumed to decompose into different types of interactions, such as hydrogen bonding, electrostatic attraction and repulsion, rotamer selection, solvent displacement, and salt bridge formation to name a few that will be defined in this dissertation. To make the search manageable, these interactions are often assumed to be local and independent, and a separate model is trained from experimental data for each type of interaction.

For an overview of model creation, integration, and maintenance, let me briefly introduce the Rosetta hydrogen bond model, which this dissertation explores in detail, especially in chapter 6. A model is created by first choosing the relevant local parameters of conformations, which I will call *features* (e.g., atomic positions for hydrogen-bond donor and acceptor atoms and their

neighbors), second choosing a *functional form,* which takes features and *tuning parameters* and returns an interaction energy (e.g., a function of the distances and angles that are characteristic of hydrogen bonding), and third choosing the values for tuning parameters to fit experimentally observed data distributions. The model is integrated into the software; in Rosetta, this is by adding it as an *energy term*. The net energy of a configuration is a weighted sum of energy terms, and various benchmarks can be used to tune the weights. The resulting computational models become extremely complex—Rosetta has dozens of energy terms that are active and hundreds that are implemented and available. The model is collaboratively maintained as dozens of researchers tweak and tune, and hundreds use the software for structure prediction and protein design.

A complex computational model, like any other form of complex software, becomes very brittle without good support for testing and refactoring. Modeling decisions made locally can have unexpected consequences, and models developed independently can have unwanted interactions. The thesis of this work is that a *features analysis tool* for visualizing and assessing distributions of features, which are distributions of geometric measures derived from samples of conformations, both from native protein structures and from predicted structures with different energy functions, supports the creation, tuning, and maintenance of energy functions, improving protein structure prediction.

1.1 Overview of Features Analysis

Before I outline the chapters that describe the tool design and case studies of its application, let me give in Figure 1.1.1 an example of a small-multiple plot from a features analysis of hydrogen bonding that illustrates the benefits of features analysis in four areas: detecting unexpected results, creating new functional forms, fitting and tuning parameters, and creating scientific

benchmarks and unit tests. The different plots show distributions of distances from donor to acceptor atoms in a hydrogen bond for different donor types (rows) and acceptor types (columns). For example, backbone/backbone H-bonds are in the last column, second-to-last row. Each plot shows three distribution curves: the red curve is estimated from experimentally determined native structures, the green is from predicted structures that minimize a Rosetta default baseline energy function, and the blue is from predictions that minimize a Rosetta energy with the hydrogen bond energy term replaced by one described in this dissertation. The numbers tell how many pairs each curve is estimated from; the total is nearly 1.5 million bonding pairs.

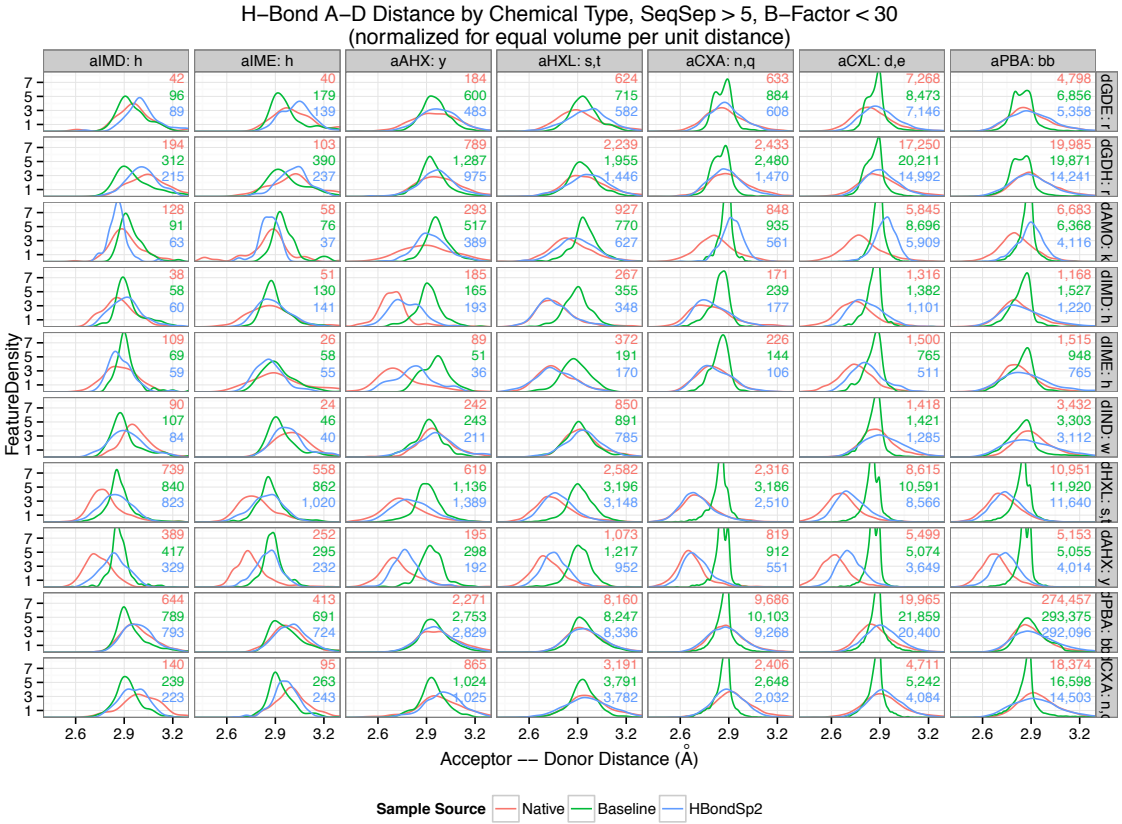


Figure 1.1.1 H-bond A-D Distance By Chemical Type: Distributions of hydrogen bond acceptor to donor distances, by acceptor and donor chemical types. Each plot shows three distributions: Native (red) is estimated from experimentally determined structures, Baseline (green) and HBondSp2 (blue) ere estimated from predicted structures using different Rosetta energy terms. Numbers indicate how many samples go into each estimation. The blue distributions fit the red more closely than the green for reasons described in the text.

Backbone/backbone bonds (last column, penultimate row) make up the majority of the data, and the good fits of both the baseline (green) and new HBondSp2 (blue) curves to the natives (red) in these cases actually makes both curves look like good fits in aggregate statistics. In many plots, however, the baseline (green) distribution is too sharply peaked because of an unexpected interaction between the minimizer and internal interpolation between long- and short-range energy terms. Looking down the columns suggest that the green peaks do not depend on donor chemistry type—it was a modeling decision that the functional form of the baseline depends only on acceptor type—but this has unwanted results in rows 7 and 8, where native hydroxyl H-bonds appear to be shorter. HBondSp2 was created and tuned to fit these cases better, and to smooth the interpolation. The shorter hydroxyl H-bonds also necessitated changes to other energy terms in Rosetta, a fact discovered by using the features analysis tool during integration. A model designer who is using the features analysis tool to create the functional form and fit the parameters that better model these types of hydrogen bond can relatively easily create a scientific unit test that can notify him or her if another modeler's changes to some other energy term moves the distribution of distances outside of acceptable ranges. Then, the two modelers can negotiate a compromise, or even a synthesis, that allows both to achieve their aims.

In my dissertation work, I develop computational tools for assessing and fitting macromolecular energy functions against experimental data. The main tool, which I call **features distribution analysis**, formalizes an intuitive approach to checking predictive models: comparing ensembles of predictions with ensembles of reference data by looking at distributions of geometric measures, or *features*. Basing the tool on distributions recognizes that ensembles, rather than single structures, should be compared. Suppose, for example, that the mean length of H-bonds with serine donors in predicted structures is two standard deviations longer than the mean observed in structures characterized via X-ray crystallography; this may indicate a problem with the energy function of the predictive model. Treating the length of the bond as a geometric feature and

comparing mean lengths tests whether the feature(s) observed in the predicted and experimental conformations are drawn from different distribution.

By choosing biophysically motivated features (e.g., hydrogen bonds lengths and angles, volumes of buried cavities, relative orientation of secondary structure elements, etc.), distributions obtained from samples can become a language for scientific 'unit tests.' These unit tests can be used to give interpretable explanations for preferring one energy function to another. To support these tests, I have created a modular database framework for extracting and managing sampled feature instances and an exploratory data analysis framework for rapidly comparing feature distributions.

I also develop tools to support focused **recovery tests** that directly measure the accuracy of prediction methods; these help to detect and prevent over-fitting in the more local feature distribution tests. A recovery test asks a model to predict an experimental observation, such as protein conformations observed through X-ray crystallography in a given conformation space. To facilitate rapid energy function evaluation, the conformation can be constrained by, for example, fixing all but a subset of the degrees of freedom. I build upon work within the structural biology community to curate and deploy a collection of new recovery tests. Specifically, I improve the computational benchmarking framework in Rosetta, assembling experimental data and prediction protocols and applying statistically rigorous analysis methods.

To exercise the features analysis tool, as well as to contribute to our understanding of the determinants of molecular structure, I evaluate and refine the **H-bond model in Rosetta** (Kortemme 2003). The partial covalent/electrostatic character of H-bonds makes them orientation specific and sensitive to their chemical environment. This makes them challenging to model and an ideal test of our ability to computationally represent complex cooperative behavior. Using the

features analysis tool, I analyze a diversity of H-bond-related feature distributions, evaluating alternative functional forms, and iteratively refining the fit of the model parameters. I use the recovery tests to measure if these modifications improve the overall predictive accuracy. Through this work, I identify and correct specific limitations of the existing model and improve the overall recapitulation of H-bond features observed in experimental data.

To demonstrate that the features analysis and recovery test tools improve molecular structure prediction, I evaluate and integrate several candidate modifications to the Rosetta energy function. Through scientific benchmarks, I recommend a **new standard energy function for the Rosetta community**, which has been accepted as the first systematic improvement in nearly a decade.

1.2 Chapter Outline

In Chapter 2, I present structural biology background to establish context for the rest of the dissertation. I first define terminology for molecular conformations (Section 2.1). I then step back to consider the centrality of building and studying models in the scientific process (Section 2.2). Finally, I consider the high computational cost of globally sampling conformations from structural biology energy functions (Section 2.3), which motivates building and studying local feature models.

In Chapter 3, I describe the features analysis tool. In Section 3.1, I lay the conceptual foundation culminating in the definition of a feature as an observable random variable that forms a small geometric model that represents a conformation sample source (Section 3.1.1). I then describe the role of features analysis in building, fitting, checking, and benchmarking computational models

(Section 3.1.2). I support my conceptual foundation by relating it to concepts in computer science, statistics, and structural biology (Section 3.1.3).

In Section 3.2, I describe the tool components and the usage of the features analysis tool by both developers and analysts. Developers implement FeaturesReporters (Section 3.2.3) that populate a features database (Section 3.2.4) and implement features analysis scripts that compare feature instances sampled from the features database through a query (Section 3.2.5). Analysts provide batches of conformation samples, report features to a features database, and run features analysis scripts and interpret the resulting plots and statistics.

In Chapter 4, I present three case studies demonstrating the use of the features analysis tool for checking computational models. I walk through a features analysis step-by-step of the N-Cα-C bond angle distribution when using the Backrub move (Section 4.1). I present three vignettes in which I identify and fix derivative discontinuities in the energy function that frustrate gradient-based minimization (Section 4.2). Finally, I develop a novel model for disulfide interactions that demonstrates the use of features analysis to create components of molecular energy functions (Section 4.3).

In Chapter 5, I elaborate on the computational and statistical methods that underlie the features analysis tool including kernel density estimation, feature transforms, and quantifying divergence of density distributions. I describe desiderata for the tool and how this shapes the design decisions (Section 5.1). I consider mathematical details of density estimation (Section 5.2) and using the features analysis tool to do exploratory data analysis and hypothesis testing.

In Chapter 6, I present three case studies to demonstrate that features analysis can improve how H-bonds are modeled in Rosetta. As context, I use feature models to define energy-based

computational models in structural biology (Sections 6.1.1,2). I then Reference Ratio Method, which provides a statistical grounding to the features distribution comparison underlying features analysis (Section 6.1.3-5) and computational methods for weighting feature models (6.1.6). I then discuss prior on work model H-bonds in structural biology (Section 6.2).

Specifically, in the three case studies, I first extend Rosetta's H-bond model by building an smooth analytic functional form over the (BAH, BA χ) angles to model the orientation dependence H-bonds for Sp^2 acceptors (Section 6.3); I second fit the parameters to recapitulate feature distributions (Section 6.4); and I third harmonize the H-bond model with the rest of the energy function by adjusting the Lennard-Jones model for hydroxyl donors and refitting the H-bond model when combined with a Coulombic potential for electrostatics (Section 6.5).

In Chapter 7, I define recovery scientific benchmarks and use them to evaluate modifications to the Rosetta energy function developed in the case studies. I present 6 recovery benchmarks of increasing difficulty (Chapter 7.1). I then use these benchmarks to evaluate the modifications to the Rosetta energy function presented in Chapters 4 and 6. Based on the positive results, the final energy function combining all the modifications, Talaris2013, has been adopted as the standard Rosetta energy function.

In Chapter 8, I conclude the dissertation with a summary of the work (Section 8.1); I emphasize the community aspect to building and testing computational models (Section 8.2); and present features analysis observations that suggest future work in improving energy functions.

1.3 Contributions

My main contribution is the features analysis tool in Rosetta, but I would also like to highlight contributions from my case studies using the tool, and list projects by others in the Rosetta community.

Using the tool for local features analysis has revealed several long-standing anomalies in the Rosetta molecular energy functions, and helped to resolve them, as seen in Chapter 4.

I have advanced the state of the art in modeling of Hydrogen bonding by developing a novel functional form to model H-bonds with Sp^2 acceptors, fitting model parameters to recapitulate native H-bond feature distributions, and integrating an electrostatic model and an explicit H-bond model as detailed in Chapter 6.

I have established community accepted recovery scientific benchmarks and used them along with the feature analyses to demonstrate that my modifications make substantial improvements to the Rosetta energy function. These improvements have culminated in a new standard energy function for the Rosetta Community, Talaris2013 that outperforms Score12, which has been the default for almost a decade.

My improvements to the H-bond model demonstrate local features analysis can guide evaluating and improving complex computational models with substantially less computational cost than globally mapping their behavior as discussed in Section 2.1.

Developers and users of the Rosetta platform features have broadly adopted the tools developed in this dissertation to further the development of Rosetta energy function and prediction methods.

Beyond the work I present here, further work that uses one or more components of my features analysis tool include:

- Development of novel feature models
    - Development of partially covalent model of hydrogen bonding using atomic orbitals; Steven Combs, Meiler Lab, Vanderbilt University
    - Development of a bond length and bond angle feature model and Cartesian space optimization algorithms; Patrick Conway, Baker Lab, University of Washington Seattle
- Use of features database for complex structure prediction and design protocols
    - Design of de novo bundle and repeat proteins using a graph of fragments from native structures; Tim Jacobs, Doo Nam Kim; Kuhlman Lab, UNC-Chapel Hill
    - Development of ligand virtual screening; Sam Deluca, Meiler Lab, Vanderbilt University
- Descriptive studies of feature patterns
    - Analysis of beta turns in anti-body loops; Brian Weitzner, Gray Lab, Johns Hopkins University
    - Analysis of cooperative H-bonding and solvation effects Kevin Houlihan; Kuhlman Lab, UNC-Chapel Hill
    - Analysis of pH dependent switches in viruses; Joseph Harrison, Kuhlman Lab, UNC Chapel-Hill
- Features database as a means for managing structure prediction data
    - Storage of scoring and sequence data for antibody study; Jordan Willis, Vanderbilt University
    - Storage of conformational representation various projects; Jared Adolf-Bryfogle, Dunbrack Lab, Fox Chase Cancer Center

2 Background Information

Before introducing the features analysis tool in the next chapter, we need a few key concepts of modeling protein structure. Because features analysis depends upon conformation space, I review the basics of protein 3D structures and define conformation space (Section 2.1). Features analysis also depends on a view of structural biology as the discipline of building computational models to extend what can determined experimentally and theoretically from a molecular structure (Section 2.2). Features analysis is a direct attack on the computational challenges of working with the complex computational models in structural biology (Section 2.3).

2.1 The Structure of Protein Molecules

I briefly review the basics of protein structure, focusing on foundational concepts that are necessary to understand the work supporting my thesis. Chemically, each protein chain is a sequence of amino acids connected by peptide bonds. The chemical identity of the protein, or *primary structure,* is determined by the type of each amino acid at each position or *residue* along the sequence. Each residue has three atoms (N, Cα, C) that connect to form the protein *backbone* and the remainder of the atoms, specific to each amino acid, branch off each Cα atom to form *sidechains*. There are 20 canonical amino acids and they are often represented by letters of the alphabet, so the primary structure of a protein forms a long word. Consider these two amino acid sequences,

```
VRDAYIAKPHNCVYECARNEYCNN  GMRLEKDRFSVNLDVKHFSPEELK
LCTKNGAKSGYCQWSGKYGNGCWC  VKVLGDVIEVHGKHEERQDEHGFI
IELPDNVPIRVPGKCH          SREFHRKYRIPADVDPLTITSSLS
                          SDGVLTVNGPRKQVSGPERTIPIT
```

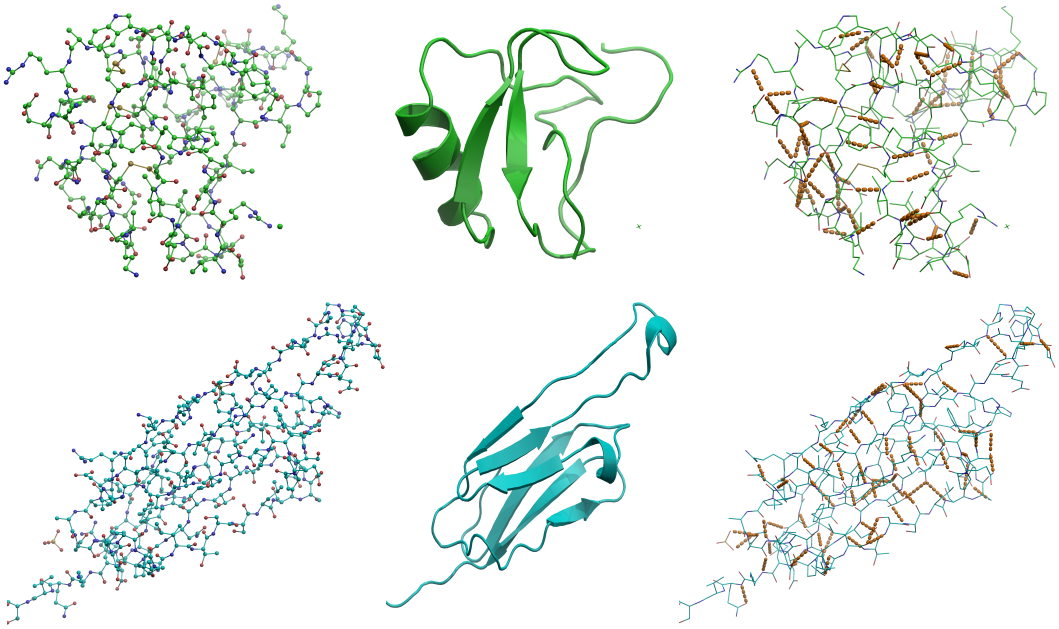


Figure 2.1.1 Representations of Protein Conformations: Three diagram styles depict conformations of two proteins with codes 1CHZ (top) and 3L1G (bottom). The underlying data for each diagram are the coordinates of the atoms in space, but each representation is a visual model, made using PyMOL (www.pymol.org), that highlights different aspects of the molecular conformation. In a ball-and-stick diagram (left) atoms drawn as colored spheres (red:oxygen, blue:nitrogen, rest:carbon; hydrogens are suppressed), and covalent bonds as lines. A cartoon or ribbon diagram (center) shows the protein backbone, highlighting α-helices and β-sheets. A wire diagram (right) suppresses the spheres to show other information: here Hydrogen bonds are depicted as orange dotted lines.

To give these—and most proteins—their characteristic biological function in nature, the chains of amino acids reliably fold into characteristic 3D structures. Figure 2.1.1 shows the structure for each of these sequences using three different visualizations where the underlying data for each is the coordinates of atoms in space. These conformations were experimentally determined through X-ray crystallography, and the details of the experimental process and the atomic coordinates were deposited into Protein Databank with accession codes 1CHZ[1] and 3L1G[2].

[1] 1CHZ is the BmK 2M neurotoxin protein from the Chinese scorpion *Buthus martensii* Karsch (Li, 1996)

[2] 3L1G is the alphaB crystallin protein from humans and helps keep the eye clear (Laganowski, 2010)

To begin to understand the spatial organization of protein structure, consider the degrees of freedom. Chemical bonds that make up the primary structure geometrically constrain the positions of the atoms in space by forcing bond lengths and bond angles to be what a biochemist would assume is "essentially rigid" (this assumption is examined in 4.1), leaving the twisting of some bonds' torsion angles as the only allowable motions. Each amino acid residue has 3 rotatable torsion angles along the protein backbone (ϕ, ψ, ω) where ω is often assumed to be 0° or 180° and, depending on the amino acid type, between 0 and 4 in its sidechain $(\chi_1, \dots, \chi_4)$. The specification of the amino acid sequence and the torsion angles can thus determine a molecular *conformation*. In nature, many proteins adopt a single, stable, highly organized fold (Lane 2013). To express this observation rigorously, a set of possible conformations of a protein forms a *conformation space*, which can be partitioned into *states;* for example, the simplest partition declares a small region around the folded conformation to be in the *folded* state, and every other conformation to be the *unfolded* state.

2.2 Model Building to Understand Protein Structure

To understand the natural world, scientists build *models:* simple systems that correspond to more complex systems of interest. Scientific models are useful because, investigation of the model system can provide information about the more complex system. Through the correspondence, the simpler system is said to *represent* the more complex system. A simpler representation should not reproduce every detail of the more complex system; the application must dictate which aspects of the more complex system to reproduce and which are not needed.

Consider the visual representations for molecular conformations shown in Figure 2.1.1. The ball-and-stick diagrams in the left column show heavy atoms and bonds and give the viewer some information about the precise location of these atoms in space; however, the overplotting obscures the spatial relations of some molecules. The right most diagrams highlight the pattern of hydrogen bonding, which contributes to stabilizing the overall fold. Upon close inspection, there appears to be regularity to the H-bonding pattern, though it is still difficult to distinguish. The center diagrams abstract the backbone-backbone H-bonding patterns and show the molecules as ribbon diagrams (Richardson 1981), which highlight the *secondary structure*: α-helices, β-sheets, and the connecting loops.

With current technology, laboratory experiments are unable to directly observe all the complex conformational and functional details of macromolecules. The gap between the rate that genes, which code for proteins, are sequenced through high throughput sequencing technology and deposited into GenBank (~1,000,000 per year) and the rate that the structures of proteins are characterized through the X-ray crystallography and deposited into the Protein Bata Bank (~4,000 per year) hints at the number of protein sequences that still have unknown structures. Thus, a grand challenge for computational modeling in structural biology remains: to reliably predict the

structure of a naturally occurring protein from its molecular sequence. Because of the relationship between structure and function, increasing our ability to predict protein structure is likely to increase our ability to understand protein function within cells.

In theory, molecular structure is fully described by the laws of Quantum Mechanics. However, solving or even approximating the solutions to the partial differential Schrödinger is computationally intractable for most biologically relevant proteins because they are such large systems and have delicate energy balances. The gold standard of approximate accuracy for feasible computational QM method scales as $O(n^7)$ in the number of atoms (Řezáč 2013).

Since neither experimentally observing protein structure nor directly solving for the structure from first principles is feasible, a primary approach to gaining insight into protein structure is by building computational models to predict protein structure from amino acid sequences. For now, a computational model can be thought of as software used to construct and predict molecular conformations; later I discuss them as realizations of abstract statistical models (Section 6.1).

Computational modeling of protein structure is not just a stopgap until we can develop experimental assays to characterize naturally occurring molecules. The insight computational models provide goes beyond what we can observe in the laboratory. For example, to design molecules with specific molecular function, such as monitoring or modulating a target biological process or synthesizing a biofuel, high-level descriptions of the desired functionality must be translated into low-level specifications, analogous to how architects work from architectural concepts to create detailed drafts for the engineers. Useful models for design facilitate encoding design constraints as patterns in molecular structure searching for sequences to fold to structures that satisfy the constraints.

The many successful computational models in structural biology define energy functions over molecular conformation space as linear combinations of *feature models*, where each feature model evaluates the energy of a type of local geometric observable. With such an energy function, a computational model makes predictions by sampling the conformation space around minima of the energy function. For example, a Markov Chain Monte Carlo sampler (MCMC), defined over a conformation space with a set of local moves, samples observations according to the Boltzmann distribution associated with the energy function as described in the next subsection.

These computational models, exemplified by the Rosetta Macromolecular Suite, are trained (Leaver-Fay 2013) with experimental data (e.g., X-ray structures deposited into the PDB). The models can then be tested by predicting stable conformations that have biological function and then testing experimentally if the predicted molecules indeed have the intended structure and function. For example, in a landmark 2011 study, Fleishmann et al. used Rosetta to design 88 different molecules to bind to Hemagglutinin, a key surface molecule from the 1918 H1N1 flu virus, and found two that showed reproducible binding activity, showing that computational design is beginning to be feasible but still a significant challenge. X-ray crystal structures of the complex demonstrated that the mode of binding was consistent with the predictions from Rosetta.

2.3 Conformational Sampling: From Energy Function to Prediction

In this dissertation, I describe my tools to assist the modeling and evaluation of energy functions in structural biology—energy functions that occur in nature probed by experiments, and those defined in computational models. To motivate modeling energy functions by building simple systems to elucidate their behavior, in this section, I discuss why globally mapping the behavior of energy functions is not feasible. The take home messages are that the complexity of the energy functions makes mapping their behavior over conformation space through computational sampling very demanding and that biased sampling can give misleading results. Additionally, I introduce the FastRelax sampling protocol, which I use in the case studies.

Two canonical sampling methods from statistical mechanics are Molecular Dynamics (MD), which computationally simulates the laws of motion in conformation space, and Markov chain Monte-Carlo (MCMC), which stochastically applies local conformational moves. Either can generate unbiased samples from the Boltzmann distribution[3] for an energy function if they satisfy detailed balance conditions and if the simulations are run long enough. However, because of the complexity of typical energy functions over molecular conformation spaces, it is often computationally infeasible to generate unbiased samples. Diverse approaches have been developed to address this challenge, including making full use of computational resources such as GPUs, distributed and parallel computer architectures, and even specialized computer hardware;

[3] In statistical mechanics, an isolated system with an energy function $E(c)$ over conformation space in equilibrium at temperature T, adopts the Boltzmann distribution $p(c) = \frac{1}{Z} e^{\frac{-E(c)}{kT}}$, where Z is the normalization constant. Once the temperature has been fixed, the relationship is one-to-one, so the Boltzmann distribution over conformation space is a lossless model for the energy function. To gain a deeper understanding of the relationship between an energy function, the Boltzmann distribution and temperature, I recommend investigating simulations of the Ising model, a particularly simple system that exhibits different statistical mechanical behavior depending on the temperature (Baxter 1984).

using coarse grained conformation spaces to reduce the complexity of the problem at the expense of modeling accuracy; and, as I do, using biased sampling strategies but being careful with how the results are interpreted.

A features analysis, which compares distributions of locally defined features that determine global geometry rather than the more global measures such as RMSD to natives, reduces the dimensionality of the problem, increasing the information gained from less costly protocols.

The main sampling strategy that I use to generate decoy sample sources from native chains is the FastRelax protocol in Rosetta (Khatib 2011). FastRelax minimizes the energy function for an input conformation by alternating between a fixed-backbone sidechain optimization (Kuhlman 2000, Leaver-Fay 2008) (*repack*) and all-atom gradient-based minimization (*min*) using the (L)BFGS minimizer (Nocedal 2006). The protocol begins with a small weight on Lennard-Jones inter-atom repulsion, to make conformational rearrangement feasible, then alternates repacking and minimizing while ramping up the weight of Lennard-Jones repulsion. FastRelax is written in the relax protocol domain language as

```
repeat 5
  ramp_repack_min 0.02  0.01     1.0
  ramp_repack_min 0.250 0.01     0.5
  ramp_repack_min 0.550 0.01     0.0
  ramp_repack_min 1     0.00001  0.0
  accept_to_best
endrepeat
```

That is, it will accept the best conformation through 5 cycles, each determined by four runs of

```
ramp_repack_min <fa_rep weight> <min_tolerance> <constraint_weight>
```

This sets the weight of the repulsive component of the Lennard-Jones model to `fa_rep weight`, applies repack, and then applies minimize using `min_tolerance` as the convergence tolerance. (Constraints are task specific terms in the energy function and not used in this study or in this dissertation.)

For the Top8000 set, FastRelax produces conformations with an all-atom RMSD of approximately 1.5 Å from the native, but with distributions of local features that mirror the distributions from more costly protocols, like ab initio. Thus, I am able to use FastRelax for all decoy distributions in the features analyses of this dissertation. For global tests, I use recovery benchmarks, as described in chapter 7.

As an illustration of the challenges of drawing global conclusions from biased samples, let me describe two studies by Tyka and coauthors in which different sampling protocols lead to opposite conclusions. The first, in 2010, found that the energy function of Rosetta was able to find deeper local minima near native structures, and concluded that sampling, rather than energy function quality, was the bottleneck for structure prediction. The second, in 2012, sampled with less bias and found no difference between the depth of local minima for structures that were near or far from native, and concluded that better energy functions were needed.

Here are the details. In 2010, Tyka et al. used Rosetta to generate predictions for a set of 111 proteins of 50 to 150 residues. For each protein, approximately 600,000 predictions were generated using a two-stage protocol known as Iterative Looprelax. The first stage, which aimed to seed the sampling with a diverse range of conformations, used a unified-atom representation of a conformation and performed MCMC optimization with fragment insertions (Simons 1997, Kuhlman 2003). The second stage used FastRelax to find low energy conformations, which are thought to represent stable structures. Depending on the size of the protein, generating a single prediction took approximately 10 CPU minutes. As a result, the whole study required approximately 10 million CPU hours.

After globally superimposing the lowest energy conformations obtained from the Iterative Looprelax protocol, showed remarkable agreement with the native conformation with 90% of the residues showed less than 0.8 Å deviation from the native structures (Figure 2.3.1) and further many of the discrepancies could be explained by un-modeled crystal packing artifacts, un-modeled interactions with small molecules, or unstructured loop regions.

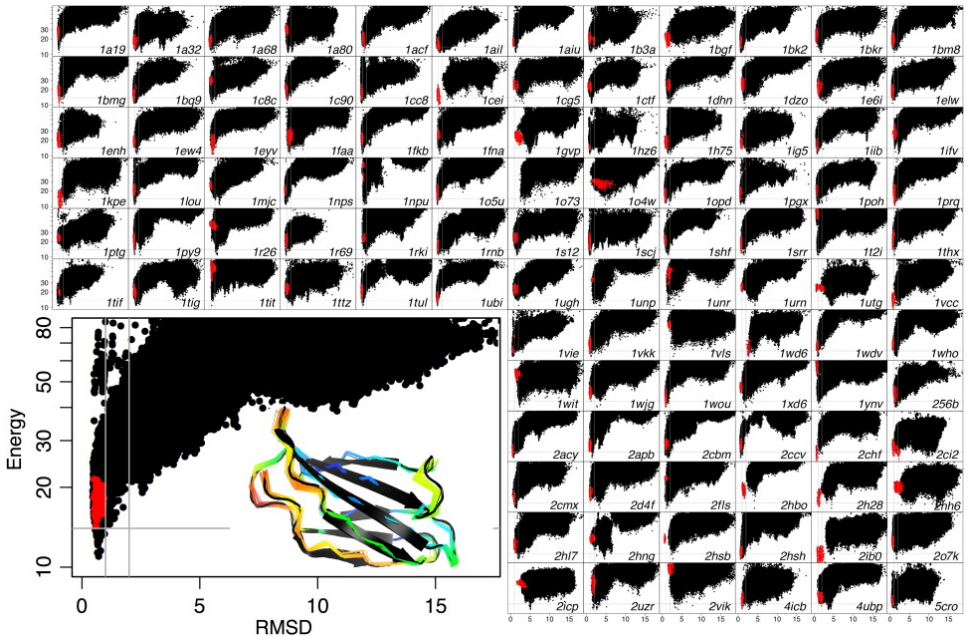


Figure 2.3.1 Decoy Discrimination (Tyka 2010): Discrimination of native and non-native conformations (measured by the root mean squared deviation of the Cα atoms (Cα-RMSD) to the native) by the Rosetta energy function. Each cell shows predictions for a native conformation, where each black point is the result of a trajectory of the Iterative Looprelax protocol and each red point is applying FastRelax to the native conformation. The inset target is 1TEN. The vertical gray bars are at 1 Å and 2 Å and the horizontal gray bars are at the lowest energy of the relaxed native conformation. For most targets, the lowest energy conformation has low Cα-RMSD, indicating the energy function is good. However, the bias towards sampling near-native conformations obscures low energy conformations with high RMSD that become visible with more extensive sampling (Figure 2.3.2).

Although Tyka et al. said that they "cannot exclude the possibility that more thorough sampling could reveal lower-energy minima further from the native structure," they conclude that the primary limitation to better modeling is sampling, not energy function accuracy: "[C]urrent macromolecular energy functions are sufficiently accurate for good structure prediction, but the available computing power and sampling algorithms are still insufficient to sample reliably within

1–2 Å Cα Root Mean Squared Deviation (RMSD) of the native structure, as needed for a model to be recognized as native-like based on its very low energy." This result even suggested that it would be possible to evaluate the quality of a prediction based on this computational benchmark, without always needing to have an experimentally determined structure. An alternative explanation, however, is that since fragments come from native structures, the seeds were biased toward native or homologous-to-native structures, and thus FastRelax did more extensive search in the neighborhoods of native structures.

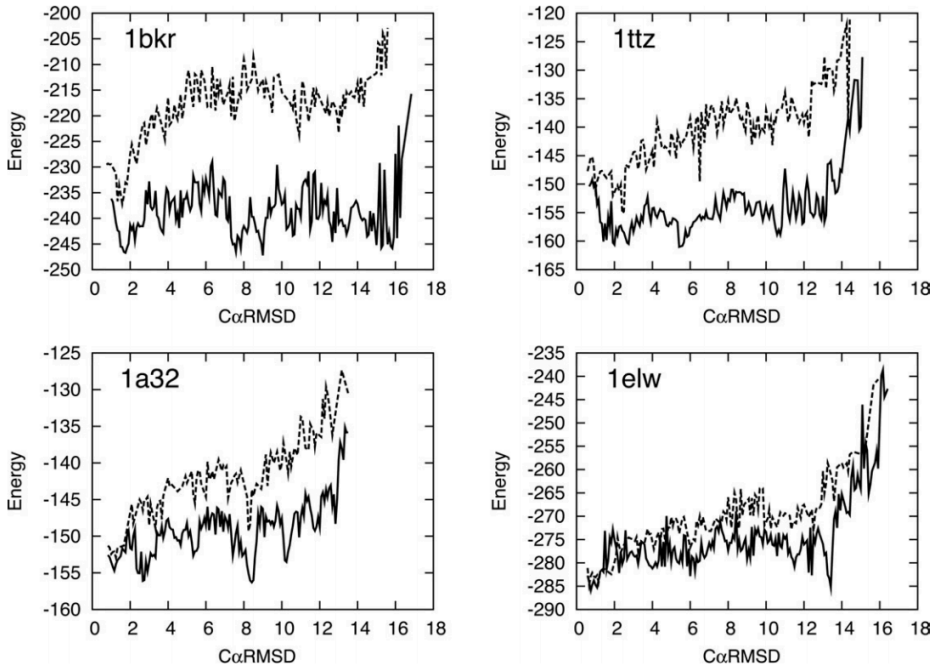


Figure 2.3.2 Deep Decoy Sampling (Tyka 2012): Predictions for four targets from the 2010 (dotted line) and 2012 (solid line) study are plotted as the lower envelops of the Cα-RMSD vs. The 2012 study samples more extensively and reveals low energy conformations with high Cα-RMSD indicating the energy function fails to discriminate the native conformation for these targets from non-native conformations.

In a follow-up study in 2012, Tyka et al. developed a substantially more sophisticated sampling protocol called Batch Relax, which aggressively culls the worst performing sampling trajectories, using a sophisticated job scheduler to maintain parallel efficiency. By sampling more broadly—and spending 50,000 hours of computer time per protein—they were able to find conformations with dramatically lower energy than with Iterative Looprelax. When the Batch Relax protocol

was applied to 31 of the original structures without biasing towards native conformations, many conformations with large RMSD were found to have as low or lower energies than the native conformation. In 14 cases, the folding funnel observed in 2010 was completely erased (Figure 2.3.2). The apparent ability of Rosetta to discriminate native conformations in the 2010 study resulted partially from heavy sampling near native conformations and under-sampling far from native conformations. They conclude, "[f]or the first time in several years, it appears in many cases that successful protein structure prediction is no longer only limited by ability to sample but by accuracy of the energy function." However, the intense computational cost of sampling prevents use of this protocol for general structure predictions.

Although the ultimate test of model quality is validating structure predictions through laboratory experiments, the time and money costs of crystallizing and determining every structure is too high. Tyka's studies demonstrate that although it is possible to evaluate the discriminative ability of energy functions, the computational costs of this evaluation may also make this infeasible. As a solution to this problem, I recommend generating candidate energy functions through features analysis and small-scale scientific benchmarks and using these large-scale discrimination tests only as a final validation step.

> [S]cientific knowledge advances by practice-theory iteration. Known facts (data) suggest a tentative theory or model, implicit or explicit, which in turn suggests a particular examination and analysis of data and/or the need to acquire further data; analysis may then suggest a modified model and may require further practical illumination and so on.
>
> Box, 1980

3 The Features Analysis Tool

3.1 Concepts and Terminology

This dissertation builds tools to assess computational models of molecular structure against experimental data. To discuss the features analysis tool and its usage rigorously, in this section I define concepts and terminology that characterize the data and how it is analyzed (Section 3.1.1). I then describe a motivating example (Section 3.1.2), which I return to in Section 6.5. I connect the concepts of features and features analysis to unit testing in computer science (Section 3.1.3), Bayesian modeling in statistics (Section 3.1.4), and knowledge-based potentials in structural biology (Section 3.1.5).

3.1.1 Sample Sources and Features

I define a *sample source* as a process from which it is possible to sample conformations, whether through laboratory experiments or generated from a computational model. By repeatedly sampling from a sample source, it defines a statistical distribution over conformation space so that sampling or generating a conformation can be thought of as observing a random variable from the distribution. The statistical distribution is an abstract model of the sample source. A set of conformations sampled from a sample source is called a *batch*.

For convenience, I use the term *native* to refer to a sample source whose conformations are experimentally observed, often through X-ray crystallography, then filtered for quality and coverage. These observations produce a model for the behavior of molecular structure in nature, albeit biased by the researchers' choice of what to experimentally observe, the limitations of the technique, and the filtering process. I use the term *decoy* to refer to a sample source whose conformations are predicted by software, such as Rosetta, with specified energy functions, optimization methods, and sampling protocols.

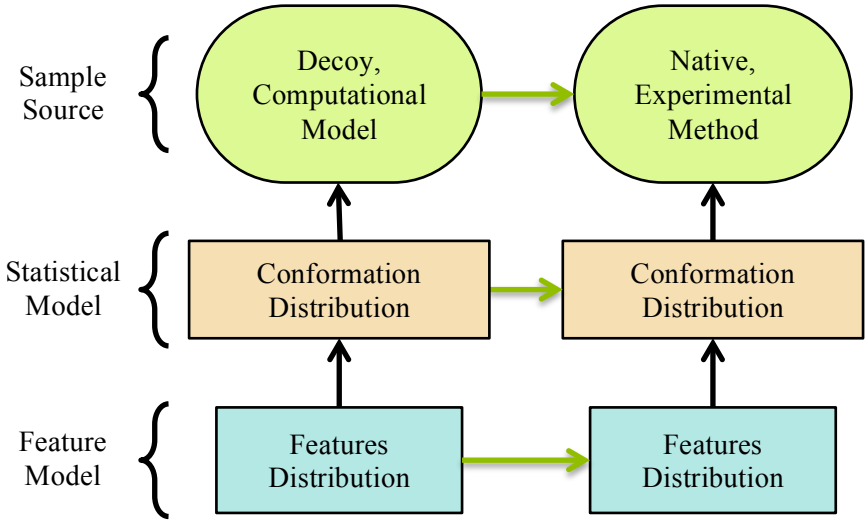


Figure 3.1.1 Model Abstraction: Modeling relationships are depicted as arrows that point from a simpler system to the more complex system it models. The green ovals depict sample sources, which are physical processes that produce molecular conformations either occurring in nature or engineered in computers. The boxes indicate abstract statistical models as probability distributions over either a conformation or a feature space. The green arrows indicate the scientifically relevant modeling relationships. We would like to assess if the decoy computational model accurately represents the native sample source observed through a specific experimental method. Although comparing sample sources or full distributions over conformation spaces are intractable, using features analysis we can compare feature distributions and infer the comparison between sample sources.

I use the term *feature* throughout this dissertation to mean a measurable quantity of a subset of the atoms in a conformation that may be biophysically relevant. For example, consider the feature *AHdist* that measures the length of a hydrogen bond (H-bond) (Section 6.1) as the distance between the main acceptor atom and the donated hydrogen atom. A realization of a feature in a conformation is a *feature instance* (each conformation may have many), and the set of all possible instances forms a *feature space*.

A sample source induces a probability distribution over a feature space, called a *feature distribution*, by first sampling a conformation from the sample source and then uniformly sampling a feature from the conformation. For each sample source, each feature is a random variable distributed with respect to the induced feature distribution.

A feature distribution is a model of a conformation distribution, and a conformation distribution is a model of a sample source. Through these correspondences, a feature distribution gives a focused view of a sample source. If a researcher detects a discrepancy between feature distributions from different sample sources (e.g., natives vs. decoys) by a two-sample test, this implies a discrepancy between the distributions over conformation space and therefore a discrepancy between the sample sources. This is similar to inferring differences in populations by comparing sample means. I define a *features analysis* as the use of batches of sampled features to assess and compare sample sources.

3.1.2 Features Analysis as Scientific Unit Testing

From a software engineering perspective, a significant challenge in building complex computational models is assuring the quality of the model. Often biochemical models are stochastic, so testing for specific numerical results is not meaningful. Instead, researchers express the desired behavior at a much higher level, often in the language of the substantive domain, which may not easily translate into the algorithmic details of the computational model.

Features analysis provides a method to express and test properties of a computational model. Interpreting the behavior of the model as a statistical distribution, individual features encode units of behavior. By defining features that are meaningful in the substantive domain, researchers can encode model tests as feature distribution tests. Since models are designed to represent a more

complex system, often feature distribution test are most naturally expressed as deviations from a reference distribution estimated from the larger system. In this sense, a features analysis becomes a minimal scientific unit test of the computational model.

Software code unit testing, such as through xUnit testing frameworks (Meszaros 2007), allow developers to record that a specific unit of functionality is important to maintain as the code evolves through refactoring and further development (Fewster 1999, Fowler 2002, Beck 2002). To create a code unit test, a developer writes small pieces of test code in a unit-testing framework that exercises the functionality and signals when behavior is broken. Then another developer, or even an end user of the software, runs the unit test through the unit-testing framework to assert that the code is functioning properly.

The process of creating and then running a features analysis scientific benchmark is analogous to creating and running a code unit test. To create a features analysis, a modeler defines a feature through the features reporter framework and creates a features analysis script through the features analysis script framework with the additional requirement that, in addition to plots and summary statistics, the features analysis must return a pass or fail condition. A user or another developer runs the features analysis through the features analysis tool by providing native and decoy samples sources to assert if the decoy feature distribution recapitulates the native feature distribution.

3.1.3 Concepts Related to Features Analysis in the Statistical Modeling Literature

Features analysis is related to posterior predictive checks in applied Bayesian statistics (Gelman 1996, 2004). In Bayesian statistics, a statistical model is created by considering a generic prior distribution over observed and unobserved random variables, and then, given observed data from

the system to be modeled, the prior distribution is updated to the posterior distribution through the Bayes rule. After a model has been fit but before it is used to make inferences or predictions about the larger system, the researcher should check that the model actually fits the data; *Posterior predictive checking* does this by computing a summary statistic once from "native" data and multiple times from "decoys" or replicas of the data simulated from the fitted model. The tail probability for the summary statistic of the native data being estimated from the decoy sampling distribution from the model is interpreted as a classical *p*-value that can be used to reject the model. By choosing summary statistics relevant to the application of the model, posterior predictive checks can detect relevant model violations. Posterior predictive checks can be implemented in the features analysis framework by partitioning the samples from the computational model into replicas.

Approximate Bayesian Computation (ABC) is a method developed primarily in ecology for using feature analysis-like methods to fit their models (Marin 2011). Typically, ABC is motivated by the observation that in complex statistical models, the probability of the data given the model (the likelihood function) is intractable to evaluate because normalizing the probability distribution requires integrating over the entire space (e.g., all of conformation space). Because of the lower dimensionality of feature spaces, evaluating the likelihood there becomes feasible. Use of ABC has been criticized because it is difficult to show if a set of feature distributions are sufficient, i.e., whether it fully encodes all of the information in the full probability distribution. A counter argument is that the scientific method works by iteratively developing and refining models based on observations of the world; if no feature is known to discriminate a model from reality, then the model is good enough to make predictions.

3.1.4 Features Analysis in Structural Biology

Features analysis has long held a prominent role in structural biochemistry, thanks to the availability of highly detailed X-ray crystallography data deposited in the Protein Databank (Berman 2000). In this subsection, I review several landmark studies and recent studies that take advantage of the continued growth in the available experimentally determined structure data. These studies can be classified as describing patterns of molecular geometry for use in tools that validate the quality of experimentally determined or computationally predicted conformations, and for use in tools to generate structure predictions.

Previous studies to identify and classify patterns in native structures suggest several features that can be the basis of features analysis. For example studies of patterns that involve a few residues include those to identify correlations between adjacent bond-lengths and angles (Engh 1991, Berkholz 2009), and torsion angles (Ramachandran 1968, Betancourt 2004). Patterns in sidechain torsion angles have been collected into Rotamer libraries (Ponder 1987, Lovell 2000, Liang 2002), and protein backbones have been classified into regular secondary structure patterns of α-helices and β-sheets, etc. (Richardson 1981, Kabsch 1983, Richardson 1988, Frishman 1995). Studies of non-bonded interactions include characterizing general patterns of charge distribution and H-bonding (Murray-Rust 1984, Baker 1984, Stickle 1992, Kortemme 2003).

Studies of features of the whole structure include measurements of volume, packing quality (Sheffler, 2009), and structural rigidity (Jacobs 2001). Studies have also focused on the boundary between the protein and solvent, including measures of solvent-accessible surface area (Lee 1971), and hydrophobic surface patches (Jacak 2012). Specific feature models have been developed for molecular interfaces, including various interface descriptors.

3.2 The Features Analysis Tool

3.2.1 Overview of the Features Analysis Tool Components

The features analysis tool allows researchers to investigate differences between molecular sample sources by comparing batches of sampled feature instances. The tool builds databases of feature values extracted from given sample sources, whether native or decoy. These can be retrieved, filtered, and compared using graphical data analysis and statistical two-sample tests. Figure 3.2.1 depicts the general workflow.

The features analysis tool is constructed from three components, each of which builds on established technology. The central component is a relational database schema that stores values for sampled feature instances. Storing sampled feature instances in a relational database allows modelers to construct samples of complex features through database queries and rely on robust relational database engines for data management.

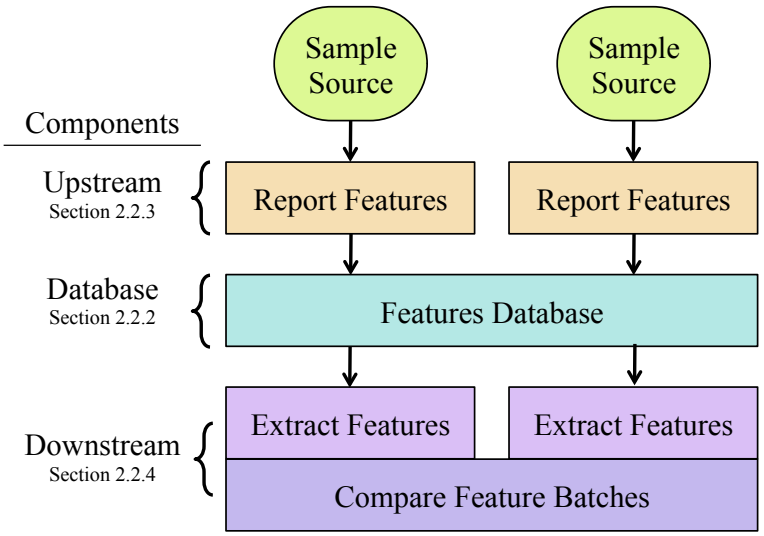


Figure 3.2.1 Components of the Features Analysis Tool: Inputs are batches of conformations sampled from sample sources. The upstream component reports elementary features to the features database. The downstream component extracts, transforms, and compares features to assess differences between sample sources.

The upstream component is the features reporter framework and it is responsible for populating the features database from batches of conformations from sample sources. A features reporter module in the features reporter framework is a C++ class that is responsible for populating a set of tables in the database for a conformation. The framework builds on the Rosetta platform, which provides support for representing conformations and making measurements. The features reporter framework is integrated into the RosettaScripts protocol specification language and job distribution system.

The downstream component is the features analysis script framework and it is responsible for creating sampled instances of complex features by querying and transforming data from the features database and analyzing these sampled feature instances. A script in the R statistical programming language is responsible for performing each single analysis and generating graphical and numerical summaries.

3.2.2 Features Analysis Use Cases

There are two types of users of the features analysis tool. The first type are the *developers* who create features reporters for the upstream framework and features analysis scripts for the downstream framework and the second type are the *analysts* who perform the features analyses by providing sample sources, executing features analysis scripts, and interpreting the resulting plots and statistics. For exploratory data analysis, one researcher may be both a developer and an analyst. For scientific unit testing, two or more researchers, working on different aspects of the computational model, may each serve as developers of their own component of the models, creating scripts and unit tests that ensure the quality of their component, and analysts of the integrated model as a whole.

The features analysis tool is straightforward for analysts to use, even with limited computational experience. An analyst provides a set of conformations representing the sample source, adapts a RosettaScripts protocol to report features and create a features database, adapts a configuration script to run the features analysis scripts, and inspects the results. The ability to relatively easily choose the sample sources and features analysis scripts allows for a variety of comparisons to be performed.

The features analysis tool is flexible for developers to create a wide range of features analyses. Even with limited knowledge of C++ and programming for the Rosetta platform, a developer is able to create simple or complex features reporters that populate the features database. Even with limited knowledge of SQL and R, a developer is able to define a wide range of features and apply a wide range of methods for comparing them. The cost of learning the C++, SQL, and R languages for developing for the features analysis tool is repaid by the expressive power and the availability of support libraries (e.g., Rosetta in C++ and the wide range of statistical and plotting packages for R).

3.2.3 Features Databases

A relational database consists of a schema that specifies data stored in each table, constraints over single tables or pairs of tables, and a relational database management system (RDMS) that is responsible for storing the data on disk and allowing clients to interact with the database using the Structured Query Language (SQL) or application programming interfaces (Codd 1970).

A features database stores sampled feature values. The schema is organized hierarchically to reflect the conceptual hierarchy of sample sources. The sample sources have conformations, which in turn have features. At the highest level of the schema is a *batch* that represents a specific

set of conformations sampled from a sample source. Associated with each batch are metadata about the sample source, including software version and command parameters used to report the features, the sampled *structures* that represent the sampled conformations themselves, and *feature* data. The feature data is grouped into sets of tables representing closely related features. The set of tables is populated by a features reporter, which is described in Section 3.2.4. Within each set, some tables store sampled values for elementary features while others help define the feature values and the possibly complex relationships to other features. Figure 3.2.2 shows the tables for H-bond related features, and the foreign key relationships between the tables populated by the `HBondFeaturesReporter` and the `batches`, `structures`, and `residues` tables.

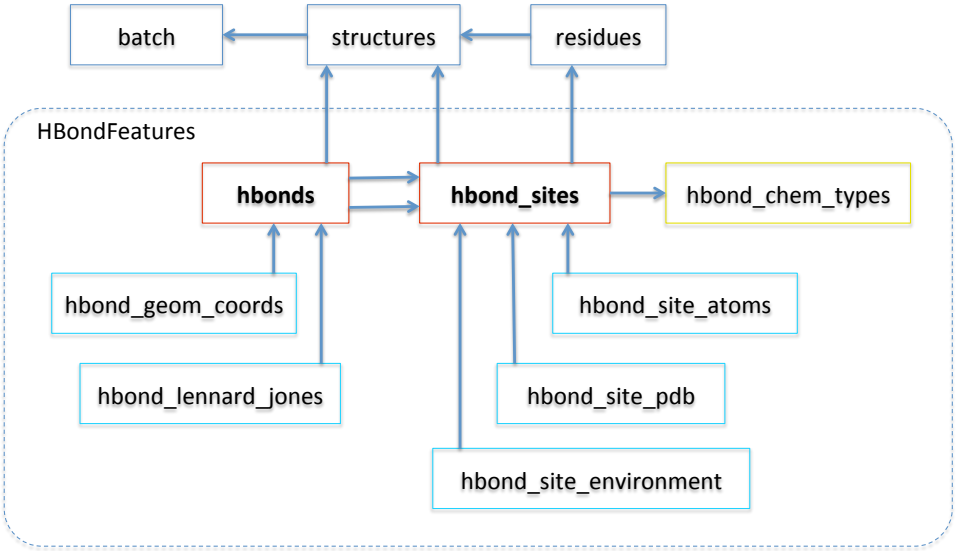


Figure 3.2.2 HBond feature database schema: Each box represents a table and each arrow represents a foreign key constraint. Associated with each H-bond site are the atomic coordinates, and experimental and solvent environment features. Associated with each H-bond are the donor and acceptor sites (stored in the `hbond_sites` table), the geometric coordinates (i.e., distances and angles), and the sum of the Lennard–Jones energies for H-bonding atoms.

Recall that a feature is defined as a random variable over a feature space; this means that features are not explicitly stored in the features database. Rather, the values stored in the database are samples from feature random variables. This distinction is important because querying the database can define new features by constraining feature spaces that are their domains. For

example, two different features defined by subsets of the same set of samples are residue types of all residues and residue types of all α-helical residues. The former is stored directly in the `residues` table of a features database and the latter can be obtained through a query that joins the residues table and the `residues_secondary_structure` table to constrain the DSSP type. Representing feature data as samples in a relational database allows a wide range of features to be accessible with minimal duplication of the actual stored values, making relational databases space efficient.

3.2.4 Reporting Feature to the Database

The upstream component of the features analysis tool facilitates reporting sampled feature values to the features database. C++ classes called `FeaturesReporters` are responsible for populating a set of tables. An analyst specifies the features reporters of interest.

I develop on the Rosetta platform a framework that can extract features either from previously generated batches of conformations or on-the-fly from Rosetta-generated predictions.

To report features, an analyst selects a set FeaturesReporters through the RosettaScripts XML-based protocol language (Fleishman et al., 2011). Implemented and proposed feature reporters are shown in Figure 3.2.1. The RosettaScripts framework allows users to specify a sequence of Movers that can each modify the conformation. I create a simple Mover, called ReportToDB, which does not actually move anything, but provides an interface to the features reporting framework to output all instances of features specified by the user. For example, including ReportToDB as the only Mover in a RosettaScripts protocol reports the features for the input. This allows conformations sampled from native and externally generated sample sources to be reported to the features database by the same mechanism. By including the ReportToDB mover at

the end of a prediction protocol, features for Rosetta generated predictions can be seamlessly reported to a features database.

From a developer's perspective, new features can be defined and reported to the database by implementing to a FeaturesReporter C++ interface and building on the functionality available through the Rosetta platform. This process is described in more detail in Section 3.2.5.b.

Batch	One Residue	Two Residue	Multi Residue
Batch	Residue	Pair	Structure
Protocol	ResidueConformation	AtomAtomPair	PoseConformation
JobData	ProteinResidueConformation	AtomInResidue-	RadiusOfGyration
PoseComments	ProteinBackboneTorsionAngle	AtomInResiduePair	SecondaryStructure
	ResidueBurial	ProteinBackbone-	SecondaryStructureSegment
Experimental Data	ResidueSecondaryStructure	AtomAtomPair	Smotif
PdbData	GeometricSolvation	HBond	HelixBundle
UnrecognizedAtom	BetaTurnDetection	Orbitals	StrandBundle
PdbHeaderData	Rotamer	SaltBridge	HydrophobicPatch
DDG	Rotamer Recovery	LoopAnchor	Rigidity
NMR	RotamerBoltzmannWeight	ChargeCharge	VoronoiPacking
DensityMap	ProteinBondGeometry	DFIREPair	
MultiSequenceAlignment	HelixCapping		Energy Function
HomologyAlignment	ResidueLazaridisKarplusSolvation	Multi Structure	ScoreFunction
	ResidueGeneralizedBornSolvation	ProteinRMSD	ScoreType
Chemical	ResiduePoissonBoltzmannSolvation	RecoveryBenchmark	StructureScores
AtomType	Pka	ResiduePairRecovery	ResidueScores
ResidueType	ResidueCentroids	ResidueClusterRecovery	HBondParameters ResidueTotalScore
			ResidueGridScoresFeatures

Table 3.2.3 Features Database Schema: Each cell is a feature type consisting of one or more tables in the database. Each table stores samples of feature instances and relations to other feature types. Features are organized into the following categories: Batch: metadata about the batch sampled from a sample source. Experimental Data: non-conformational data obtained along with experimental assays. Chemical: Assignments of chemical classifications to covalently bound arangments of atoms. One/Two/Multi Residue: Features defined over one, two, or multiple residues. Multi Structure: Special features that relate multiple conformations within a batch. Energy function: values of computational feature models evaluated over features. Features in gray are proposed, but not yet implemented.

3.2.5 Feature Batch Comparison

The downstream framework of the features analysis tool facilitates comparing batches of features stored in a features database. The tool provides a framework for running R-based features analysis scripts, methods to assist in features analysis tasks, and a community repository for developed features analysis scripts. The prototypical features analysis script specifies input features databases, extracts and filters feature instances, normalizes and transforms them, then compares the batches of feature instances through either summary statistics, visual comparison of distribution functions, or quantitative two-sample tests. I describe details of the functionality supported by the downstream framework in Section 3.2.5.c, below, and demonstrate examples of use in the case studies (Sections 4.1-2 and 6.3-5).

The analyst can run a sample source comparison by minimally specifiing a features database and one or more analysis scripts to run and format the analysis output. Additionally, the analyst can specify analysis script specific parameters, such as whether a sample source is a reference or not, or parameters of the kernel density estimation bandwidth selection method and adjustment factor. The dowstream framework runs each features analysis script, initializing it with the sample source and output parameters and additional parameters.

4 Case Studies Demonstrating Usage of the Features Analysis Tool

In this chapter, we walk through the first three of six case studies that demonstrate different aspects of the usage of features analysis tool. The first case study (Section 4.1) is a detailed tutorial to show the use of the tool step by step. The second case study (Section 4.2) demonstrates use of the tool to diagnose pathologies in a prediction method. The third case study (Section 4.3) demonstrates the use of features analysis to create a complete feature model and integrate it into an energy function. The remaining three case studies, in Sections 6.3–5, following a discussion of energy based computational models, demonstrate how the tool can be used to create and assess energy functions, tune their parameters, and determine how they should be combined.

Each case study follows the features analysis workflow outlined in Section 3.2—specifying one or more sample sources, storing sampled features into a features database, extracting relevant features and possibly applying transformations, and analyzing feature samples through exploratory data analysis. Through the analysis, I make observations that suggest further sample sources and feature analyses to explore, which I pursue through iterating the workflow.

Each case study not only demonstrates the features analysis tool, but also addresses a research question in structural biology, showing how the tool helps the researchers to assess and improve their computational models.

4.1 Bond Angle Variation with the Backrub Move

This case study walks through a complete features analysis with my tool. I investigate if a local sampling move called the Backrub distorts N-Cα-C bond angles. After elaborating on this

question (Section 4.1.1), I define the sample sources and extract features (Sections 4.1.2-3). I specify a features analysis script and use it to compare feature distributions (Sections 4.1.4-6). Additionally, I demonstrate how the features analysis tool can be used to investigate the causes of the observed features by iterating the process (Section 4.1.7-8).

In this case study, I compare N-Cα-C bond angle features from Rosetta decoys generated using the Backrub move against the angles observed in Natives. I confirm the observations from the literature that native N-Cα-C bond angles vary up to 6.5° and depend on the local backbone conformation. I further observe that the recommended sampling bias and bond angle restraint used with the Backrub Move in Rosetta systematically predict N-Cα-C bond angles to be 1.5° too tight (110° vs. 111.5°). Based on this features analysis, I propose an alternative bond angle restraint model that does not have this bias. Further features analysis shows, however, that neither model is able to recapitulate variation by secondary structure classification.

4.1.1 Does the Backrub Move Distort Bond Angles?

The Backrub Move, designed by Ian Davis (2006), is a procedure to locally modify the backbone conformation while keeping the remainder of the structure fixed and he used it as a tool for refining crystal structures by capturing local motion observed in native protein backbones. The Backrub move was adapted in 2008 for the Rosetta platform (Friedland 2008, Smith 2008), since its locality makes it suitable move for Markov chain Monte Carlo conformation sampling algorithms.

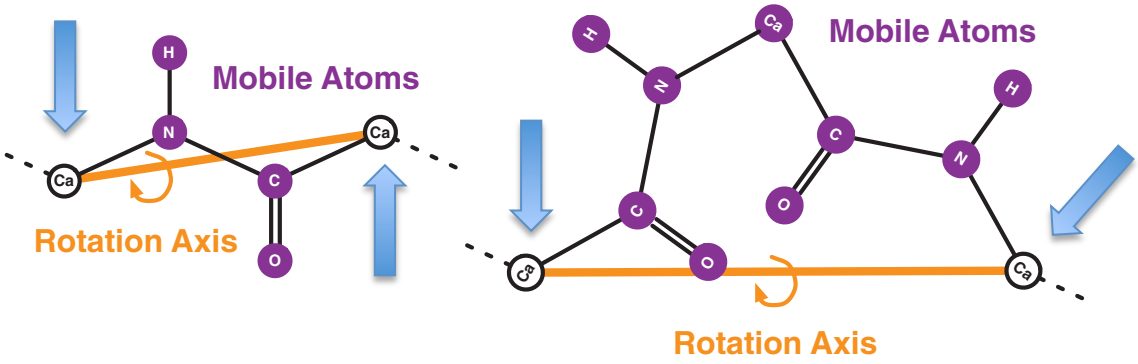


Figure 4.1.1 The Backrub Move (Freidland 2008): The Backrub Move selects pivot atoms (white circles labeled Ca) and rotates the mobile atoms (purple circles) about the rotation axis (orange line). The blue arrows point to the bond angles that may be distorted by a Backrub Move.

In Rosetta's implementation of the Backrub Move, two Cα atoms close in primary sequence are chosen as pivot atoms and a rotation axis is drawn through them. The atoms in the sequence between the pivots, called the mobile atoms, rotate as a rigid body about the rotation axis (Figure 4.1.1). If the bonds extending beyond the pivot atoms lie on the rotation axis, then the backrub motion does not change the bond angles at the pivot atoms (the dotted lines Figure 4.1.1). If these bonds do not lie on the rotation axis, however, then the backrub motion bends the bond angles. In Davis (2006), secondary axes of rotation in the mobile segment are used to relieve bond angle strain caused by the torque about the primary axis of rotation; however, this extra behavior was not implemented in Rosetta.

Protein modelers typically assume that bond lengths and angles are fixed at ideal values. Evidence to support this assumption is that the N-Cα-C bond angle (the angle bent by backrub moves) is seen to vary only up to 6.5° in native conformations (Figure 4.1.2). With this assumption, however, no backrub moves would be possible; To make backrub motions possible, the N-Cα-C bond angle must be allowed vary, but then care must be taken that the resulting angles are still physically plausible. Using features analysis, we can assess whether the

distribution of angles after a Backrub move is consistent with the distribution of angles observed in the Native sample source. The bond angle restraint, sampling bias, and aggregate factors from rest of the energy function and prediction protocol can all affect the bond angles of predicted conformations; I would like to assess not only distorted angles, but also the cause.

Approaches proposed to actively control the N-Cα-C bond angle distribution include biasing the Backrub sampling to propose moves conditional on amount that they bend the N-Cα-C bond angle (Betancourt 2005) or by incorporating a restraint on the angle in the energy function (Smith 2008). In implementing the Backrub move in Rosetta, Smith et al. combined both approaches, taking a bond angle restraint, the `mm_bend` term from AMBER ff94 (Cornell 1995) and CHARMM22 (MacKerell 1998), and biasing the acceptance ratio as a function of bond angle deviation.

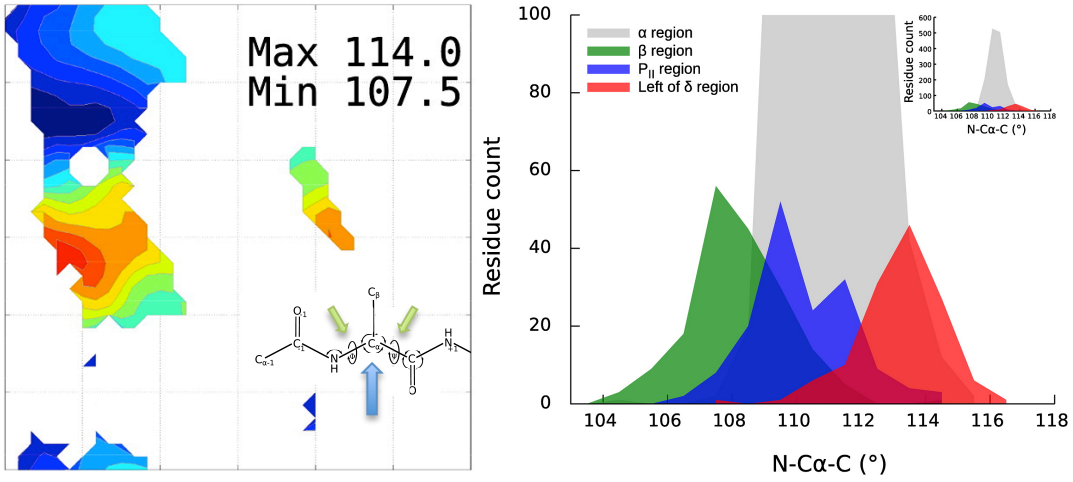


Figure 4.1.2 N-Cα-C Bond Angle Variation in Natives (Berkholz 2009): (LEFT) The N-Cα-C bond angle (inset diagram; blue arrow) was measured conditional on adjacent phi/psi torsion angles (inset diagram; green arrows) in Berkholz 2009. The mean N-Cα-C bond angle is plotted conditional on the backbone φ angle (x-axis) vs. ψ angle (y-axis) with red representing tight angles (min 107.5°) to blue representing wide angles (max 114.0°). (RIGHT) N-Cα-C bond angle variation histogram by region of the plot on left.

In the following sections, I detail a features analysis to compare the distribution of N-Cα-C bond angles from a Native sample source, called the Top8000, and Rosetta sample sources that use backrub sampling to generate structures. As the first step, I collect the data from sample sources and populate features databases. As the second step, to analyze the feature data, I specify a features analysis script.

4.1.2 The Native Sample Source

To obtain a sample of conformations that represents naturally occurring proteins, I use the Top8000 chains dataset from (Keedy 2012, Richardson 2013). This is a set of protein chains of at least 37 residues taken from structures determined by X-ray crystallography at reported resolution of at most 2 Å and deposited with electron density maps into the Databank on or before March 29, 2011. Each chain was processed with Reduce (Word 1999) to place H-atoms and resolve flipped Asn/Gln/His sidechains. These chains were filtered to have MolProbity scores (Chen 2010) better than 2.0, and angle, bond length or C-beta deviation outliers in at most 5% of their residues. These conformations were then hierarchically clustered by sequence identity at the 70% level, and the chain with the best average resolution and MolProbity score from each cluster was selected. This gives 6,656 chains and 1,582,859 residues, which I call the Native sample source.

```
sql_query <-"
  SELECT max_temperature
  FROM residue_pdb_confidence;"
f <- query_sample_sources(
  sample_sources, sql_query)
qs <- compute_quantiles(
  f, c(), "max_temperature", 1000)
ggplot(data=qs) + theme_bw() +
  geom_line(
    aes(y=probs, x=quantiles, color="red")) +
  scale_y_continuous(
    limit=c(0,1), breaks=0:3*.25) +
  scale_x_continuous(
    breaks=c(10, 20, 30, 50, 75, 100)) +
  coord_trans(
    x="sqrt", y="identity")
save_plots(self, "befactors", ...)
```

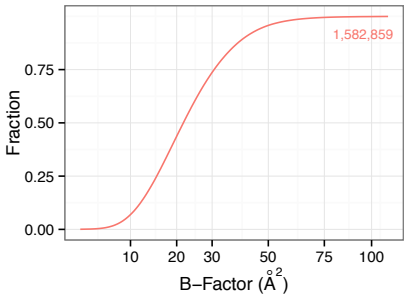


Figure 4.1.3 Native B-Factors: (left) Emperical cumulative distribution function for B-Factor from the Native sample source. The X-axis is the maximum B-factor over all atoms the residue plotted on the square root scale. The Y-axis is the fraction of residues that have at most that B-factor. (right) The features analysis script to generate the plot. The script defines an SQL query to retrieve the B-factor values and queries the sample source to store the data in **`f`**. The cumulative distribution function is computed with the `compute_quantiles` function with no grouping covariates. The result is stored as a `data.frame` with the columns `probs` and `quantiles`. These columns then are mapped using the grammar of graphics (Section 5.2) to the x- and y-aesthetics of the line geometric primitive of the `ggplot`. The plot is scaled, and the coordinates are transformed. For clarity, the count indicator, axis labels, and standard arguments to `save_plots` have been omitted.

4.1.3 The Backrub Sample Source

To generate conformations comparable with the Native sample source and to exercise the backrub prediction protocol, for each chain I perform a 10,000-step MCMC simulation with unit temperature, beginning at the native conformation and consisting of backrub moves (3/4 of the time) and sidechain moves (1/4 of the time). Both of these moves have been designed to satisfy a detailed balance, which means that as the number of steps approaches infinity, the sample approaches an unbiased sample from the probability distribution related to the energy function by the Boltzmann equation. A 10,000 trial simulation does not move a conformation substantially from its starting conformation, but it does adjust enough local features to support features analysis.

To execute the MCMC simulation and extract bond length and angles, I used the RosettaScripts interface to Rosetta with the following script

```
<ROSETTASCRIPTS>
  <SCOREFXNS>
    <s weights=score12_full>
      <Reweight scoretype=mm_bend weight=1/>
    </s>
  </SCOREFXNS>
  <TASKOPERATIONS>
    <ExtraRotamersGeneric name=extra_chi ex1=1 ex2=1 extrachi_cutoff=0/>
    <RestrictToRepacking name=rtrp/>
    <PreserveCBeta name=preserve_cb/>
  </TASKOPERATIONS>
  <MOVERS>
    <MetropolisHastings name=mc scorefxn=s trials=10000>
      <Backrub sampling_weight=.75/>
      <Sidechain
        sampling_weight=0.25
        task_operations=extra_chi,rtrp,preserve_cb/>
    </MetropolisHastings>

    <ReportToDB
      name = features_reporter
      database_name = "features_top8000_backrub_r50086.db3"
      batch_description = "Backrub Ensemble">
      <feature name = ResidueFeatures/>
      <feature name = PdbDataFeatures/>
      <feature name = ResidueSecondaryStructureFeatures/>
      <feature name = ProteinBondGeometryFeatures/>
    </ReportToDB>

  </MOVERS>
  <PROTOCOLS>
    <Add mover_name = mc/>
    <Add mover_name = features_reporter/>
  </PROTOCOLS>
</ROSETTASCRIPTS>
```

The <SCOREFXNS/> block defines a score function s using the score12_full weight set, which is the standard score function in the Rosetta community prior to the research reported in this dissertation. It enables the mm_bend score term to restrain bond angle deviations, as recommended by the implementers of the Backrub Move. The <TASKOPERATIONS/> block initializes configuration parameters for the sidechain moves. The <MOVERS/> block initializes the MCMC protocol and the ReportToDB mover, which will write to an SQLite3 database and report 4 types of features.

To extract features from the Native sample source, I simply remove the `mc` mover from the `<PROTOCOLS/>` block and store features extracted from the input structures in another SQLite3 database, `features_top8000_r50086.db3`.

I have now demonstrated how populate the features databases with sampled features instances from different sample sources. By providing Rosetta scripts protocols and features to extract, the feature databases can be populated with different sorts of feature data. For the rest of this case study I compare the features from the Backrub sample source with several variants of the bond angle restraint. In section 4.1.7 I consider varying the strength of the mm_bend restraint and in 4.1.8 I consider varying the strength of the `cart_bonded` restraint.

4.1.4 Specification of a Features Analysis Script

I describe conducting a features analysis top down, starting with specifying the sample sources, the analysis scripts to run, and the output formats in a configuration file. This is then passed to `compare_sample_sources.R`, to run it.

```
{"sample_source_comparisons" : [{
    "sample_sources" : [{
      "database_path" : "features_top8000_r50086.db3",
      "id" : "top8000",
    }, {
      "database_path" : "features_top8000_backrub_r50086.db3",
      "id" : "top8000_backrub",
    }],
    "analysis_scripts" : [
      "scripts/analysis/plots/backbone_geometry/bond_angles.R"
  ],
  "output_dir" : "build",
  "output_formats" : ["output_print_pdf"]
  }]
}
```

Features analysis scripts are collected in the Rosetta repository, but to demonstrate how to make a new features analysis script, I begin by making a script that generates Figure 4.1.3 and then extend it to a more in-depth investigation.

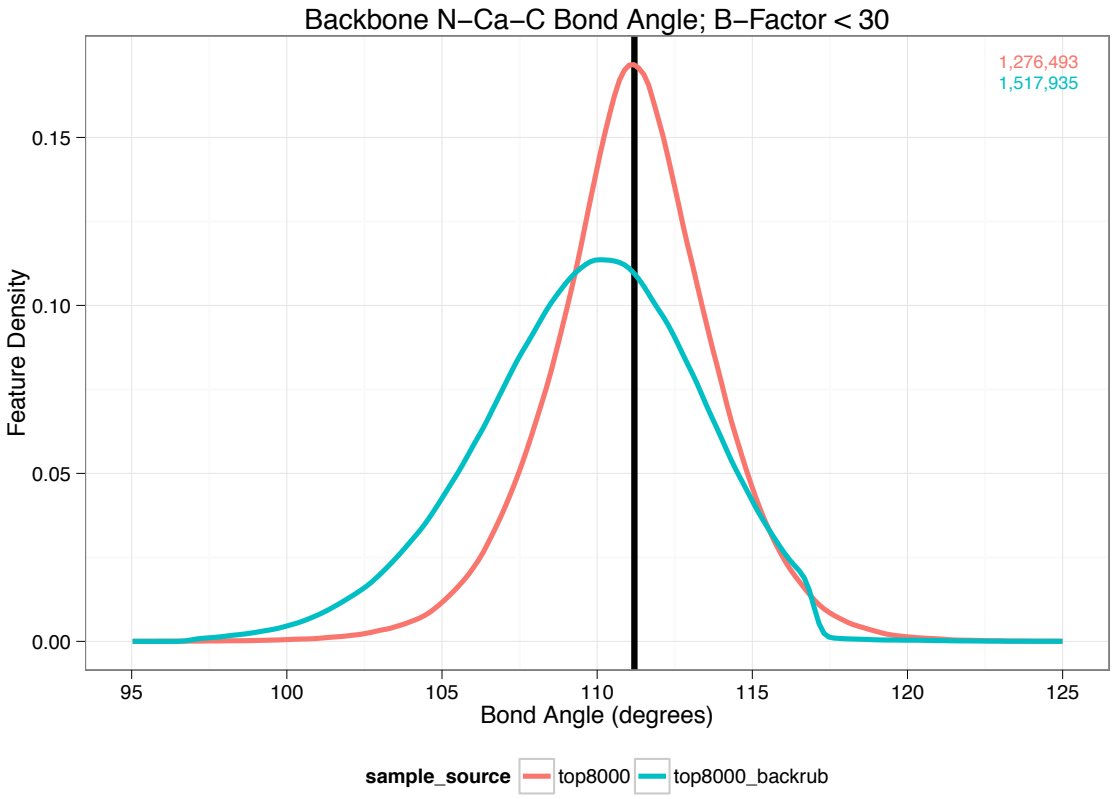


Figure 4.1.4 N-Cα-C Bond Angle Native vs. Backrub: Kernel density estimation of N-Cα-C bond angle for Native (red) and Backrub (cyan) sample sources. The numbers in the upper right indicate the sample size for each sample source. The plot is generated by the features analysis script `scripts/analysis/plots/-backbone_geometry/bond_angles.R` described below.

I start writing the `bond_angles.R` features analysis script with a standard script template whose bold and dotted parts must be filled in.

```
run=function(self, sample_sources, output_dir, output_formats){

f <- query_sample_sources(sample_sources, sql_query)
dens <- estimate_density_1d(f, id_columns, measure_column)
p <- ggplot(dens) +
  geom_line(…) +
  geom_vline(…) +
  scale_x_…(…) +
  scale_y_…(…) +
  theme(…)
save_plots(self, plot_id, output_dir, output_formats)

}
```

I initially specify the SQL query to extract the ideal N-Cα-C bond angle and the observed angle from the `bond_intrares_angles` table (reported by the `ProteinBondGeometry-Features` reporter) for each residue in each sample source.

```
sql_query <- "
SELECT
  b_ang.ideal,
  b_ang.observed
FROM
  bond_intrares_angles AS b_ang
WHERE
  b_ang.outAtm1Num = 1 AND b_ang.cenAtmNum = 2 AND b_ang.outAtm2Num = 3;"
```

The experimental data I use is from X-ray crystal structures and one characterisic of the data is that uncertainty of the atomic coordinates is not uniform for all atoms. Therefore, when analysing feature distributions it is important to filter out residues that have high uncertainty. One way of doing this is to filter on the experimentally reported B-factors. Since this filtering will be used in all further case studies I will describe it in detail, which will be clearer after the first reading of this case study.

B-Factors attempt to capture the observed disorder for an atom in the electron density, and are sometimes called temparature factors because usually, at higher temperatures, there is more disorder. B-Factors have units Ångstroms squared, with lower values being more ordered. Their use is somewhat problematic becuase there are many causes for observed disorder, ranging from molecular motion in the crystal to experimental error. Further, their values are not effectively normalized across different deposited structures in the protein databank. However, they do communicate some information and can be used to filter unobserved residues from well-resolved residues. The cutoff of 30 that I use keeps approximately 75% of the residues, whereas a cutoff of 20 is more stringent, keeping less than 50% of the residues (Figure 4.1.3).

```
sql_query <- "
SELECT
  b_ang.ideal,
  b_ang.observed
FROM
  residues AS res,
  residue_pdb_confidence AS res_conf,
  bond_intrares_angles AS b_ang
WHERE
  res_conf.struct_id = res.struct_id AND
  res_conf.residue_number = res.resNum AND
  res_conf.max_temperature < 30 AND
  b_ang.struct_id = res.struct_id AND
  b_ang.resNum = res.resNum AND
  b_ang.outAtm1Num = 1 AND
b_ang.cenAtmNum = 2 AND b_ang.outAtm2Num = 3;"
```

I specify the use of 1-dimensional kernel density estimation, using default parameters, to estimate the density distribution for the observed N-Cα-C bond angles for each sample source.

```
   dens <- estimate_density_1d(f, c("sample_source"), "observed")
```

Returning to filling in the template, I then extend this SQL query to filter by the experimentally reported B-Factor values for each residue, which are stored in the `residue_pdb_confidence` table reported by the `PdbDataFeatures` reporter. Although the `residue_pdb_confidence` table and the `bond_intrares_angles` table could be joined directly in the `WHERE` clause, I do the join through the residues table because I will need it in the next step.

To separate the specification of the data and the specification of the visualization of the data I specify the plot using the grammar of graphics (Wilkinson 2005, Wickham 2010), which is a rigorous method for creating plots by mapping data to layers of aesthetic elements on the page. While use of the grammar of graphics is not essential for the use of the features analysis tool, I highly recommend it for its power and flexibility.

I choose to plot 3 layers of data: first, the ideal bond angle as a reference vertical line; second, the distribution of observed bond angles for each sample source as a line where the color corresponds to the sample source; and third, an indicator displaying how many counts were observed to help calibrate the user's confidence in the estimated density. To do this, I map columns in the `dens` and `f` `data.frame` objects to aesthetic elements in each layer. The `geom_indicator` is a geometric element I have implemented:

Layers:

- geom_line: data=`dens`
  - x=`x`, y=`y`, color=`sample_source`
- geom_vline: data=`f`
  - x=ideal
- geom_indicator: data=`dens`
  - indicator=`counts`
  - color=sample_source

I deposit my new features analysis script into the Rosetta git code repository under `main/tests/features/scripts/analysis/plots/backbone_geometry/bond_angles.R`, so it can be run and made available to others.

4.1.5 Preliminary Features Analysis

Running the `compare_sample_sources.R` features analysis driver script with the above configuration file gives this output, with the generated plot shown in Figure 4.1.3.

```
~/rosetta/rosetta/rosetta_tests/features/compare_sample_sources.R \
  --config analysis_configurations/bond_angles.json

Sample Source Comparison:
  Output Directory <- '/path/build/top8000_top8000_backrub'
    Output Formats <- output_print_pdf

  Sample Sources:
  top8000 <- features_top8000_r50086.db3
  top8000_backrub <- features_top8000_backrub_r50086.db3

  Analysis_scripts:
  scripts/analysis/plots/backbone_geometry/bond_angles.R

Features Analysis: bond_angles
loading: top8000 ... 23.33 s
loading: top8000_backrub ... 22.55 s
Saving Plot:
build/top8000_top8000_backrub/bond_angles/output_slide_pdf/backbone_geometry_bo
nd_angle_NCaC_120727.pdf ... 0.33s
```

Already in Figure 4.1.3 there are several observations that can be made:

1. With over a million samples from each sample source to generate the smoothed distributions, we can trust the visual representation of the distributions.
2. The center of the Native distribution aligns closely with the ideal value.
3. The width of the simulated distribution is qualitatively in agreement with the width of the Native distribution.
4. The peak of the simulated distribution is shifted to tighter angles.
5. There appears to be an artificial discontinuity in the simulated distribution occurring around 117°.

To show the flexibility of the features analysis tool, it is easy to recreate the plots from Berkholz 2009 by simply extending the above SQL to measure the φ/ψ angles for each residue as covariates:

```
sql_query <- "
SELECT
  b_ang.ideal,
  b_ang.observed,
  b_tor.phi,
  b_tor.psi
```

```
FROM
  residues AS res,
  residue_pdb_confidence AS res_conf,
  bond_intrares_angles AS b_ang,
  protein_backbone_torsion_angles AS b_tor
WHERE
  res_conf.struct_id = res.struct_id AND
  res_conf.residue_number = res.resNum AND
  res_conf.max_temperature < 30 AND
  b_ang.struct_id = res.struct_id AND b_ang.resNum = res.resNum AND
  b_ang.outAtm1Num = 1 AND b_ang.cenAtmNum = 2 AND b_ang.outAtm2Num = 3 AND
  b_tor.struct_id = res.struct_id AND b_tor.resNum = res.resNum;"
```

Next, I hypothesize that there is dependence on the amino acid type of the residue and residue secondary structure. To investigate this hypothesis, I use DSSP (Kabsch 1983) to annotate residues by their secondary structure, and then extend the SQL query to extract two additional columns, `res_type` and `dssp`. For each of these two covariates, separately and combined, I estimate densities for each subgroup,

```
dens_res <- estimate_density_1d(
  f, c("sample_source", "res_type"), "observed")

dens_dssp <- estimate_density_1d(
  f, c("sample_source", "dssp"), "observed")

dens_res_dssp <- estimate_density_1d(
  f, c("sample_source", "res_type", "dssp"), "observed")
```

I then create several plots mapping the identifying covariates (`sample_source`, `res_type`, and `dssp`) to visual elements of the plot, in this case, the color and the facet cell. For each covariate, I first plot all levels on top of one another to get a gestalt view of the variation and formulate a mental hypothesis for dependence (Figure 4.1.4, 4.1.6), which I then check by mapping the levels to small multiple plots (Buja 2009).

4.1.6 Analysis of Features Data

In a typical exploratory data analysis campaign, the identification of hypotheses, and investigation of high bandwidth measure of the data such as density distribution plots iterated

interactively. To give the flavor of this investigation I show a representative set of plots and discuss their interpretation. I later case studies I do not always show the breadth and depth of plots.

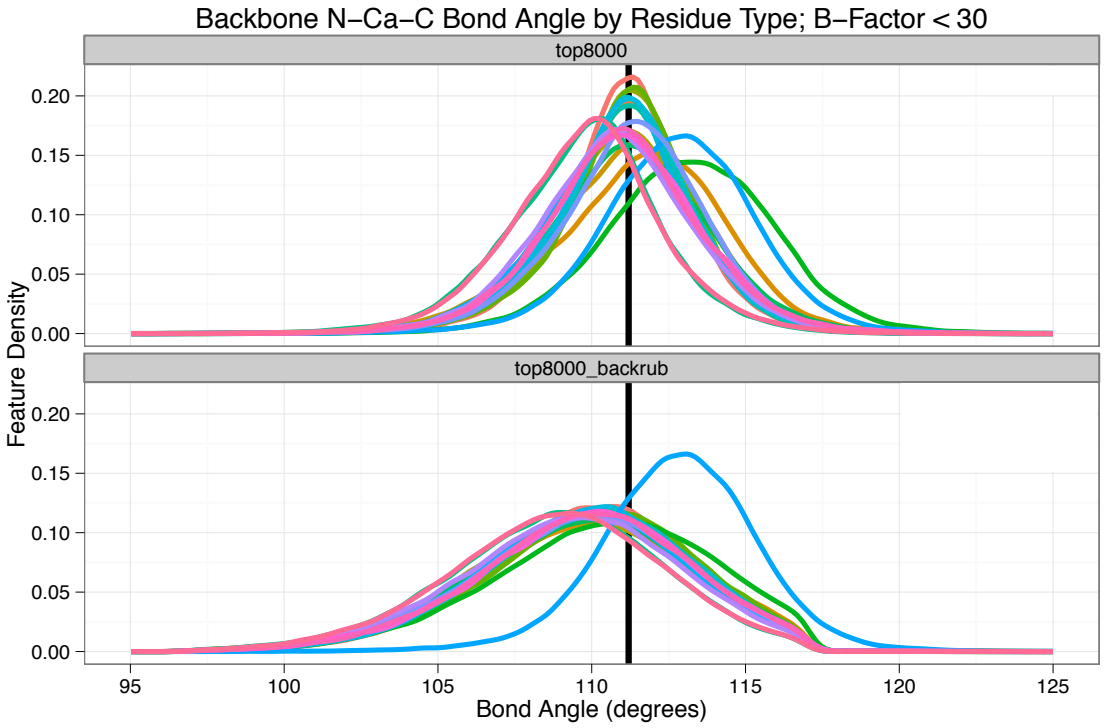


Figure 4.1.5 N-Cα-C Bond Angle by Sample Source and Residue Type: Kernel density estimation of N-Cα-C bond angle from Native and Backrub sample sources conditional on the residue type plotted by color in each facet cell. N-Cα-C bond angles from the Native sample source appear to vary by amino acid type, which is not recapitulated in the Backrub sample source. The outlying blue line is the striking exception. Inspection of Figure 4.1.5 reveals that the outlier is proline, and inspection of the code reveals that Backrub Moves are not considered for proline residues (since proline sidechains make a second attachment to the backbone at the N-atom).

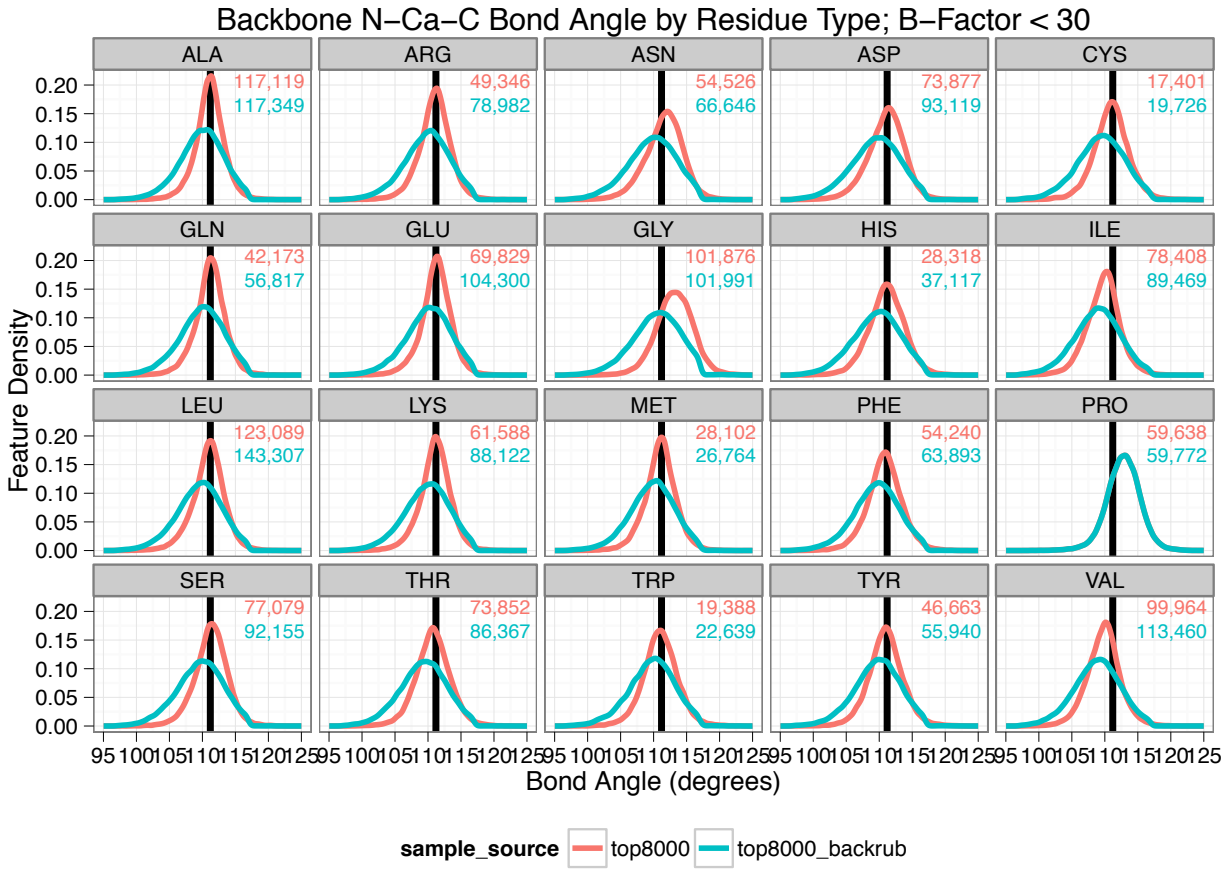


Figure 4.1.6 N-Cα-C Bond Angle by Residue Type and Sample Source: Kernel density estimation of N-Cα-C bond angle by residue type conditional on the sample source plotted by color in each facet cell. N-Cα-C bond angles from the Native sample source show variation by amino acid type. The proline distribution from the Backrub sample source angle distribution is identical to (and hides behind) the predicted, because Backrub Moves do not apply to proline.

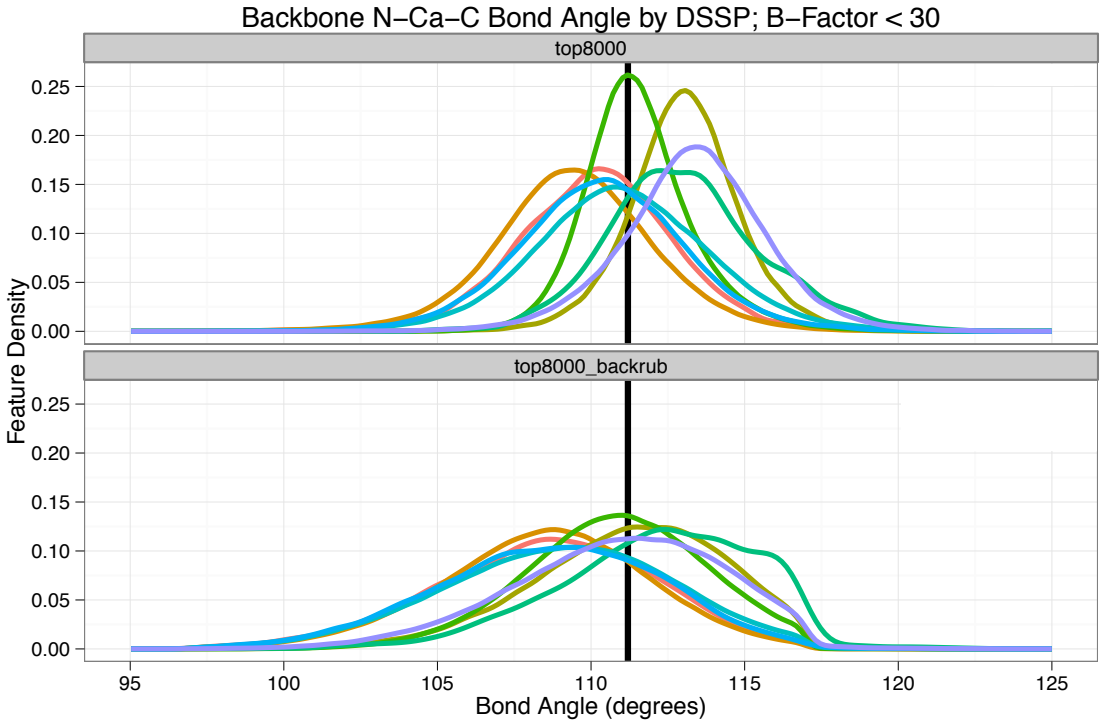


Figure 4.1.7 N-Cα-C Bond Angle by Secondary Structure and Sample Source: Kernel density estimation of N-Cα-C bond angle from Native (above) and Backrub (below) sample sources, with DSSP type plotted by color in each facet cell. Both Natives and backrub predictions show bond angle variation correlated with secondary structure.

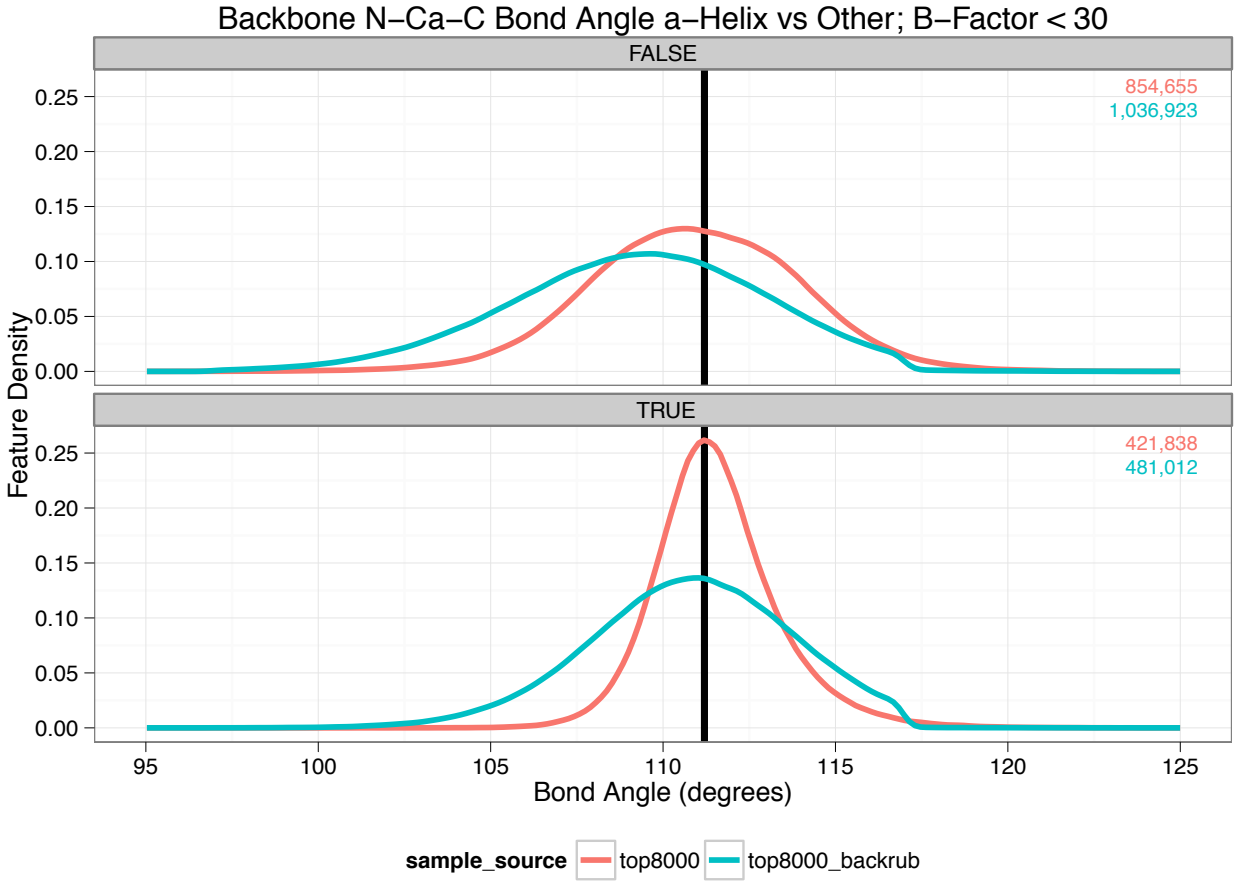


Figure 4.1.8 N-Cα-C Bond Angle by in α-helix and Sample Source: Kernel density estimation of N-Cα-C bond angle by DSSP==a-helix by sample source plotted by color in each facet cell. Since a large fraction of residues are in α-helix secondary structures. This figure tests bond angle variation is conditional on being in an α-helix. For non α-helical residues from the Backrub sample source, the previously observed shift to tighter angles (i.e. distribution shifted to the left) is visible. For α-helix residue form the Backrub sample source, the shift is not observed, this is consistent with the observation α-helices are highly constrained secondary structures that may not readily admit Backrub sampling. However, the bond angles in the backrub sample source have a wider variance than Natives.

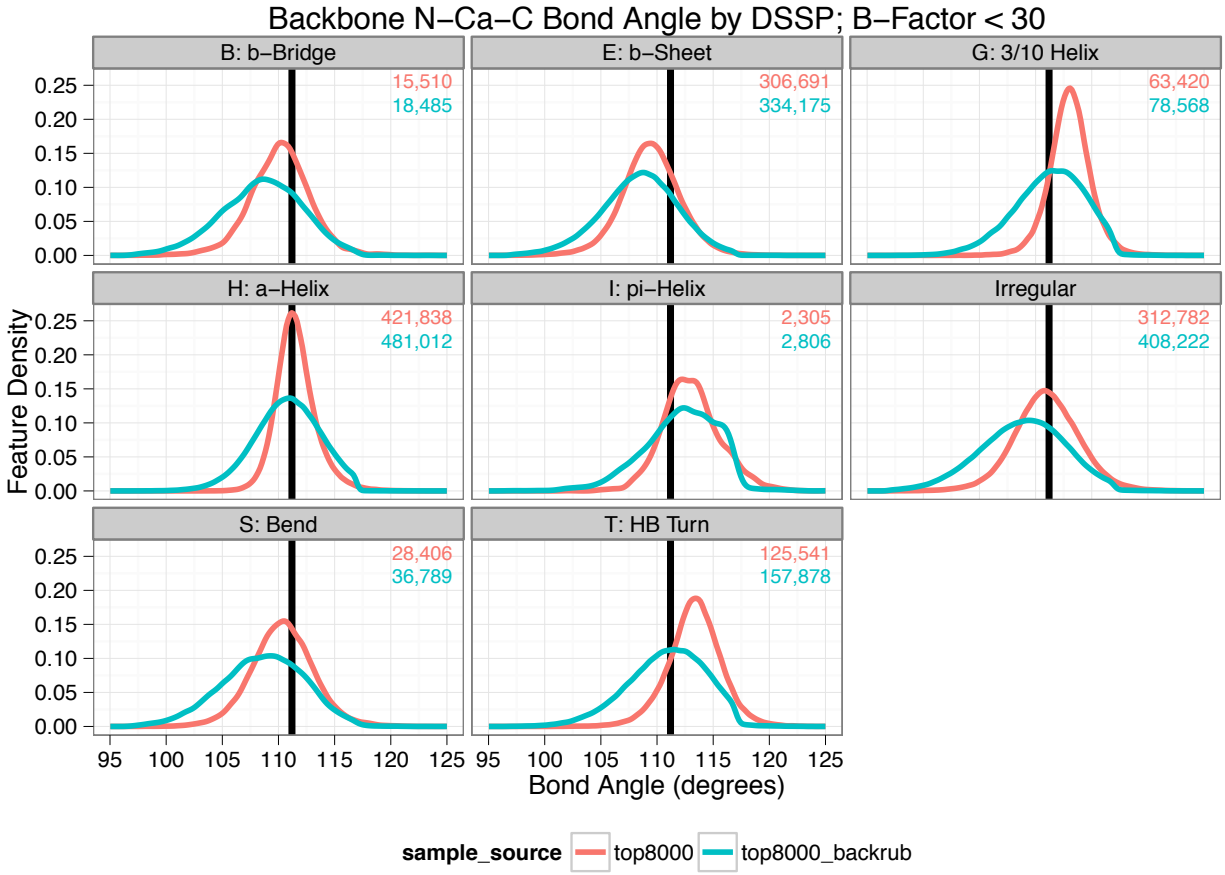
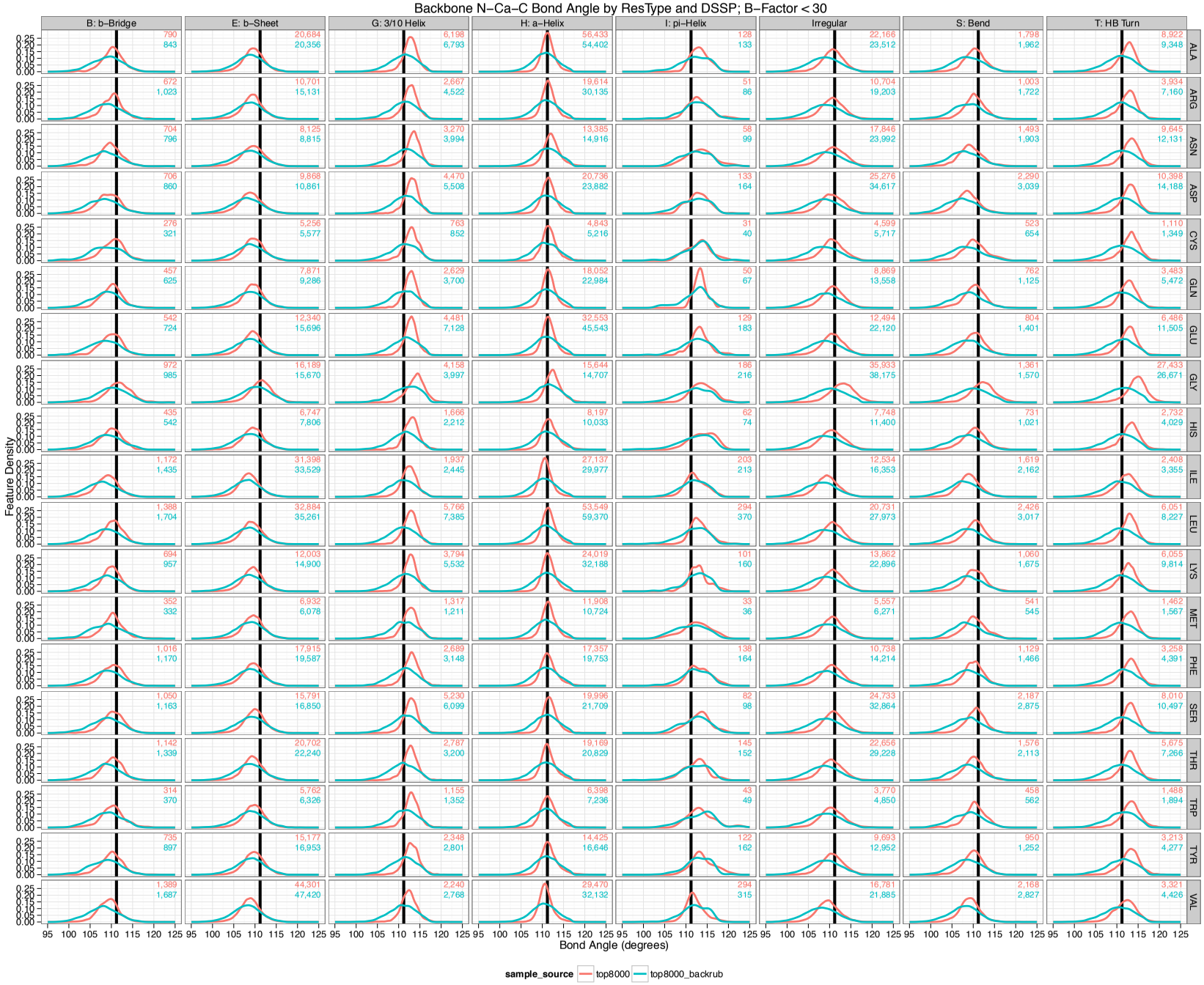


Figure 4.1.9 N-Cα-C Bond Angle by Secondary Structure and Sample Source: Kernel density estimation of N-Cα-C bond angle by DSSP type by sample source plotted by color in each facet cell. For the Backrub sample source the distribution visibly broadens and shift to tighter angles. N-Cα-C bond angles for the DSSP types of HB Turn and 3/10 helices from the Native sample source have particularly wide angles (relative to the "ideal" bond angle) and narrow distributions. Note in the N-Cα-C bond angles distributions for pi-Helix from the Backrub sample source the artificial shoulder in the density at 117°. The cause of this shoulder is described in Smith (2008), and it is a result of rejecting backrub moves that straighten bond angles too far from the ideal. Although this thresholding of Backrub Move acceptance does not affect the N-Cα-C bond angle distributions for most types of DSSP codes, it does impose limits on certain subpopulations, such as in pi-Helices. The discontinuity makes thresholding seem an inelegant way to narrow a distribution.

Figure 4.1.10 N-Cα-C Bond Angle by Residue Type, Secondary Structure, and Sample Source (on next page): Kernel density estimation of N-Cα-C bond angle by DSSP type (columns) and residue type (rows) by sample source plotted by color in each facet cell. The previously observed bond angle deviation appears to be more strongly due to the secondary structure type than the residue type (confirming the observations in Berkholz (2009)). Exceptions to this trend include N-Cα-C bond angles for glycine residues with Irregular DSSP codes; these appear to have a wider angle than other residues with the Irregular DSSP code. This makes biological sense because glycine is unique in that it has only a hydrogen atom in place of the sidechain and therefore is more flexible than other residue types and commonly found in less structured linker regions.


Backbone N–Ca–C Bond Angle by ResType and DSSP; B–Factor < 30
B: b–Bridge
E: b–Sheet
G: 3/10 Helix
H: a–Helix
I: pi–Helix
Irregular
S: Bend
T: HB Turn
ALA
ARG
ASN
ASP
CYS
GLN
GLU
GLY
HIS
ILE
LEU
LYS
MET
PHE
SER
THR
TRP
TYR
VAL
Feature Density
Bond Angle (degrees)
sample_source
top8000
top8000_backrub

In the preceding sections, I demonstrate that the features analysis tool can be used to identify patterns of feature distributions in order to compare differences between sample sources. I show that the visualization of the feature distributions facilitates rapid assessment at different levels of details for a large quantity of data. This analysis suggests that there is a systematic discrepancy with the backrub sample source with respect to the N-Cα-C bond angle distribution. In the next section, I transition to exploring why this occurs by adjusting the prediction protocol. This demonstrates the flexibility of the features analysis tool.

4.1.7 Investigating the Cause of the Bond Angle Distribution Discrepancy

To show how investigating different sample sources can multiply the utility of creating features analysis scripts, I investigate varying the default bond angle restraint in Rosetta, `mm_bend`.

The tightening of N-Cα-C bond angles from backrub predictions may indicate that the `mm_bend` term actively restrains the bond angle too tightly, or it may be caused by a combination of other factors in the energy function and sampling protocol. For example, the desolvation term disfavors exposing atoms to the solvent, thus favoring compacted conformations, and compactness may be facilitated by having more sharply bent backbone angles.

To test the hypothesis that tight bond angles are caused by the `mm_bend` term, I create two additional Backrub sample sources with different weights on the `mm_bend` term; one with weight 0 (weak) and one with weight 5 (strong); the original has weight 1 (medium). I regenerate the plots of Section 4.1.6; some of which are shown in Figures 4.1.10-12.

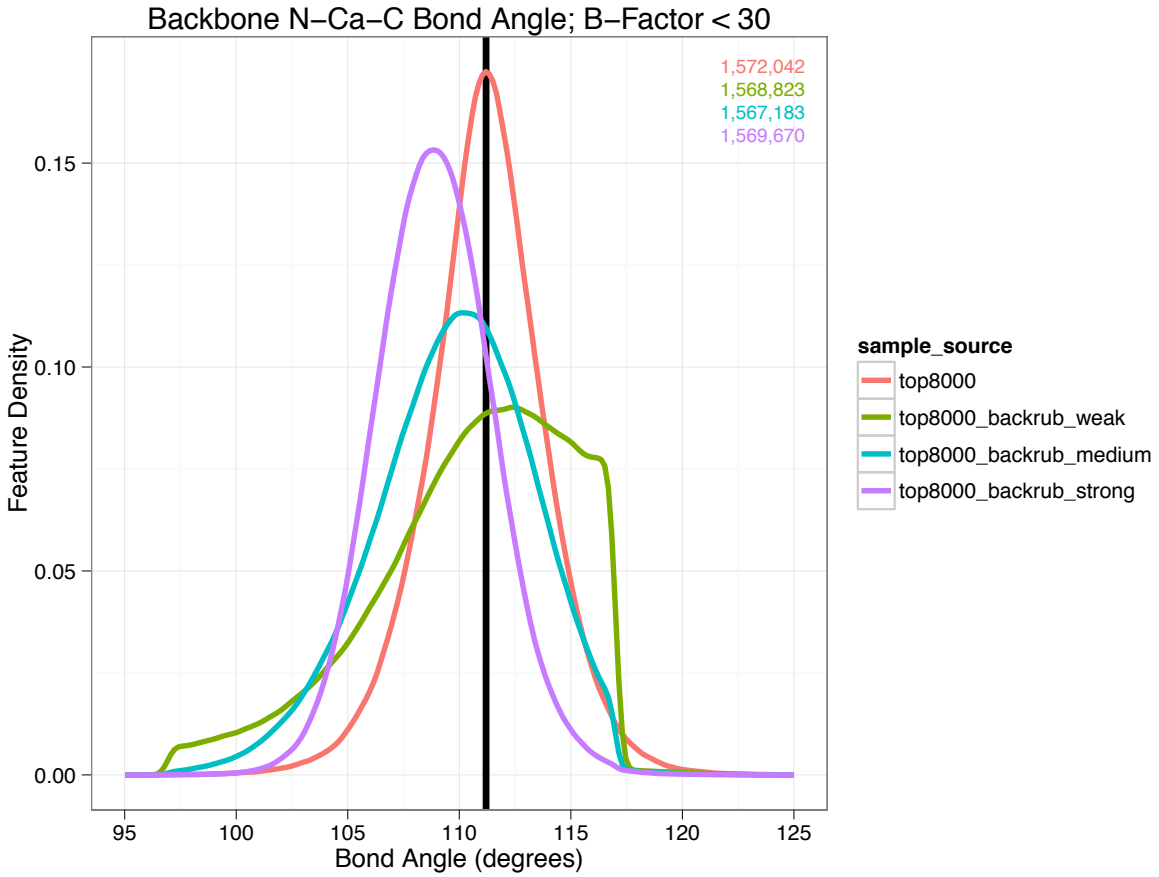


Figure 4.1.11 N-Cα-C Bond Angle by `mm_bend` Strength: Kernel density estimation of N-Cα-C bond angle by sample source plotted by color. (RED) Top8000, the Native sample source, (GREEN, BLUE, PURPLE) top8000_backrub_[weak, medium, strong], the Backrub sample source with the `mm_bend` term set to [0, 1, 5]. Increasing the weight of the `mm_bend` term shifts peak of the N-Cα-C bond angle distribution to tighter angles and makes the distribution narrower.

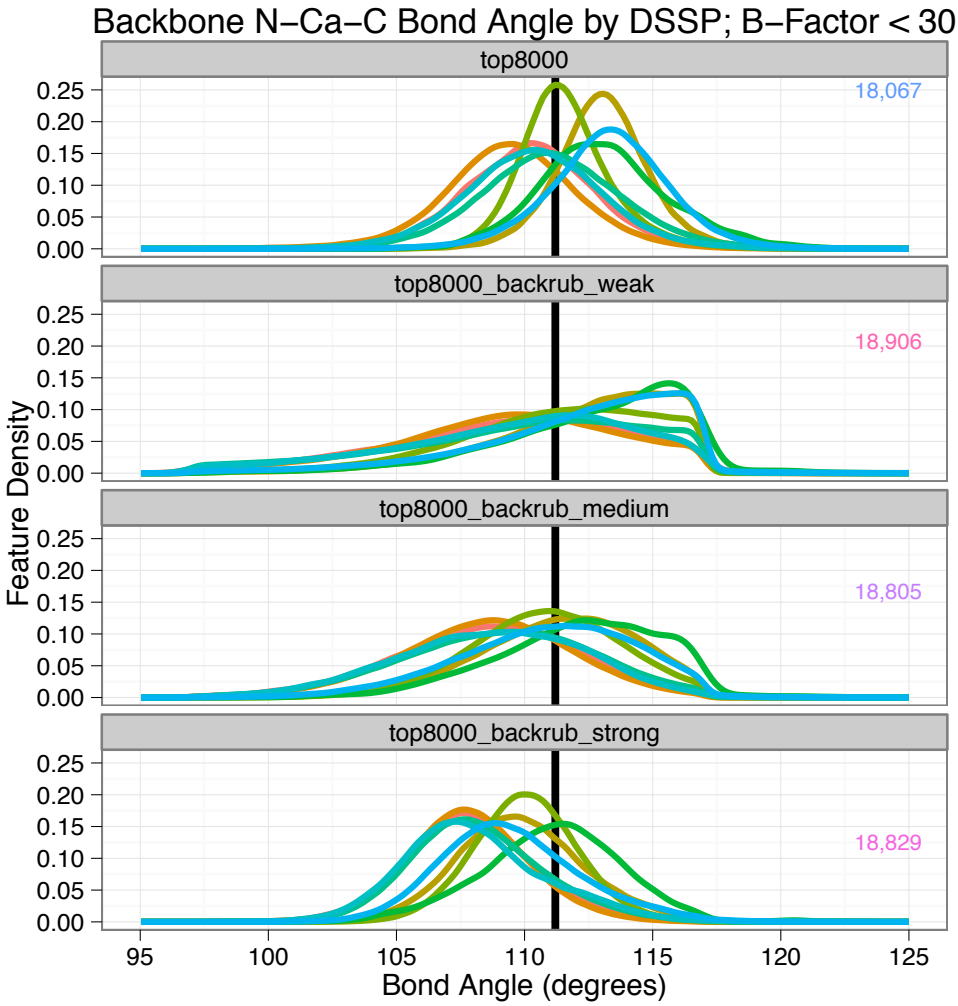


Figure 4.1.12 N-Cα-C Bond Angle by `mm_bend` Strength and Secondary Structure: Kernel density estimation of N-Cα-C bond angle by sample source for Native and Backrub sample sources with varying `mm_bend` strength by DSSP type plotted by color.

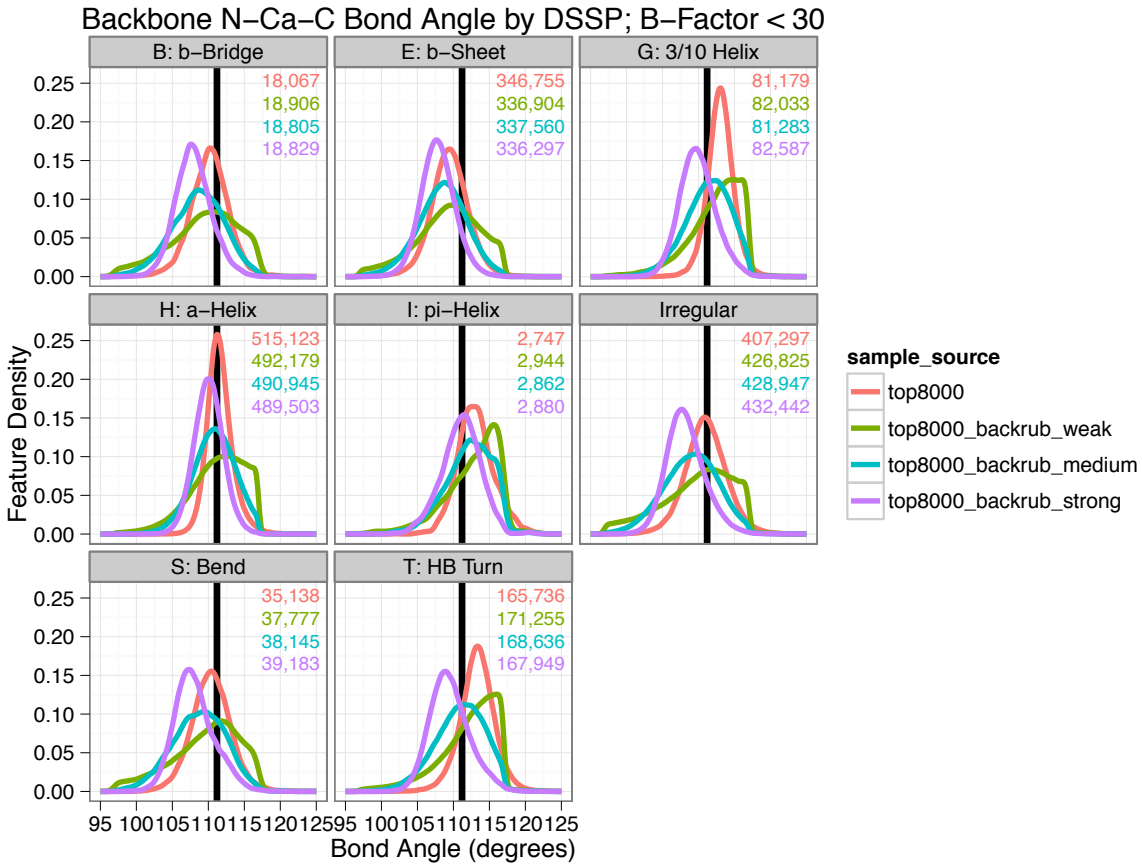


Figure 4.1.13 N-Cα-C Bond Angle by Secondary Structure and `mm_bend` Strength: Kernel density estimation of N-Cα-C bond angle by DSSP by sample source plotted by color.

Figures 4.1.10-12 show that without the `mm_bend` term (weak), the distribution of bond angles has a mean shifted to wider angles and a broader distribution. When the `mm_bend` term is set to 5 (strong), the bond angles are tighter and the distribution is narrower. Even though the weak Backrub sample sources have no energetic restraint on the N- Cα-C bond angle, their distribution still shows some angular preference. This may indicate bias due to starting at the Native conformation or contributions from other aspects of the simulation. The shift in mean with increasing the strength of the `mm_bend` term, however, indicates that the `mm_bend` term responsible for giving too tight bond angles in the Backrub sample source.

Possible actions to resolve this discrepancy with the use of the Backrub move modifying the `mm_bend` bond angle restraint, using an alternative restraint such as the `cart_bonded` term, under development for use with Cartesian space minimization in Baker Lab by Dr. Dimaio and Mr. Conway, or adjusting the MCMC simulation parameters.

4.1.8 Evaluation of the `cart_bonded` for Backrub bond Angle Restraint

In this section, I evaluate the `cart_bonded` angle restraint (using the version available 12/8) for use with the Backrub Move. I again experiment with different weights, where I set the `cart_bonded` weight to 1, which I call the cartbonded sample source and to 5, which I call the cartbonded_strong sample source. The Native and Backrub sample source with no angle restraint (i.e. setting `mm_bend` weight to 0 is equivalent to setting the `cart_bonded` weight to 0) are reused.

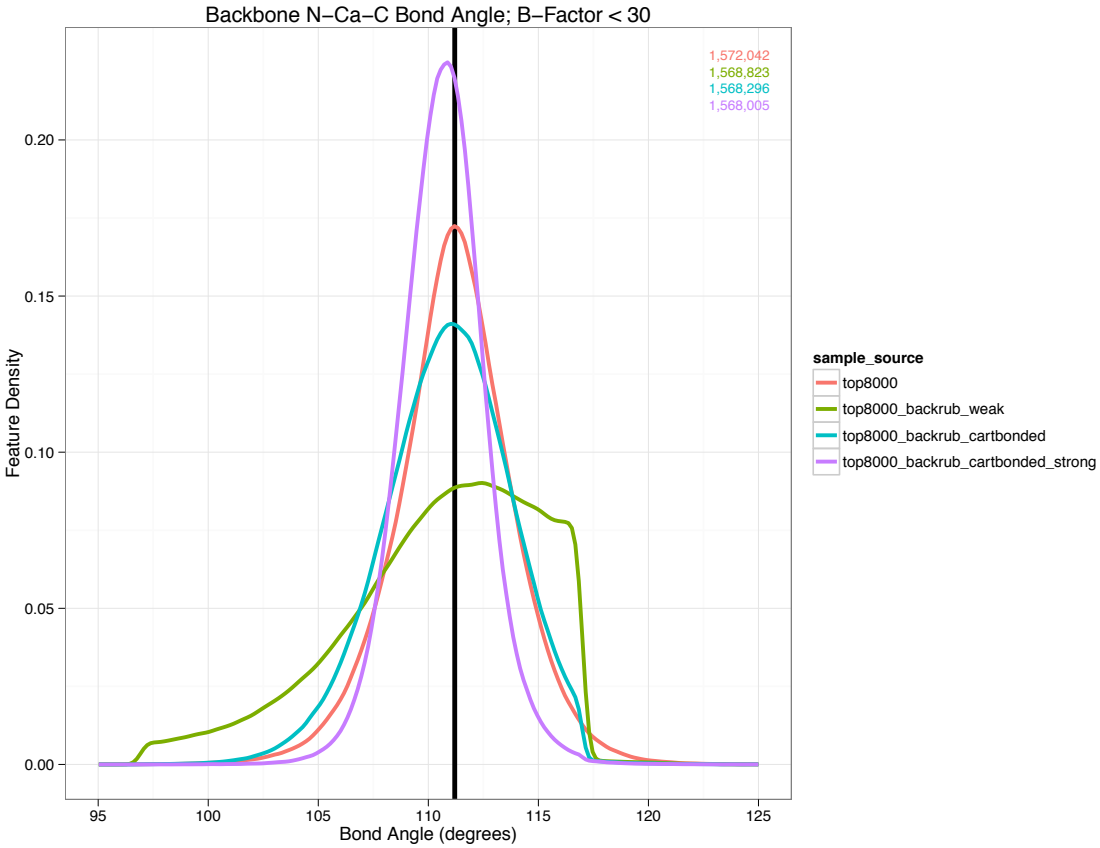


Figure 4.1.14 N-Cα-C Bond Angle by `cartbonded` Strength: Kernel density estimation of N-Cα-C bond angle by sample source plotted by color. (red) Top8000, the Native sample source, (green, blue, purple) top8000_[weak, cartbonded, cartbonded_strong], the Backrub sample source with the `cart_bonded` term set to [0, 1, 5]. Increasing the weight of the `cart_bonded` term does not shift the peak of the N-Cα-C bond angle distribution and while it does make the distribution narrower.

Inspection of Figures 4.1.13 reveals that the `cart_bonded` term does not have the bias toward tighter bond angles that the `mm_bend` has, which appears to be an improvement. Increasing the weight of the `cart_bonded` term decreases the variance of the angle distribution, with variance of the Natives lying between the variance for cartbonded and cartbonded_strong. In Figure 4.1.15 below, the variances in cartbonded_strong agree closely with Natives for α-helix and 3/10-helix residues, and the variances in cartbonded are closer to Natives for the remaining DSSP types.

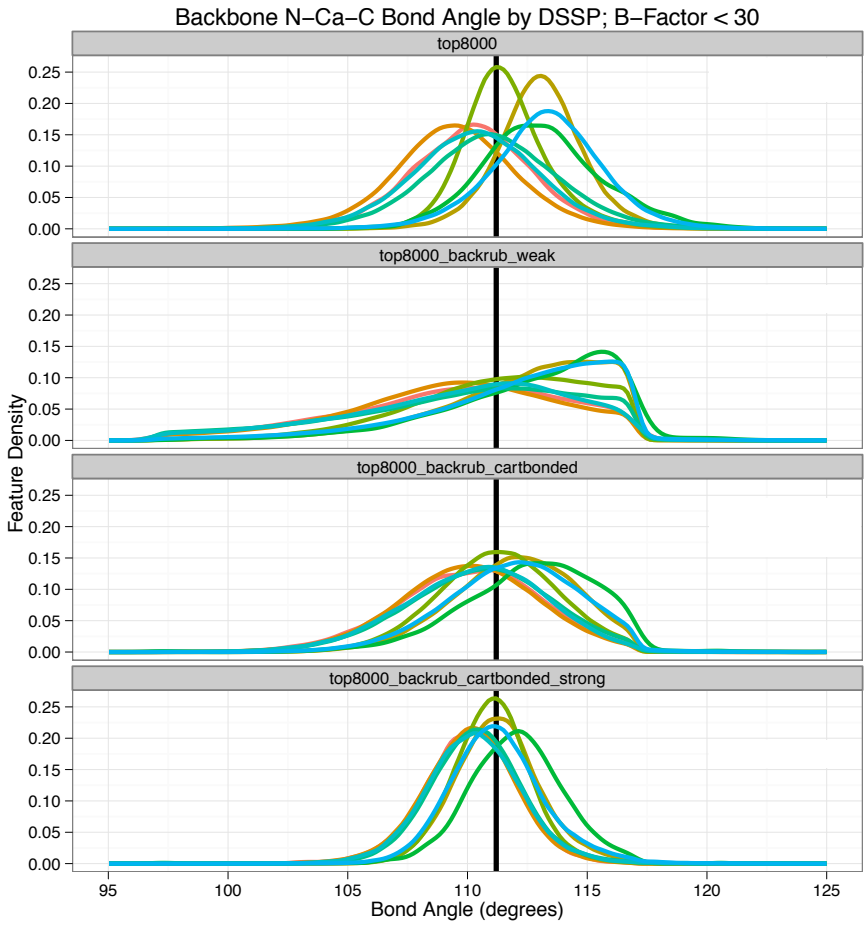


Figure 4.1.15 N-Cα-C Bond Angle by `cartbonded` Strength and Secondary Structure: Kernel density estimation of N-Cα-C bond angle by sample source for Native and Backrub sample sources with varying the weight of the `cart_bonded` term by DSSP type plotted by color.

The weight of the `cart_bonded` term also appears to affect the observed variation conditional on DSSP type. Increasing the weight decreases the variance of the means for different DSSP types (Figure 4.1.13-15). For cartbonded and cartbonded_strong, the variance of the means is substantially less than that observed in Natives. This makes sense because the bond angle is determined not only by the bond angle restraint but by a combination of the other terms in the energy function and sampling protocol. Thus, lowering the weight of the bond angle restraint relative to the other terms allows the other terms have greater influence on determining the bond angles.

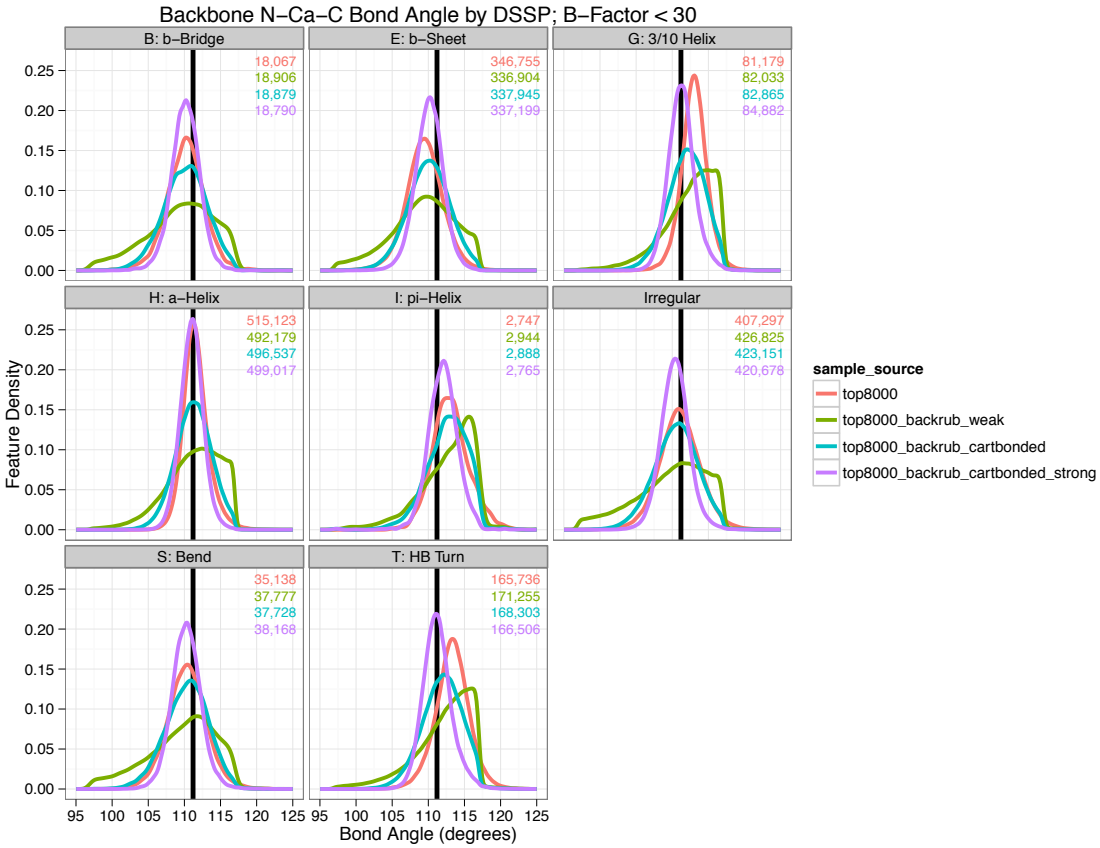


Figure 4.1.16 N-Cα-C Bond Angle by Secondary Structure and `cartbonded` Strength: Kernel density estimation of N-Cα-C bond angle by DSSP by sample source plotted by color; (red) Top8000, the Native sample source, (green, blue, purple) top8000_[weak, cartbonded, cartbonded_strong].

Through Sections 4.1.5-8 the features analysis investigation has revealed discrepancies between Rosetta simulations with the Backrub move and a bond angle restraint. I will now summarize these finding and show in the next section how these can lead to new hypotheses, thus completing the iterative cycle of hypothesis-investigation that the features analysis tool facilitates.

- Figures 4.1.11 and 4.1.14 show that the `cart_bonded` mean N-Cα-C bond angle agrees more closely with the native mean N-Cα-C bond angle than the `mm_bend` restraint.
- Figures 4.1.11 and 4.14 show the deviation about the mean varies with the strength of the restraint and to recapitulate the native deviation, the strength should be increased above the default.

- Figures 4.1.12 and 4.1.15 show that the native variation by secondary structure is not recapitulated by the Backrub sample sources. Figure 4.1.13 and 4.1.16 show that reducing the strength of the bond angle restraint increases variation by secondary structure.

4.1.9 Challenges of Resolving Distributional Discrepancies

The above analysis indicates that the strength for `cart_bonded` is too low to recapitulate the variance of the angle distribution and too high to recapitulate the variance of the means by secondary structure. To demonstrate the challenge of resolving these conflicting discrepancies, I consider two approaches in modifying the computational model. For further background on the energy functions in structural biology see Chapter 6 especially Section 6.1 and 6.5.

The first approach is to add DSSP-specific parameters; however, this problematic both from a computational perspective and a biophysical perspective. First, the DSSP assignment is determined by non-local patterns of hydrogen bonding. Any feature model that evaluates DSSP makes sampling protocols that rely on incrementally updating the total energy more computationally expensive. Second, the DSSP are discrete, so evaluation of the DSSP would cause discontinuities in the angle restraint functional form as the conformation crosses a definitional boundary. As I discuss in the next case study (Section 4.2), derivative discontinuities cause problems with gradient-based minimization. Third, from a biophysical perspective, making the bond angle restraint term dependent on the secondary structure type seems inappropriate because the forces that govern the bond angle at a highly localized interaction and should not depend on more global structural phenomena.

The second approach to resolving the N-Cα-C bond angle distribution discrepancy is to adjust the temperature parameter of the MCMC simulation.

From the theory of statistical mechanics, in nature at low temperatures, the distribution of a molecular conformation over conformation space is concentrated in low energy states. If the total energy depends on the value of a feature, then the variance of the feature will decrease with decreasing temperature.

To assess the effect of temperature in nature on the distribution of N-Cα-C bond angles, one could compare the distributions of N-Cα-C bond angles for sample sources of conformations experimentally characterized at high and low temperature. X-ray crystallography experiments are typically performed at cryogenic temperatures (i.e., much colder than room temperature) to limit radiation damage (Fraser 2011), while nuclear magnetic resonance (NMR) experiments are typically conducted at room temperature (Kay 1998). So, features from an X-ray crystallography sample source and an NMR sample source could be compared to assess the affect of temperature.

To model temperature in MCMC simulations, the temperature parameter controls the rate of accepting candidate moves that make the energy higher. Consequently, the variance of an energetically restrained feature—such as the N-Cα-C bond angle—should decrease with decreasing temperature. So decreasing temperature would make the distribution have a lower variance without increasing the weight of the bond angle restraint. Having the weight on the N-Cα-C bond angle restraint model to be low would allow other terms in the energy function that may be responsible for the observed dependence on secondary structure to emerge. Therefore I hypothesize that lowering the temperature and lowering the bond angle restraint would allow the distribution of the N-Cα-C bond angle feature distribution from the backrub sample source to recapitulate the native N-Cα-C bond angle distribution.

Since a lower simulation temperature means on average fewer candidate moves are accepted, to effectively sample conformation space at a lower temperature, one should consider lengthening the number of simulation steps or using an annealing algorithm that ramps the temperature down through the course of the simulation to the final temperature. Further, adjusting the temperature and sampling protocol will affect other feature distributions. Therefore, consideration of the N-Cα-C bond angle distribution should be just one piece of information that leads to adjusting MCMC temperature.

4.1.10 Conclusion

This case study demonstrated how to use the features analysis tool to do exploratory data analysis. It describes a native source and computational model sample sources exploring the consequences of using different energy functions on the feature distributions. It demonstrates how to extract features from a sample source and how to create a features analysis script. It demonstrates the use of kernel density estimation and the grammar of graphics to explore and compare variation in the data by different covariates.

In describing the Backrub Move, Davis et al. (2006) say

> A small amount of distortion is introduced into the τ angles (N-Cα-C), but they generally remain well within the range of values seen in typical crystal structures.

This case study shows the features analysis tool makes it is easy to precisely measure these distortions, that the choice of bond angle restraint significantly impacts the recapitulation of native bond geometry, and that the recommended choice of restraint is systematically biased.

This case study examined the consequences of the backrub sampling heuristic on the distribution of a single feature—the N-Cα-C bond angle. I observed that the published and recommended

bond angle restraint, `mm_bend`, causes Rosetta simulations using backrub to systematically predict bond angles that are too tight, but that this can be resolved by the use of the `cart_bonded` term. However, neither the `mm_bend` nor the `cart_bonded` terms seem to recapitulate the variation of distributions by secondary structure (as measured by DSSP). This observation motivates a future follow up study considering the role of temperature on feature distributions variation.

4.2 Energy Function Smoothness

This case study demonstrates how to use the features analysis tool to check that computational models adhere to the modeling decisions made by the developer. It demonstrates the flexibility of the features analysis tool to assess focused hypotheses about the model.

A central method for generating conformation predictions is to minimize the energy function over conformation space through iterative search. If the energy function has continuous first derivative, then gradient-based minimization algorithms, which take steps "downhill," can be used. When the local gradient accurately represents the larger topographical features, e.g., by being smooth, then following the local gradient will lead to the closest local minima. When combined with randomization, gradient-based minimization can rapidly identify diverse local minima, and facilitating minimizing the energy function. If, the energy function has a discontinuous gradient or is very rough, robust alternatives such as subgradient optimization can be used, but these are in general less efficient (Shor 1985, Yu 2010, Goffin 2012) than popular methods like BFGS that require continuous gradients (Nocedal 2006).

Ensuring at the coding level that the energy function is continuous can be difficult. Derivative discontinuities can be difficult to spot in code review because they can occur at the boundaries of cases that are handled in separate blocks of code. It is possible to construct unit tests to assert that the derivative is continuous. The empirical gradient at a point is computed by constructing a hyperplane from points nearby. The analytic gradient is compared with the empirical gradient and if the difference exceeds a threshold, the test fails. However, the high dimensionality of the energy function makes it challenging to effectively test at enough points to guard against discontinuities.

This case study tells 3 vignettes for how the features analysis tool can be used to detect and resolve derivative discontinuities in the Rosetta energy function. The fact that these had not been identified prior to this dissertation—some for many years—indicates the pressing need for better tools to effectively check model consistency.

4.2.1 Backbone Terms on ϕ, ψ Grids

Rosetta defines three energy function terms that evaluate the protein backbone ϕ, ψ torsional angle features. Each term has a functional form with additional parameters tuned to probability distributions observed in Native structures. The Ramachandran term, which computes an energy over ϕ, ψ for each residue conditional on the identity of the sidechain, *aa*, has the functional form,

$$E_{rama}(\phi, \psi) = -\ln p(\phi, \psi | aa).$$

The *p_aa_pp* term (sometimes called the design term), which computes an energy of the identity of the sidechain conditional on ϕ, ψ, has the functional form

$$E_{paapp}(aa|\phi, \psi) = -\ln \frac{p(aa|\phi, \psi)}{p(aa)}.$$

The rotamer term (almost always called the Dunbrack term, or *fa_dun*), which computes an energy over sidechain torsion angles, χ_i, conditional on ϕ, ψ and the identity of the sidechain, has a more complex functional form. Rotamers are binned by χ angles, and the functional form includes a term for the probability of being in a particular rotamer bin, *rot*, conditional on ϕ, ψ and the identity of the sidechain, and a term that measures the deviation from the ideal rotamer geometry:

$$E_{dun}(\chi|\phi, \psi, aa) = -\ln p(rot|\phi, \psi, aa) + \sum_i \left(\frac{\chi_i - \overline{\chi_{i,aa,rot}}(\phi, \psi)}{\sigma_{i,aa,rot}(\phi, \psi)} \right)^2.$$

In the Rosetta *Score12* energy function, each of these terms is defined by bilinear interpolation on data stored on a 10° grid in ϕ, ψ space. For the p_aa_pp and Ramachandran terms, the grid origin is (5,5), and for the $p(rot|\phi, \psi, aa)$, $\overline{X_{i,aa,rot}}(\phi, \psi)$, and $\sigma_{i,aa,rot}(\phi, \psi)$ functions in the rotamer term, the grid origin is (0,0), to agree with the 2002 Dunbrack library. Bilinear interpolation makes the functions continuous, but the derivatives are discontinuous at grid lines, and the origin shift causes derivative discontinuities every 5° in ϕ, ψ space.

Plots of the ϕ, ψ distribution for non-helical residues (Figure 4.2.1B) clearly show that the minimizer accumulates density at these discontinuities. Indeed, *Score12* predictions place 23% of all ϕ, ψ angle pairs within .05° of a grid boundary. It seems unlikely that nature would have the same preference.

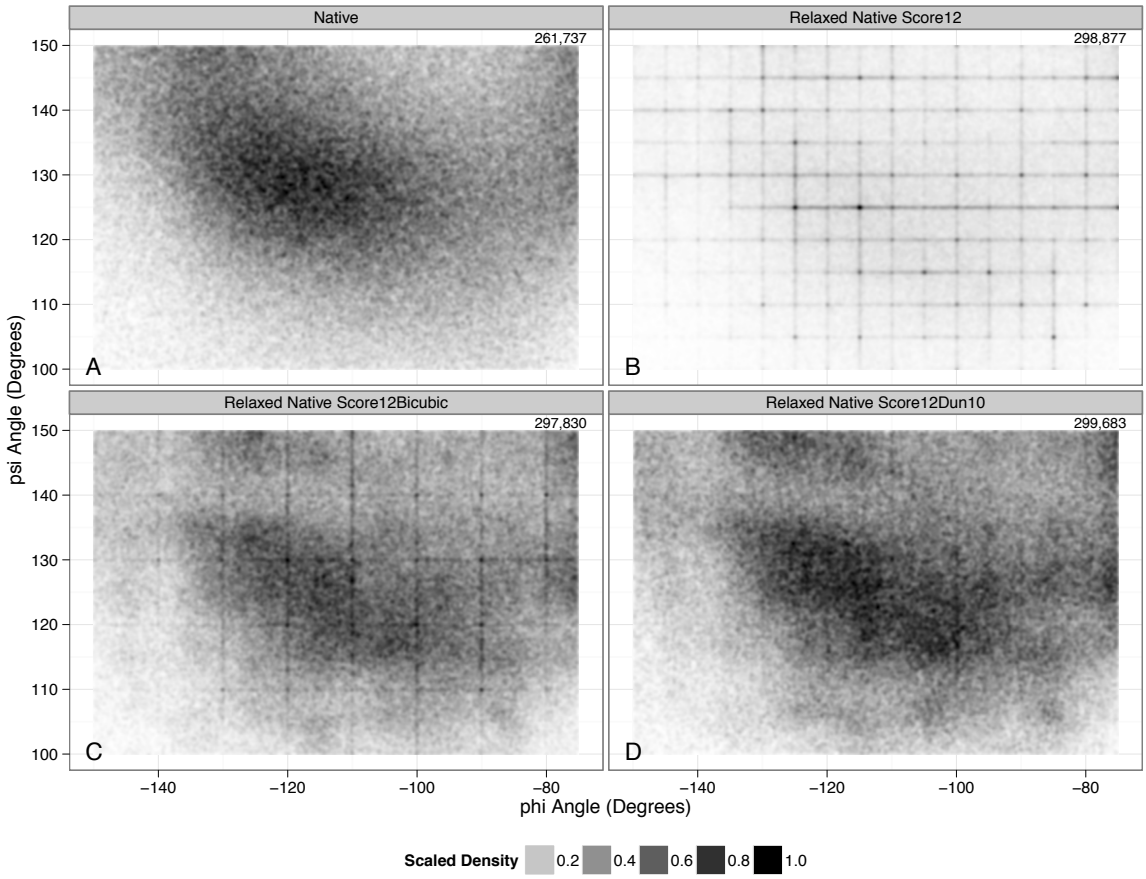


Figure 4.2.1 Bicubic Smoothing of Backbone Potentials: These plots show density in the β-region of ϕ, ψ-space for four sample sources (A) Native, (B) predicted with *Score12*, (C) with *Score12Bicubic*, and (D) the *Score12Dun10* Rosetta energy functions. Derivative discontinuities in (B) cause substantial accumulation every 5°. In (C), bicubic interpolation for Ramachandran and p_aa_pp terms improves the density distribution, leaving some accumulation at the rotamer term's 10° bin boundaries. In (D), moving to the 2010 Dunbrack rotamer library all but eliminates the binning artifacts. Density is calculated by scaled kernel density estimation with a very tight kernel. Conformations are from the Top8000 dataset, filtered for B-Factors < 30 Å^2, see Section 4.1. The number in the upper right corner in each facet cell counts the feature instances.

In collaboration with Andrew Leaver-Fay, I replaced the bilinear interpolant with a bicubic spline that guarantees continuous derivatives at grid boundaries. I fit bicubic splines with periodic boundary conditions for both the Ramachandran and p_aa_pp energy terms, interpolating in energy space. Bicubic splines require storing not only the value at each grid point but also 3 partial derivatives. To conserve memory for the Dunbrack energy, I fit bicubic splines only for the $-\ln\big(p(rot|\phi,\psi,aa)\big)$ portion, and continue to use bilinear interpolation for the mean and standard deviation of χ.

To test that these changes correct for the artificial density accumulation, I generate new relaxed natives with the bicubic interpolation using the FastRelax protocol detailed in Section 2.3. In Figure 4.2.1C, we no longer see the accumulation produced by the Ramachandran and p_aa_pp terms at the 5° boundaries. Accumulation on the 10° grid boundaries persists because bicubic splines were not used to interpolate the mean and standard deviation of χ. In Figure 4.2.1D, the 2010 Dunbrack rotamer library further reduces the accumulation at the remaining derivative discontinuities, probably because the underlying model was fit with a smoother kernel bandwidth to the experimental data.

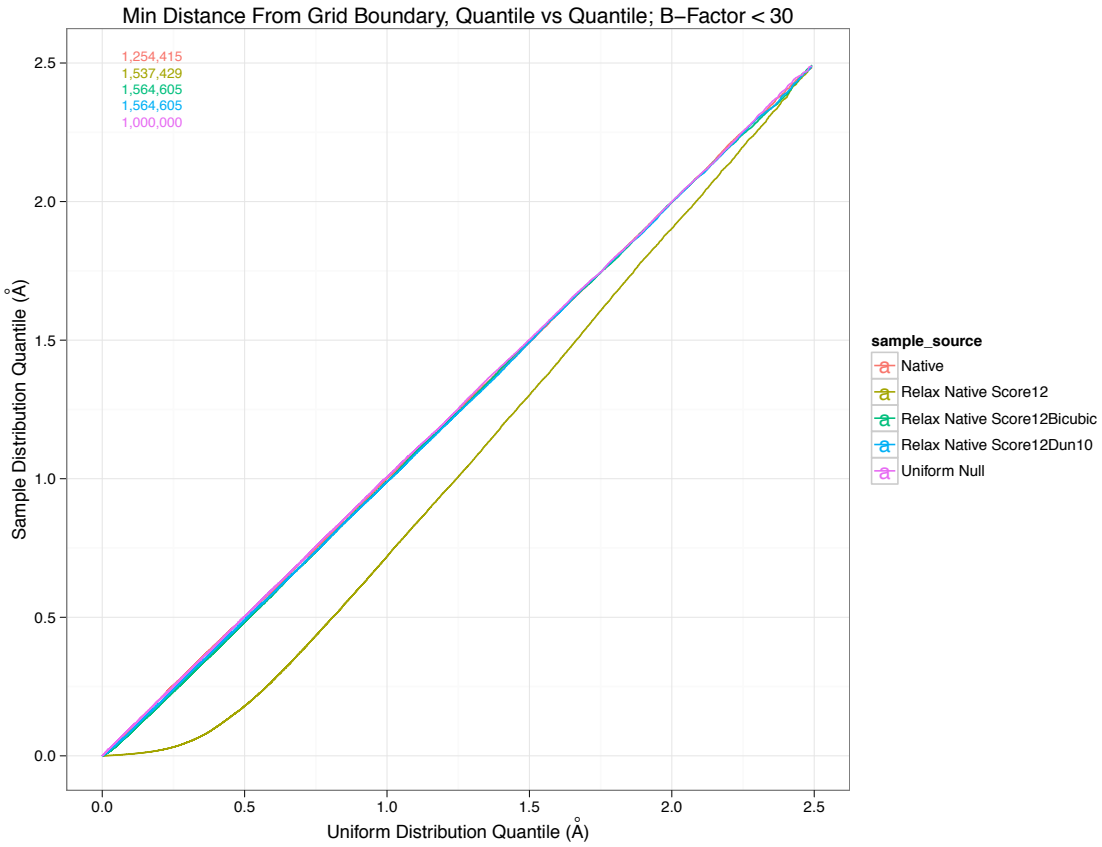


Figure 4.2.2 Min Distance From Grid Boundary Q-Q Plot: Quantile-Quantile plot measuring the distance to the closest 5° grid line in either the ϕ or ψ direction. The purple line represents points sampled uniformly at random from the (ϕ,ψ) plane. The ochre curve represents Score12 and is below the diagonal near 0, indicating an accumulation near the grid boundary.

The visually striking figure 4.2.1, easily generated through features analysis, gives strong evidence that the accumulation along the grid lines is unnatural. The level of accumulation can be quantified by defining a feature that measures the distance the nearest grid line and plotting the empirical cumulative distribution function against the theoretical quantiles for the null model that assumes points are uniformly distributed on the ϕ, ψ feature. The resulting QQ plot quantifies the accumulation of density at the grid lines. This allows objective assessment that smoothing the p_aa_pp, Ramachandran, and $-\ln p(rot|\phi, \phi, aa)$ portion of the rotamer term in Score12 is necessary, but smoothing the 3D distribution for the Dunbrack library is not a pressing concern at least in the aggregate.

I considered other methods to achieve smooth backbone models, such as B-splines, which do not require additional memory for storing pre-computed derivatives. However, the evaluation of B-splines is more computationally expensive and potentially not worth the cost. A future direction could be to evaluate backbone dependent models that have functional forms that are Fourier series, wavelets, or mixtures of bivariate von Mises distributions (Boomsma 2010). Although these models also have higher evaluation cost than bicubic splines, they can be represented more compactly by values on a grid. Therefore, they may be particularly appealing for use on computational architectures where memory constraints are even more stringent, such as GPU chips.

4.2.2 Fade-Functions in the HBv1 H-bond Model

This vignette demonstrates the use of the features analysis tool to investigate the modeling decisions that lead to creating derivative discontinuities as fade function in the Rosetta H-bond Model (HBv1). Rather than simply smoothing them as in 4.2.1, I simplify the functional form to not require the fade functions and therefore resolve the derivative discontinuities.

The default energy term for hydrogen bonds, HBv1, derived from the work of Kortemme et al. (2003) has discontinuities in fade functions. Before I can define fade function I give more details on the HBv1 H-bond model, which will be used as a baseline for further modifications to the H-bond model in the case studies in Sections 6.3-6.

The HBv1 function depends on the positions of four atoms (Figure 4.2.3)—the donor heavy atom D, the hydrogen H, the acceptor A, and the adjacent heavy atom called the base B—and the acceptor chemical type. Three features, the distance *AH* and angles *BAH* and *AHD*, serve as the basis for five functions determined from observed values of these three features in native structures. One function depends on AHdist alone. Because Kortemme et al. (2003) observed that H-bond orientation dependence is more pronounced at short interaction distances, they implemented short- and long-range functions of each angle feature, using linear interpolation in the range of AHdist where both short- and long-range functions are defined.

In this model, an H-bond is identified when the sum of the five energy functions is negative. But because each function is of a single variable, and not all go sufficiently positive to overcome the others, unwanted interactions may have negative energy, such as a donor and acceptor pair with good angles at an arbitrary distance. Thus, before summing, the functions are also multiplied by "fade functions" that linearly go to zero as other parameters leave their valid ranges.

Figure 4.2.3 shows the functions and fade functions for one chemical type in Score12. Each of the three columns represents a feature and each of the five rows represents an H-bond energy function. Multiplying along each row, and summing the resulting column gives the H-bond energy term. In this example, fade function knots cause derivative discontinuities in the AHdist feature at discontinuities 1.4 Å, 1.5 Å, 1.9 Å and 2.3 Å; other chemical types have fade functions that cause derivative discontinuities at 2.1 Å.

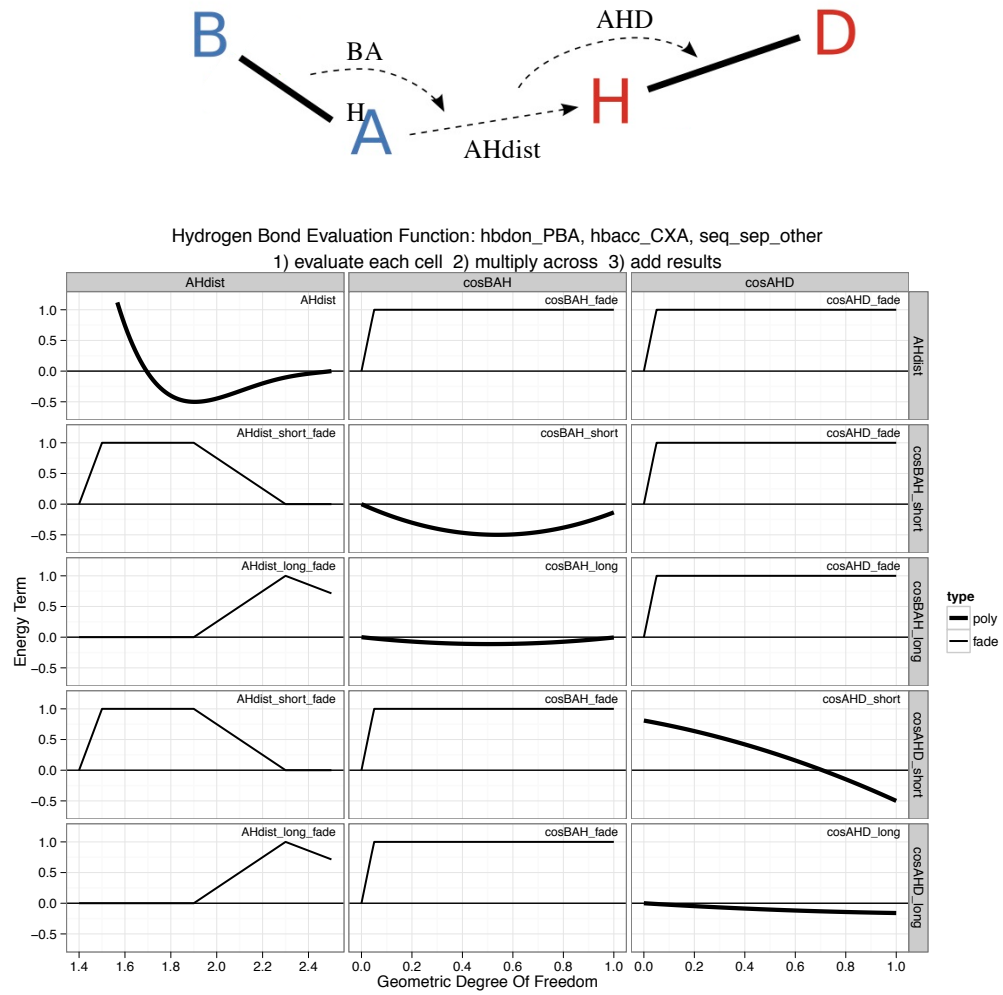


Figure 4.2.3 Score12 H-Bond Model Evaluation: The Score12 H-bond model evaluates the following 3 geometric features defined on the Base (B), and Acceptor (A) atoms of the acceptor group (whose definition depends on the hybridization type of the acceptor), and the hydrogen (H) and donor (D) atoms of the donor group: The A-H distance (AHDist), the B-A-H angle (BAH), and the A-H-D angle (AHD) (top). Score12 H-bond functional form (bottom): Evaluation of the Score12 H-Bond energy between a protein backbone donor (hbdon_PBA) and an asparagine or glutamine acceptor (hbacc_CXA) where the separation in primary sequence is greater than 4 (seq_sep_other). For a given arrangement of donor and acceptor group atoms, each column represents a geometric feature that each row contributes additively to the energy. Cells with bold lines are the primary functions and the remaining cells are piecewise-linear fade functions. To compute the H-bond energy for a given arrangement of atoms, evaluate the geometric features to find the x-value in each column of cells. Evaluate the function in each cell by measuring the y-value of the line. Then, multiply the function values across each row and sum the results.

In Figure 4.2.4, a histogram for the Native sample source and Score12 relaxed native sample source of the length of H-bonds (AHdist) with a bin width of 1/600 Å clearly shows density accumulation at fade function knots. Accumulation at 2.3 Å is minor because there is less density in the neighborhood of the discontinuity.

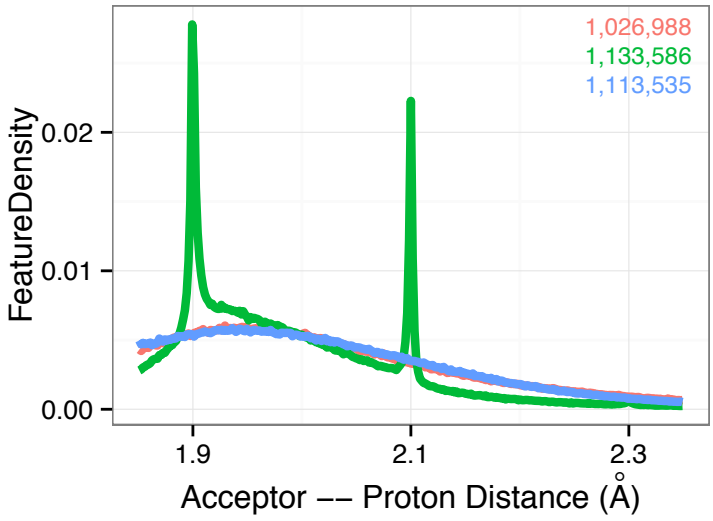


Figure 4.2.4 Score12 H-Bond AHdist Fade Derivative Function Discontinuity Spikes: Histogram for AHdist H-bond feature restricted to regions where the fade functions have derivative discontinuities for the following sample sources: Native (red), Score12 relaxed natives (green), and Talaris2013 relaxed natives (blue), which are described in Section 7.2. The histogram is computed with a bin width of 1/600 Å and is estimated for the H-bond AHdist feature with volume normalization (Section 5.3). Note the Native curve is partially obscured by Talaris2013. The numbers in the upper right are the feature instance counts. The labeled points on the x-axis, (1.9, 2.1, 2.3) are points of derivative discontinuity in the Score12 H-Bond function due to knots in the piecewise linear functional form of the fade functions.

To resolve derivative discontinuities, rather than simply using splines to smooth the piecewise linear functional of the fade functions, I examine whether fade functions are needed at all. The motivation for the Kortemme model inclusion of short- and long-range angle potentials was the observed dependence of the angles on distances in Native distributions. I therefore test whether the strength of the orientation dependence correlates with the length of bond in nature, and whether encoding this dependence is necessary to recapitulate the behavior in Rosetta predictions. To do this, I remove dependence of angle features on AHdist (in the Talaris2013 H-bond energy term), and use a features analysis to plot empirical cumulative distribution functions conditional on sliding windows of the AHdist feature. I use a generic definition of an H-bond as any polar contact with an AHdist less than 4 Å (Figure 4.2.5) to facilitate comparing feature distributions across sample sources.

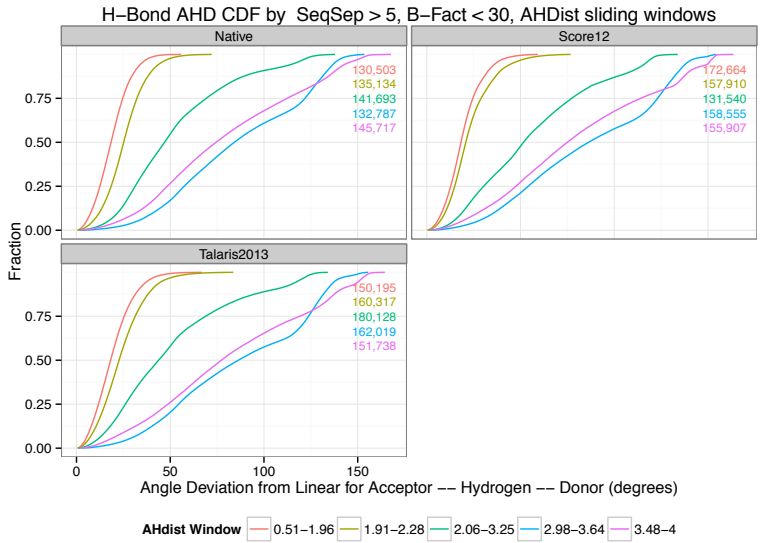


Figure 4.2.5 H-Bond AHD CDF by Sample source and AHdist Sliding Windows: Empirical cumulative distribution functions for the H-bond AHD angle feature by sample source conditional on sliding window values of the AHdist feature plotted as the color. For AHD angles from the Native sample source, short H-bonds (AHdist < 2.28 Å) are significantly smaller than for long H-bonds (AHdist > 2.98 Å), confirming the observations in Kortemme 2003. However, the AHD angle distributions from the Score12 sample source and the Talaris2013 sample source recapitulate the dependence of AHD angles on AHdist.

Figure 4.2.5 shows that the AHD angle features from the Score12 and Talaris2013 sample sources recapitulate the dependence on AHdist without accounting for the interaction between AHD and AHdist in the H-bond term. This indicates that the effects are caused by other terms of the energy function or sampling protocol. A possible explanation is that for short H-bonds, a large AHD angle causes the donor and acceptor heavy atoms to come in close enough contact that they experience strong repulsion from the Lennard-Jones model.

I next consider if fade functions are needed to disable interactions. An H-bond is defined when the total energy is negative and the minimum value of each term is -.5. By extending each term to achieve a value of 1.0 at unfavorable geometries, each term can overcome the contribution from the other terms and disable the interaction. Based on these observations, I simplify the H-bond functional form by removing the angular dependence on AHdist.

4.2.3 AHD Pole in the HBv1 H-Bond Model

The third vignette shows that derivative discontinuities can arise even in the "gold standard" experimental data. Artificial accumulation is observed in Native and Score12 sample sources when H-bonding atoms are spatially collinear. In nature, the H-atom in an H-bond predominantly lies on the line connecting the acceptor and donor atoms. The Rosetta H-bond functional form, as well as many other orientation dependent functional forms, includes a component that penalizes the AHD angle away from exterior angle of 0. In Score12, at the pole AHD=0, the derivative of the AHD term is non-zero, causing a derivative discontinuity. Projecting the (AHD, AHχ) feature onto polar coordinates using the Lambert-Azimuthal projection (Section 6.3), the accumulation at the pole is clearly visible (Figure 4.2.6). Interestingly, in X-ray crystal structures with H-atoms placed with Reduce, the artificial accumulation at the pole is also visible, indicating bias in the processing of X-ray crystal structures.

I impose a Lagrangian constraint that the derivative at the pole must be zero, while fitting the polynomial function for the AHD term. This removes the derivative discontinuity and removes the spike in density at the pole (Figure 4.2.6 D).

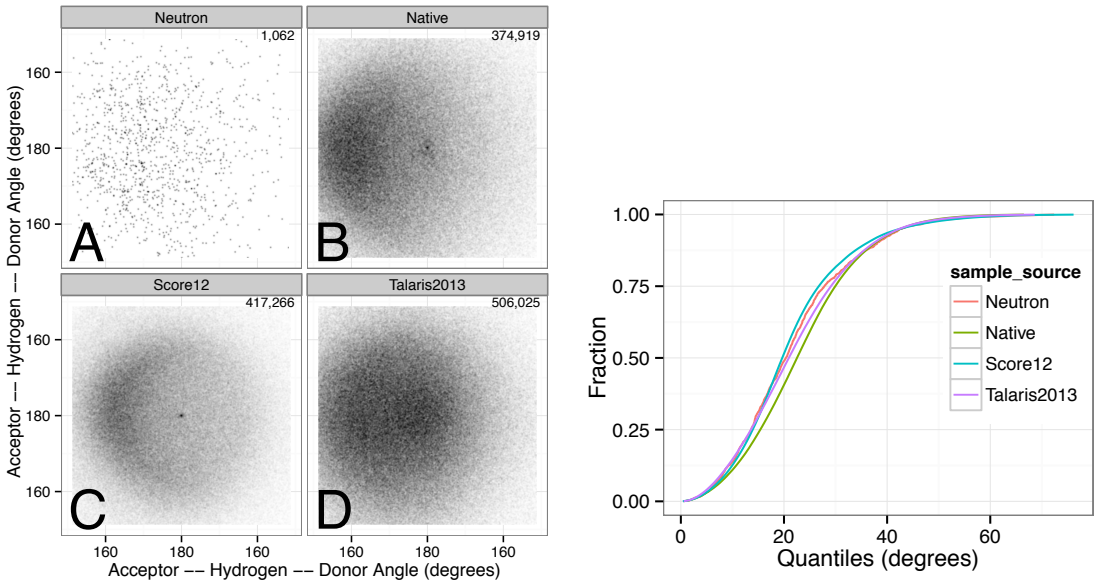


Figure 4.2.6 Score12 H-Bond AHD Pole Accumulation: Sample sources include Neutron: 70 structures determined neutron diffraction experiments selected from the PDB for good MolProbity Scores (A), Native: described in Section 4.1.2 (B), Score12: Natives relaxed (Section 2.3) with the Score12 energy function (Section 7.1) (C), Talaris2013: Natives relaxed with the Talaris2013 energy function (Section 7.1) (D). (left) H-bond (AHD,AHχ) feature density distribution is plotted over the Lambert-Azimuthal projection using kernel density estimation and a tight kernel. (right) H-bond AHD empirical cumulative distribution function. The dot at the center shows density accumulation at the AHD=0° pole in Native and Score12 sample sources. The appearance of the accumulation at the pole in the Native dataset indicates there may be artifacts introduced during the refinement of the X-ray crystal structures. High-resolution neutron diffraction studies are able to resolve H-atom coordinates, which may be less suseptable to these artifacts, though the availability of these structures is too limited to fully resolve the presense or absense of the artifact. The absense of the pole in the Talaris2013 is due to the constraint that the AHD term have a continuous derivative at AHD=0°.

4.2.4 Conclusion

This case study demonstrates how the features analysis tool can be used to assess the consequences of modeling decisions. In the theme tying the three vignettes together the decision to use gradient-based minimization, which requires the energy function to be free of derivative discontinuities. I detect several live instances of problematic derivative discontinuities and resolve them by smoothing the offending terms, evaluating and simplifying the functional form and constraining parameter optimization, thus demonstrating the flexibility of the features analysis tool.

4.3 Creation of the `dslf_fa13` Disulfide Model

As a self-contained example of using features analysis to specify the parameters of a feature model, in collaboration with Frank Dimaio in the Baker Lab, I develop a new model for disulfide interaction, which occurs when the sulfur atoms of two cysteine residues form a covalent bond.

This case study shows how features analysis can be used to create a term in an energy function, clarifying the motivation for the features analysis tool. Furthermore, the term created, dslf_fa13, is one of the modifications that is incorporated into Rosetta's new default energy function, Talaris2013. Thus, this case study also demonstrates that features analysis can advance the state of the art in protein structure prediction.

Dimaio (2013) took Rosetta-refined crystal structures and evaluated, for each atom, the gradient of each energy term. By the maximum likelihood principle, the native structure for a given sequence is assumed to be the global minimum of nature's energy function. The net gradient at any minimum is zero, so Dimaio suggested that large gradients on individual Rosetta individual terms might indicate discrepancies between Rosetta and nature's energy functions. The Score12 disulfide-bond energy term was one term with large gradients.

To investigate whether this is a problem, in collaboration with Frank Dimaio, I create a *native* sample source by taking cysteine pairs in the Top8000 chains, yielding 1920 disulfide bonds, identified as SG-SG pairs within 2.3 Å. For a first decoy sample source, called *old*, I apply Rosetta FastRelax to 50 small disulfide-containing proteins, yielding 191 disulfide bonds. I compare distributions of disulfide-bond features (SG-SG distance, Cβ-SG-SG angle, Cα-Cβ-SG-SG dihedral, and Cβ-SG-SG-Cβ dihedral), and observe a lack of fit in both SG-SG distance and Cβ-SG-SG angle features (Figure 4.3.1).

To improve the disulfide energy term, here is a new model for all four features: For the SG-SG distance feature, a skewed Gaussian is used to capture the asymmetry. For the Cβ-SG-SG angle, the Cα-Cβ-SG-SG dihedral, and the Cβ-SG-SG-Cβ dihedral, mixtures of 1, 2, and 3 von Mises functions, respectively, are used. Initially, the parameters are fit to the native distribution, and then a constant offset is added to help balance against the other terms in the energy function. I select the offset value by refining native and non-native conformations for the 50 small disulfide-containing proteins, and choose the smallest offset that favors the disulfide-linked conformations, yielding a value of -2 energy units. Finally, the parameters are adjusted to recapitulate the native distribution with conformations refined using this new disulfide model (*new*; Figure 4.3.2).

The new disulfide model is incorporated into the Talaris2013 energy function, which is Rosetta's new default that is detailed in Chapter 7.

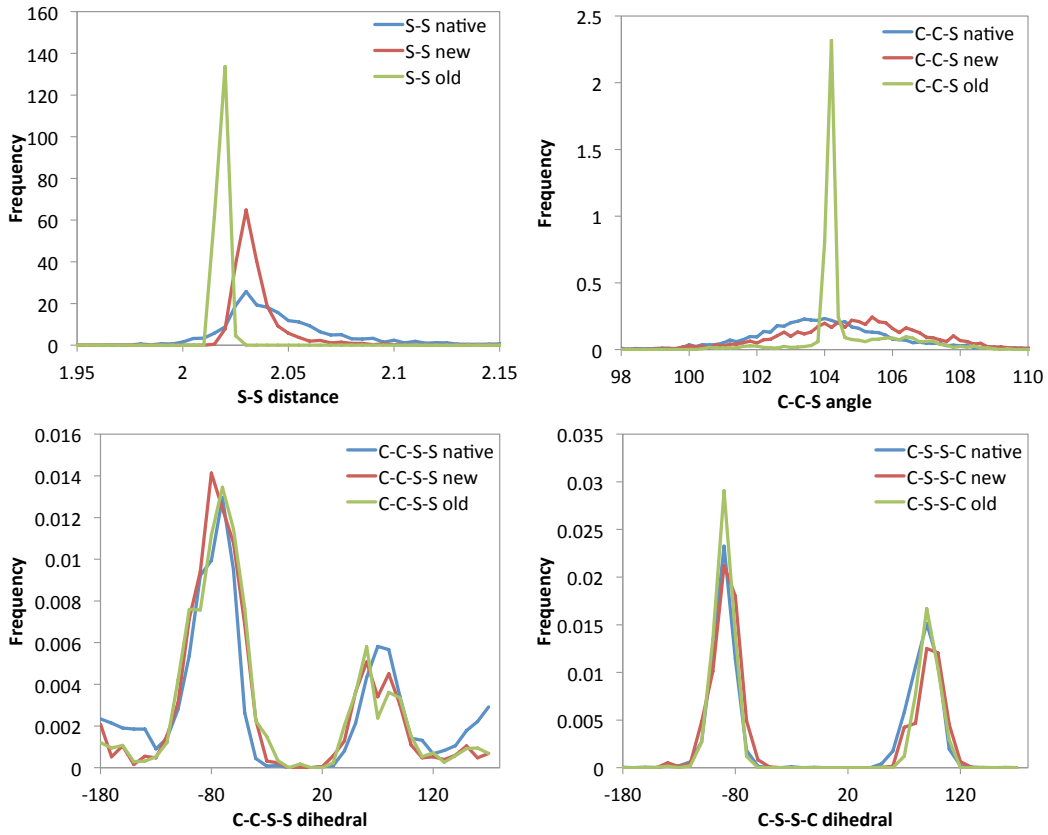


Figure 4.3.1 `dslf_fa13` Disulfide Model Features by Sample Source: Disulfide-bond feature distributions comparing the Top8000 chains sample source (native) against 50 small disulfide-containing conformations refined with Rosetta using the Score12 disulfide model (old) and the updated disulfide model (new). Each cell plots the estimated distributions of a disulfide bond feature: SG-SG distance, Cβ-SG-SG angle, Cα-Cβ-SG-SG dihedral angle, or Cβ-SG-SG-Cβ dihedral angle.

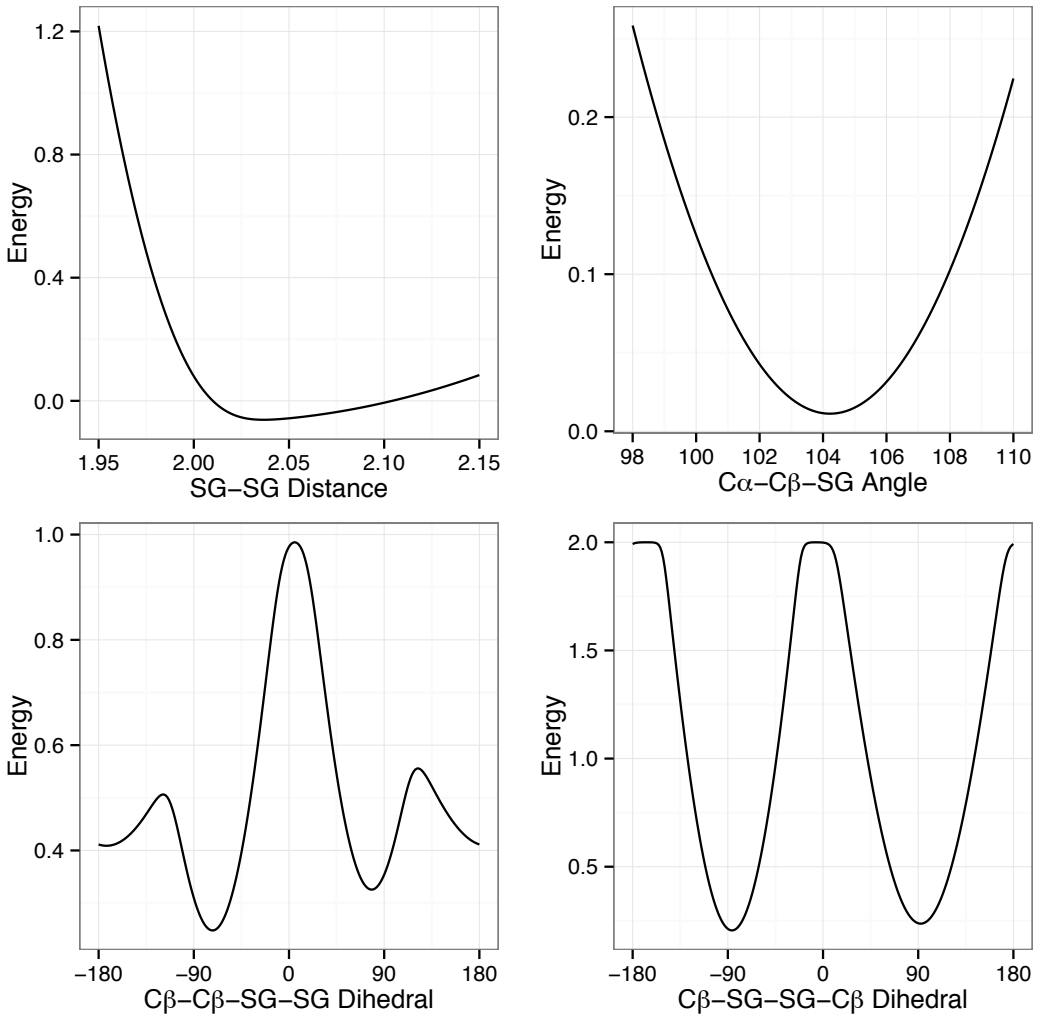


Figure 4.3.2 `dslf_fa13` Model Component Terms

5 Computational and Statistical Methods

The design of the features analysis tool builds on existing technology to achieve a high level of functionality and usability. This could mean, however, that accessibility is limited to those with some experience with software development: Rosetta, relational databases, and R are extremely powerful software tools, but the learning curve for each can be steep. To increase usability for analysts I have developed the components to be self-contained and to facilitate reuse. This allows less experienced users to still use the Features analysis tool productively. I have also invested considerable effort in working with the Rosetta community to encourage their adoption into routine research.

In order for the features analysis tool to aid researchers in gaining a deeper understanding of their computational models, the tool must be easy to use and highly functional. Here are desiderata of the tool for

- Analysts:
    - Easy for non-experts to get meaningful results
    - Facilitates interactive exploratory data analysis
    - Computationally efficient Expressive language for defining features supports repurposing features analyses
    - Extensible and flexible for complex or custom analyses
- Developers:
    - Easily extendable to define new features and new features analysis scripts
    - Compact and robust management of feature data
    - Modular components to be useful beyond the features analysis tool
    - Clear specification of feature database schema
    - Wide and robust support for cluster and database platforms

To achieve these objectives, in the four sections of this Chapter, I detail four technologies and statistical methods that underlie components of features analysis tool, namely:

5.1: SQL database schema, and the design decisions that lead to it,

5.2: kernel density estimation, focusing on bandwidth estimation,

5.3: normalization and boundary conditions for reparameterizing composite feature values,

5.4: statistical hypothesis testing and exploratory data analysis.

5.1 Features Database

5.1.1 Features Database Architecture

Since features are random variables they cannot be stored directly in the features database, instead samples from features and their relationships from sampled conformations are stored. In the features analysis scripts, a database query constructs samples for new features by conditioning and filtering using the relationships stored in the database. The flexibility of the features database allows developers to easily create different features analyses by changing the database query.

Each FeaturesReporter module is responsible for defining and populating a set of tables in the features database. Creating a FeaturesReporter requires specifying the schema for a set of tables in the features database and implementing code to populate the database from the molecular conformations. To capture the feature database specification, a developer creates an SQL schema for the portion of the features database and documents the design decisions in the developer Wiki (wiki.rosettacommons.org/index.php/FeaturesScientificBenchmark). Geometric descriptions of the feature, configuration information such as definition thresholds, and literature references

should be included in the Wiki to facilitate analysts in interpreting the results of features analyses. Further, this work of ontology building can be reused if feature data is to be represented in another form, such as for serialization or a NoSQL, document based storage, or the development of other tools to populate the features database.

This modularity of the features database schema allows feature analysis developers to use small number of FeaturesReporters for a specific analysis, thus significantly reduce the storage requirements over compressed, full-conformational data representations. Additionally, since feature databases can be populated once and then used for multiple features analyses, the preprocessing of conformational representations into feature representations saves the computational expense of computing elementary features for each analysis. By putting basic features into a relational database along with other supporting data, researchers can perform the expensive task of extracting features from an input batch of structures just once, while retaining the ability to examine a variety of conditional feature distributions in the future.

Usage of a standard relational database schema allows for integration with other data sources that may be relevant to a scientific prediction task, including biological activity data or data from other high-throughput experimental assays.

To demonstrate the utility of features reporting beyond features analysis, I extend it to allow reconstruction of conformation representations from feature data. Each features reporter can optionally implement a `load_into_pose` method to populate conformational parameters. This makes the feature database into a flexible storage platform for conformations. This has proven useful for users of Rosetta: Many prediction protocols iterate multiple rounds of prediction and analysis. Using a features database as a common representation allows for a seamless workflow, using features analysis to drive the prediction, for example by specifying the

protocol input conformations by an SQL query that selects conformation containing certain feature characteristics.

The design of the features analysis scripting framework encourages keeping each features analysis script conceptually self-contained while building on common support functionality such as estimating densities, saving plots, etc. I have implemented this component building on R because of the array of statistical and data processing packages. The tool, however, could be ported to another analysis environment, such as Matlab or Python if the researcher has more experience with those, with much less effort than recreating everything from scratch.

As a future direction, I would like to add some automatic parameter optimization by features analysis. This would require more efficient, tighter integration with the evaluation of the feature models. The features analyses that I have developed externally serve as prototype and reference implementations for the development of computationally optimized counterparts.

5.1.2 Primary Keys in the Features Database

Relational databases do not represent objects explicitly; rather, they represent objects implicitly through relations defined by tables. Therefore, a central aspect of relational database schema design is clear specification of how the objects are identified by database users. In the features database schema, there are three types of objects batches, structures and features: each batch of features has a 32bit integer `batch_id` identifier, each structure is associated with a batch and has a 64bit integer `struct_id` identifier and each feature instance associated with a structure and is identified by a composite primary key (`struct_id`, `<feature_id>`).

To support features analysis on distributed computational clusters, feature databases support technical partitioning (sharding) to represent a single conceptual batch in multiple database instances where each shard contains the features for a disjoint subset of the structures in the batch. This allows merging of feature databases shards, which is essential for reporting features from distributed prediction protocols. To implement this, the lower 32 bits of the struct_id identifier represent an auto-incrementing integer primary key within a shard, and the upper 32 bits represent the shard index. An alternative approach to implementing sharding that I rejected is to use universally unique identifiers (UUID) for the `struct_id`. The form of a UUID standardized by the international telecommunications union uses 128bits, which is double the size of the struct_id field. Since each feature instance is includes the struct_id in the composite key, this adds 64 bits per row in the database, leading to an unacceptable increase in storage requirements and consequently an unacceptable decrease in usage performance.

5.1.3 The `FeaturesReporter` framework

Feature reporters, which report elementary feature instances within Rosetta to a feature database, implement the `FeaturesReporter` interface, which is a C++ pure virtual base class that minimally specifies two methods: `write_schema_to_db` that defines the database schema of the tables that it is responsible for populating and `report_features` that, given a conformation and a database connection, populates the tables with feature data for that conformation. Additionally functionality that the features reporter framework recognizes in a features reporter includes initialization from a RosettaScripts specification, reconstruction of a Pose from feature instances, and removing data from the database associated with the `FeaturesReporter` for a given `struct_id`. More detail of this interface is described in Appendix A.

To use the features reporter framework, the user initializes the ReportToDB mover with database connection information, initializes and attaches derived `FeaturesReporter` classes to the ReportToDB Mover, and repeatedly calls the ReportToDB apply method with conformations (represented by Rosetta Pose objects). To facilitate the usage of the features reporting framework, I have integrated them into several Rosetta frameworks, including the `JD2` job distributor framework to support reading and writing conformations from/to a relational database for job distribution, and the RosettaScripts protocol language mentioned above.

In collaboration with Tim Jacobs in the Kuhlman lab at UNC Chapel Hill and Sam Deluca in the Meiler lab at Vanderbilt University I have extended the database interface for the FeaturesReporter framework to be a general database interface for the Rosetta platform. In developing the database interface, I set out the following goals:

- to fully support directly interaction with the SQLite3 databases and client interactions with PostgreSQL and MySQL database servers.
- to create layers of abstraction and support methods to facilitate consistent database interaction while not prohibiting usage of backend specific functionality.
- to implement robust testing to ensure correct functionality under relevant use-cases.

To achieve these goals, I build on the `cppdb` library (Bellis 2010) that exposes a common interface to many database engine APIs. I develop

- Rosetta-specific wrappers for the `cppdb` session class and basic database interaction functions that catch and throw database specific failures,
- a module to facilitate specifying database schemas using native Rosetta types that abstracts database backend inconsistencies. A schema object is built by instantiating and appending instances of columns and constraints objects and then writing the schema to an SQL string. To specify the data type for column objects, at construction, an instance of a class corresponding to a Rosetta primitive data type is passed in.

- a module to optimize batching of database insertions into transactions. Writing data to a database in atomic transaction allows the database backend to optimize the resolution of constraints and recover from failure by rolling back to before the transaction. However, when inserting a large number quantity of data, on certain cluster architectures it can be more efficient to split the transactions into moderately sized pieces. The insertion statement generator facilitates splitting up data to be inserted into user specified batches that are each written as a transaction to the database.
- unit and integration testing suites built on the SQLite3 engine (it has a permissive license, so we can distribute it with Rosetta), and
- distributed BuildBot testing slaves to test PostgreSQL and MySQL functionality

By building the `FeaturesReporter` framework on the Rosetta platform, I can use already developed infrastructure for molecular conformations. Rosetta has low level libraries for numeric and geometric computation, including graph and spatial decomposition data structures, several molecular conformational representations (including spatial eletron density, atomic, and a unified atom centroid mode), and a wide range of feature models developed primarily for structure prediction and evaluation. In practice, to use these feature models within the `FeaturesReporter` framework often requires refactoring to expose access to individual feature instances. This effort is repaid as using a features model for both prediction and features analysis makes the code for feature models more robust. Further, performing features analysis using representations that are consistent with the feature models used for prediction means structural insight gained can be directly applied to improving the feature models in Rosetta. This theme will be revisited in the case studies.

5.2 Support for Statistical Analysis Methods

I develop methods to assist common analysis tasks in developing features analysis scripts. Each method implements the split-apply-combine methodology (Wickham, 2011). The concept is to *split* the instances into groups, *apply* an analysis method, and then *combine* the results with the group identifier. Therefore, the user specifies the a table of feature instances, the identifying and measurement columns, and additional analysis parameters. Methods developed for computing summary statistics include:

- Estimating 1- and 2-dimensional density, including support for histograms and kernel density estimation (Section 5.1).
- Estimating the location of the primary mode as the location of the peak for the roughest kernel density estimation that retains a single peak.
- Calculating cumulative distribution functions and quantile-quantile functions.
- Sliding windows with the number of windows and fraction of overlap as parameters.

Methods for graphically comparing feature instances and summary statistics are based on the grammar of graphics, as implemented in the ggplot2 package. The concept is to construct plots by describing a mapping of properties of data to graphical elements on the page.

Additionally, the features analysis tool provides support for computing quantitative measure of similarity. Comparison matrices for a given metric over all pairs of groups Comparison statistics for each group between two sample sources and for all pairs of groups from a single sample source. Supported metrics include two sided t-tests, bootstrapped and non bootstrapped Kolmagorov-Smininov test, Anderson-Darling 2-sample test, and the earth mover distance with L_1 divergence, histogram and KDE-based Kullback-Leiblier divergence, and Maximum-Mean Discrepancy measures.

5.2.1 Kernel Density Estimation

A central method for comparing distributions is to estimate the probability density function form samples and compare for each sample source and compare the estimate densities. Kernel density estimation (KDE) is a method to estimate from a given set $\{x_1,\ x_2,\ \dots,\ x_n\} \in X$ sampled independently from an unknown distribution f over X, a smooth probability density function $\hat{f}$ (Rosenblatt 1956, Parzen 1962). The KDE depends upon kernel function $K: X \to \mathbb{R}^+$ that is a symmetric probability distribution and is parameterized by a bandwidth, h,

$$\hat{f}_h(x) = \frac{1}{n}\sum_{x_i} K\left(\frac{x - x_i}{h}\right)$$

When X is a set of real numbers, a common kernel function is the Gaussian with unit variance.

The choice of the bandwidth parameter values reveals different aspects of the data (Chaudhuri 1999). When the bandwidth parameter is large, the influence of each point is spread out, biasing the estimation to smoother functions. When the bandwidth parameter is small, the influence of each point is narrow, making the estimation have higher variance. In general, it is better to have a large bandwidth parameter when there is less data and a narrow bandwidth parameter when there is more data. Despite this, the bandwidth parameter must still be specified, so significant effort has gone into developing reasonable default bandwidth selection methods. The so-called first generation bandwidth estimation methods (Jones 1996), include biased and unbiased cross-validation, and Silverman's "rule of thumb" (Silverman 1986), which is a simple function of the sample variance and the number of samples,

$$h = \hat{\sigma}\left(\frac{4}{3n}\right)^{\frac{1}{5}}$$

Data-driven second-generation methods, which have been claimed to be superior based on real data examples, simulation studies, and asymptotic analysis (Jones 1996), include Sheather-Jones (1991), which optimizes the integrated mean square error through the "solve-the-equation plug-in method" (Woodroofe 1970). However, the claims of superiority have been called into question (Loader 1999).

To determine an appropriate default bandwidth selection method, I compare five methods from the MASS package, Silverman's rule of thumb (`bw.nrd0`), biased cross-validation (`bw.bcv`), unbiased cross-validation (`bw.ucv`), Sheather-Jones solve-the-equation, (`bw.SJ(method = "ste")`), and Sheather-Jones direct plug-in (`bw.SJ(method="dpi")`). To compare these methods, I sample from the Native sample source AHdist features subsets of sizes of 1, 10, 20, …, 90 thousand (representative of the sample sizes estimated in dissertation). Then for each method I compute the bandwidth and measure the running time (Figure 5.2.1). I observe that both first and second-generation methods agree reasonably well over the range of features analyses required for structural biology predictions in this dissertation. Because the Sheather-Jones method fails for some of the subsets, and because other methods have higher computation time, I choose the default bandwidth by Silverman's rule of thumb, and make other methods available through the `estimate_density_1d` function.

In most cases, the default bandwidth is appropriate. However, in certain applications, such as detecting derivative discontinuities, which appear as high-frequency spikes in the density distribution, choosing a very tight kernel can reveal the pathology; see Section 4.2 and (Leaver-Fay 2013).

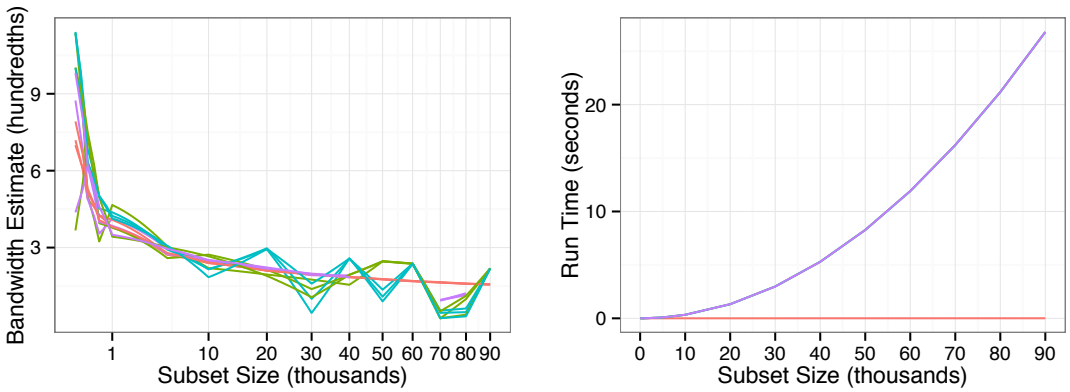


Figure 5.2.1 Kernel Density Estimation Bandwidth Selection Methods: Four kernel density bandwidth selection methods are applied to different size subsets of HBond lengths from the Top8000 dataset. Red: Silverman's rule of thumb, light green: biased cross-validation, teal: unbiased cross-validation, purple: Sheather-Jones solve-the-equation. (left) Estimated bandwidth, averaging three runs for each method. The Sheather-Jones method failed for the 50k, 60k, and 90k subsamples. (right) Running time of bandwidth selection method. All methods except for Silverman's rule of thumb follow the increasing curve. Since the rule of thumb is fast, stable, and tracks the more expensive estimates for representative samples of features, that is my default.

5.2.2 Normalization for Kernel Density Estimation

In defining features for density estimation features analysis, it is important to determine the appropriate normalization for any change of variables and treatment of boundary behavior.

I can best illustrate the importance of normalization by an example using features analysis in exploratory visualization. Consider the BAH angle in an Hydrogen bond as defined in Section 4.2.3; the exterior angle defined by the Acceptor-Base (B), Acceptor (A), Hydrogen (H) atoms, which is zero degrees when these atoms are co-linear. We would like to see if the H-bond causes a certain BAH angles to be preferred. The natural null hypothesis is not, however, that all angles are equally likely, but instead that the H atom is equally likely to be placed anywhere on a sphere about the acceptor atom, A. A range of BAH angles defines a ring or annulus on the sphere; the surface area of a ring is quite small for angles near zero.

Figure 5.2.2 shows in red the feature density of angle BAH for points chosen randomly from a unit sphere, and in teal the density normalize by dividing each angle by sin(BAH), the circumference of its circle on the unit sphere. This experiment shows that the normalization flattens the expected angle from the null hypothesis, which would make it easier to recognize systematic affects caused by hydrogen bonding. It also shows that the variance is not even across the domain even after the normalization. On the far left, there are few counts so the variance is still high, while on the right, there are many counts so the variance is lower.

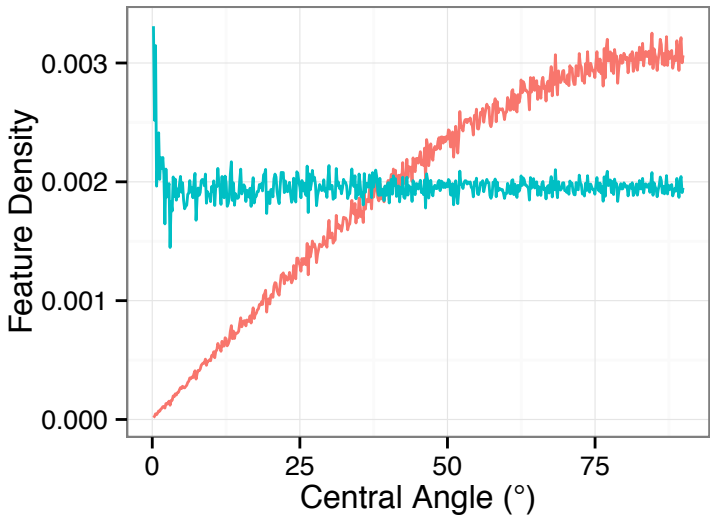


Figure 5.2.2 Normalization of the BAH Angle Null Density Distribution: 100,000 points are sampled from a 3D Gaussian with unit variance and measure the angle with the z-axis. Each curve is a histogram of the samples with 512 bins. The red curve has no normalization; the blue curves is normalized by weighting each angle by 1/sin(BAH). The user can simply pass the `weight_fun=canonical_3d_normalization` option to the `estimate_density_1d` method in the features analysis tool. Normalizing so that the null-hypothesis angle distribution (from the spherically symmetric 3D Gaussian) becomes flat in the density estimation makes it easier to identify anisotropy in angle distributions.

In general, to normalize a feature distribution $F(x)$ over the domain $[a, b]$, a weight function $w(x)$ is sought so that a given a reference distribution $G(x)$ has a given target probability density function $t(x)$,

$$\int_a^x G(u)w(u)du = t(x)$$

When the density function for $G(x)$ is $t(x)$, then the weight function is not needed, i.e., $w(x) = 1$. However, in applying a transform to a feature for either geometric reasons as above or for statistical reasons (Wand 1991), the reference distribution in under the transformation may not have the appropriate target density function. In this case, the weight function will be non-trivial. Once the weight function is found, weight function is used to define the weighted density function $f_w(x)$,

$$\int_a^x F(u)w(u)du = f_w(x)$$

Then comparing the plots of $f_w(x)$ and $t(x)$ is reasonable.

5.2.3 Boundary Conditions for Density Estimation

Explicit boundary conditions on the domain of the feature require special treatment because typical kernels such as Gaussian kernels is defined over the whole real line, and would assign density to undefined regions of the feature space (Silverman 1986, Wand 1995). One technique is to compute the density estimation over a change of variables (Marron 1994, Koekemoer 2008). For example, if the feature is defined over positive real numbers, the logarithm of the feature is defined over the whole real line, where Gaussian kernels can be applied. Other methods include reflecting the data across the boundary (Jones 1993). An alternative approach to comparing distribution near a boundary is compare cumulative distribution functions or QQ-plots when comparing sets of feature instances (Wilk 1968)

For circular domains, linear kernels can be wound around the domain leading to the wrapped or kernels defined on the circular domain such as the von Mises distribution can be used (Mardia 2000, Taylor 2007).

5.2.4 Statistical Hypothesis Testing and Exploratory Data Analysis Visualization

Exploratory data analysis championed by Tukey (1977) uses diverse graphical and summary statistics to identify patterns in data without an explicit model. In contrast, in classical statistical hypothesis testing, a researcher specifies a fixed model up front, designs an experiment, collects data, and uses the data to determine whether or not to reject the model. Recently Buja, et al. (2009) have bridged this divide by formalizing exploratory data analysis in the statistical hypothesis-testing framework. The central idea is that even though a model is not explicitly defined in exploratory data analysis, the context of the exploration leads the analyst to develop an implicit null hypothesis in their mind, which they then attempt to reject by inspecting the data. This insight can help feature analysis developers to establish context in presenting results that guide the analyst to specific null hypotheses. For example, in creating the small multiple plots, the developer guides the analyst to the implicit null hypothesis is that the graphic in each cell is consistent across the different cells. When a trend or single cell deviates, this contrast allows the analyst to reject the implied null hypothesis. As another example, in using kernel density estimation, variable transformations and normalization can make the null distribution have a flat line in the plot. When the density deviates from the flat line, this signifies deviation from the null hypothesis, thus associating visually interesting curves with rejection of the null hypothesis.

Graphical display of the features analysis summaries is a high bandwidth way of communicating the data. This is particularly effective for identifying unexpected behavior in the data because the analyst is able to inspect a large amount of data with minimal compression. Once a potential deviation from the null hypothesis is identified from the graphical representation, more concise summaries facilitate more efficient monitoring of improvements to the energy function. Quantitative summary statistics are one way to more concisely summarize the features analysis results.

To generate quantitative summary statistics, the features analysis tool provides support for computing similarity metrics and test statistics, T-tests, Anderson-Darling non-parametric tests, or Maximum Mean Discrepancy tests. As an example, the Maximum Mean Discrepancy (MMD) computes the distance between kernel density estimations by interpreting the kernel density estimate as the mean value of the data lifted to a Reproducing Kernel Hilbert Space (RKHS). The MMD is simply the distance between the means in the RKHS (Gretton 2006). To compute the statistics in table 5.2.3, after querying the database and transforming coordinates so that the `data.frame` `feature_data` has columns (`sample_source`, `don_chem_type`, `acc_chem_type`, `hx`, `hy`, `hz`), call

```
stats <- comparison_statistics(
  sample_sources=sample_sources,
  f=feature_data,
  id.vars=c("don_chem_type", "acc_chem_type",
  measure.vars=c("hx", "hy", "hz"),
  comp_funs=c("mmd_comparison"))
```

The `mmd_comparison` function uses the `kmmd` function with default arguments from the `kernlab` R package (Karatzoglou 2013).

Maximum Mean Discrepancy
H-Atom Position in Acceptor Coordinate Frame
MMD(Talaris2013, Native) - MMD(Score12, Native)

		aAHX	aCXA	aCXL	aHXL	aIMD	aIME	aPBA
		y	n,q	d,e	s,t	h	h	bb
dAHX	y	-5.66	-6.35	-7.74	-2.31	-2.37	-1.53	-10.16
dAMO	k	-5.12	-1.89	-12.21	-0.45	-2.21	-0.04	-3.89
dCXA	n,q	-6.44	0.55	0.44	-4.4	-2.28	-0.33	-3.10
dGDE	r	-2.63	-1.82	-3.85	-1.97	-0.22	0.42	-8.78
dGDH	r	-4.79	1.36	-3.94	-2.12	-0.18	-1.24	-3.88
dHXL	s,t	-5.99	-4.55	-6.28	-4.88	-2.06	-0.69	-11.93
dIME	h	-2.16	-2.15	-6.37	-4.13	-1.30	-1.12	-2.90
dIME	h	-7.77	-6.68	-6.78	-5.83	-4.2	-0.55	-7.06
dIND	w	-3.50		0.96	-2.05	0.87	-2.25	-2.99
dPBA	bb	-5.74	1.54	4.36	-6.06	-2.88	-0.18	-1.24

Table 5.2.3 Maximum Mean Discrepancy to Native: Score12 vs. Talaris2013: (MMD) for H-bonds (Section 6.2-3) over H-atom position in the acceptor coordinate frame. For each combination of donor and acceptor type, the MMD values between Talaris2013 (Section 7.2) and Natives (Top8000 filtered for B-Factor < 30) and between Score12 and Natives are computed and the difference times 100 is displayed. The majority of results are less than zero, indicating that the H-bonds from the Talaris2013 sample source are more consistent with Natives than Score12.

6 Feature Models and Computational Models

The features analysis tool presented in Chapters 3-5 assists researchers in comparing batches of structural features to make inferences about discrepancies between sample sources. In this Chapter, I elaborate on how the features analysis tool can be used to drive improvements to predictive computational models by reducing detected discrepancies. In the introductory section 6.1, I first develop statistical terminology and concepts for features models and computational models and discuss other work related to improving computational models in structural biology. To demonstrate the utility of the features analysis tool, I improve the H-bond model in Rosetta (Section 6.3-6). This requires a detailed discussion of H-bond models (Section 6.2) and several case studies in different types of corrections. Following this chapter, as an independent assessment of the validity of features analysis as a means of improving computational modeling, I develop and use recovery tests to test the predictive accuracy of the energy functions under consideration (Chapters 7, 8).

6.1 Computational Models in Structural Biology

Recall that a *statistical model* is a scientific model (Section 2.1.2) that represents a physical process or phenomenon as an abstract parametric family of probability distributions over a space. The probability can be interpreted either as the researchers' uncertainty in how the system behaves (Good 1950, Jeffreys 1961, de Finette 1974, Berger 1985, Jaynes 2003) or as representing the variation of the system over repeated measurements (Pearson 1920, Fisher 1974, Neyman 1937).

Recall that a *sample source* (Section 3.1.1) in structural biology is represented by a statistical model that abstracts the sometimes-messy details of a process to generate molecular conformations. For a native sample source, the process involves the laws of physics guiding molecular stability, the molecular evolution in the species, the researcher's decisions of which molecules to assay, the limitations of the experimental assay, and work to harmonize and validate available data into a reference data set such as the Top8000. The statistical model hides these details by representing the process of making an observation as a mathematical function that assigns to conformational states (regions of conformation space that form a σ-algebra) the probability of observing a conformation in the state.

To the extent that the statistical models correspond to physical processes, comparison of statistical models conveys information about the comparisons of the physical processes themselves. In particular features analysis can help tease apart the influence of the various factors that contribute to determining the sample of native molecular conformations. Comparing a sample source of X-ray crystal structures vs. NMR structures may reveal influences of the assay. For example, we know that structures obtained by crystallography can by influenced the process of crystallization, the interpretation of the observable diffraction pattern into electron density maps, and fitting of the atomic conformations to the electron density. However, features analysis can reveal additional factors that should be taken into account. Comparing a sample source made of antibody loops vs. other cellular proteins may reveal differences associated with the special antibody maturation process vs. standard molecular evolution.

When a statistical model represents the behavior of software developed to run simulations or generate predictions, the software is called a *computational model*. The computational model is a process implemented in a physical computer that produces actual data and is separate from the mathematical representation that the statistical model affords. A statistical model that represents a

computational model is defined by a distribution over conformation space in the same way that a statistical model that represents a sample source that generates native conformations.

A computational model is a scientific model in the sense that it is a simple system to represent a more complex system; for example, protein prediction software represents how proteins in nature fold. The flexibility and precision of writing and executing computer code allows computational models to be engineered from simple, understandable parts to achieve complex behavior. Just like a statistical model, a computational model has a *functional form* and *parameters* that can be adjusted to tune the correspondence to the process it models. Unlike a statistical model, which is a mathematical concept, the functional form and parameters of a computational model are constrained by the processing power of the computers the software is implemented on. For example, as I discussed in Section 2.4, the smoothness of the objective function impacts the performance of the gradient-based minimization algorithm and therefore impacts the sampling distribution in practice, while for statistical models these considerations do not come into play as the distribution is defined by the energy function abstractly.

6.1.1 Energy Functions in Computational Models

In structural biology, a computational model that defines a conformational distribution through an energy function and the Boltzmann equation is called an energy model. The energy function is often a linear combination of feature models: Let x be a conformation, containing a set of feature instances $\{f_{ij}\}$ in which features $\{f_{i1}, f_{i2}, \ldots, f_{in_i}\}$ are of type F_i. For each feature type F_i, there is a *feature model* M_i with parameters θ_i that assigns an energy to each feature instance of the type, $M_i: (\theta_i, F_i) \rightarrow \mathbb{R}$. For a given set of parameters Θ and weights W, the energy of a conformation x, $E(x|W, \Theta)$, is the linear combination of the energy of each feature with parameters θ_i from Θ, and weights w_i, one for each feature type,

$$E(x|W,\Theta) = \sum_{x \to f_{ij}} w_i M_i(f_{ij}|\theta_i) \qquad \text{(eq. 6.1.1)}$$

Given a temperature factor *kT*, the Boltzmann equation from statistical mechanics transforms an energy function into a probability distribution, where *Z* (sometimes called the partition function) is the normalizing constant to ensure that the integral over the conformation space is equal to one,

$$p(x|W,\Theta,kT) = \frac{1}{Z} e^{\frac{-E(x|W,\Theta)}{kT}} \qquad \text{(eq. 6.1.2)}$$

As an example of an energy function in structural biology, the default and community standard in Rosetta before this dissertation was *Score12* (Rohl *et al*. 2004), which consisted of the following core terms, each of which are feature models:

- Lennard-Jones: Models van der Waals forces, which are short-range attractive and repulsive force between non-bonded atoms not due to electrostatics; the features are pairs of atoms that are at least four bonds apart, but within an interaction cutoff.
- Implicit desolvation (Lazaridis & Karplus, 1999): Models the energetic cost of removing atoms from polar solvent (e.g., water); the features are pairs of atoms that are at least four bonds apart, but within an interaction cutoff.
- Orientation-dependent H-bond (Kortemme et al., 2003): Models H-bond interactions; the features are H-bond donor and acceptor groups within an interaction cutoff.
- Sidechain and backbone torsion (Rohl 2004, Dunbrack 2011): Models feature distributions observed in experimental data collected in the Protein Databank; the features are residue sidechains and the backbone torsion angles of the residue and adjacent neighbors.
- Residue-pair interaction: Models specific interactions due to electrostatic forces; the features are pairs of charged residues with the "action centers" within an interaction cutoff.
- Disulfide bond (Leaver-Fay 2011): Models disulfide interactions

- Reference energies: Models the energy of an amino acid in a generic unfolded state; the features are the amino acid identity of residues.

In addition to these core feature models, the Rosetta platform permits developers to define other feature models for use in prediction protocols and approximately 50 feature models have been developed. These include alternative models for basic biophysical forces, such as sidechain conformations, electrostatic interactions, and H-bonding; models that evaluate non-protein macromolecules such as RNA and DNA, small molecules, and non-canonical amino acids; models that evaluate alternative molecular representations such as low-resolution conformations (centroid mode) and spatial grid based models; and models that are not routinely used because of their high computational cost such as evaluation of core-packing through spatial voids (Sheffler & Baker, 2009), solvent accessible surface area (Lee 1971), and the area of contiguous hydrophobic surface patches (Jacak 2012).

6.1.2 Specifying and Improving Energy Functions

In the functional form of the energy function (eq. 6.1.1), the adjustable parameters are the weights, w_i, and the feature model parameters, θ_i. Ideally these parameters are specified to achieve good scores at a broad range of scientific benchmarks that measure the accuracy of the computational model in representing native sample sources (See sections 7.1 and 7.2 for more on Scientific Benchmarking). However, because of the interdependence of the energy function parameters, and the extreme cost of most scientific benchmarks, specifying the parameters is challenging. My features analysis tool addresses this problem by helping researchers to fit parameters to reduce discrepancies between decoy and native feature distributions. Before I describe this in detail and demonstrate it through case studies, let me discuss other practical and theoretical studies on specifying energy function parameters.

Recently, there has been renewed interest in improving energy functions for molecular models in the structural biology community. The increasing availability of computational resources allows for more extensive molecular simulations that highlight discrepancies between structure predictions and both experimental and theoretical reference data and highlight the effects of energy function parameterization on the simulations. In 2012, Piana et al. used Anton (Shaw 2009), a special-purpose computer for molecular dynamics simulations, to evaluate the effect of electrostatic interaction cutoffs in computing the free energy of folding for the villin headpiece domain (Kubelka). In 2012, Tyka et al. (discussed in Section 2) evaluate the Rosetta energy landscape near native conformations. In 2012, Beauchamp et al. evaluate 11 energy functions against 5 different explicit solvent models in MD simulations to recapitulate 524 NMR measurements. In 2012, Vymětal and Vondrášek evaluated the backbone and sidechain energy landscape for 6 energy functions over each amino acid capped to form a dipeptide.

The increasing availability of new experimental data in the Protein Data Bank and databases of high quality ab initio QM calculations (Hobza 2011) can now serve as reference data for more extensive scientific benchmarking (Leaver-Fay 2013). Finally, there has been substantial methodological progress in fitting heterogeneous models for molecular structure prediction (Hamelryck 2010, Song 2010, Li 2011, Leaver-Fay 2013, Wang 2013a), each based on iteratively adjusting parameters to increase model fit (See section 5.6).

An alternative strategy, and the primary motivation for the tools developed in this dissertation, uses features analysis to tune the parameters of an energy function to improve feature distribution recapitulation. If an energy function generates a feature distribution that does not recapitulate the native feature distribution then the energy function over-predicts the frequency of certain feature configurations and under-predicts the frequency others.

To see why over-prediction is problematic consider the use of structure prediction to design a conformation with specific biological function and assume that it contains a feature instance that is rarely observed in nature. From statistical mechanics, feature configurations are rarely observed because they are unstable. So, when the design is tested in the lab, it is probable that the designed conformation will structurally reorganize itself so that the feature has a more stable configuration. This reorganization may be inconsequential, or it may destabilize whole the conformation, wrecking the designed function of the molecule. For naturally occurring proteins, the difference in free energy between the folded state and the unfolded state is typically on the same order as 5 kcal/mol, which is about the free energy of a single hydrogen bond.

6.1.3 Reference Ratio Method as a Theoretical Framework for Energy Function Optimization by Features Analysis

The reference ratio method (RRM) of (Hamelryck 2010) and (Mardia 2010) formalizes an iterative approach to fitting parameters in structural biology energy functions. The method is closely related to the features analysis tool. Here, I describe the RRM, bridging the notational and contextual differences between their work (Mardia 2010) and mine.

Let X be a conformation thought of as a random variable over a conformation space and let Y be a feature also thought of as a random variable related to X by the function $m: X \rightarrow Y$. For example let Y be sets of H-bond BAH angles, so that m returns the set of BAH angles observed in a conformation. Let $N_X(x)$ and $D_X(x)$ be native and decoy probability distribution functions over the conformation space X. We want $D_X(x)$ to equal $N_X(x)$. However, since the feature Y is a deterministic function of the conformation X this is equivalent wanting $D_{X\times Y}(X,Y)$ to equal $N_{X\times Y}(X,Y)$. Consider the joint distributions using the rules of probability,

$$N_{X\times Y}(X,Y) = N_Y(Y)N_{X|Y}(X|Y)$$

$$D_{X\times Y}(X,Y) = D_Y(Y)D_{X|Y}(X|Y)$$

The distributions $N_Y(Y)$ and $D_Y(Y)$ are simply the feature distributions for Y for the native and decoy sample sources, which can be estimated from samples obtained from sampled conformations. The distribution $N_{X\times Y}(X,Y)$ is not readily accessible because samples from the native distribution are too sparse. The distribution $D_{X\times Y}(X,Y)$ is accessible only through the computational model, but not in closed form. In particular we don't have access to the distributions of native and decoys conditional on having a given feature value; $N_{X|Y}(X|Y)$ and $D_{X|Y}(X|Y)$. Since we have only limited access to the native and decoy distributions, the RRM method suggests considering the modified the decoys distribution,

$$D'_{X\times Y}(X,Y) = \frac{N_Y(Y)}{D_Y(Y)}D_{X\times Y}(X,Y) = N_Y(Y)D_{X|Y}(X|Y).$$

In other words, the method suggests modifying the computational model by multiplying the distribution by the ratio of the native over the decoy feature distributions. If the decoy feature distribution differs from the native feature distribution, then this adjusts the decoy distribution to make it more like the native feature distribution. The feature distribution for the updated computational model in general does not perfectly recapitulate the native feature distribution $D'_Y(Y) = \int N_Y(Y)D_{X|Y}(x|Y)dx \neq N_Y(Y)$, which suggests iterating the procedure.

The RRM can be seen as taking a step in the space of probability distributions over X with the Kullback-Leibler divergence metric, from $D_X(X)$ towards $N_X(X)$ in the Y direction. This suggests finding a set of features $Y_1, \ldots Y_n$ that span the space of probability distributions and using the

RRM as a partial derivative to perform gradient-based optimization. Like most iterative optimization procedures, work must be done to clarify under what conditions (theoretically and practically) the decoy distribution converges to the native distribution.

A large hurdle for realizing the RRM as a method for global energy function optimization is the complexity of the computational feature models. Many features are biophysically motivated but do not fully encode these biophysical assumptions into the model. For example, when two feature models give conflicting energy for a feature complex, biochemical intuition may indicate that one is more likely to be at fault than the other. In theory, this can be resolved by encoding the biochemical intuition as constraints or prior distributions on the parameters as part of the model specification. The difficulty of this task is compounded by the fact that computational feature models are often designed to be computationally efficient. This may require specific constraints such as smoothness that may be difficult to encode (See section 4.2 for examples).

I take the approach of building tools to enhance the researcher's abilities rather than replace their involvement. Specifically, I seek to reduce the discrepancy between feature distributions by optimizing parameters, but rather than doing so in an automated or formulaic fashion, I facilitate the researcher in diagnosing the cause of the problem. To do this, the tools aid in assessing if candidate modifications resolve the problem and identifying the extended consequences of the modifications in other feature distributions.

6.1.4 Weight Optimization

Once the individual model values are known, equation 6.1.1 can be viewed as a linear combination of weights, and weight values can be chosen to optimize the results on scientific benchmarks, such as amino acid and rotamer recovery.

One strategy, developed in the YASARA molecular modeling framework (an extension of the WHAT-IF project) with version of the AMBER force field, is to refit a subset of the parameters using a Monte Carlo search of the parameter space so that experimentally validated conformations do not deviate (measured by the root mean squared deviation to the native conformation) when optimized. They call the resulting parameterization YAMBER (Krieger 2004).

A second strategy, developed by the Wang group with the ZAPP energy function (Vreven 2012) is to fit weights to recapitulate experimentally measured binding free energies using the downhill simplex optimization (Nelder 1961).

A third strategy, developed for machine learning is to directly maximize the probability (or equivalently minimize the energy) assigned to the experimental training data in the computational model using maximum likelihood or maximum a posteriori estimation (Koller 2009). To prevent assigning arbitrary large probability values to the training data, the distribution must be normalized so the total probability is 1. But because of high dimensionality of the conformation space, estimating an unbiased normalization constant is computationally expensive even with sophisticated methods such as tree-reweighted believe propagation (Yanover 2008). Therefore studies that use maximum likelihood estimation often consider a reduced conformation space, for example by assuming the backbone is fixed and optimizing the weights while only considering sidechain degrees of freedom; this includes studies with the ORBIT (Sharabi 2010, Sharabi 2011) energy function and with the Rosetta energy function (Yanover 2008, Leaver-Fay 2013).

6.2 Modeling Hydrogen Bonding

In the following three case studies, I use the H-bond model to demonstrate how the features analysis tool can be used to create and refine feature models. Biochemically, an H-bond is a common interaction between a positively charged donor group that contains a hydrogen atom and a negatively charged acceptor group. Figure 6.2.1 shows a schematic of an H-bond, and H-bonds in the context of a complex macromolecule. I will formally define a hydrogen bond in the H-Bond model that I will develop in this section. The motivation for examining hydrogen bonds (H-Bonds) is 1) they have a complex and interesting geometry, and 2) they are biologically relevant.

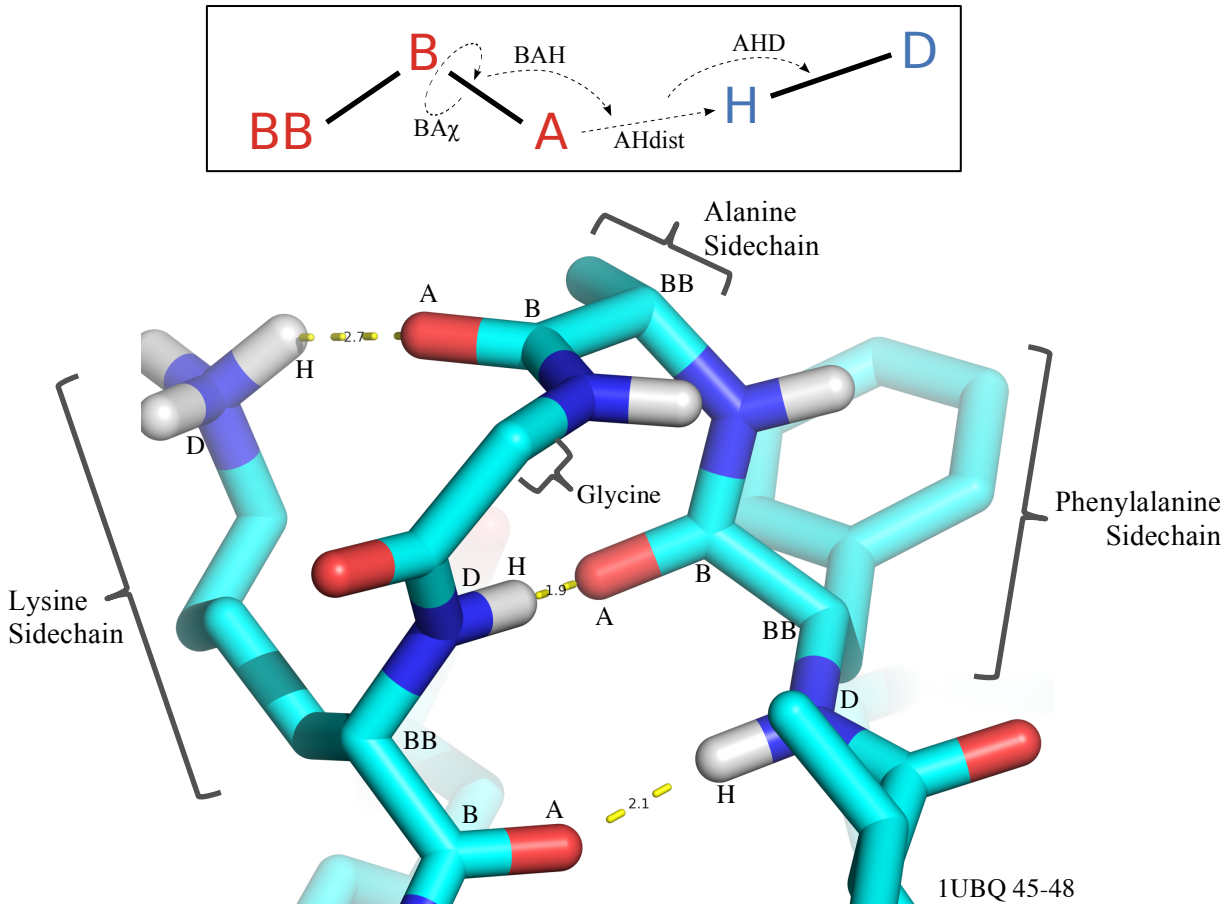


Figure 6.2.1 H-bonding in Molecular Structure: (top) Atoms that constitute a hydrogen bond, BBase (BB), Base (B), and Acceptor (A) atoms of the acceptor group, and the hydrogen (H) and donor (D) atoms of the donor group. (bottom) Residues 45-48 of Ubiquitin (1UBQ) form a β-turn motif. The lysine residue on the left and the phenylalanine residue on the right form the last two H-bonds of the β-sheet. The lysine residue donates a hydrogen atom that is accepted by the backbone of the alanine residue.

6.2.1 Hydrogen bonds and protein folding

Hydrogen bonds are essential for protein folding. However, the properties of hydrogen bonds that make them essential for biological function also make them challenging to model (Müller-Dethlefs 2000). At short ranges, polar contacts form partially covalent, geometrically specific H-Bonds (Fersht 1985, Fleming 2005, Arunan 2011, Arunan 2011a). These allow biological macromolecules to adopt overall geometrically specific conformations that are necessary for forming molecular signaling, cellular structures, organizing catalytic active sites, and forming multi-domain biological assemblies. Effectively capturing the geometric specificity of H-Bonds seems to require building complex models. At long ranges, polar contacts interact through slowly decaying electrostatic interactions. In nature, these forces can drive macromolecules to form stable, folded states. To capture these long-range interactions seems to require evaluating a large number of interactions, leading to computationally expensive models. Lastly, relative to other forces in macromolecules, polar contacts are relatively weak. In nature, this allows polar contact to form and break on reasonable timescales without catalysis, thus allowing for diverse H-Bonding patterns without extensive cellular support machinery. Additionally, this property allows for H-bonds to be modulated by the environmental context, which is necessary for macromolecules to function in response to external signals. The weakness of H-bonds, however, means that modeling them should not be done in isolation; instead, creating energy functions that capture H-bond behavior requires considering all the forces that influence the geometry.

6.2.2 Hydrogen bonds in Rosetta

The Rosetta energy function (Score12) has allowed researchers to make stunning molecular structure predictions (Kuhlman 2003, Ashworth 2006, Siegel 2010, Fleishman 2011, Koga 2012, Khare 2012). However, these and other projects have revealed that accurate prediction of polar

contacts is a significant limitation in the state of the art. A recent survey of protein-protein interaction design shows that researchers have predicted interfaces that are significantly more hydrophobic than those observed in nature. Moreover, of the molecules created and assayed in the lab, predictions with more polar interfaces were more likely to fail (Stranges 2012). I use features analysis to improve the modeling of polar contacts by resolving discrepancies between observed H-Bonds and those predicted in Rosetta and by integrating an explicit H-Bond model with a Coulomb potential.

I revisit the H-Bond model in Rosetta by assessing whether optimizing native conformations for the Rosetta score function introduces systematic discrepancies with experimentally characterized conformations. The first implementation of the Kortemme 2003 model in Rosetta does not adequately capture important chemical details and introduces several non-physical artifacts. In resolving these, I create a simple, chemically motivated functional form that can be parameterized to capture a diversity of molecular features (section 6.3). Then, based on the promising results in Morozov 2003, I investigate complementing the explicit H-Bond model with a short-range Coulomb model (section 6.5). Through extensive scientific benchmarks, I confirm that these models work together to improve predictive accuracy (Chapter 7).

6.3 H-bond Orientation at the Acceptor

In the next five subsections, let's walk through several applications of features analysis to investigate and improve the modeling of hydrogen bonds with Sp^2 hybridized acceptors in Rosetta. In these analyses the elementary features are H-bond defined by the Talaris2013 energy function (Section 7.2) and the BAH and BAχ angles conditional on donor and acceptor chemical types and secondary structure.

The native sample source is again the Top8000 chains dataset, filtered so donor and acceptor heavy atoms have B-factors at most 30 described in Section 4.1.2. The decoy sample sources are the same chains, optimized under variations of the Rosetta energy function with the FastRelax protocol (Section 2.3). Rosetta does not estimate B-factors, so no filtering is performed, however note that this means that there are more residues in the decoy sample sources than the native sample source.

The features analyses in the five subsections:

6.3.1: Describe Sp^2 acceptors H-bonds

6.3.2: Define the H-bond orientation features,

6.3.3: Explore Native H-bond Sp^2 feature distributions

6.3.4: Compare Native and a *Baseline* Sample sources via H-bond Sp^2 feature distributions

6.3.5: Create and tune an *HBondSp2* energy function to address observed discrepancies,

6.3.6: Compare Native, Baseline and HBondSp2 sample sources.

This case study supports my thesis by demonstrating how the feature analysis tool can be used to develop the functional form for a feature model. I consider the joint H-bond BAH and BAχ angle

distribution by acceptor hybridization. In natives, for H-bonds with Sp^2 acceptors, we will see that this feature has strong bimodal character, consistent with observational studies of H-bond geometry. Unfortunately, the distribution for the Score12 Rosetta H-bond model does not recapitulate the bimodal Sp^2 character, since it is iso-energetic in the BAχ angle. After evaluating orientation-dependent models of H-bonding, I develop a novel potential to model the Sp^2 character and demonstrate that it recapitulates the Sp^2 character and improves prediction of β-sheet shear.

6.3.1 Character of H-bond angles for Sp^2 acceptors

Molecular orbital theory predicts the geometry of covalent bonds based on the orbital hybridization of the bonding atoms. Since H-bonds are mediated through lone-pair orbitals, the theory predicts that H-bonds with Sp^2 acceptors (e.g., the backbone carbonyl oxygen) should be in the plane of the functional group and make an angle of 120° at the acceptor atom. Figure 6.3.1 shows the H-bond acceptor sites in canonical amino acids that have Sp^2 hybridization.

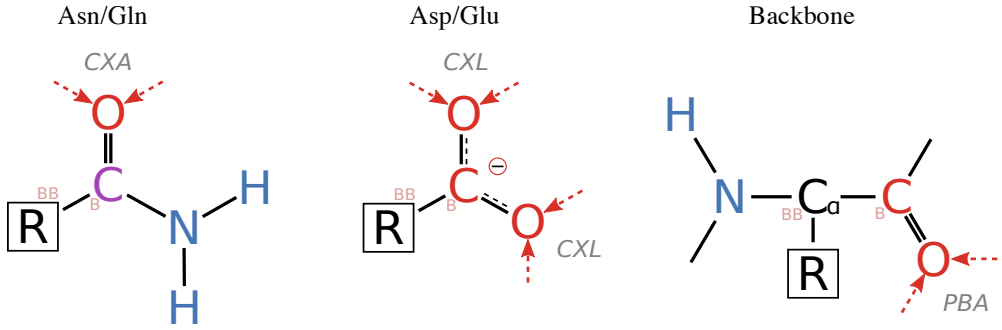


Figure 6.3.1 Acceptor Sp^2 Functional Groups in Canonical Amino Acids: Atoms are **C**arbon, Oxygen, **N**itrogen, and **H**ydrogen. Covalent bonds are represented as solid lines, single or double; bonds in resonance are solid and dashed lines together. H-bond acceptor sites (lone pairs) are red arrows. The rest of the residue attaches at R. Asparagine and glutamine residues (left) contain **c**arbo**xa**mide groups (CXA). Aspartate and glutamate (center) contain **c**arboc**xyl** groups (CXL). **P**rotein **b**ackbones (right) contain **a**mide groups (PBA). Acceptor Base and Base-Base are also labeled.

6.3.2 H-Bond Acceptor Orientation Features

The BAH angle feature is a random variable that measures the exterior angle defined by the Acceptor-**B**ase, **A**cceptor, and **H**ydrogen atoms. When the atoms are co-linear, BAH := 0°. To compute density estimates of the BAH angle, I normalize by cos(BAH) so that atoms uniformly distributed in space have a flat distribution and reflect the data across BAH=0. (See Section 5.5.4 on normalization.) Figure 6.3.2 (top) plots BAH for PBA, CXA, and CXL acceptors.

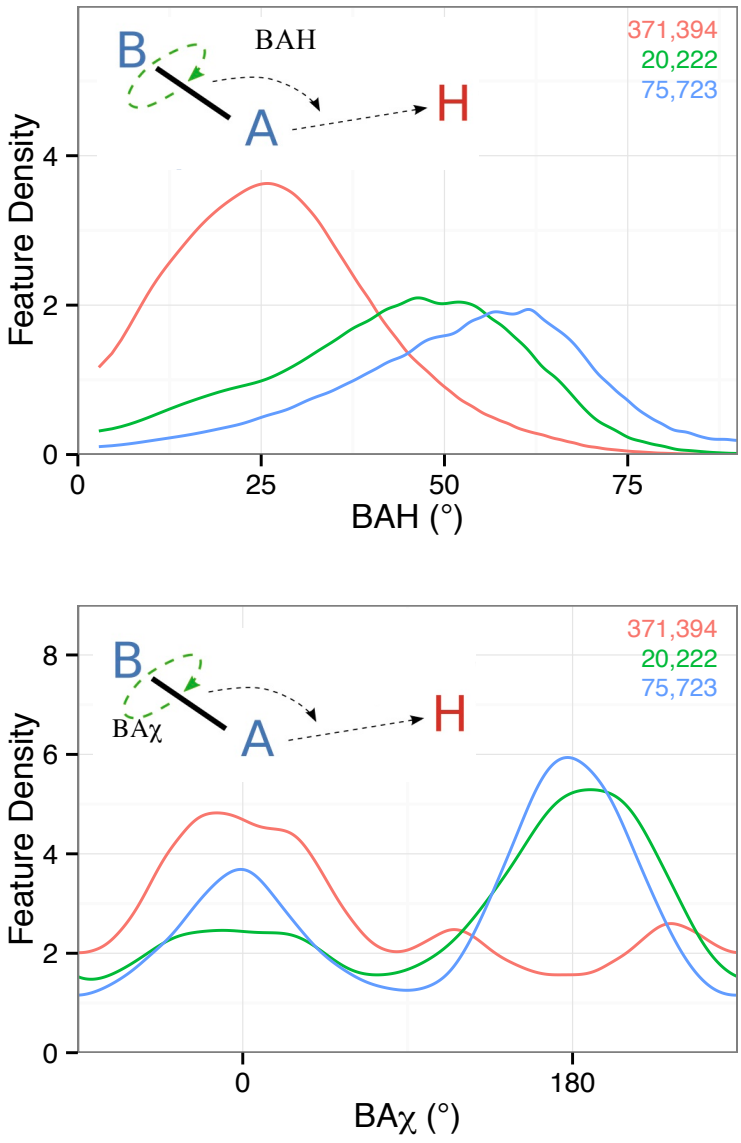
The BAχ angle feature is a random variable that measures the torsion angle defined by the Acceptor **B**ase-**B**ase, Acceptor **B**ase, **A**cceptor and **H**ydrogen. The *acceptor plane* defined by the atoms (BB, B, A) sets the zero axis. When the H-atom is in the acceptor plane and the 4 defining atoms form a "boat" (cis), BAχ := 0; when and the atoms form a "chair" (trans), BAχ := 180°. I define the domain of BAχ as [-90°, 270°] so that in-plane conformations are not at the boundary of the domain. To compute density estimation of the BAχ angle I use the wrapped normal to account for the circular domain. (See Section 5.5.4 on normalization.) Figure 6.3.2 (mid) plots BAχ for PBA, CXA, and CXL acceptors.

Plotting BAH and BAχ distributions separately does not reveal the coupling between these features. Since the (BAH, BAχ) angles define a sphere the coupling is especially tight at the BAH=0° pole where a small change in position on the sphere may correspond to a large change in BAχ. I plot the joint distribution on a plane in polar coordinates (R, Θ) using the Lambert-Azimuthal projection,

$$(R, \Theta) = \left(2\ \cos\frac{BAH}{2}, BA\chi\right).$$

This projection is area preserving and therefore does not require normalization for sphere surface effects. Figure 6.3.2 (bottom) illustrates the projection; the BAH angle is along the radial axis

(demarcated in degrees) and the BAχ angle is measured counter-clockwise about the origin from the positive x-axis.



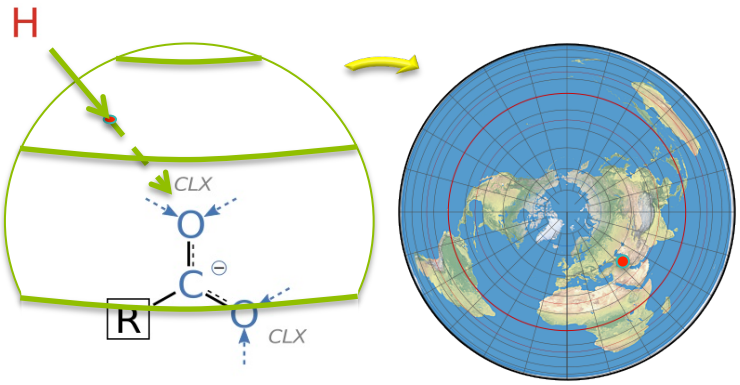


Figure 6.3.2 H-Bond BAH, BAχ, and (BAH,BAχ) Feature Definitions: BAH and BAχ, show acceptor orientation hybridization effects. Each curve is a kernel density estimation for all H-bonds with sequence separation greater than or equal to 5 and B-factor of the donor and acceptor heavy atoms at most 30 A^2. The red curve represents H-bonds with PBA acceptors; green, CXA; blue, CXL. Numeric counts for each density estimate are in the upper right. BAH angle (top) BAχ angle (mid). Details of the angle definition and density estimation are in the text.

(bottom left) Schematic of Lambert Azimuthal map projection: The H-atom is projected to the unit sphere centered on the A-atom, which is then (right) flattened to the plane, Projecting the world (Mapthematics, 2012) with the "north pole" mapped to (0,0), we see that the projection is 1-to-1 except at the "south pole" and that coordinates below the "equator" are visible.

6.3.3 Native Sp^2 Acceptor Orientation Features Analysis

With these features defined, we can use the features analysis tool to estimate and plot Native 1D and 2D distributions, then dissect them by estimating conditional distributions based not only on acceptor, but also on donor chemical type, and, for donor backbone atoms, on DSSP type, similar to the backrub bond angle case study in Section 4.1. The goal of the exploration of the native distribution is to gain understanding of the distributional characteristics, which we will try to replicate in the decoy distributions.

Although I focus on the distribution for H-bonds with Sp^2 acceptors in this case study, to establish context, let me first plot in figure 6.3.3 the Lambert-Azimuthal projection of the BAH-BAχ feature distributions for all acceptor hybridization types: ring, Sp^3, and Sp^2 (sidechain and backbone Sp^2 acceptors are plotted separately).

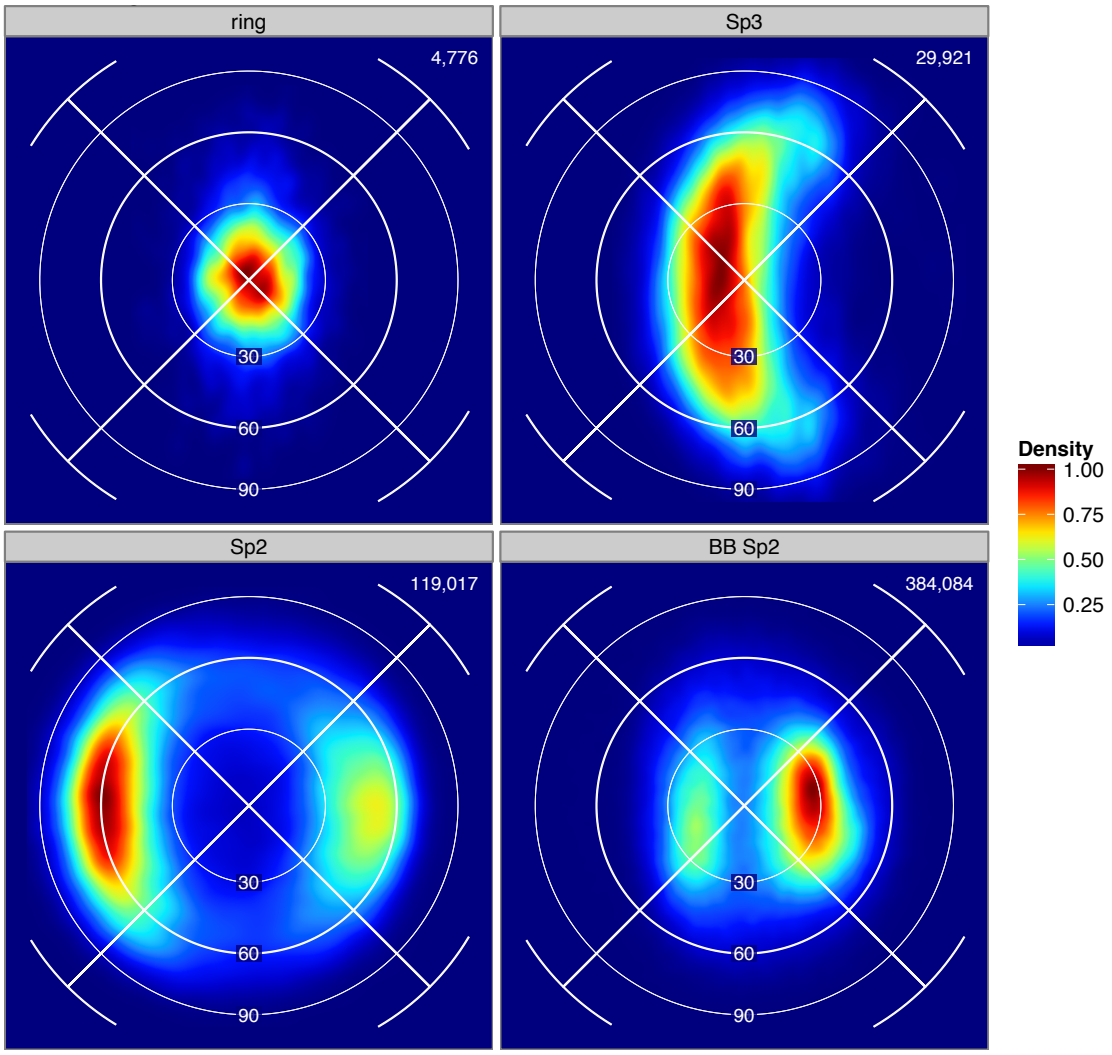


Figure 6.3.3 Native H-bond Orientation at the Acceptor by Acceptor Hybridization: Density is computed over the Lambert Azimuthal projections of the (BAH, BAχ) H-bond angles using kernel density estimation independently for each plot or *facet cell* (Section 5.3), where blue is less density and red is more density. (top left) Ring acceptors: His or Trp sidechains; (top right) Sp^3 acceptors: Ser, Thr, Tyr; (bottom left) Sp^2 sidechain acceptors: Asn, Asp, Gln, Glu; (bottom right) Backbone acceptors. For Ring and Sp^3 acceptors, BAH := Angle((BB+B)/2, A, H) and BAχ := Torsion(BB, (BB+B)/2, A, H). In canonical proteins, Sp^3 acceptors are hydroxyl groups so the BB atom is the hydroxyl H-atom and lies on the positive x-axis.

Observe in the bottom row that H-bonds with Sp^2 acceptors show a bimodal distribution with peaks in the plane of the acceptor, and in the top right cell that H-bonds with Sp^3 acceptors show a unimodal distribution perpendicular to the plane of the acceptor.

Following the pattern for features analysis setup in Section 4.1, I investigate the Native distribution of each elementary feature conditional on chemical types. In figures 6.3.4-5, for each acceptor type that has Sp^2 character I plot the BAH and BAχ distributions conditional on donor chemical type overlaid to get a sense of the variation by donor chemical type.

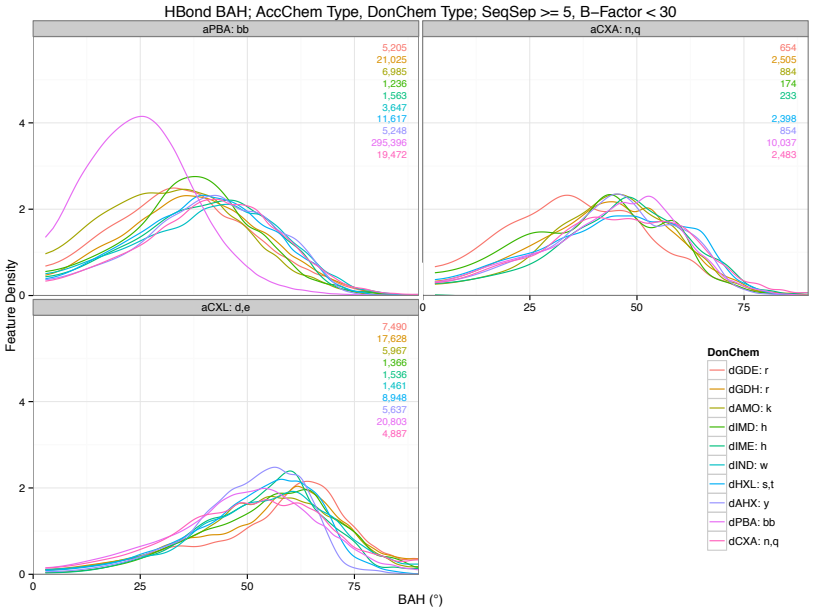


Figure 6.3.4 Native H-bond BAH Angle by Acceptor and Donor Type: Backbone-backbone H-bonds appear to be an outlier, but otherwise there is little variation by donor chemical type. For H-Bonds with sidechain donors and sequence separation at least 5, the peaks occur at BAH angles of 46.8°(PBA), 50.9°(CXA), and 59.9°(CXL).

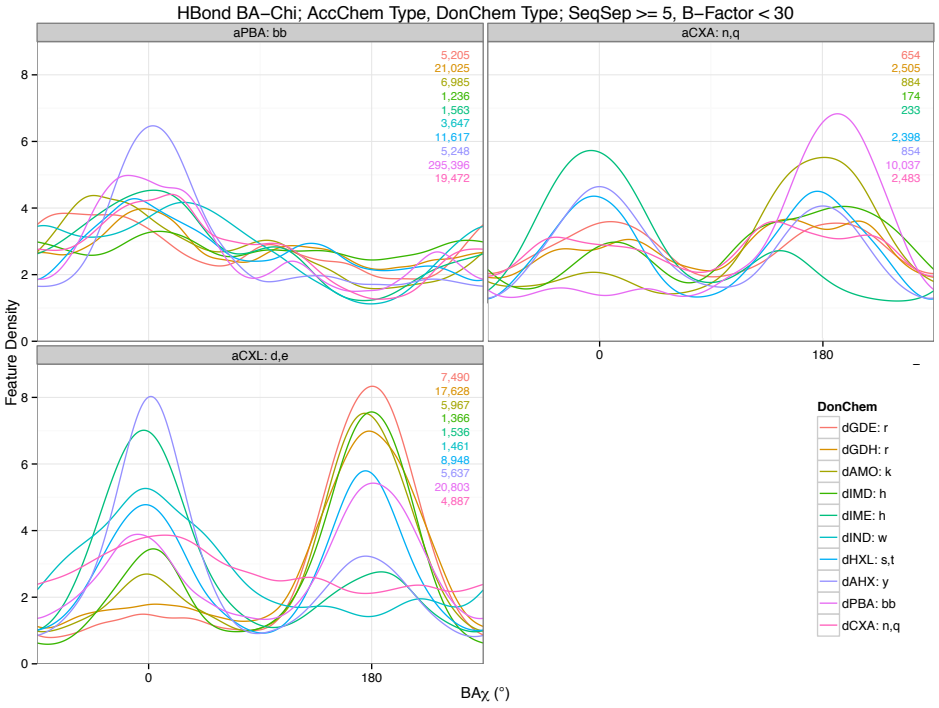


Figure 6.3.5 Native H-bond BAχ Angle by Acceptor and Donor Type: Peaks in the plane of the acceptor are visible at 0° (*syn* orbital) and 180° (*anti* orbital) for all acceptor types except for the anti orbital for backbone acceptors. There is substantial variation by donor chemical type, which is dissected in the next plot (Figure 6.3.6).

Since the BAχ distribution shows some variation by donor chemical type, I plot in Figure 6.3.6 the same data in Figure 6.3.5 to but faceting on donor chemical type.

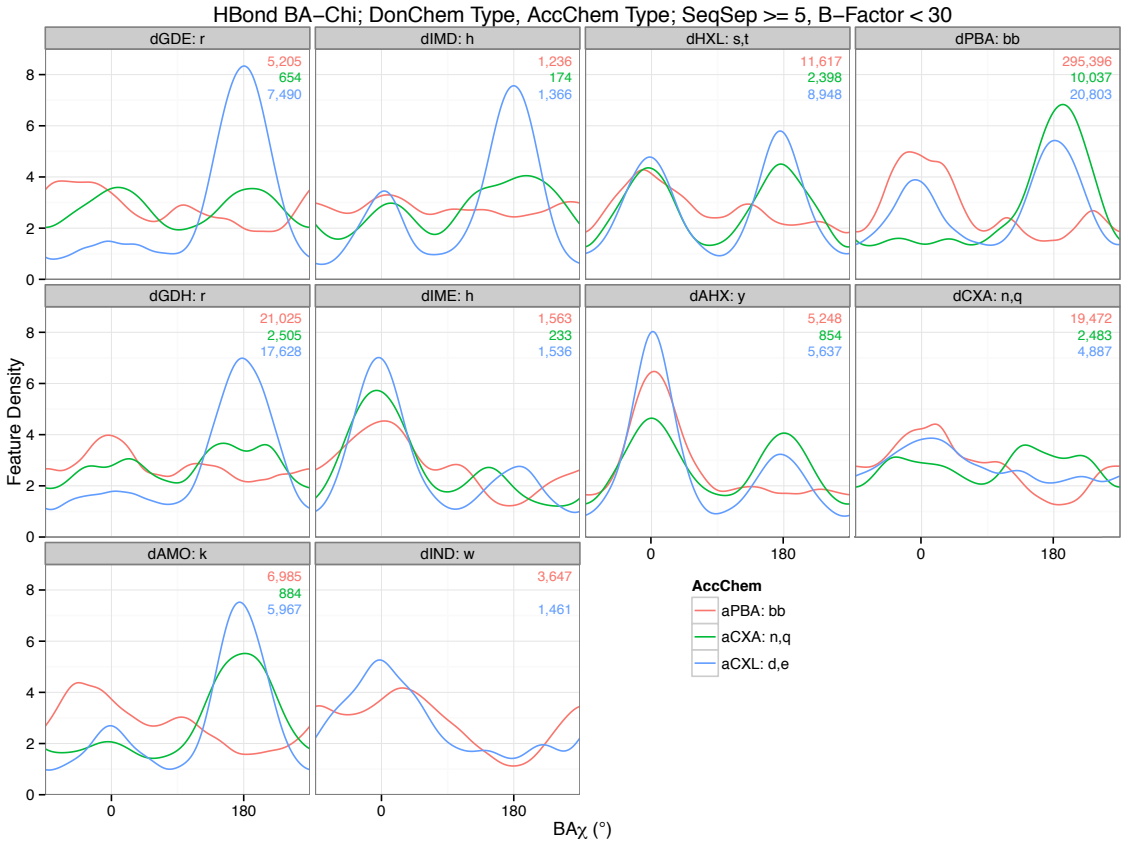


Figure 6.3.6 Native H-bond BAχ Angle by Donor and Acceptor Type: Preference for the syn orbital between aCXL and dGDE/dGDH/dAMO is consistent with salt bridge formation (Figure 6.3.7 right), and the interaction between aPBA and dCXA is consistent with a β-sheet like motif (Figure 6.3.7 left). The lack of preferences for the syn-orbital between aCXA/dPBA, for the syn-orbital for backbone acceptors to all donor types (Figure 6.3.5), and for all Sp^2 acceptors with the relatively inflexible donors, dAHX/dIME/dIND, is consistent with steric hindrance.

Inspection of Figure 6.3.6 reveals that some of the variation can be explained by donor dependent motifs, two of which are shown in Figure 6.3.7. These environmental effects should still be present in the computational model with out explicitly encoding for them.

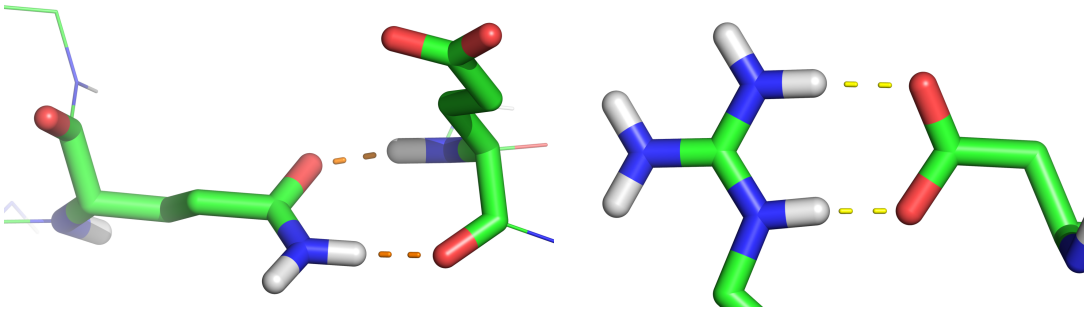


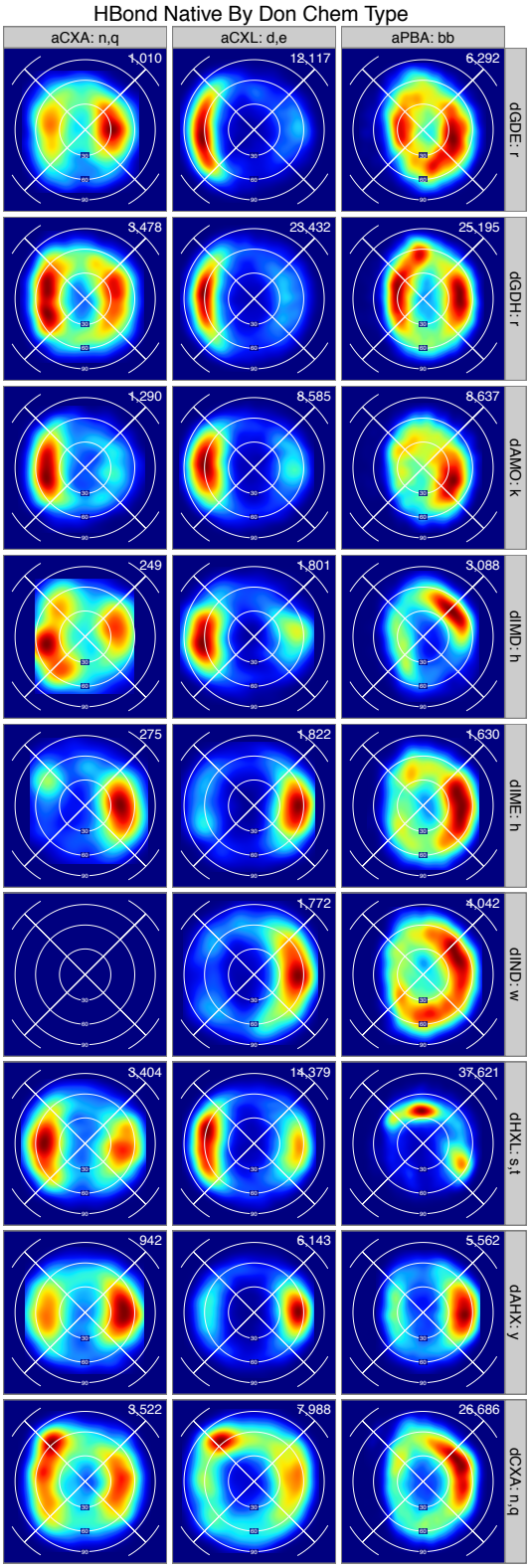
Figure 6.3.7 Common donor dependent motifs for H-bonds to Sp^2 acceptors (Branden 1999): H-bonds from aCXA to dPBA can form a β-sheet like motif, e.g., in 3q23 between residues 469 to 557 (left). H-bonds from aCXL to dGDH can form a salt-bridge motif, e.g., in 1yq2 between residues 120 to 187 (right).

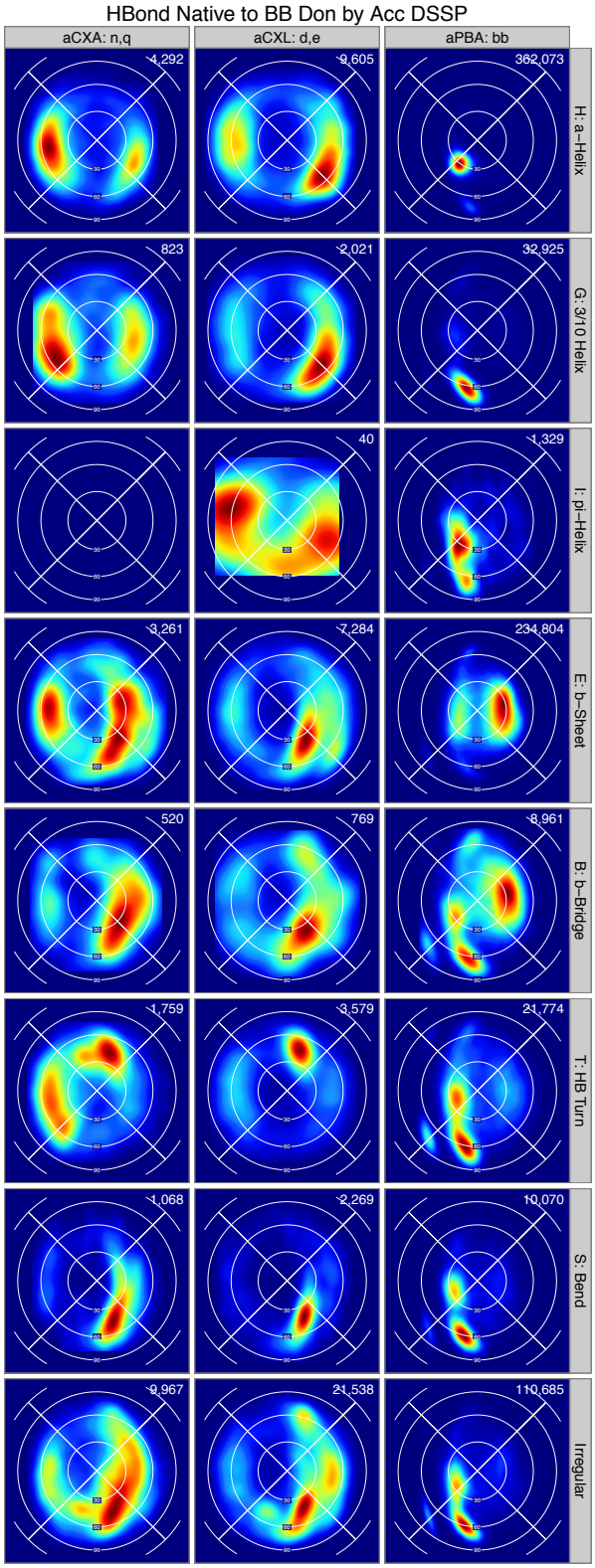
Investigating Single dimension feature distributions can obscure the interaction between the features, which can be revealed by plotting the joint density distribution. Therefore I plot the joint (BAH, BAχ) distribution using the Lambert-Azimuthal projection (defined in Section 6.3.2) in figure 6.3.8.

Figure 6.3.8 Native (BAH, BAχ) Distribution by Sp^2 Acceptor Type (next page): Sidechain donors by donor chemical type (left) backbone donors by acceptor DSSP(right). Donor dependent patterns visible in the 1D distributions are visiable. The aPBA/dHXL shows a sharp peak at 90°, which is consistent with a Serine motif seen in α-helices. The relative preference for the syn orbital with aCXL for salt-bridge-forming donor groups (dGDH, dGDE, dAMO, dIMD, dIME) is noteworthy. The relative lack of orientation dependence between aPBA and dAMO is also noteworthy, as it may indicate the electrostatic character of the interaction

The interactions by DSSP show striking patterns. The multimodality of the backbone-backbone interactions indicate that DSSP may not accurately cluster backbone secondary structure. Other secondary structure definitions exist, such as Stride (Frishman 1995) or KAKSI (Martin 2005) that could be encoded as features. Nonetheless, DSSP still shows orientation dependent behavior.

The distribution for dCXL in DSSP type of I: pi-Helix is unusually squarish because it is estimated from only 40 instances.


HBond Native By Don Chem Type
aCXA: n,q
aCXL: d,e
aPBA: bb
dGDE: r
dGDH: r
dAMO: k
dIMD: h
dIME: h
dIND: w
dHXL: s,t
dAHX: y
dCXA: n,q
1,010
12,117
6,292
3,478
23,432
25,195
1,290
8,585
8,637
249
1,801
3,088
275
1,822
1,630
1,772
4,042
3,404
14,379
37,621
942
6,143
5,562
3,522
7,988
26,686


HBond Native to BB Don by Acc DSSP
aCXA: n,q
aCXL: d,e
aPBA: bb
H: a−Helix
G: 3/10 Helix
I: pi−Helix
E: b−Sheet
B: b−Bridge
T: HB Turn
S: Bend
Irregular
1,292
9,605
362,073
823
2,021
32,925
40
1,329
3,261
7,284
234,804
520
769
8,961
1,759
3,579
21,774
1,068
2,269
10,070
9,967
21,538
110,685

6.3.4 Rosetta Baseline Sp^2 Acceptor Orientation Features Analysis

To assess Rosetta predictions, I compare the Top8000 set sample source (native) against the *Baseline* sample source (decoy) that applies the FastRelax protocol with Baseline energy function to each structure in the Top8000 set. The Baseline energy function is the standard Rosetta energy function, Score12 (Rohl 2004), with the several well-established corrections. It is described in detail in Section 7.2. Note the Baseline energy has the HBv1 H-bond model described in Section 4.2.2.

We observed in Section 6.3.3 that native H-bonds have strong orientation preference that depends on the acceptor hybridization type. This behavior is modulated by the acceptor and donor chemical type for H-bonds involving sidechains, and secondary structure type for H-bonds involving the backbone. In this section I assess whether the decoys recapitulate these patterns.

Before we consider the data, let us establish our expectations. The Score12 H-bond model is based on the H-bond model presented in Kortemme 2003, which has functional form based on four parameters to capture the orientation preferences of H-bonds,

$$E_{HB} = E(AHdist) + E(AHD) + E(BAH) + E(BA\chi).$$

However, the model implemented with the Rosetta Score12 energy function is based on only three parameters; it does not evaluate BAχ (see Section 2.4.3 for more details). Since the energy function is iso-energetic in and out of the acceptor plane, we should therefore not expect Score12 to adequately recapitulate the orientation preferences. The Score12 model does not distinguish H-bonds by their donor chemical type (only sidechain vs. backbone), however as is described in the previous section, the variation in BAχ distribution by donor type (Figure 6.3.6) can be partially

explained by other parts of the energy function, namely steric hindrance and several donor dependent motifs. Therefore, we may expect that the donor dependent patterns are recapitulated.

I will now show the sample plots as Figures 6.3.{3,4,5,6,8} using the Baseline sample source instead of the Native sample source.

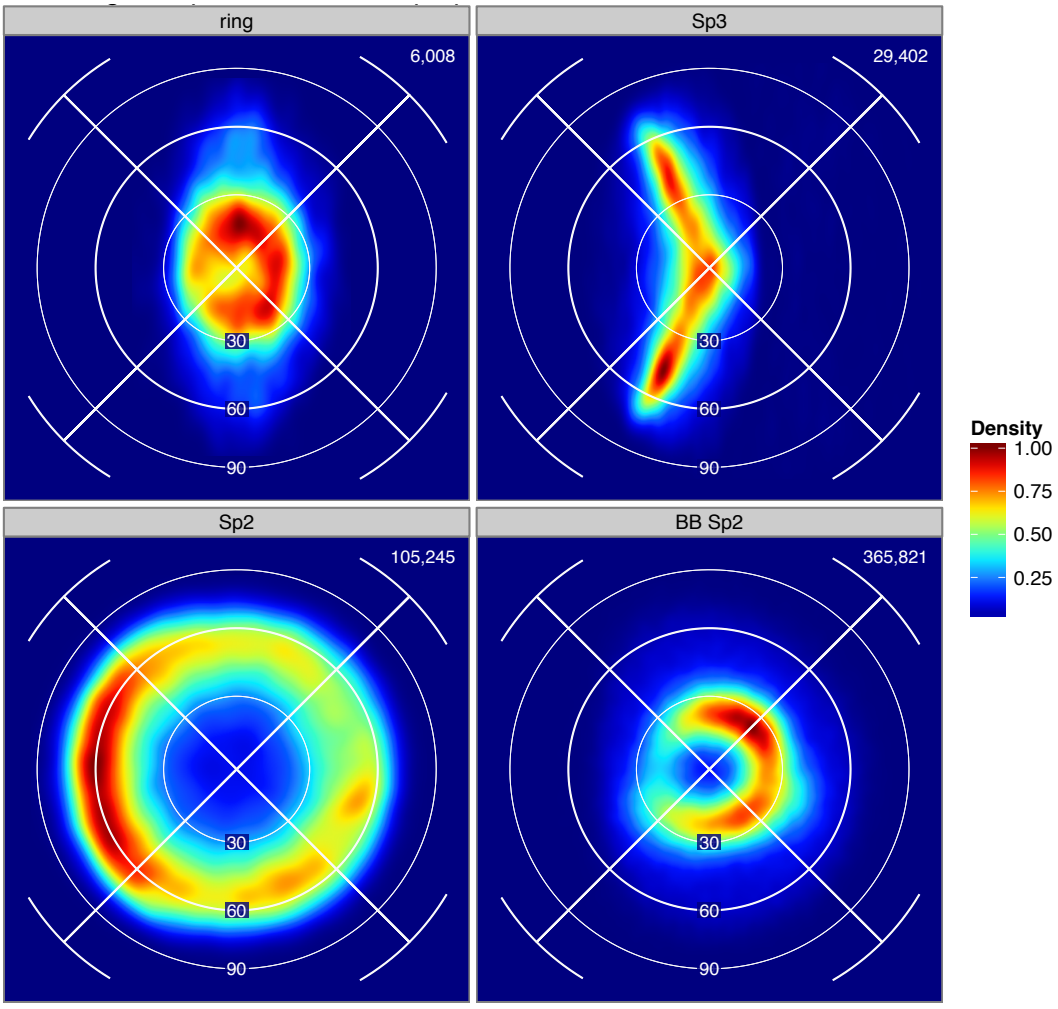


Figure 6.3.9 Baseline H-bond Orientation at the Acceptor by Acceptor Hybridization: Compare with Natives in Figure 6.3.3. (top left) for Ring acceptors, the BAH angle is less concentrated at zero. (top right) for Sp^3 acceptors, the distribution is more concentrated perpendicular to the acceptor plane (the curve towards 180°, rather than towards zero, is because the Score12 H-bond model defines the BAH angle with Base not as the adjacent heavy atom, but as the acceptor's hydroxyl hydrogen to prevent collision with the donor H-atom.) (bottom) for Sp^2 acceptors, there is markedly less preference for bonding in the plane of the acceptor, leading to a characteristic donut shape.

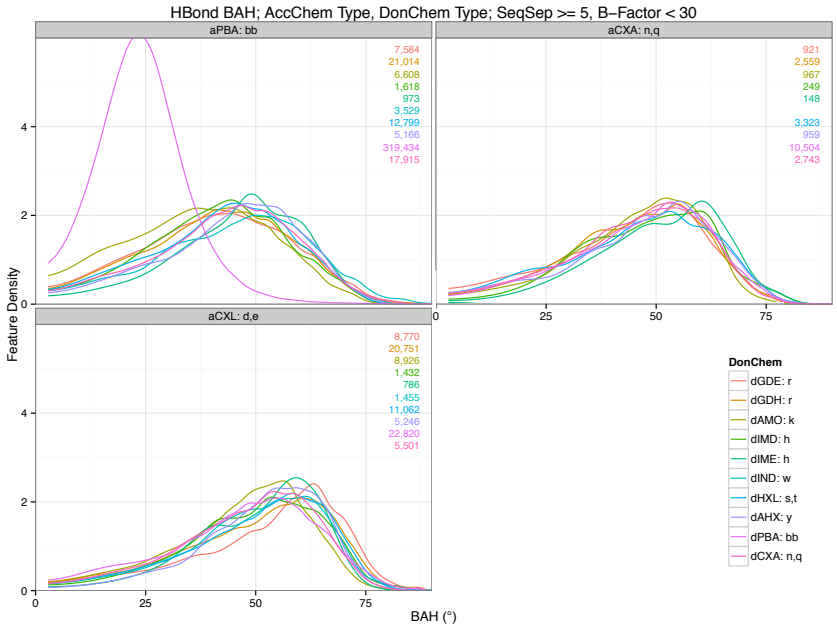


Figure 6.3.10 Baseline H-bond BAH Angle by Acceptor and Donor Type: Compare with Natives in Fig. 6.3.4. The BAH angle distribution shows little variation in donor chemical type (except for Backbone-backbone types) and the principal modes occur at 53.7°(aPBA), 48.9°(aCXA), and 51.8°(aCXL).

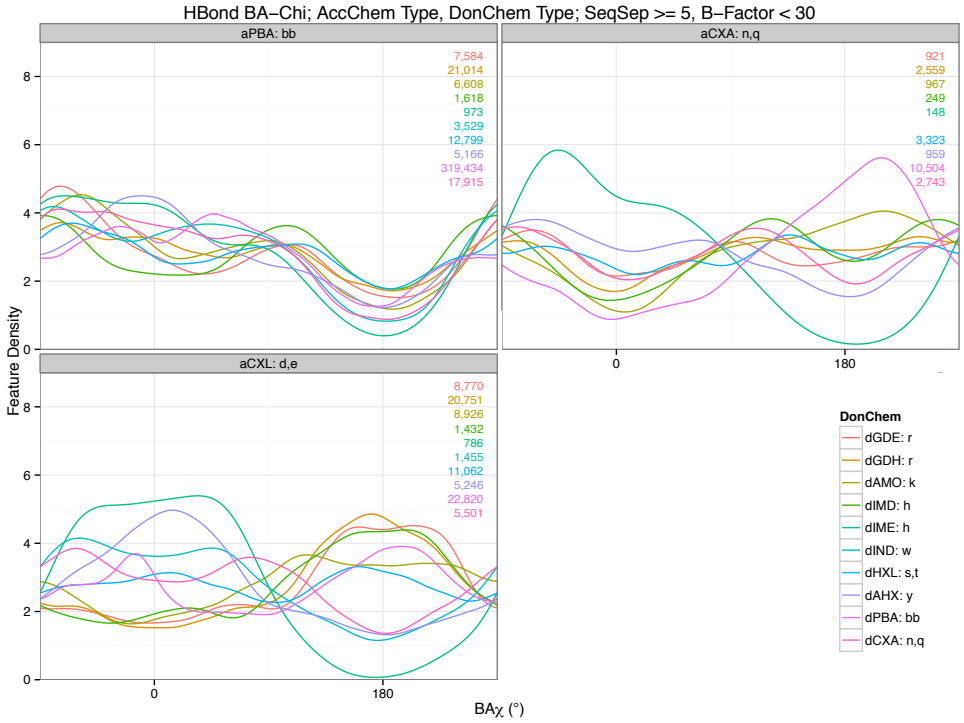


Figure 6.3.11 Baseline H-bond BAχ Angle by Acceptor and Donor Type: Compare with Natives in 6.3.5. The pronounced in-plane peaks are not as visible here. Mild preference away from BAχ=180 for backbone acceptors is visible.

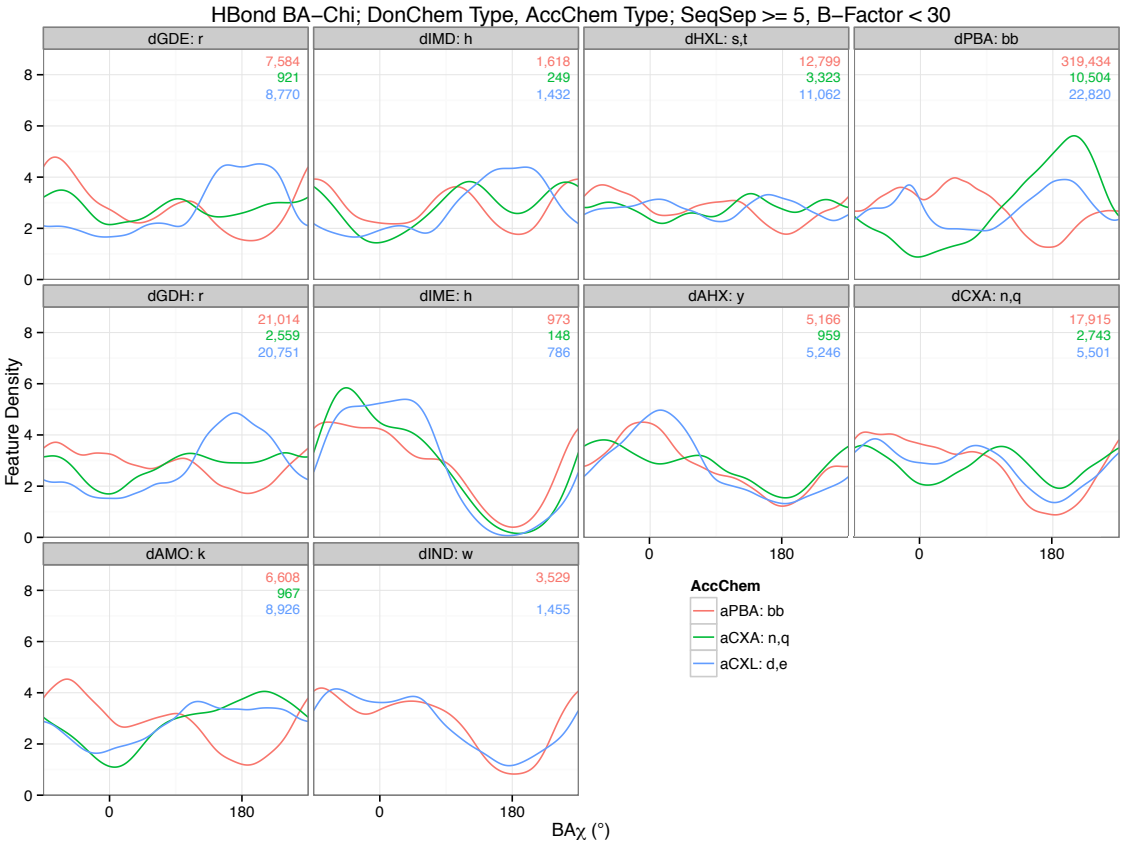
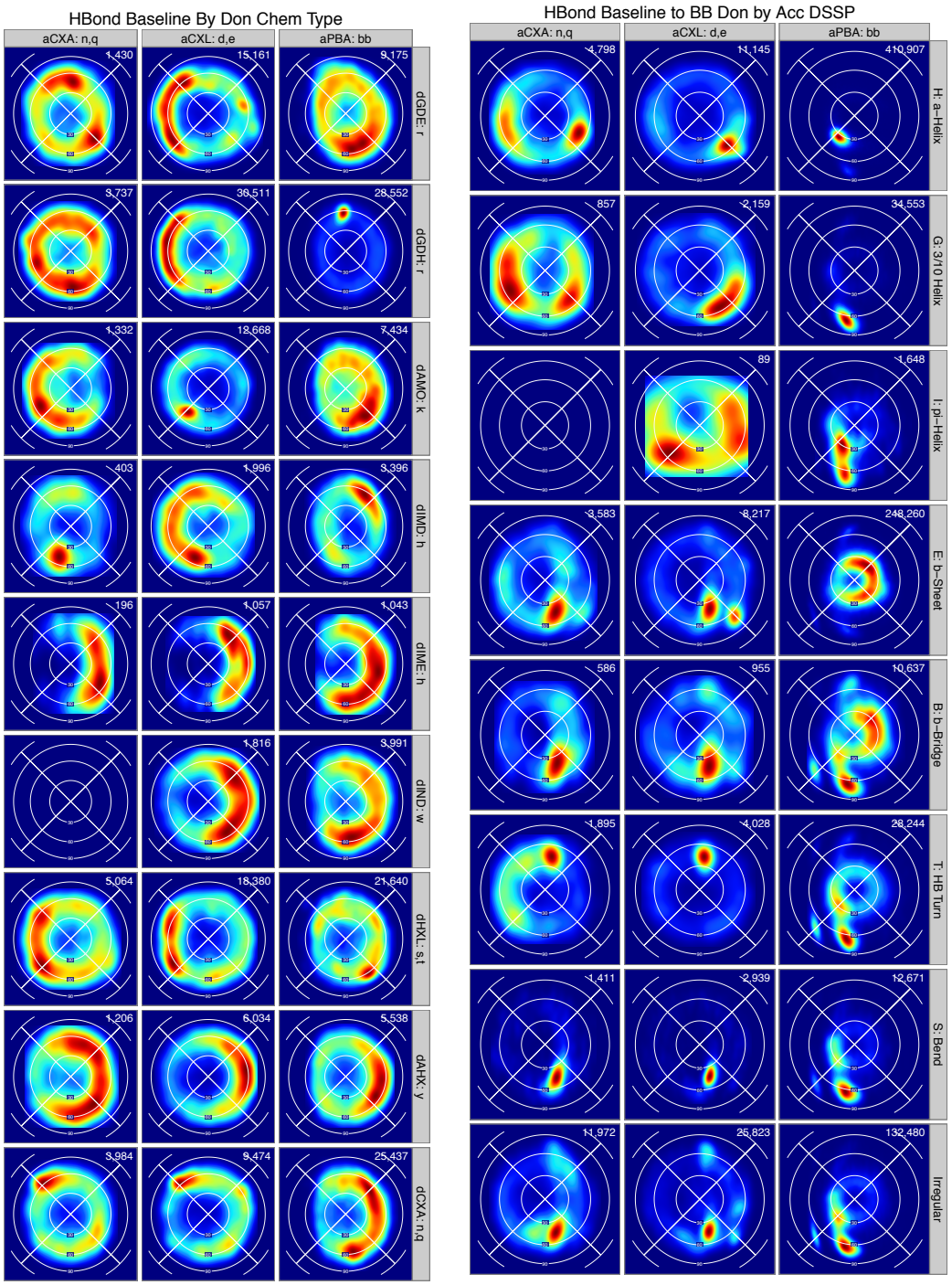


Figure 6.3.12 Baseline H-bond BAχ Angle by Donor and Acceptor Type: Campare with Natives in 6.3.6.

Figure 6.3.13 Baseline (BAH, BAχ) Distribution by Sp^2 Acceptor Type(next page): Compare with Natives in 6.3.7. (left) sidechain donors by donor chemical type (right) backbone donors by acceptor DSSP.



The Baseline distribution largely confirms our hypothesis. The Baseline distribution is largely unable to recapitulate the Native BAχ angle distribution. As an example compare the [dPBA: E b-Sheet] cell in Figures 6.3.8 and 6.3.13, an example I will return to in Figure 6.3.19.

6.3.5 Developing a new Sp^2 Potential

In this section, I present a simple, numerically stable parametric model for Sp^2 H-bonding that, incorporated with the Rosetta energy function, better recapitulates the Native distributions better than the Baseline H-bond model.

Before developing the Sp^2 function form, lets first consider how other H-bond models approach this problem. Widely used models of hydrogen bonding, such as the one in DSSP (1983), the non-bonded term in OPLS (1988), AMBER (1995), and polar contacts in and MolProbity (1999), are iso-energetic about the base-acceptor bond vector BAχ. Thus, if these models are able to recapitulate Sp^2 character, it must be through indirect interactions such as steric packing, secondary structure constraints, etc., and not through direct H-bonding interactions.

Three recent models of hydrogen bonding have included orientation dependence, but none is a numerically stable, simple parametric model. The Kortemme et al. model (2003) includes dependence on BAχ but assumes BAχ is independent of the BAH angle. This leads to an instable model as BAH approaches zero: A small positional displacement can lead to a large change in BAχ, which in turn causes a large change in the evaluated energy. This is a 2D version of the "gimbal lock" phenomena (Hoag, 1963). The Hooft et al. model (1996) considers the joint distribution of BAχ and BAH, but their energy is piece-wise constant over angle bins, which causes discontinuities at bin boundaries. The Grizhaev and Bax model for backbone-backbone hydrogen bonding (2003) considers the distribution of an amide hydrogen in the coordinate frame of the acceptor as a single 3D distribution. The model uses kernel density estimation to create a smooth function. However, this model is only for backbone-backbone interactions, which comprise most, but not all of the Sp^2 acceptor H-bonds in proteins.

Since the Baseline sample source does not recapitulate the BAχ torsion angle dependence for Sp^2 acceptor H-Bonds, we could consider simply adding a BAχ term to the Score12 model as proposed by Kortemme et al. (2003). However, this again results in an instability as BAH approaches zero. Instead, I replace the BAH term with an analytic, smooth, joint function of (BAH, BAχ).

I observed that for Sp^2 acceptor H-Bonds, many of the dependencies on chemical types could be explained by specific binding motifs, so I consider a simple functional form that favors the ideal lone pair directions for all chemical types. The functional form consists of a cosine function of BAχ that interpolates between two functions of BAH (Function: H), one, in the plane of the Sp^2 group (Function: F), has a minimum at 30° degrees (the ideal locations of the lone pairs) and a local maximum at 0°, while the other (Function: G), which is perpendicular to the plane of the Sp^2 group, has a single minimum at 0 (Figure 6.3.14). Overall the functional form has three parameters (Shown as blue bars): the value at (BAH=30°, BAχ=0°), the value at (BAH=0°), and the outer width of the BAH potentials.

$$F = \begin{cases} \frac{1}{2}\left(d\cos\left(3(\pi - \mathrm{BAH})\right) + d - 1\right) & \mathrm{BAH} > \frac{2\pi}{3} \\ \frac{1}{2}\left(m\cos\left(\pi - \frac{1}{\ell}\left(\frac{2\pi}{3} - \mathrm{BAH}\right)\right) + m - 1\right) & \frac{2\pi}{3} \geq \mathrm{BAH} > \pi\left(\frac{2}{3} - \ell\right) \\ m - \frac{1}{2} & \pi\left(\frac{2}{3} - \ell\right) \geq \mathrm{BAH} \end{cases}$$

$$G = \begin{cases} d - \frac{1}{2} & \mathrm{BAH} > \frac{2\pi}{3} \\ \frac{m-d}{2}\cos\left(\pi - \frac{1}{\ell}\left(\frac{2\pi}{3} - \mathrm{BAH}\right)\right) + \frac{m-d+1}{2} + d & \frac{2\pi}{3} \geq \mathrm{BAH} > \pi\left(\frac{2}{3} - \ell\right) \\ m - \frac{1}{2} & \pi\left(\frac{2}{3} - \ell\right) \geq \mathrm{BAH} \end{cases}$$

$$H = \frac{\cos(2\mathrm{BA}\chi) + 1}{2}$$

$$E = HF + (1 - H)G$$

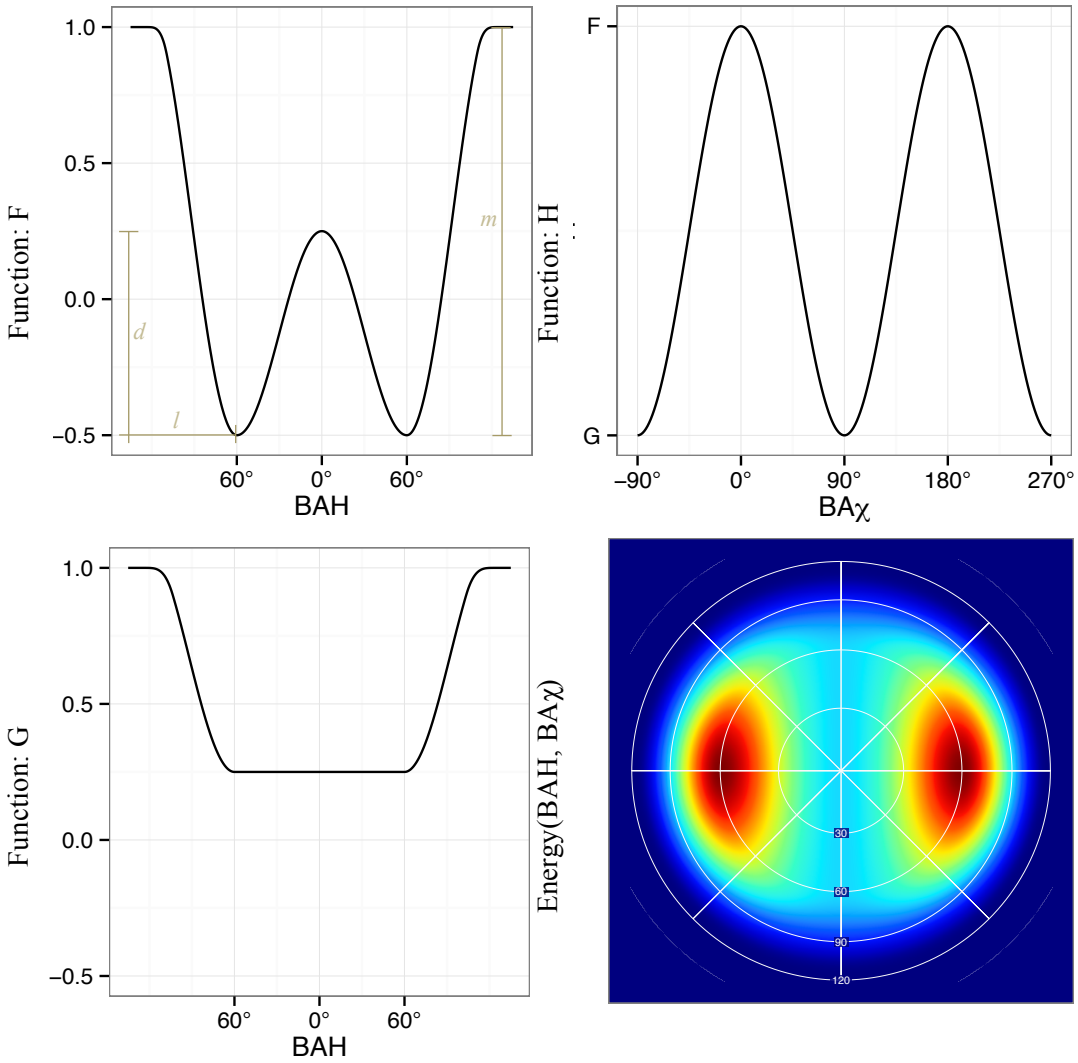


Figure 6.3.14 The Sp^2 Model: Energy(BAH, BAχ) for Sp^2 Acceptors (upper left) The F function over the BAH angle when BAχ is in the plane. The parameters for the Sp^2 model are ℓ, d, and m, show as blue bars (lower left) The G function over the BAH angle when BAχ is out of the plane. (upper right) The H function over the BAχ angle interpolating between the F and G functions. (lower right) The Energy(BAH, BAχ) function over the (BAH, BAχ) angles shown as an area-preserving Lambert-Azimuthal map projection.

6.3.6 HBondSp2 Sp^2 Acceptor Orientation Features Analysis

To assess how the new Sp^2 functional form impacts the feature distributions I plot the same set of figures as for Native and Baseline but with the *HBondSp2* sample source.

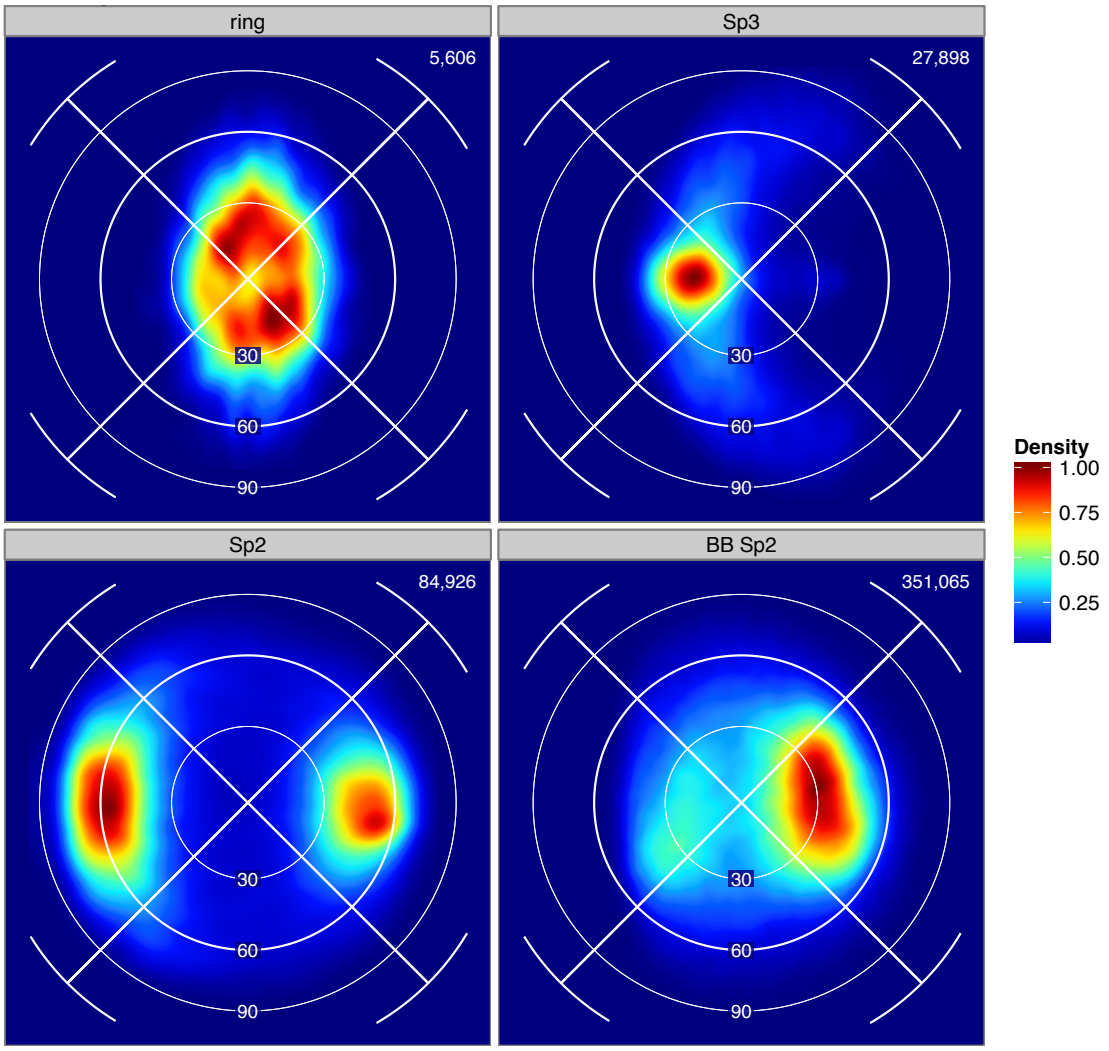


Figure 6.3.15 HBondSp2 H-bond Orientation at the Acceptor by Acceptor Hybridization: Compare with Natives in Figure 6.3.3 and Baseline in Figure 6.3.9. (top left) for Ring acceptors, the BAH angle is less concentrated at than the Native distribution, similar to the Baseline distribution. (top right) for Sp^3 acceptors, the distribution is tightly concentrated with BAχ about 180° and BAH about 15°. See Figure 6.5 (bottom) for Sp^2 acceptors, the distribution is concentrated in two peaks.

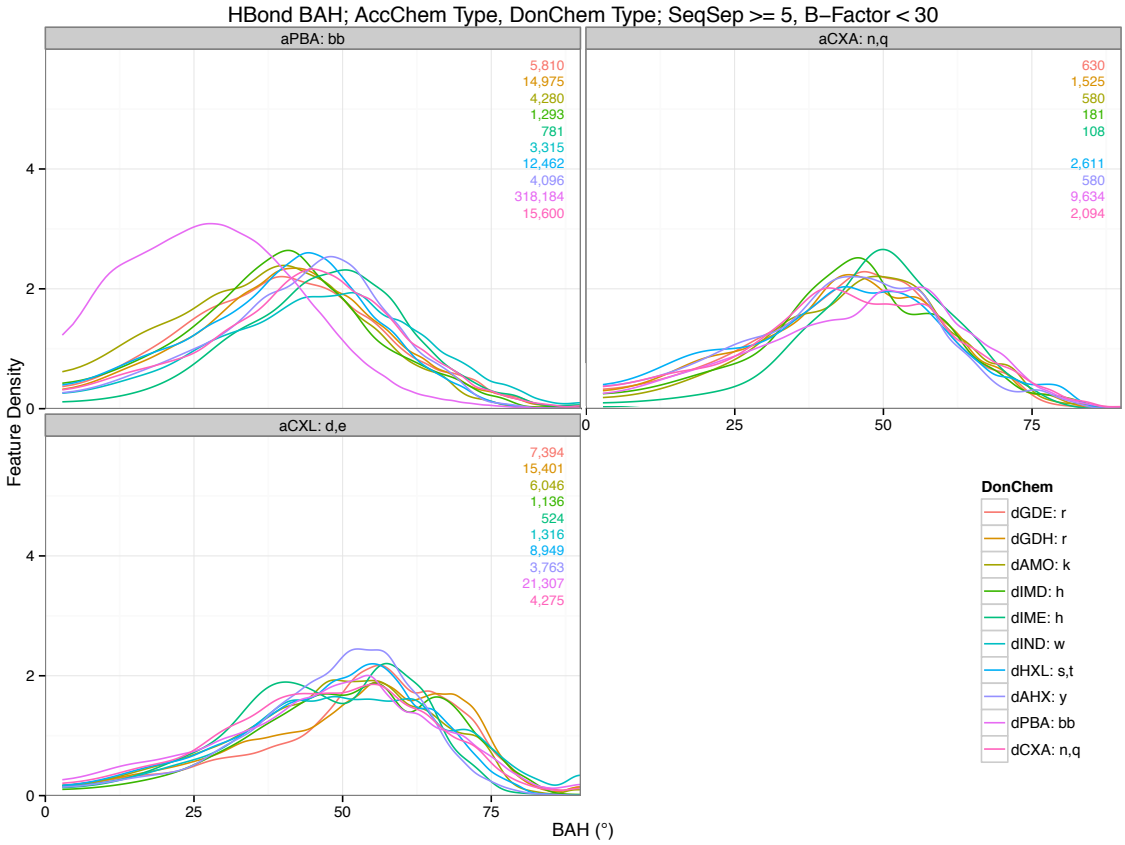


Figure 6.3.16 HBondSp2 H-bond BAH Angle by Acceptor and Donor Type: Compare with Natives in Figure 6.3.4 and Baseline in Figure 6.3.16

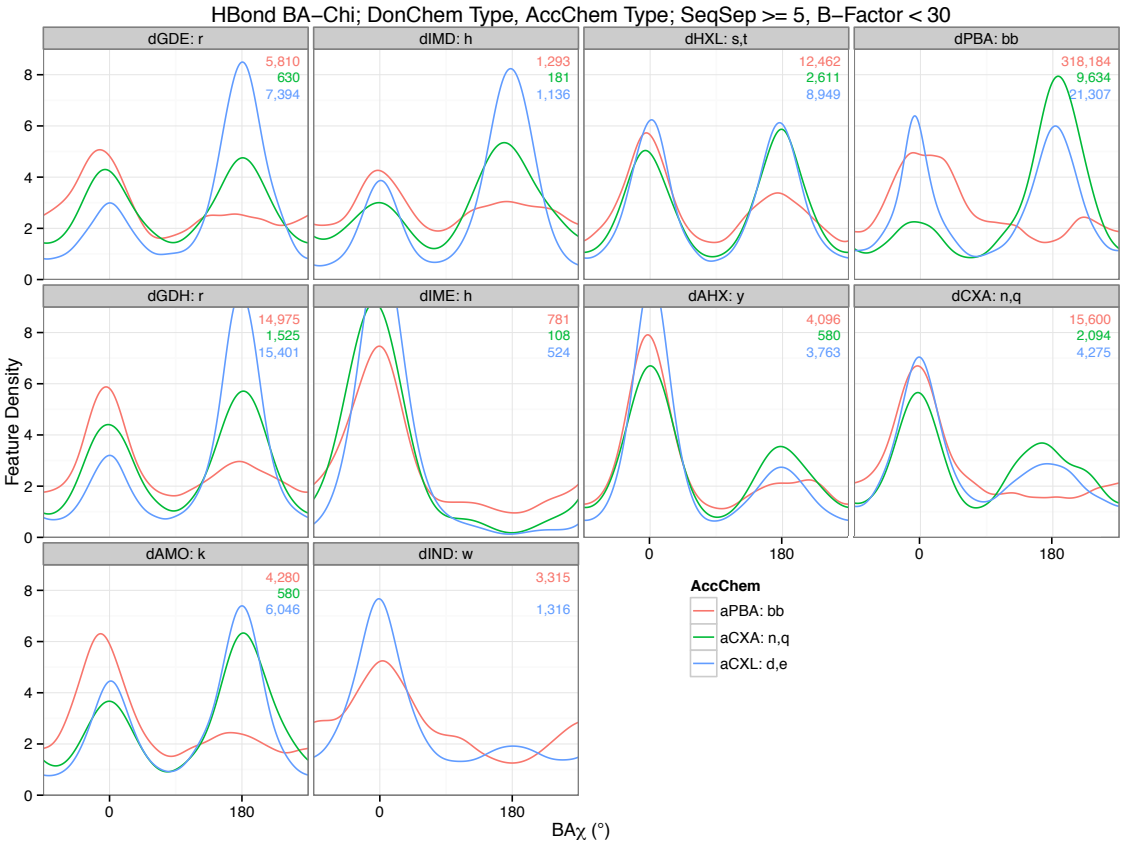
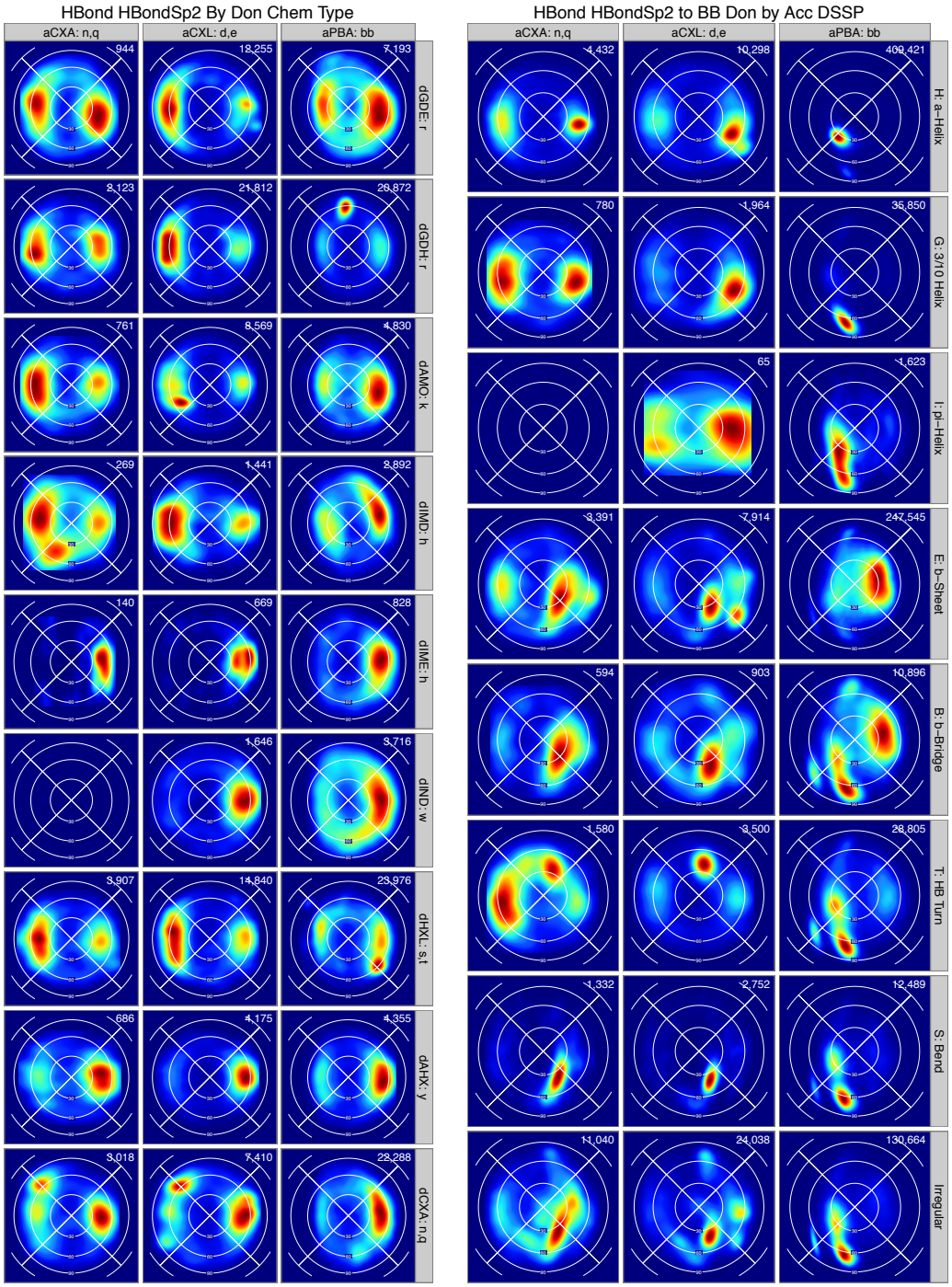


Figure 6.3.17 HBondSp2 H-bond BAχ Angle by Donor and Acceptor Type: Compare with Natives in 6.3.5. and Baseline in Figure 6.3.12.

Figure 6.3.18 HBondSp2 (BAH, BAχ) Distribution by Sp^2 Acceptor Type(next page): Compare with Natives in 6.3.7 and Baseline in Figure 6.3.13. (left) sidechain donors by donor chemical type (right) backbone donors by acceptor DSSP.



These plots show that the HBondSp2 potential has significantly more density in the plane of the acceptor, which corrects the central failure of the Baseline (BAH, BA χ) feature distribution. Additionally the diverse pattern of feature distributions can be recapitulated with this simple

potential, including donor dependent and secondary structure dependent patterns (e.g. seen in Figure 6.3.18).

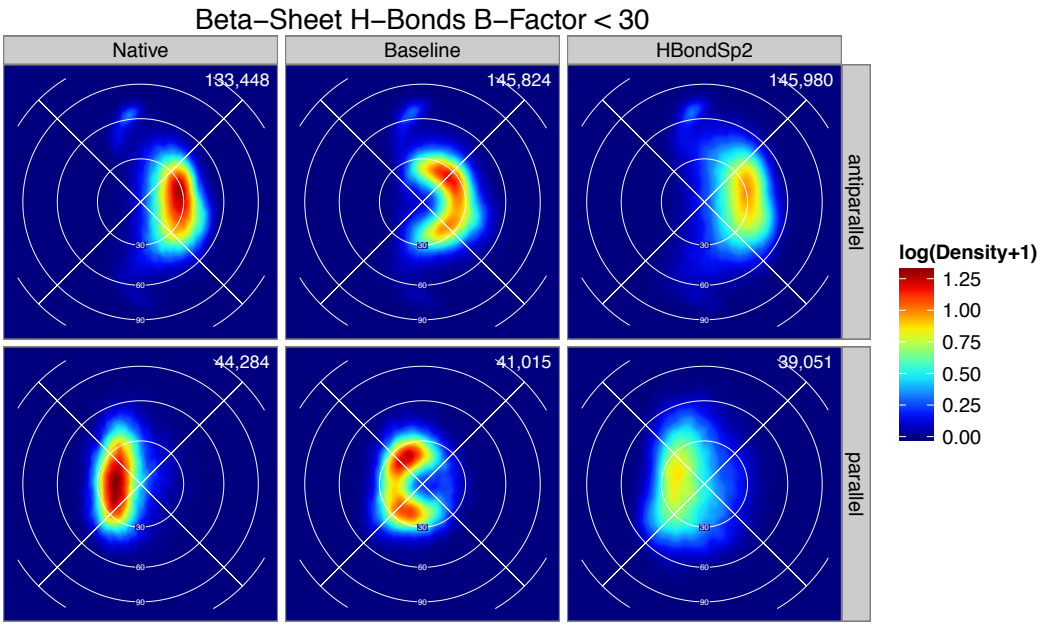


Figure 6.3.19 H-Bond Acceptor Orientation in β-sheets by Sample Source and Strand Orientation: Lambert Azimuthal projection of (BAH, BAχ) showing log(density+1) for (top) anti-parallel and (bottom) parallel β-sheets from (left) Native, (center) Baseline, and (right) HBondSp2 sample sources.

This simple potential also better recapitulates β-sheet shear: As seen in Figure 6.3.20, parallel and anti-parallel β-sheet H-Bonds populate the N-term (syn) and C-term (anti) orbitals, respectively, consistent with what we observe in Natives. In anti-parallel β-sheets, the close contact of the Cα-bound H-atom to the backbone carbonyl oxygen is a measure of shear, which has caused some to postulate carbon H-bond effects. I observe that Score12 does not recapitulate native shear, even with an electrostatics potential; one must include an explicit carbon H-bond term to make Score12 recapitulate shear. Surprisingly, the Sp2 potential recapitulates native shear, with no need for an explicit carbon H-bond term. Anisotropy at each carbonyl is enough to drive the β-sheet shear.

The Score12 H-bond model uses separate parameters for backbone-backbone H-bonds with sequence separation less than 5, as a proxy for helical secondary structure. I observe that, although Natives show clear patterns of (BAH, BAχ) feature distributions conditional on secondary structure (as defined by DSSP), Decoys generated with the new H-bond Sp^2 potential largely recapitulate these patterns without needing this sequence separation dependence.

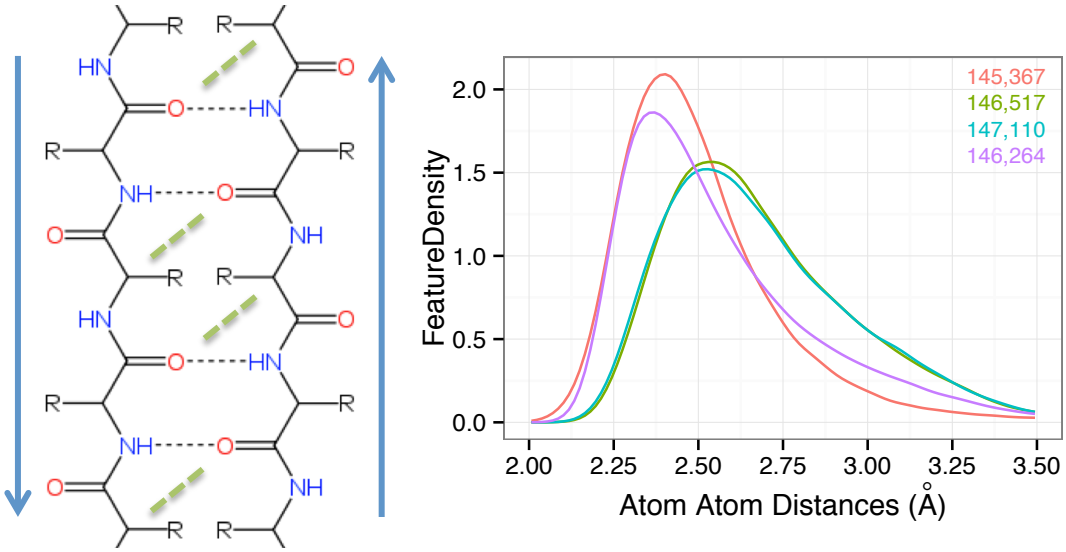


Figure 6.3.20 Anti-Parallel β-sheet close contact (Hα,O) distance: Thick dashes at left. Sample sources at right: Natives (red), Baseline (green), Score12 + explicit electrostatic Coulomb term (green), HBondSp2 (purple). The HBondSp2 model recapitulates Hα,O distance without an explicit carbon hydrogen bond model.

6.4 Parameter fitting

In this subsection, I demonstrate using the features analysis tool to fit parameters of a feature model. I refine the chemical types in the Kortemme et al. (2003) HBond model to increase the specificity of the features analysis. I iteratively fit the AHdist and AHD angle conditional on the donor and acceptor chemical type to recapitulate Native feature distributions.

6.4.1 Chemical types and Geometric Dimension Features

The Score12 H-Bond model is conditioned upon only a few chemical types; acceptor hybridization (Sp2/Sp3/Ring), backbone vs. sidechain for both the donor and acceptor, and short-range/long-range backbone-backbone interactions defined by sequence separation. With the increasing availability of experimental data, I can distinguish patterns of feature distributions with greater refinement. By basing chemical types on the functional group of the donor and acceptor, even though I investigate hydrogen bonds primarily in proteins, my scheme can be generalized to non-protein macromolecules, including RNA, DNA, and small molecules. To visualize patterns based on chemical type, I use small multiple plots, where the rows and columns show distribution variation by donor or acceptor chemical type.

I first refine the chemical types based on the functional group into 7 acceptor types and 10 donor types (Figure 6.4.1). I label the geometric dimensions by the atoms in the donor acceptor groups used to define them.

To be consistent with the Kortemme 2003 H-bond model, I identify the donor group by the coordinates of the (H)-atom and the (D)onor atom. I identify the acceptor groups by the coordinates of the (A)cceptor atom, the acceptor (B)ase, and the acceptor (BB)ase atom, whose definition depends on the acceptor hybridization type. The parent of an atom is the first atom on the shortest bond-path to the residue root atom, which, for canonical amino acids, is the C-α atom. For Sp^2 groups (PBA, CXA, CXL), the Base is the Acceptor parent and the BBase is the Base parent. For Ring groups (IMD, IME) and Sp^3 groups (HXL, AHX), the Base atom is the average of the atoms bound to the Acceptor and the BBase is the parent of the Acceptor. In Score12, for Sp^3 groups, the Base atom is the hydroxyl hydrogen and in Talaris2013, the Base is the Acceptor parent and the BBase is the hydroxyl hydrogen.

I identify the H-Bond geometric features by the atoms used to define them, where lengths are defined by 2 atoms (AHdist, ADdist), exterior bond angles are defined by 3 atoms (BAH, AHD) and torsion angles are defined by 4 atoms but identified by the atoms defining the torsional axis, (BAχ, AHχ). Additionally, for Sp^2 and Sp^3 acceptor groups, I measure the Quarter BA-χ (QBA-χ), the angle of the hydrogen out of the plane defined by the (BBase, Base, Acceptor).

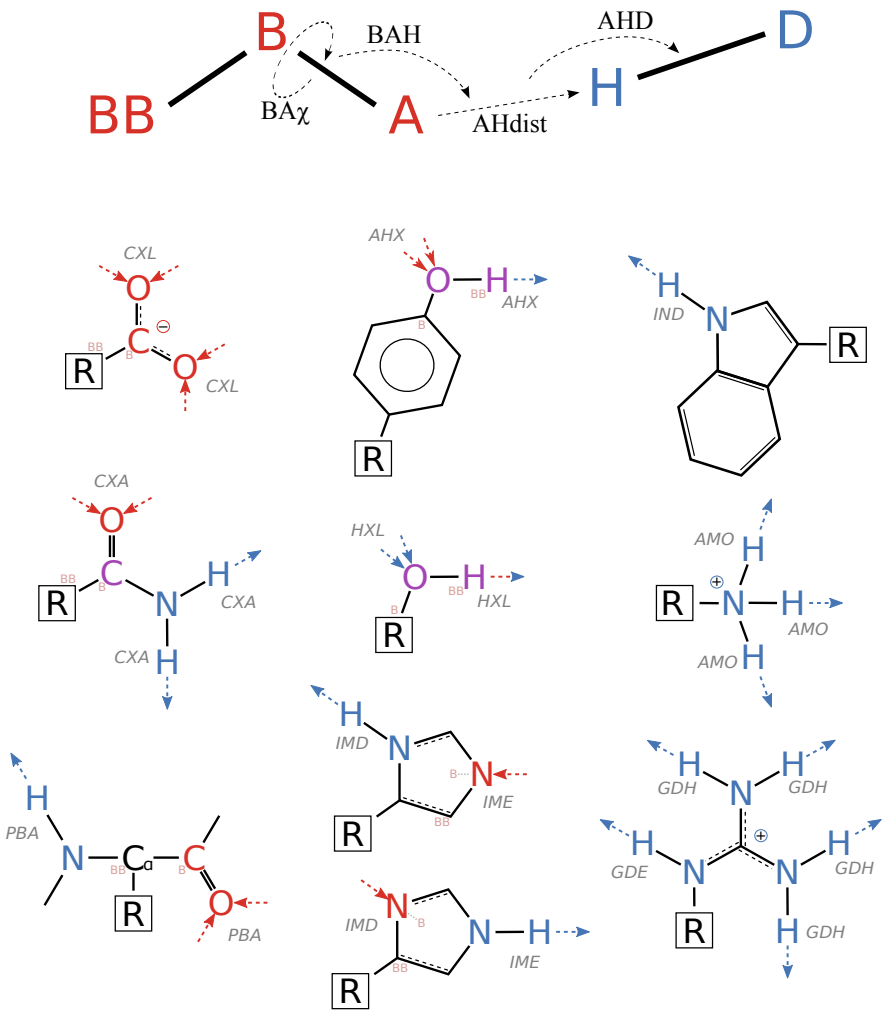


Figure 6.4.1 HBondSp2 Model Geometric Features and Chemical Types: (top) The H-Bond model evaluates 4 geometric features defined on the **BB**ase, **B**ase, and **A**cceptoratoms of the acceptor group (whose definition depends on the hybridization type of the acceptor, and are labeled for each chemical type), and the **H**ydrogen and **D**onor atoms of the donor group: The A-H distance (AHdist), the B-A-H angle (BAH), the BB-B-A-H torsion angle (BA-χ), and the A-H-D angle (AHD). An H-bond in the HBondSp2 model is defined when the computed energy is less than zero. (bottom) The H-bond sites in canonical amino acids. Blue outward facing arrows are positively charged donor groups and inward facing red arrows are negative charged acceptors. The BBase and Base atoms are labaled for each acceptor group.

6.4.2 Parameterization of AHdist and AHD components

Using the features analysis tool, I can observe that, for AHdist and AHD features, the Baseline HBond model does not recapitulate the native distributions (Figure 6.4.2-3).

To visualize the AHdist feature distributions, I use kernel density estimation, normalizing by $1/x^2$ to account for volumetric effects using the `weight_fun= radial_3d_normalization` option to `estimate_density_1d` function (See Chapter 5 for a discussion of normalization for density estimation).

To visualize the AHD feature distribution, I use cumulative distribution functions. The H-atom in an H-bond predominantly lies between the donor and acceptor heavy atoms, making the AHD angle cluster near zero. To use kernel density estimation would require treatment of the boundary behavior or a change of variables. Instead, viewing the cumulative distribution function for the AHD angle makes it easy to measure deviation from linear.

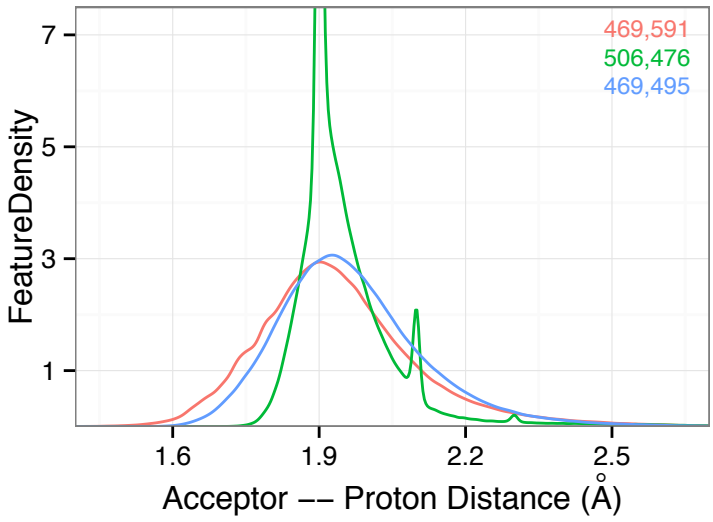


Figure 6.4.2 H-bond AHdist by Sample Source: Kernel density estimation for Natives (Red), Baseline (green), and HBondSp2 (blue) the peakiness of the Baseline is discussed in Section 4.2.2.

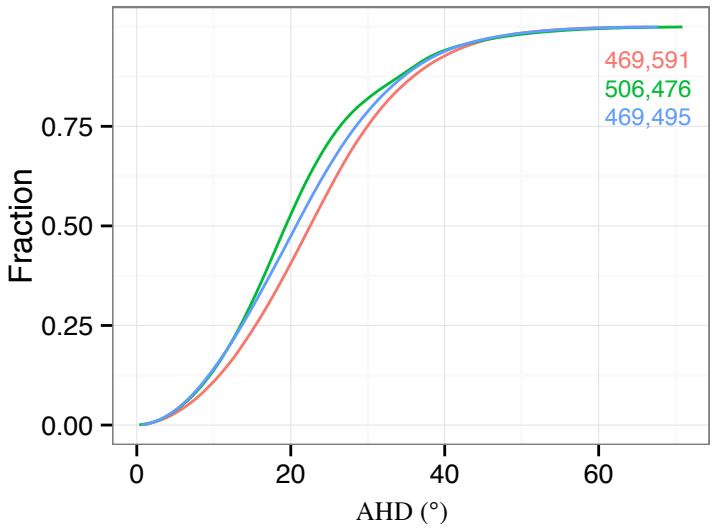


Figure 6.4.3 H-bond AHD feature by Sample Source: Emperical cumulative distribution function for Natives (Red), Baseline (green), and HBondSp2 (blue).

Using the refined chemical types, I am able to show that there is variation in the AHdist feature depending on the donor and acceptor chemical type (Figure 6.4.4-5). The donor type has more variation than the acceptor type. The Score12 model does not evaluate different potentials based on donor chemical type; therefore, Score12 may not recapitulate this variation.

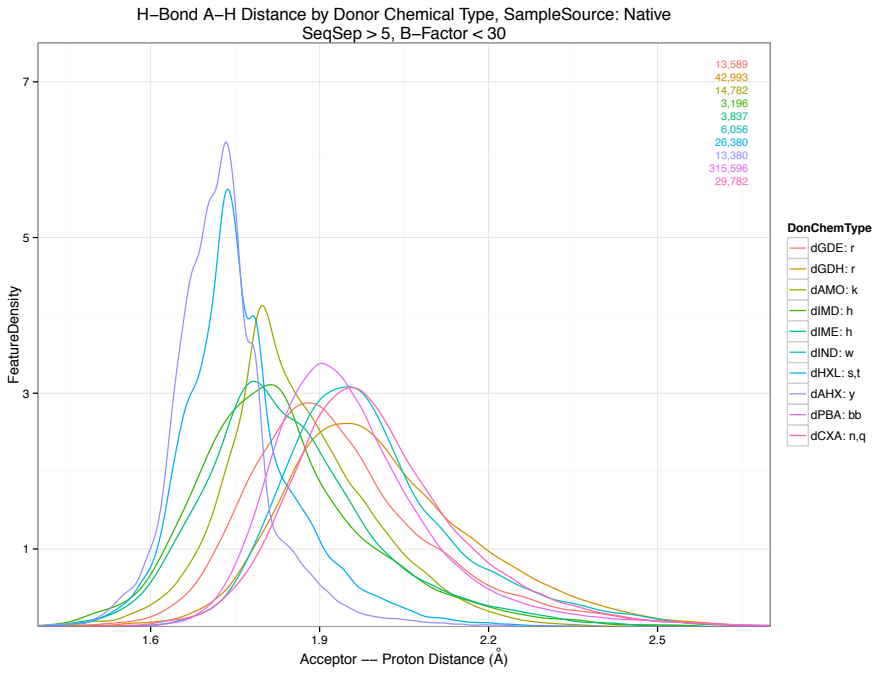


Figure 6.4.4 Native H-bond AHdist Feature by Donor Chemical Type: The dHXL (hydroxyl) and aAHX (aromatic hydroxyl) are the two tight distirbutions on the left and modeling them is discussed in 6.5.1.

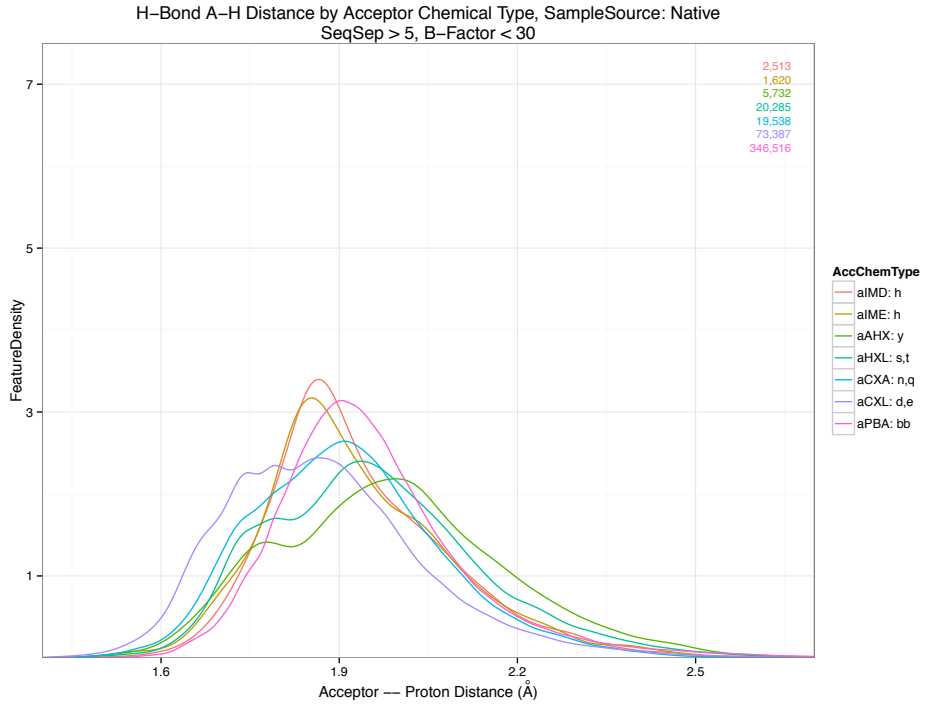


Figure 6.4.5 Native H-bond ADdist Feature by Donor Chemical Type: The H-atom is typically unobserved in X-ray crystallography experiments, checking the dependence on donor chemical type shows the dependence is not an artifact of the H-atom placement.

The functional form of the AHdist term in the Baseline HBond model is a 10-degree polynomial with a single minimum that achieves a value of -0.5. The high degree of polynomial is necessary to achieve a derivative of zero at AHD=0° and tight angular distribution. Using Horner's rule, the value and derivative of a polynomial can be evaluated efficiently with a small, constant number of additions and multiplications. This leads to computationally efficient evaluation in the energy function, which is necessary for prediction protocols that explore conformation space through repeated evaluation of the energy function.

For each combination of donor and acceptor chemical type, I iteratively adjust the polynomial to recapitulate the native distribution. To do this, I constrain the polynomial using Lagrangian multipliers, set control points, interpolate the polynomials of different degrees, and select the best fit.

Incorporation of the new polynomials in the HBondSp2 model gives better recapitulation of the native AHdist feature for each donor and acceptor chemical type (Figures 6.4.6-8).

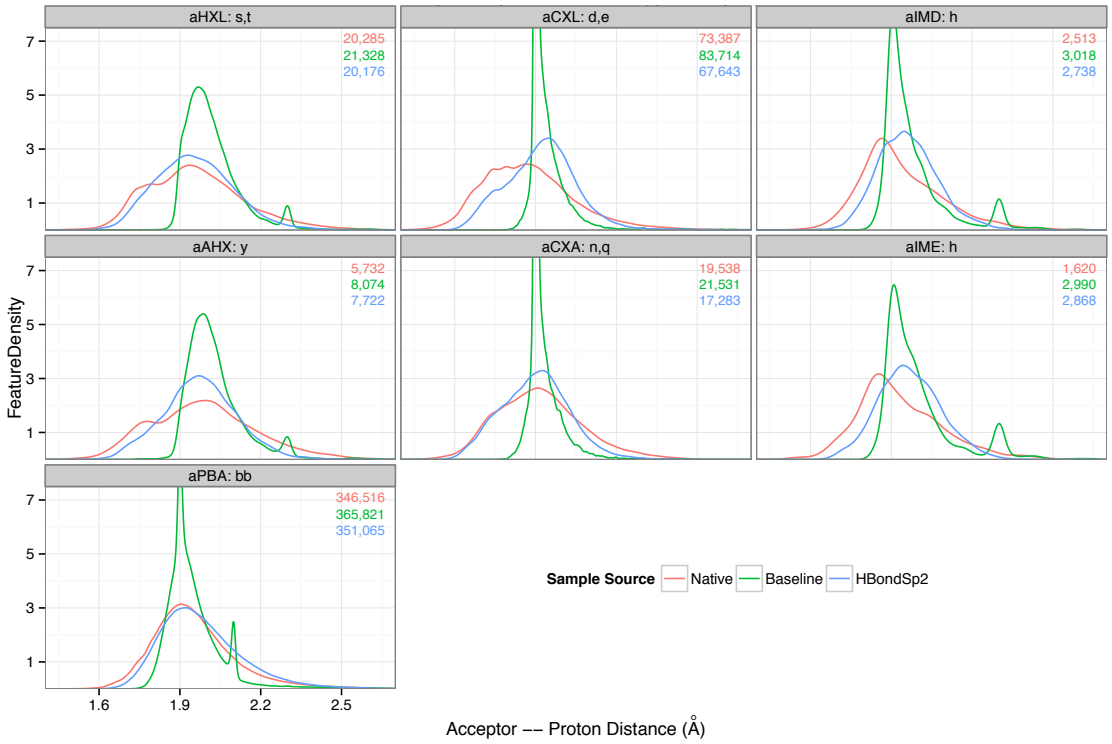


Figure 6.4.6 H-bond AHdist Feature by Acceptor Chemical Type and Sample Source: Natives (Red), Baseline (green), and HBondSp2 (blue).

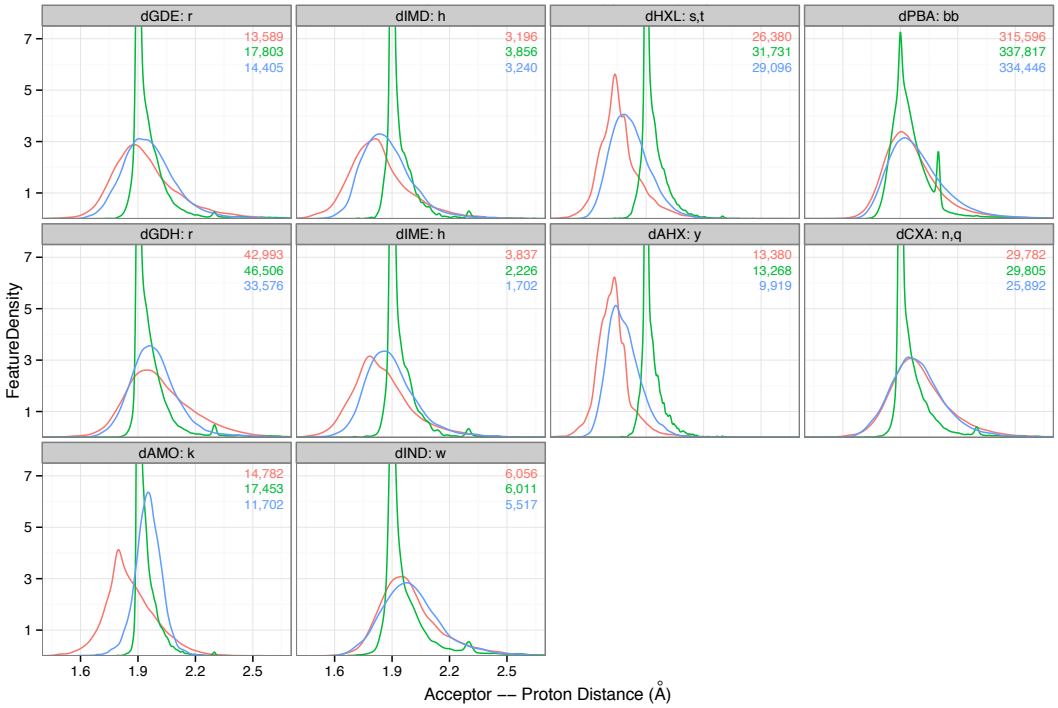


Figure 6.4.7 H-bond AHdist Feature by Donor Chemical Type and Sample Source: Natives (Red), Baseline (green), and HBondSp2 (blue).

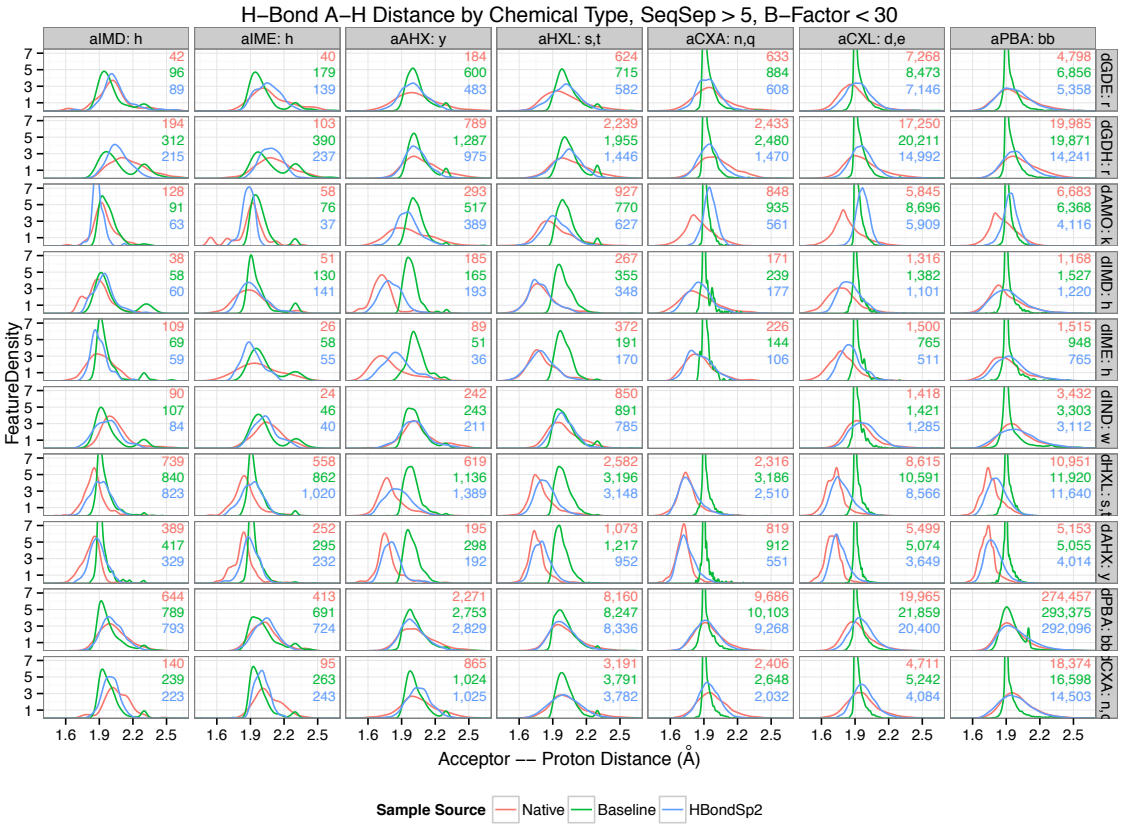


Figure 6.4.8 H-bond AHdist Feature by Donor / Acceptor Chemical Type and Sample Source: Natives (Red), Baseline (green), and HBondSp2 (blue).

Since the placement of the hydrogen in X-ray crystal structures is a judgment call, either by a crystallographer or a program, and not supported by data, I check the ADdist feature, which is the distance between the acceptor heavy atom and the donor heavy atom (not the donor hydrogen) (Figure 6.4.9).

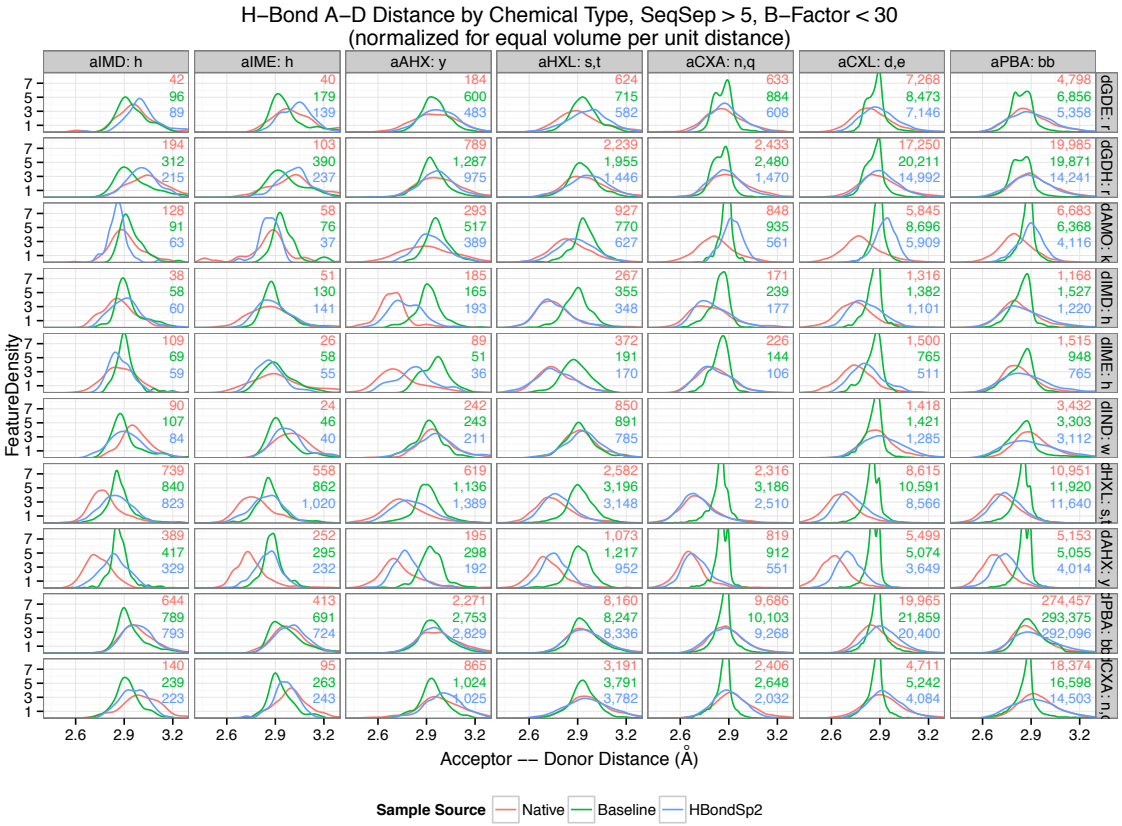


Figure 6.4.9 H-bond ADdist Feature by Donor / Acceptor Chemical Type and Sample Source: Natives (Red), Baseline (green), and HBondSp2 (blue).

These figures show with adjusting the parameters the majority of the decoy AHdist feature distributions can recapitulate the Native distribution. Interestingly some of the dAMO (lysine; amino group) to Sp^2 acceptors are not recapitulated. The amino group is negatively charged, so rather than strongly adjusting the potentials, I consider combining the H-bond term with an electrostatics potential in Section 6.6.

I now turn to the AHD angle distribution conditional on donor and acceptor chemical types in following plots (Figures 6.4.10-11). As I discuss in 6.4.2, I use empirical cumulative distribution functions to compare conditional distributions. I begin with gauging the level of variation natives by chemical type.

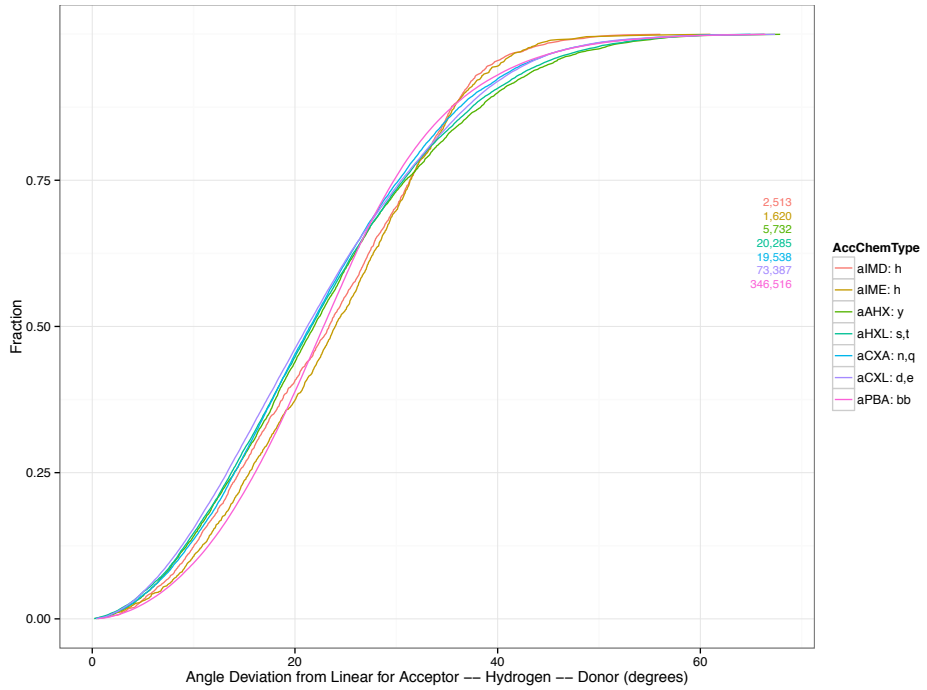


Figure 6.4.10 Native H-bond AHD Feature by Acceptor Chemical Type: There is less variation by acceptor chemical type than by donor chemical type shown in Figure 6.4.11.

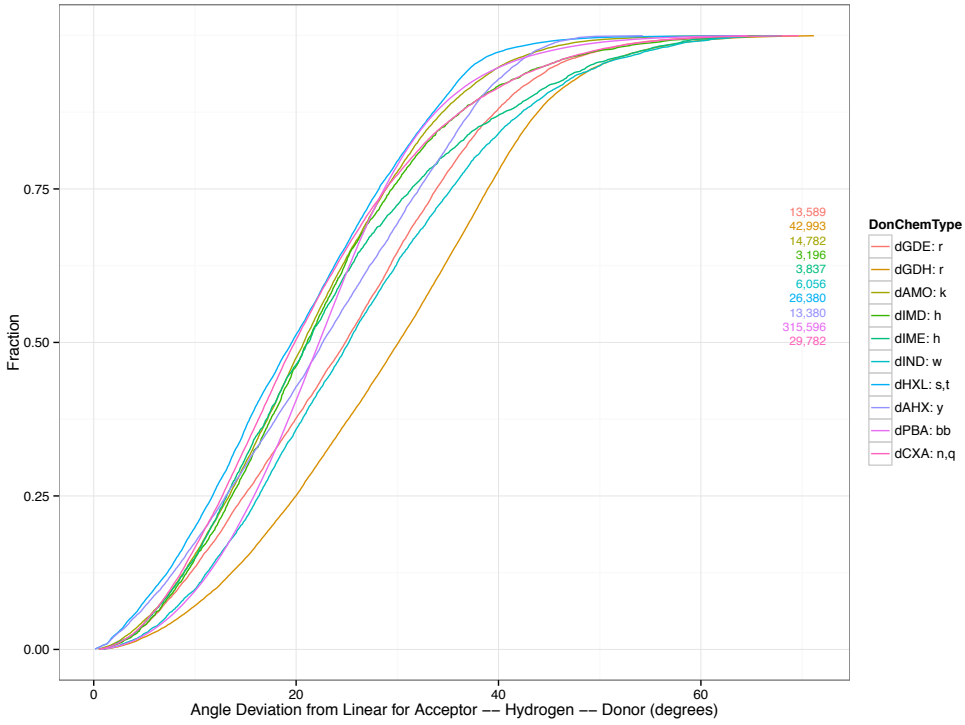


Figure 6.4.11 Native H-bond AHdist Feature by donor Chemical Type: The three widest distributions dIND (indol), dGDH and dGDE (guanidinium) occur on bulky inflexible sidechains indicating this may be caused the challenge of aligning these groups to have narrow AHD angles (See Figure 6.4.13 for another perspective).

In Score12, the AHD angle term is a polynomial function of the cosine of the AHD angle, since the cosine of the angle is easily computed as the dot product of the length-normalized vectors. The cosine is also suggested by the volume normalization for azimuthal angles in spherical coordinates. However, we already know that we are not dealing with evenly spaced points on the sphere, but concerned with the variation of angle around zero. Since the derivative of the cosine function at zero is zero, it compresses the signal, and a high degree polynomial is needed to expand the signal again. This increases the computational cost and difficulty of fitting the entire distribution.

To fit the AHD angle term in the HBondSp2 model, I fit a polynomial to the AHD angle directly. This allows for a lower degree polynomial. To make the potential smooth at the AHD=0 pole, I use a Lagrangian constraint on the derivative to be zero at the pole. This process is time consuming and requires hand editing of the parameters. Even after multiple rounds of fitting, the distributions of AHD angles in the HBondSp2 model do not completely capture the distributions observed in the natives, especially for the acceptor types (Figure 6.4.12-14). Future directions include automating the fitting process to achieve better fit.

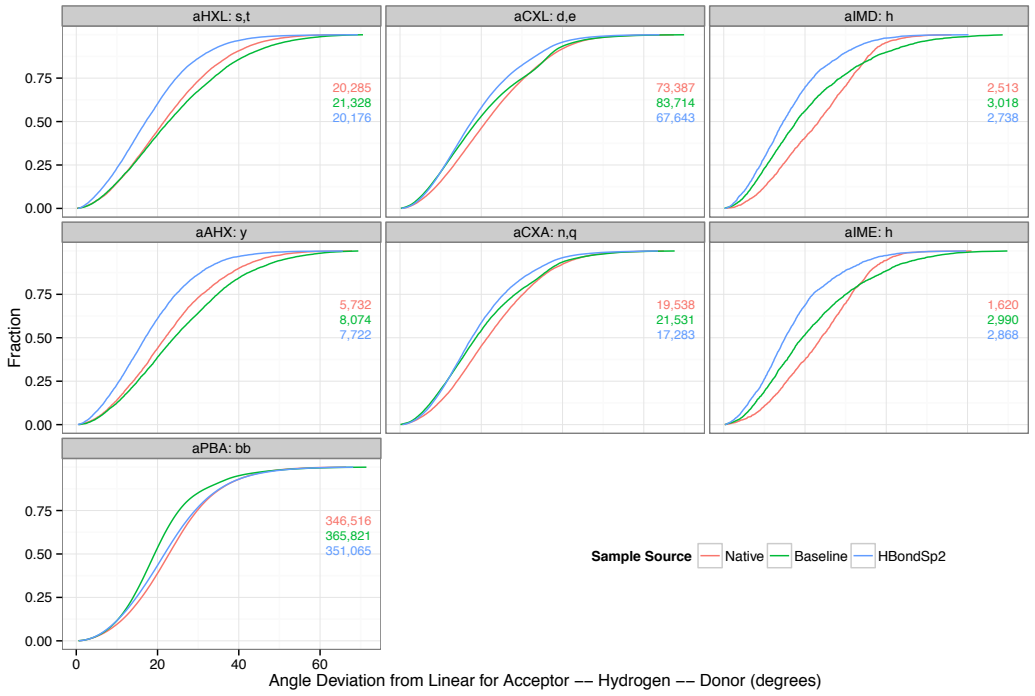


Figure 6.4.12 H-bond AHD Feature by Acceptor Chemical Type and Sample Source: Natives (Red), Baseline (green), and HBondSp2 (blue).

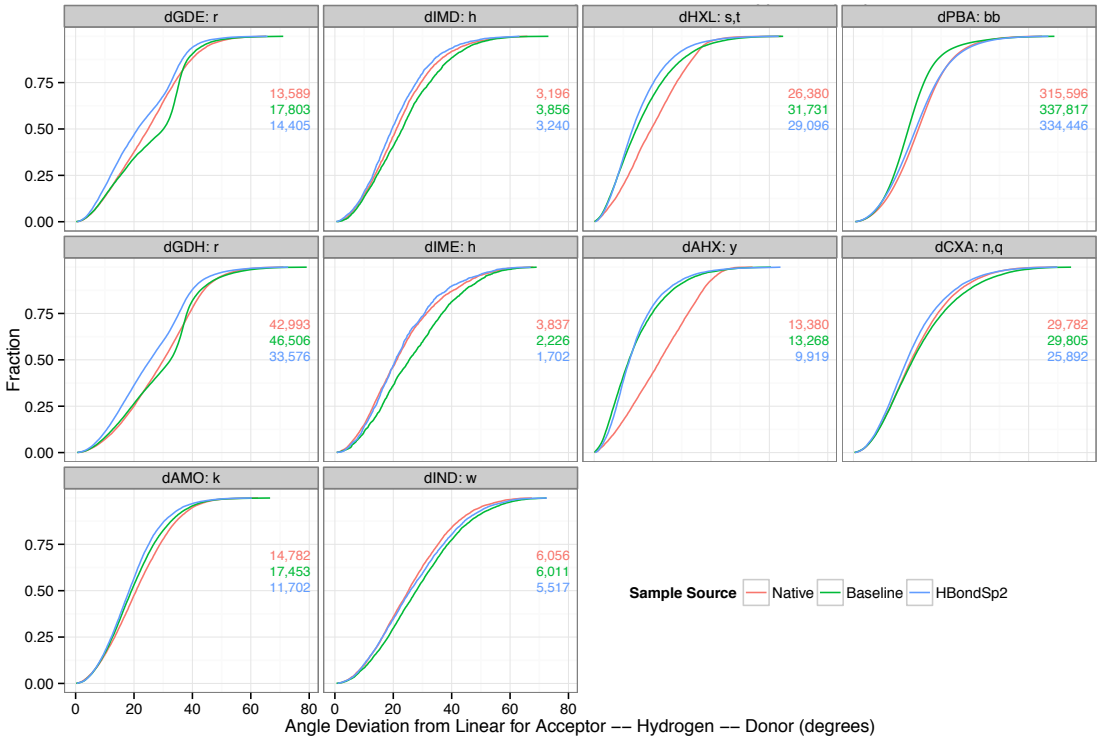


Figure 6.4.13 H-bond AHD Feature by Donor Chemical Type and Sample Source: Natives (Red), Baseline (green), and HBondSp2 (blue). The hydroxyl donor groups (HXL, AHX) are able to rotate the position of the hydrogen around the hydroxyl ogxygen, this should allow greater freedom make the angle tighter, however, in the Natives with Reduce placed hydrogens the mean angle is wider, indicating this deserves further investigation.

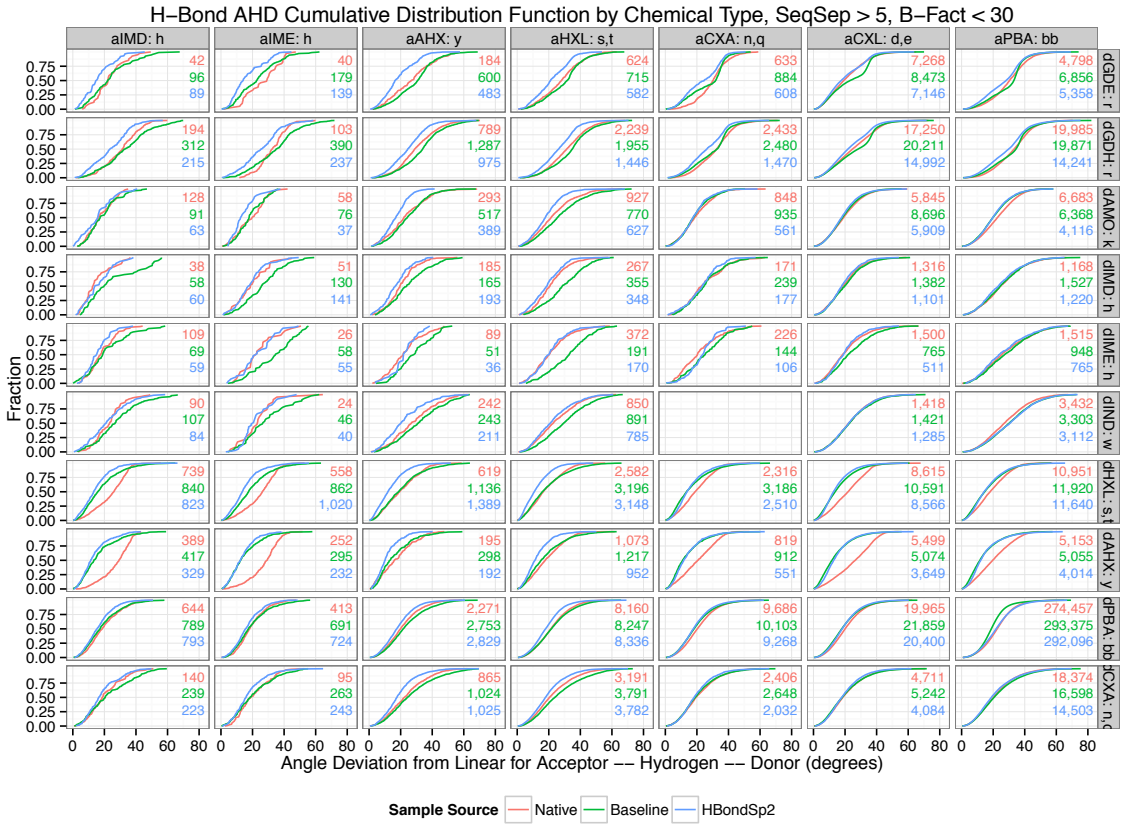


Figure 6.4.14 H-bond AHD Feature by Donor / Acceptor Chemical Type and Sample Source: Natives (Red), Baseline (green), and HBondSp2 (blue).

6.4.3 Discussion of H-Bond chemistry revealed by features analysis

Features analysis reveals details of underlying biophysical properties of hydrogen bonding. In the next section, I discuss the insights that features analysis gives to understanding the complex phenomenon of hydrogen bonding.

I have shown differences in H-Bond related feature distributions for Native and Score12 Relaxed Native sample sources relating to variation by chemical type, Sp2/Sp3 character at acceptor groups, and pathologies due to misspecification of the H-Bond functional form.

For mean AHdist by donor type in the Native sample source, I observe 3 ranges: 1.76 Å for aromatic and non-aromatic hydroxyl donors, 1.89 Å for imidazole and amino donors, and 1.98-2.04 Å for the remaining donors, with each group having a standard deviation within 0.2 Å (Figure 6.4.15). For Score12 relaxed natives, I observe less variation, 1.92-2.02 Å for all donor types with standard deviation for each group within 0.18 Å.

I observe modest effects of donor chemical type on the AHD angle; however, these trends correlate with the flexibility of the donor group (e.g., mean AHD angle dHXL: 19.29°, dIND: 26.89°) and are largely recapitulated by Score12 (dHXL: 19.02°, dIND: 31.14°). This is consistent with the notion that it is more challenging—for both Nature and Rosetta—to place less flexible residues when forming (i.e., linear AHD angle) H-Bond geometries. An exception is that in Natives, tyrosine has similar mean AHD angle as a donor and as an acceptor; 22.7° vs. 23.4°, while Score12 has a larger difference; 15.4° vs. 26.3°. The aromatic ring in tyrosine residues make it relatively inflexible, which explains the trend for both Native and Score12 to use tyrosine as an acceptor. One explanation for the lack of recapitulation of AHD angles for tyrosine as a donor is the ambiguity of the H-atom placement in X-ray crystal structures, and subsequent placement of H-atoms using Reduce can obscure the tighter AHD angles. A second explanation is that although in Natives there is a strong preference to place the hydroxyl H-atom in the plane of the aromatic ring, in Score12, the torsional angle of Tyrosine is sampled in the plane of the aromatic ring but is energetically unconstrained. This allows Rosetta to give mediocre H-Bonds artificially good scores by twisting the H-atom out of the plane.

6.5 Integration of Overlapping Feature Models

This case study demonstrates the use of the features analysis tool to harmonize feature models. Feature models that overlap violate the assumption of independence that motivates the additive

functional form of base energy functions. Overlap of feature models may be difficult to anticipate or detect when models are developed by different researchers, each with their favorite features. Checking the recapitulation of feature distributions can reveal unanticipated lack of fit that may be caused by overlap between different terms.

Interaction between feature models can be difficult to detect, primarily because people do not look for it, but also because the overlap may only occur in special cases, and good fit in general cases can mask very poor fits in special cases. Investigating conditional distributions can reveal the special cases and guide adjustment of the model's functional form or parameters.

In the remainder of this section, I consider two instances of feature model interactions. The first (section 6.5.1) is the overlap between the models for Lennard-Jones attraction between atoms and for Sp^3 donor H-bonds. Because these H-bonds make particularly close contacts, the repulsive component of the Lennard-Jones model competes with the preferred distance of the H-bond model. The second (section 6.5.2) is the overlap between the distance dependent Coulomb model for atom-centric electrostatic interactions and the H-bond model. Morozov, Kortemme and Baker (2003) suggested combining these two models and showed promising preliminary results, but the consequences of combining these two closely related models in Rosetta has never been fully investigated.

6.5.1 Lennard-Jones and Sp^3 donor H-bonds

In section 4.2, where we fit the AHdist term in the H-bond model to recapitulate native distributions, we could observe substantial variation by donor chemical type. In particular, H-bonds with the Sp^3 donor groups (Ser, Thr, Tyr) form particularly tight H-bonds. Panels A-C of figure 6.5.1 summarize the fitting process from section 4.2 for Ser and Thr donors (dHXL). The

black curve in each panel is the target native distribution. In panel A, the red and teal curves show exp(-E_{AHdist}(AHdist)), the Boltzmann distribution for the (dXHL-dCXA) chemical type with Score12 (red) and with newHB (teal). This is an energy term for Ser and Thr donors with peak shifted to match the native distribution. The corresponding curves in panel B show distributions after relaxing native structures: Score12 produces a peak too far at 1.9 Å as we expect, although the peak is higher due to a derivative discontinuity; newHB creates a flatter distribution, as designed, but does not shift all the way toward the native distribution. The root cause is that H-bonds are not the only energy terms; on such close contact, the repulsive component of the Lennard-Jones model dominates.

First, however, we should confirm that the misalignment of the peak is not due to automated placement of the Hydrogen atom. Figure 6.5.2 shows that the heavy atom distance, ADdist, undergoes a similar fraction of an angstrom shift. The shift looks smaller because the ADdist is correlated with the AHdist because the covalent bond attaching the hydrogen to the donor is essentially rigid and the AHD angle is relatively straight. The correlation, however, is not perfect, so it washes out some of the signal of the AHdist feature distribution.

The functional form of the Lennard-Jones model evaluates the distance, *r*, between two atoms of types *i* and *j* using two parameters, the well depth, d_{ij}, which is typically the geometric mean of the well depths for the two atom types, and the well minimum, σ_{ij}, which is typically the sum of the van der Waal radii for the two atom types,

$$E_{LJ}(r|d_{ij},\sigma_{ij}) = d_{ij}\left(\left(\frac{\sigma_{ij}}{r}\right)^{12} - 2\left(\frac{\sigma_{ij}}{r}\right)^{6}\right)$$

Adjusting well minima σs for these A-D types from 3.0 Å to 2.6 Å and σs for A-H from 1.95 Å to 1.75 Å allows the decoy feature distribution to recapitulate the native feature distribution, as seen in panel C of Figure 6.5.2.

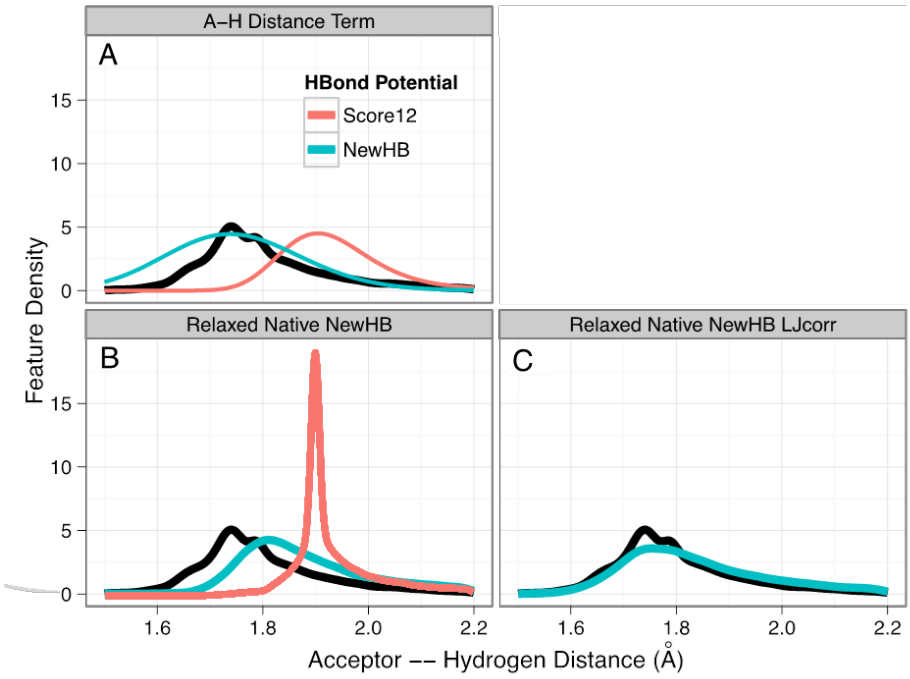


Figure 6.5.1 Hydryoxyl Donor to Backbone Acceptor H-bonds AHdist Correction: The thick curves are kernel density estimations from observed data normalized for equal volume per unit distance. The black curve in the background of each panel represents the Native sample source. (A) Thin curves show Boltzmann distributions for length in an isolated Rosetta H-bond model with the Score12 and NewHB energy terms. (B) Relaxed Natives with the Score12 and NewHB energy terms. The excessive peakiness of Score12 is due to a derivative discontinuity; the misaligned peak of NewHB is due to Lennard-Jones repulsion. (C) Relaxed Natives with the NewHB energy function and the Lennard–Jones minima between the acceptor and hydroxyl heavy atoms adjusted from 3.0 to 2.6 Å, and between the acceptor and the hydrogen atoms adjusted from 1.95 to 1.75 Å.

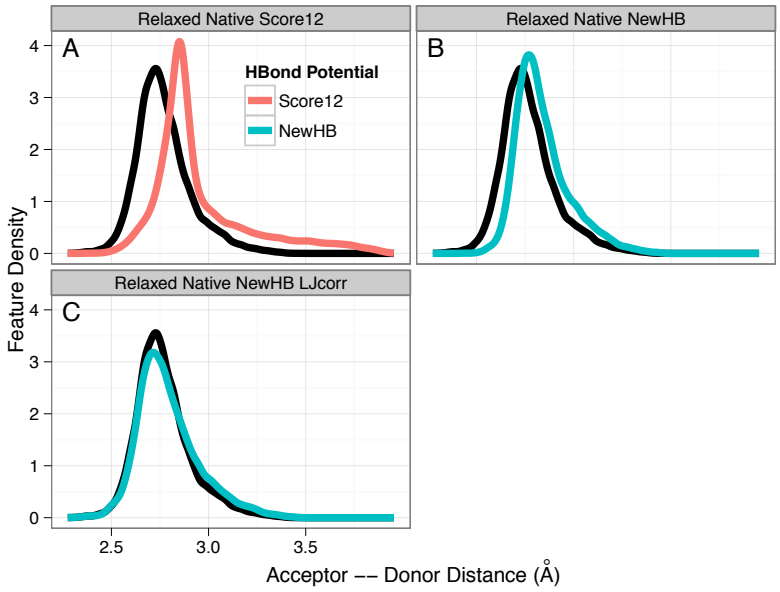


Figure 6.5.2 Hydryoxyl Donor to Backbone Acceptor H-bonds ADdist Correction: Score12 (A), NewHB (B), and NewHB with LJcorr (C) show shifts of magnitude similar to Figure 6.5.1 (B,C).

It would be possible to fit the distribution by adjusting just the H-bond model, and not touching the Lennard-Jones model, but this should be avoided for at least three reasons. The first is mathematical: the repulsive component of the Lennard-Jones model increases as a $1/r$ to the 12th power, so overcoming this would require an even stronger attractive H-bond potential. Not only is this not biophysically sensible, but the evaluation of the energy function becomes numerically unstable; it is better to have less force than strong forces in opposite directions. The second is for model maintenance: correcting the Lennard-Jones in the H-bond model introduces a dependency that makes the model more brittle and resistant to change. The third is scientific: when two models are competing against each other then there is an opportunity to understand each more deeply, or to understand if there is a synthesis that is better than either. Of course, the only way to dare to change other models is to have a robust suite of scientific unit tests and benchmarks, as I discuss in the next chapter.

6.5.2 Coulomb and Explicit models of H-Bonding

Two methods for capturing H-bond interactions are to model electrostatic interactions generally, which includes H-bond interactions as a subset, or to model H-bonds explicitly. In this section I consider combining these two models and the resulting feature distributions.

A molecular conformation specifies the co-ordinates of the atoms, but the positions of the electrons are only represented implicitly as being delocalized throughout the molecule. Interactions between the positively charged atoms and the negatively charged electrons are the electrostatic interactions, and are classically evaluated using Coulomb's law. Figure 6.5.3 shows a simple model of charges for electrostatics.

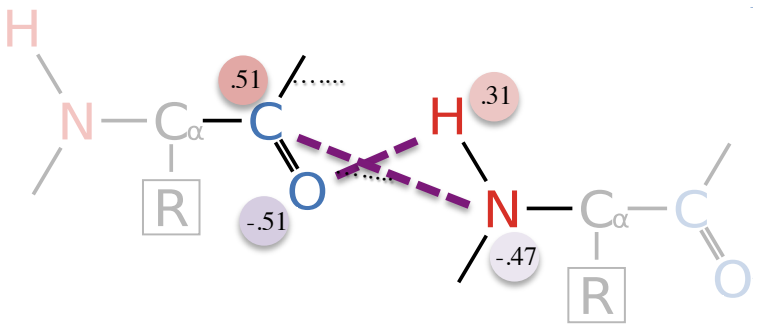


Figure 6.5.3 Coulombic Model with Atomic Point Charges: Explicit electrostatic models evaluate or approximate Coulombs law over a set of point charges. Here, charge is assigned to each atom (red and blue circles). Repulsive interactions between like charges (green dotted lines) and attractive interactions between opposite charges (purple dashed lines) contribute to the energy.

Electrostatic models typically assign point charges at atom or lone pair positions and for a given pair of point charges (q_i, q_j) with inter-charge distance $r_{q_i q_j}$ evaluate Coulomb's law using a dielectric constant, ε, that may depend on the local environment:

$$E(q_i, q_j) = \epsilon \frac{q_i q_j}{r_{q_i q_j}}.$$

For computational efficiency, electrostatic models often evaluate Coulomb's law at a subset of charge pairs (e.g., those within an interaction cutoff), and the energetic contribution of long-range interactions may be approximated (e.g., via an implicit solvent model, particle mesh Ewald summation, or multi-grid methods). One method of implementing a smoother distance cutoff is to assume that the dielectric constant is proportional to 1/r (distance dependent dielectric (Warshel 1984, Hingerty 1985)) so that the energetic contribution decays more quickly before the interaction cutoff. I consider a Coulomb model with partial charges defined by CHARMM 22 (MacKerell 1998) and a distance dependent dielectric with short and long-range cutoffs that are smoothed using spline interpolation.

Explicit H-bond models, in contrast, do not evaluate interactions over all pairs of atoms, or even all pairs within a cutoff, but instead identify specific interactions sites that form H-bond type interactions and evaluate a parameterized model on these sites. The Rosetta H-bond model is an example of an explicit H-bond model.

Most computational molecular energy functions include an electrostatic model (Vinogrado 1971, Hagler 1974, Cybulski 1989, Kaminski 2001, Ponder 2003, Liu 2012, Wang 2013b), an explicit H-Bond model (Lippencott 1955, Kabsch 1983, Vedani 1989, Gavezzotti 1994, Jain 1996, Hooft 1996, Grzybowski 2000, Lii 1994, Lii 1998, Langley 2002, Lii 2008, Kortemme 2003, Grishaev 2004, Paulsen 2008, Wang 2008, Choi 2009, Choi 2010, Liu 2011, Řezáč 2011) or both (Reid 1959, Boobbyer 1989, Vedani 1989, Fabiola 2002, Morozov 2003).

Because of the challenge of fitting molecular energy functions, many researchers select one model over the other, usually accompanied with justification for the superiority of their choice: one argument is that electrostatics are the dominant energetic component of H-bonding, and that

electrostatics models are generically applicable to all charge-charge interactions and are physically grounded in Coulomb's law (e.g. the CHARMM and AMBER computational models). An opposing argument is that the observable geometric specificity of H-bonding is difficult to capture with purely electrostatics models (Murray-Rust 1984, Kortemme 2003), and that the localized nature of H-bond models makes them more computationally efficient than the long-range nature of electrostatic models (Bradley 2005).

Dannenberg (1999) discusses the dichotomous nature of H-bonding in QM simulations, which suggests including both models is physically motivated. However, including both models requires jointly fitting the model parameters, or else the energetic contribution of H-bond formation would be counted twice (e.g., the caveats in Grishaev 2004). This can perhaps partially explain why some have seen increased predictive accuracy when both models are included (Morozov 2003), while others have not (Hagler 1974, Gavezzotti 1994, MacKerell 2000).

6.5.3 Integrating a Coulombic and an Explicit H-bond models

To integrate the Coulomb and explicit H-bond terms, I observe that H-bonds are a subset of Coulomb interactions and that the Coulomb model has a relatively rigid functional form. Therefore, I decide to modify the H-bond term to account for the overlap between the Coulomb term and the H-bond term.

In collaboration with Andrew Leaver-Fay, I begin with the H-Bond model fit into the native distributions, as described in Section 6.4. I then construct idealized two-residue systems for each pair of residue types that form H-bond interactions (using glycine to represent backbone interaction sites) and evaluate the total residue pair energy by positioning the residues to sample

along the AHDist axis. This is implemented as the Rosetta app, `sweep_respair_energies` and takes matcher input files (Richter 2011).

I modify the H-Bond AHDist term by subtracting the energetic contribution of the Coulomb term and rescale the minimum H-Bond energy to achieve -0.5. This modification allows the native feature distributions to be recapitulated. The energy components and total energy are plotted for each pair of residues in Figure 6.5.4.

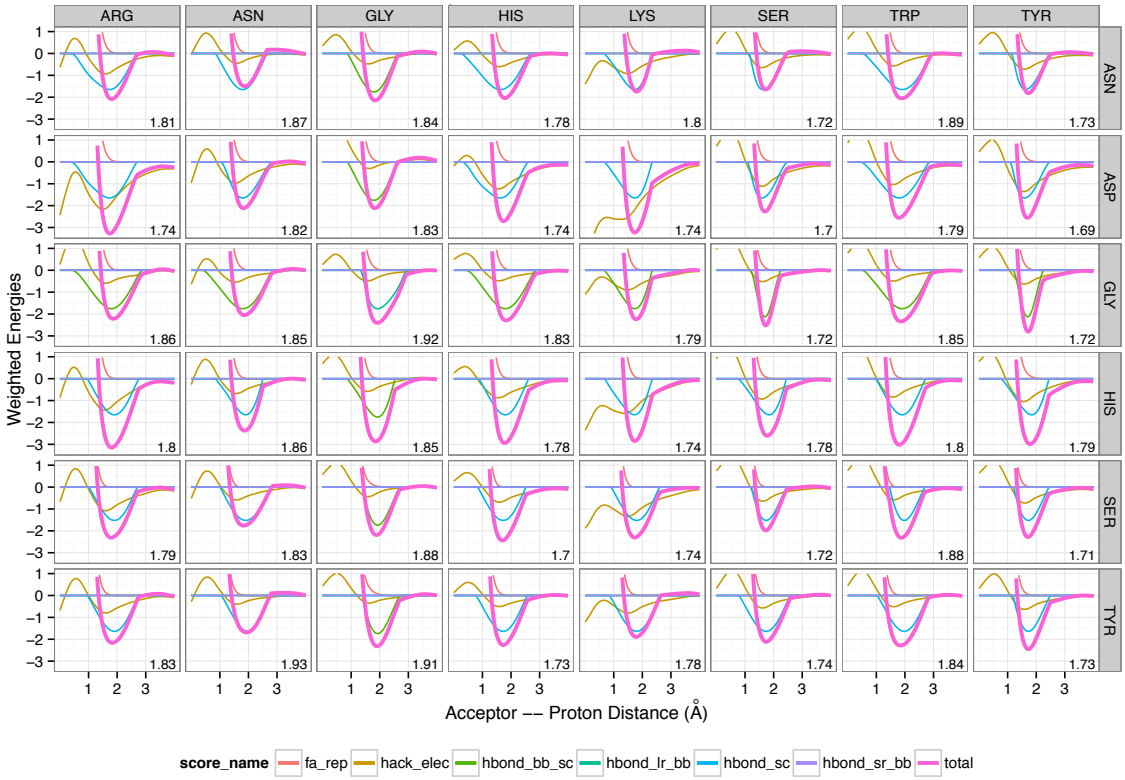


Figure 6.5.4 Rosetta Residue Pair Energies by AHdist: The number in the lower right is the AHDist that achieves the minimum value.

6.5.4 Evaluation of the Combined Electrostatics and H-bond Models

To evaluate the feature distributions of combining the Coulomb and H-bond models, I consider the following sample sources: Native, Elec (i.e., Coulomb model but no H-bond model), HBv1

(i.e., Baseline; corrections with Score12 H-bond model), HBv2 (i.e., HBondSp2; Baseline with update H-bond model), and ElecHBv2 (i.e., Coulomb model integrated with HBondSp2 as described above).

I update previous plots with these sample sources to evaluate first if the combined feature distributions are consistent with the Native feature distributions, and second attribute distributional characteristics to the different models or their combination.

The first plot is the AHdist feature distribution conditional on the donor chemical type, Figure 6.5.5. The mean of the Elec distribution is larger and the distribution is less concentrated. The ElecHBv2 looks similar to HBv2, except for lysine donors (dAMO) where the distribution looks even closer to the native distribution than HBv2.

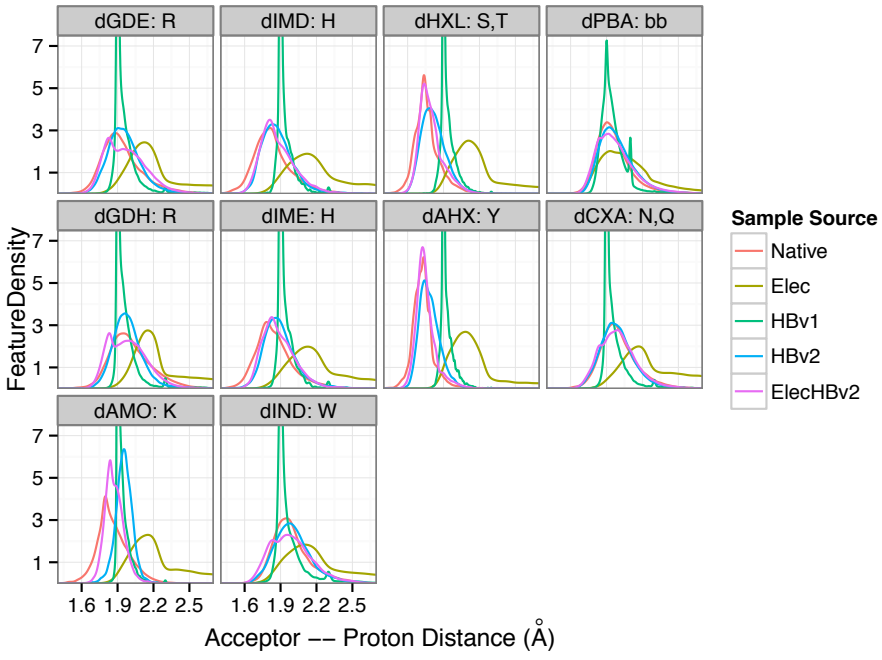


Figure 6.5.5 Elec + HB: H-Bond AHdist by Sample Source: For the five sample sources described in the text and colored as in the legend, H-bond AHdist feature distributions are plotted for each donor chemical type. This extends plot 6.4.7.

Next, to assess the orientation at the acceptor as in Section 6.3, I plot the joint (BAH, BAχ) distribution for representative class of H-bonds with Sp^2 acceptor types, aspartic acid and glutamic acid (aCXL) to charged sidechain donors. For each donor residue type and each sample source I plot the scaled kernel density estimation of the Lambert-Azimuthal projection as Figure 6.5.6.

Figure 6.5.6 shows that the Elec distribution lacks the orientation specificity of the explicit H-bond models. When combined with HBv2, Elec slightly broadens the distributions to make them more consistent with the native distributions.

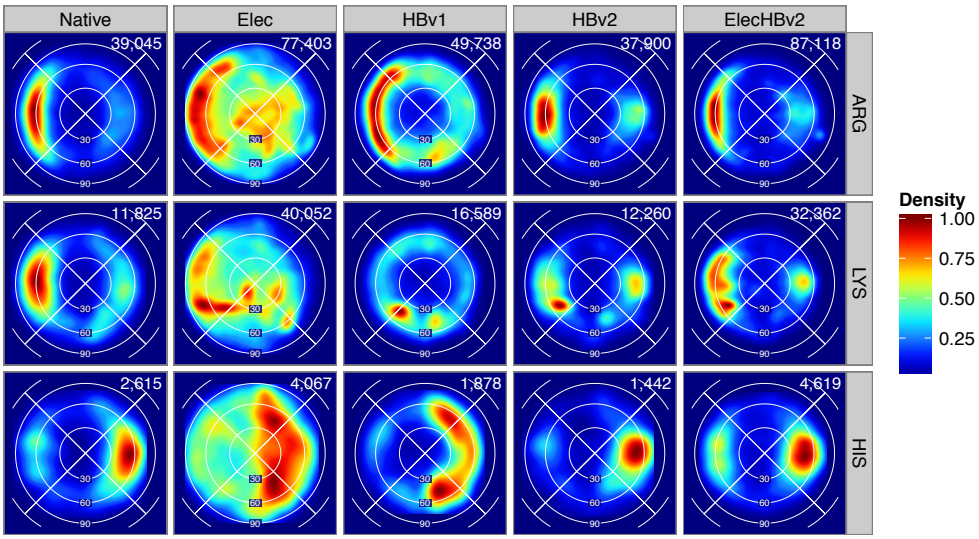


Figure 6.5.6 Elec + HB: Carboxyl H-Bonds (BAH,BAχ) by Donor Type and Sample Source: For the sample sources described in the text, for H-bonds with CXL acceptors to Arg, Lys, and His donors, (BAH, BAχ) angles are plotted using the Lambert-Azimuthal projection. Each cell plots the scaled kernel density estimation from the number of instances in the upper right corner. This extends plot 6.3.2.

Next, I plot the joint (BAH, BAχ) distribution for anti-parallel and parallel β-sheets in Figure 6.5.7, extending Figure 6.2.20, and the O-Hα distance distribution in Figure 6.5.8, extending Figure 3.2.21.

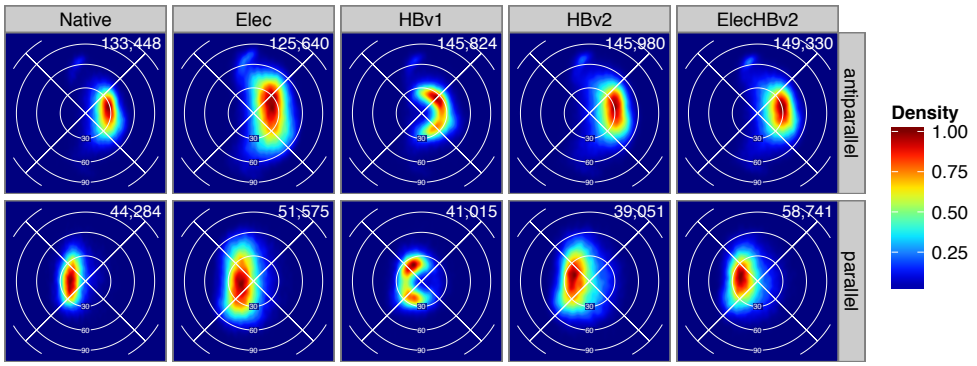


Figure 6.5.7 Elec + HB: β-Sheet H-Bonds (BAH,BAχ) by Sample Source: For the sample sources described above, for H-bonds forming parallel- and anti-parallel β-sheets the (BAH, BAχ) angles are plotted using the Lambert-Azimuthal projection. Each cell plots the scaled kernel density estimation and the number of instances considered in the upper right corner. This extends plot 6.3.19.

Interesting, the Elec distribution recapitulates the general Sp^2 character of the Native distribution, even if it is less concentrated. This indicates the lack of the Sp^2 character for Score12 is at least partially caused by the Score12 H-bond model.

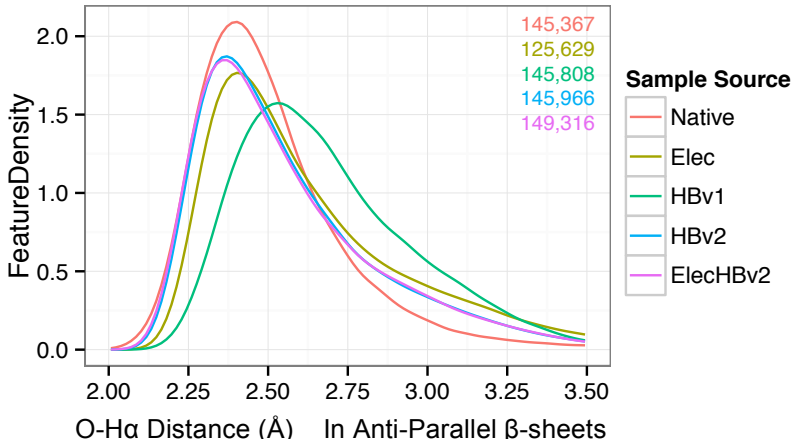


Figure 6.5.8 Elec + HB: Anti-parallel β-Sheet H-Bonds O-Hα Distance by Sample Source: For the sample sources described above and H-bonds in anti-parallel β-sheets, the close O-Hα distance distribution is plotted. This extends plot 6.3.20.

Next, I plot the (BAH, BAχ) distribution for Sp^3 acceptors. Sp^3 acceptors in canonical proteins occur in hydroxyl groups, which is an –OH group. For example, serine has an –OH groups and is shown as a H-bond acceptor is shown in Figure 6.5.5(A). In defining the BAH and BAχ angle for hydroxyl groups, there are different choices that can be made for the (B)ase atom shown Figure 6.5.5(A): The heavy atom, e.g. C_β for serine, (arc 1), hydroxyl H-atom (arc 2), or a virtual atom (V) on axis of symmetry through the hydroxyl oxygen (arc 3). For the purposes of plotting Figure 6.5.5(B) using V for the (B)ase atom highlights deviations from symmetry.

Score12 chose to measure angles between hydrogens rather than from the experimentally determined heavy atom to hydrogen in order to prevent the HAH angle distribution from becoming too tight. In Native vs. HBv1 in Figure 6.5.5(A), the native distribution curves to right (towards the H-atom) while the HBv1 distribution curves to the left (away from the H-atom). This motivates using the heavy atom for the HBondSp2 potential, and controlling the HAH angle distribution by the simple cosine potential on BAχ where the hydroxyl H atom is at 0°. The HBv2 distribution is too concentrated, and the distribution away from the peak is difficult to distinguish but curves to the right. (Inclusion of the BAχ restraint introduces a derivative discontinuity at the BAH=0° pole, but the lack of density at the pole means that this is not a problem.).

In the Native vs. Elec vs. HBv2, the Elec distribution is too broad and the HBv2 is too concentrated. Combining the Coulomb term and the HBondSp2 potential allows the ElecHBv2 distribution to more closely reproduce the Native distribution than either can separately.

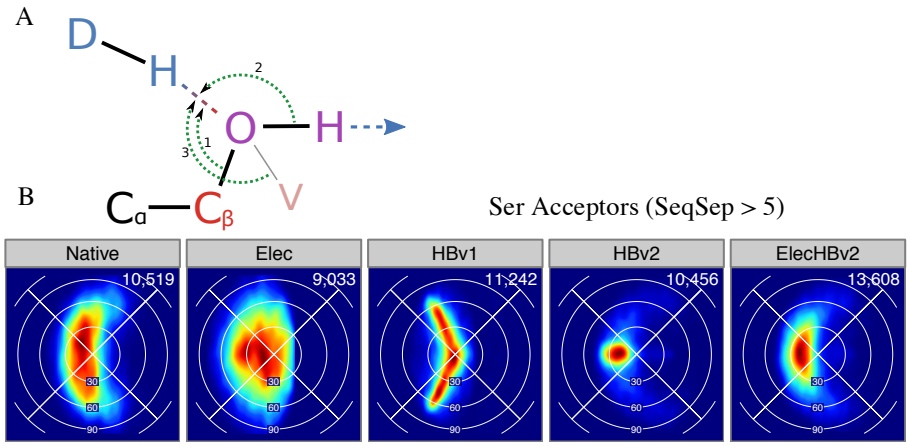


Figure 6.5.9 Elec + HB: Serine Acceptor H-Bonds (BAH,BAχ) by Sample Source: A: Arcs 1-3 show BAH angles with different choices of the (B)ase atoms. V is in the $C_\beta OH$ plane so the angles $C_\beta OV$ VOH are equal. B: The sample sources are Native, Elec, HBv1, HBv2 and ElecHBv2 described above, the estimated feature distribution of (BAH, BAχ) Lambert-Azimuthal projection using V as the base atom is plotted for serine acceptor H-bonds with Abs(don – acc) > 5 and B-factor < 30 $Å^2$. Each cell plots the scaled kernel density estimation and the number of instances considered in the upper right corner.

6.5.5 Conclusion

To summarize this case study, we see that the geometry of Native H-bonds is strongly influenced by the acceptor hybridization. We focus on Sp^2 acceptors and observe a bimodal distribution for bonding in the plane of the acceptor. Using the features analysis tool, we observe that the baseline H-Bond features in Rosetta do not recapitulate this Sp^2 character. To address this discrepancy, I develop a model of H-Bonds that evaluates the angles BAH, BAχ as a 2-dimensional term. With this simple model, H-Bond features from the (decoy) HBondSp2 sample source recapitulate Native patterns, including conditional dependence on donor type and secondary structure type and β-sheet geometries.

Together these plots indicate that the combined ElecHBv2 recapitulates the Native distribution the best, in part because the Elec model spreads out the overly concentrated HBv2 distribution. Additionally, the AHdist distribution remains consistent with the Native distribution because the model is fit to recover this distribution.

7 Structure Recovery Scientific Benchmarks

Recall that a scientific model is a simple tractable system that represents a more complex system of interest. A scientific benchmark is any method for assessing the accuracy of this correspondence. Clear scientific benchmarks define measurable objectives for improving scientific models. In the context of computational models in structural biology, widely used molecular mechanics energy functions, such as Amber and OPLS, were originally parameterized with experimental and quantum chemistry data from small molecules and benchmarked against experimental observables such as intermolecular energies in the gas phase, solution phase densities, and heats of vaporization (Jorgensen 1996; Weiner 1984). More recently, thermodynamic measurements and high-resolution structures of macromolecules have provided a valuable testing ground for energy function development. Commonly used scientific benchmarks include discriminating the ground state conformation of a macromolecule from higher energy conformations (Novotny 1984; Park 1997; Simons 1999), and predicting amino acid sidechain conformations (Bower 1997; Jacobson 2002) and free energy changes associated with protein mutations (Gilis 1997; Guerois 2002; Potapov 2009).

From a computer science perspective, the scientific benchmarks encapsulate biochemical modeling objectives and provide an interface for non-biochemical study of the models. In practice, the biochemical perspective is focused on problems in the life sciences; therefore, it becomes the role of the computer scientist to work with the biochemist to establish appropriate benchmarks that can then be translated into achievable computer science tests. The role of a model is to act as a simpler representation of a complex problem; therefore, it is important to capture just the important aspects of the system and not all of the minutiae. The close collaboration between computer scientists and biochemists is essential for scientific benchmark

development. Creating tools to help the biochemist effectively establish appropriate benchmarks facilitates the connections between these two disciplines.

To automatically notify developers, scientific benchmarks creators specify thresholds on results that signal more attention is needed. This allows the benchmarks to be run automatically through a testing server, such as rosettatests.graylab.jhu.edu, which performs both the commit level unit and integration testing, manages distributed BuildBot testing for specialized platforms, and runs scientific benchmarks on a best effort basis.

A features analysis scientific benchmark is a features analysis and one or more reference sample sources that can be run through a scientific benchmarking framework to evaluate a computational model. Features analysis is vital to this scientific benchmarking process because it allows biochemists to quantify their biochemical intuition about molecular geometry. In the language of a features analysis, it becomes possible to create computational models that have a defined goal of recapitulating native feature distributions.

Because features analysis is used to thoroughly explore sample sources in the case studies that I have described, separate sources are needed for scientific benchmarks to provide an independent assessment of predictive accuracy. Otherwise there is a danger of overfitting the computational model: improving recapitulation of observed feature distributions by worsening recapitulation of unobserved feature distributions. Therefore it is helpful to use recovery tests to directly measure the predictive accuracy.

In this chapter, I discuss a class of scientific benchmarks called *recovery benchmarks* that directly measure the predictive accuracy of the model by asking it to recover a full conformation from a subset of the atomic coordinates. Whereas features analysis scientific benchmarks could be called

model centric, in that they look at specific slices of a large number of conformations to evaluate a specific behavior of the model, recapitulation tests could be called *prediction centric*—they look at one conformation at a time to evaluate the accuracy of a specific prediction. By using only one data instance at a time, it is easier construct test sets that are independent from training data.

7.1 Recovery Benchmarks Evaluated

There is a spectrum of recovery benchmarks that range from computationally intense but accurate assessments of the prediction method to smaller, faster tests that give approximate assessments of the prediction method. To test H-bond models I consider 6 types of recovery tests. The *ab initio* structure prediction, which asks to predict the conformation of a protein using just the sequence information, would be the most important, but the computational cost of running it is extreme (Discussed in detail in Section 2.3). Fixed-backbone sequence recovery tests are important for design applications; I test monomer (single-chain) (Section 7.1.3) sequence recovery and interface sequence recovery (Section 7.1.4). Finally I consider fixed-backbone side chain prediction called Rotamer recovery (Section 7.1.5). Three variants include predicting all sidechain conformations simultaneously, predicting a cluster of 4 residues at once, and predicting one residue at a time.

To support building and executing recovery benchmarks, I have developed a recovery benchmark tool in Rosetta. Given a conformation, a recovery benchmark specifies a benchmark *protocol* for selecting the portion of the structure to fix or randomize and the method for predicting the randomized portion, a *comparer* for assessing the deviation from the native and whether the instance counts as recovered, and a *reporter* for summarizing the results. This conceptual framework is flexible enough to describe all but the ab initio recovery benchmark. The ab initio

benchmark has a multi-stage decoy generation process that requires a more sophisticated job distribution framework.

The recovery benchmark tool composes protocol, comparer and reporter modules in an object-oriented framework and runs the benchmark through the Rosetta job distribution framework. Documentation for implemented modules is in the appendix.

7.1.1 Ab initio Conformation Recovery Benchmark

The goal of the ab initio benchmark is to test whether a particular energy function can discriminate near-native and far-from-native conformations, as measured by RMSD. Achieving this goal would allow researchers to use that energy function to predict near native structures using only the sequence information.

In collaboration with Mike Tyka, I survey a diverse set of 87 small, ligand-free protein structures (between 57 and 260 residues), listed in (Table 7.1.1). Almost all are monomeric (3 are homodimers, 1 is a heterodimer). I generate 1000 predictions using a reference energy function for each sequence based on biased and unbiased Loophash sampling (Tyka 2013, Discussed in Section 2.3). These predictions are intended to cover the range of RMSD values. To assess the discrimination of the candidate energy function, each conformation is optimized using the FastRelax protocol (Section 2.3), and the resulting energy for each conformation is recorded for a total of 425k predicted conformations. This process requires 30k-140k cpu hours per protein, depending on the size of the protein.

1a32	1acf	1agy	1bk2	1bkr	1bm8	1cc8	1cei	1ctf	1elw
1enh	1ew4	1fna	1hz6	1iib	1jbe	1kf5	1lou	1mjc	1nps
1o8x	1opd	1pgx	1poh	1prq	1r69	1sau	1sen	1t2i	1t3y
1ttz	1tul	1ubi	1vcc	1vkk	1wdv	1x6x	1z2u	1zma	2acy
2chf	2dfb	2fi1	2h28	2he4	2hhg	2i24	2i4a	2i6c	2iay
2igd	2jek	2nqw	2nr7	2nwd	2oml	2oss	2ppp	2r2z	2ra9
2v1m	2vq4	2vwr	2wwe	2zib	3b79	3co1	3cx2	3d4e	3dke
3ess	3ey6	3f2z	3fk8	3gbw	3hp4	3hyn	3ich	3klr	3nbm
3q6l	4lzt	1tig	1ugh	2qsk	2gzv	2icp			

Table 7.1.1 Proteins selected for the ab initio Conformation Recovery benchmark. Proteins shown in grey are excluded because they greatly increased the variance in the results.

The benchmark computes an average recovery score, which can be compared for different energy functions. To compute the aggregate recovery score, I normalize the scores for each target by mapping the inter 90% quantile to [0,100]. The recovery score predictive accuracy corresponds by the correlation between energy and RMSD to the native. In analyzing the results, 5 proteins were responsible for substantial variation to the results and were excluded (Table 7.1.1).

The limitations of this test include the high computational cost as well as the small number of sequences evaluated. Moreover, the sequences do not necessarily represent the range of conformations and protein sizes that occur in nature. Additionally, there is potential bias coming from the initial candidate conformations. However, this is currently the gold standard for evaluating the quality of the energy function.

7.1.2 Monomer Sequence Recovery Benchmark

The monomer sequence recovery benchmark fixes the conformation of the backbone and asks to recover the native amino acid at each position. The test set consists of 38 large proteins from Ding and Dokholyan (2006). Sequence recovery is performed with the discrete, full-protein rotamer-and-sequence optimization protocol called PackRotamers. Rotamer samples are taken from the given rotamer library (the 2002 or the 2010 library) supplemented with rotamers at plus

and minus one standard deviation for the first two sidechain torsion angles (Leaver-Fay 2011a). I use the multi-cool simulated-annealing protocol (Leaver-Fay 2011b). To measure the predictive accuracy, I measured the sequence recovery rate, for which higher values are better.

In collaboration with Andrew Leaver-Fay, I use a variant of the sequence recovery test to automatically fit amino acid reference energies using the OptE protocol (Leaver-Fay, 2013). The limitations of this test are that it considers only a fixed backbone, which prevents using backbone remodeling to accommodate mutations.

7.1.3 Interface Sequence Recovery Benchmark

Interface sequence recovery tests the prediction of the native amino acid identity given the native backbone for residues at transient protein-protein heterodimeric interfaces. An interface is transient if the partners are stable both as monomers and as a complex. In nature, transient protein-protein heterodimeric interactions form one of the primary modes of cellular signal transduction and therefore they are critically important for structure prediction and design.

In collaboration with Kevin Houlihan in the Kuhlman lab, I select 96 transient protein-protein heterodimeric interfaces from X-ray crystal structures deposited in the Protein Databank having resolution less than 2 Å, bond length outliers in less than 5% of the residues (as defined by MolProbity), MolProbity score of less than 2.0 (Chen 2010), no missing density for interface residues, and no small molecules at the interface (Table 7.1.2). Interface residues are those having a sidechain heavy atom within 5.5 Å of any atom of the partner chain or having a Cβ within 9.0 Å a Cβ of the partner chain and the Cα-Cβ bond vector pointing at it (making an angle of at least 105°). For glycines, virtual Cβ atoms based on the geometry of alanine are used. A residue is

buried if it has at most 15 neighbors and exposed if it has at least 24 neighbors, where residues are neighbors if their Cβ (Cα for glycine) atoms are within 10 Å of each other.

To perform the sequence recovery benchmark, for each interface residue I randomize the amino acid identity and conformation and run the PackRotamers design algorithm 10 times. PackRotamers performs multi-cool simulated annealing over a discrete set of rotamers defined by the Dunbrack rotamer library (the 2002 or 2010 library) supplemented with rotamers at plus and minus one standard deviation for the first two sidechain torsion angles (Leaver-Fay 2011). To measure sequence recovery, I compute the percent of interface residues recovered across all runs.

By fixing the docking orientation and backbone conformation, the interface sequence recovery test is an efficient assessment of predictive accuracy. Further, since the monomers must be soluble, the interface contact surfaces must be hydrophilic and therefore form substantial polar contacts in the complex, making this a good test of H-bond and electrostatic contact prediction.

PDB	Chains	PDB	Chains	PDB	Chains	PDB	Chains	PDB	Chains	PDB	Chains
1DPJ	AB	1YDI	AB	2QWO	AB	3CJS	AB	3LPE	GH	4BJW	AB
1GL4	AB	1Z0J	AB	2VLQ	AB	3CPT	AB	3MMY	EF	4DH2	CD
1H32	AB	1Z3E	AB	2VN6	AB	3D3B	AJ	3MXN	AB	4DRI	AB
1JKG	AB	1ZHH	AB	2VOH	AB	3DBO	AB	3N1F	AD	4EGC	AB
1M45	AB	2ES4	AD	2VPB	AB	3DGP	AB	3NHE	AB	4EQA	BD
1OEY	BK	2FCW	AB	2VSM	AB	3DLQ	IR	3NV0	AB	4EUK	AB
1OO0	AB	2FHZ	AB	2WBW	AB	3EGV	AB	3NY7	AB	4G1Q	AB
1PXV	AC	2HQS	DE	2WY3	BA	3EP6	BA	3QF7	AC	4G6T	AB
1QAV	AB	2NPT	AD	2XTT	AB	3F6Q	AB	3QN1	AB	4G7X	AB
1R0R	EI	2ODE	AB	2ZA4	AB	3FJU	AB	3RNQ	BA	4GED	AB
1T0P	AB	2OMZ	AB	2ZFD	AB	3IXS	IJ	3SBT	AB	4GEH	CD
1T6G	AC	2OZN	AB	2ZSI	AB	3KCP	AB	3SHG	AB	4GF3	AB
1TA3	AB	2P1M	AB	3A4U	AB	3KF6	AB	3TZ1	AB	4HPM	AB
1UW4	CD	2P45	AB	3A8K	AE	3KJ0	AB	3ZEU	AB	4I0X	KL
1WMH	AB	2PTT	AB	3AWU	AB	3KSE	BE	4APX	AB	4IU3	AB
1XG2	AB	2QKH	BA	3C9A	BD	3KYJ	AB	4AT7	AB	4JHP	BC

Table 7.1.2 Interfaces for the interface sequence recovery scientific benchmark

7.1.4 Rotamer Recovery Benchmarks

Rotamer recovery asks to recover the conformation of a set of sidechain conformations given the conformation of the backbone and a subset of the remaining sidechains. Given a similarity metric, a rotamer is recovered it is within some distance of the native sidechain. These form some of the smallest, most efficient scientific benchmarks. Variants of the rotamer recovery test have long been used to evaluate molecular structure energy functions (Petrella 1998, Liang 2002), including extensive use to evaluate the Rosetta energy function (Kortemme 2003, Dobson 2006, Dantas 2007, Jacak 2012), the ORBIT energy function (Sharabi 2010), and the SCWRL energy function (Shapovalov 2011).

I use three variants of the rotamer recovery benchmark, which I call *One*, *Cluster*, and *All*.

The Rotamer Recovery One benchmark fixes the protein structure except for one residue, and asks the prediction protocol to predict the native sidechain conformation. The native rotamer is recovered if all sidechain χ angles are within 20° of their native angle. The benchmark consists of 9,452 residues that are not glycine or alanine (because they have only a single conformation) and have B-factor < 30 $Å^2$ in complexes from the Top8000 set where the total number of residues in the complex is less than 5,000. Filtering by the number of residues is used limit the total memory usage. To model crystal contacts (interactions across an interface that exist only in the crystal and not in their native environment) for each conformation the symmetry mates in the crystal structure are built. To predict a sidechain conformation, the RTMin protocol (Wang 2005) examines each discrete rotamer in turn and optimizes its conformation using gradient-based minimization, selecting the resulting conformation with the lowest energy.

The Rotamer Recovery Cluster benchmark, created in collaboration with Amelie Stein in the Kortemme lab, fixes the backbone conformation an all sidechain conformations except for a cluster of four residues and all residues with 4 Å of the cluster. A cluster and its neighbors are defined when each pair of residues have some pair of atoms within 8 Å. The benchmark asks to simultaneously recover the sidechain conformation for each residue in the cluster. A cluster is counted as recovered if at least two of the residues have all of their χ angles within 10° of the native angles. The benchmark identifies 76,811 clusters in the Top8000 set after filtering for residues with B-factor < 30 $Å^2$. Residues within 4 Å of the cluster are randomized and all other residues are fixed in their native conformation. To predict the sidechain conformations, the PackRotamers protocol is used.

The Rotamer Recovery All benchmark fixes the backbone conformation and asks to predict all sidechain conformations simultaneously. A rotamer is counted as recovered if all of its χ angles are within 20° of the native angles. The benchmark identifies 466,797 residues in the Top8000 set that have B-factor < 30 $Å^2$. To predict conformations, I use the MinPack protocol, which is an extension of the PackRotamers protocol that includes gradient-based minimization before a Monte-Carlo acceptance check.

7.2 Results from Scientific Benchmarks

In this section, I evaluate the modifications to the Rosetta energy motivated by features analysis though prediction recovery benchmarks. This provides additional confirmation that these changes are beneficial and should be adopted by the scientific community. In collaboration with Andrew Leaver-Fay, I collect and analyze 5 energy function corrections culminating in the Talaris2013 energy function, which we recommend and has been adopted as the new community standard energy function.

7.2.1 Energy Functions Evaluated

Score12 was the standard Rosetta energy function up to this work (2003-2013) (Rohl 2004, Leaver-Fay 2013). *Baseline* is the Score12 energy function with the following corrections:

- Analytic evaluation of the Lennard-Jones model for van der Waals forces (fa_rep, fa_atr) and EEF1 model for implicit solvent desolvation forces (fa_sol) rather than table lookup evaluation.
- Residue-level idealized bond length and angle coordinates from feature distributions estimated for the Native sample source described in (Song 2010).
- A new disulfide potential `dslf_fa13` described in Section 4.3.
- Smooth backbone potentials as described in Section 4.2
- reversion of atomic dgfree parameters to the EEF1 solvation parameters (Lazaridis-Karplus, 1999): NH20(HN2), -10 to -7.8; Narg(NC2), -11 to -10; OH(OH1), -6.77 to -6.70; ONH2(O) -10 to -5.85,
- Updating the dun10 rotamer library to the dun10 rotamer library (fa_dun) (Shapovalov 2011, Leaver-Fay 2013)
- Refit amino acid reference weights using the OptE protocol (Leaver-Fay 2013)

The HBondSp2 is the Baseline energy function with the following modifications to the HBv1 H-bond model making the HBv2 H-bond model.

- Functional Form
    - $E_{HB} = E_1(BAchi, BAH) + E_2(AHdist) + E_3(AHD)$
    - 7 acceptor chemical types, 10 donor chemical types
    - bb/sc exclusion rule
    - Down weight solvated H-Bonds to 1/5 strength (consistent with Score12 H-bond model)
- Smoothness
    - AHD smooth at pole
    - Do not use Score12 fade functions
- Sp^2
    - (BAH, BAχ) potential
    - One potential for all Sp^2 acceptors
- Sp^3
    - Acc: BAH from heavy atom, cosine potential on BAχ
    - Don: LJ_{HBond} hdis: 1.75 $LJ_{OH\text{-}D}$ dis: 2.6
    - Rotameric sampling for OHχ for (Ser/Thr): from (-60, 60, 180) to (0, 20, 40, … , 340)
- Refit amino acid reference weights using the OptE protocol (Leaver-Fay 2013)

Coulomb includes the Baseline corrections and a smoothed distance dependent dielectric Coulomb potential.

- min_dist 1.6, max_dist 5.5
- Smoothed at min_dist/max_dist using spline interpolation

- Refit amino acid reference weights using the OptE protocol

Talaris2013 includes the Baseline, HBondSp2, and Coulomb corrections

- Subtract out Coulomb double counting from H-Bond AHdist
- Fade H-bond energy smoothly to zero [-.1, .1] -> [-.1, 0]

- Refit amino acid reference weights using the OptE protocol

7.2.2 Test Results

In this section I report the results of the recovery tests described in Section 7.1 for the energy functions described in Section 7.2.1. The baseline energy function gives a clear improvement over Score12. The rotamer recovery improvements are qualitatively consistent with the improvements to SCRWL with the use of the improved 2010 Dunbrack rotamer library (Shapovalov 2011). The improvement of HBondSp2 relative to the Baseline is limited, and in the case of the ab initio decoy discrimination test, the results are worse. The results of Coulomb relative to the Baseline are clearly better. Lastly, the Talaris2013 energy function improves over both HBondSp2 and Coulomb to give the best results for each test overall (Table 7.2.1).

	H-Atm	**Relax**	**Rotamer Rec**			**Seq Rec**		**Decoy**
Energy Function	**MMD vs Nat**	**Native RMSD**	**One (%)**	**Clust (%)**	**All (%)**	**Mmr (%)**	**IFace (%)**	**Discrim Score**
Score12	5.28	1.56	81.54	71.69	76.69	37.0	34.8	-2.89
Baseline	4.48	1.55	82.98	75.77	78.81	36.9	39.1	-3.56
HBondSp2	3.87	1.56	83.13	76.45	78.85	36.9	40.5	-3.99
Coulomb	5.52	1.48	83.58	76.11	79.50	39.0	39.2	-5.77
Talaris2013	**4.91**	**1.47**	**84.46**	**77.17**	**80.32**	**39.3**	**40.8**	**-6.31**

Table 7.2.1 Feature distribution and recovery benchmark results for incremental modifications to Score12, culminating in the Talaris2013 energy function. H-Atom MMD vs. Native is described in Section 5.3, the rest are described in Section 7.1.

The improvements that the HBondSp2 energy function makes to the recapitulation of native feature distributions indicate that, even with middling recovery benchmarks scores, the modifications should be considered further. Indeed, the improvements of Talaris2013 over Coulomb give evidence that the HBondSp2 corrections are contributing to the increased predictive accuracy. To further assess that this improvement is legitimate and not a result of uncontrolled factors or over-fitting, I consider the results of the Rotamer Recovery One benchmark in more detail. The trends for the energy functions appear to be consistent across the recovery benchmarks, and the rotamer recovery test is the most computationally efficient.

I first consider the sensitivity of the rotamer recovery tests to the weight of the H-Bond model in the energy function. I perform the Rotamer Recovery One benchmark, sweeping the H-bond weight from 0.1 times the usual weight of 1.17 to 1.5 times the usual weight (Figure 7.2.1).

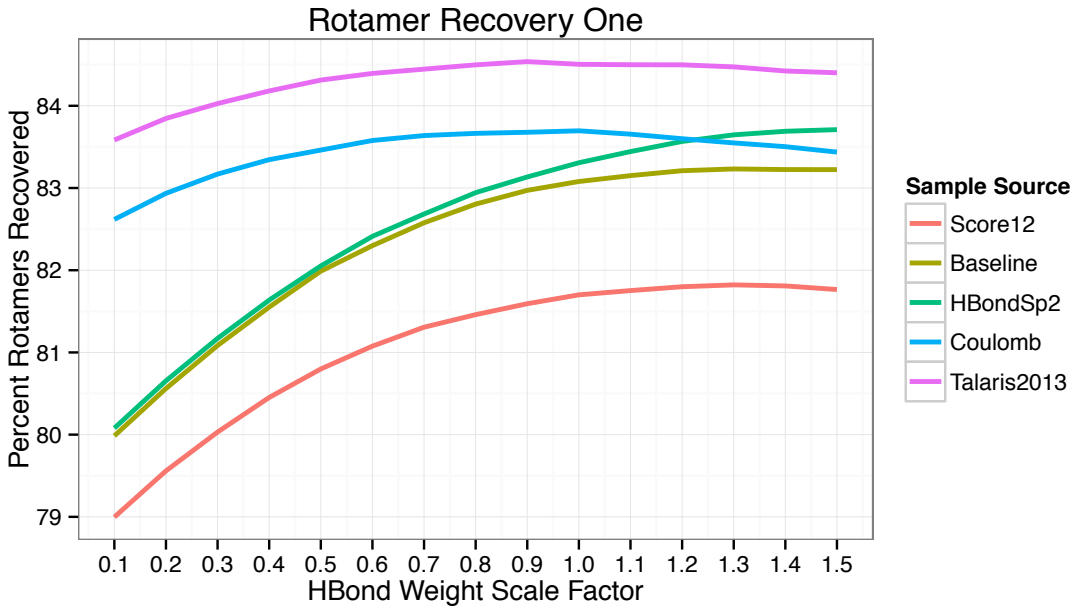


Figure 7.2.1 Rotamer Recovery One Benchmark by H-Bond weight: The x-axis is the weight relative to the weight in the energy function and the y-axis is the percent of rotamers recovered.

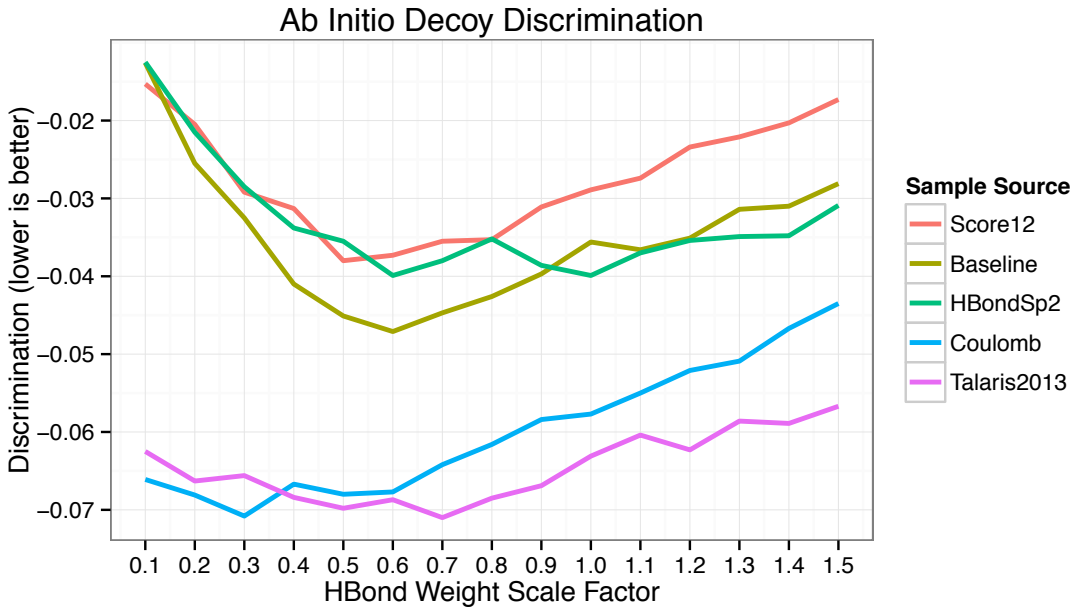


Figure 7.2.2 Ab Initio Decoy Discrimination by H-Bond weight: The x-axis is the weight relative to the weight in the energy function and the y-axis is the ab initio decoy discrimnation score where lower scores are better.

The smoothness of the curves indicates that the run-to-run variation of the test is very low. The recovery score is the relatively stable around the scale factor of 1, with HBondSp2 showing increasing performance up a weight of 1.5 and Coulomb showing maximal performance around .75. The recovery rate consistently falls with lower H-bond weights. The Coulomb and Talaris2013 terms are less sensitive to changes in H-Bond weight, especially at low weights. This is consistent with the observation that the Coulomb term overlaps with the H-bond model, so at low H-bond weights, H-bond interactions are at least somewhat evaluated. However, the consistent improvement of Talaris2013 over Coulomb confirms the general result that the changes to the H-bond model are contributing to improvements in predictive accuracy.

8 Conclusions and Future Directions

8.1 Summary

In this dissertation, my goals were to build a tool to help researchers develop computational models for protein structure, and to demonstrate through case studies various ways that this tool has already aided in improving the Rosetta energy function. To build computational models, researchers take observations of molecular features and encode them into the computational model. As I discussed in Section 2, a good computational model should represent phenomena but also be simple enough to manipulate and examine hypotheses about the phenomena. In this case, biochemical researchers use computational models to make experimentally testable predictions of molecular conformations.

However, the complexity of the molecular structure and the resulting complexity in the model mean that it is non-trivial to ensure correspondence between the data and the model. A central type of computational model in structural biology is the energy-based model, which defines an energy function; often as a linear combination of feature models that each captures a type of geometric or chemical pattern in molecular structures. The subtlety of the individual feature models and the intricacies that they represent, in addition to the interaction between multiple feature models, results in a highly complex model.

The challenges of modeling complex systems are not unique to structural biology. The availability of large quantities of data and the power of computers means the primary limitation of in building scientific models is our ability to manage the complexity of the model.

A contribution of computer science to scientific modeling is developing methods and tools to help researchers work with the computational models they want to create. The first step in this process is to create a common language between the experimental data and the computational model. Therefore, in Chapter 3, I developed a conceptual language of sample sources, features, feature distributions, and features analysis to describe the correspondence between experimentally observable data and the generative computational models. Development of the features analysis tool required many decisions for appropriate techniques.

I then described my features analysis computational tool. The tool has three parts, a relational database schema to store samples of feature instances from experimental and computational model sample sources; a framework built in Rosetta for reporting features to the database from batches of conformations to populate features databases; and a framework and support tools for writing and executing features analysis scripts in the R statistical programming language. The components of the tool are modular and can be used independently. They build upon robust and established technologies to have a high level of functionality.

Next I demonstrated how this features analysis tool could be used. In Chapter 4, I used two case studies to demonstrate the workflow of a features analysis. The first case study investigated the impact of the use a popular conformational sampling heuristic on the deviation of a bond angle typically assumed to be fixed. Going step-by-step through the features analysis I showed how features analysis could be used to observe that the assumed bond angle restraint model is inadequate while a new bond angle restraint model under development appears to be an improvement. The second case study considered the impact of discontinuities in the derivative of the energy function when using gradient-based minimization algorithms. I observed severe spikes in feature distributions, which are resolved when the discontinuities are smoothed. This case study showed how the features analysis tool can be used to check computational aspects of the

model that is otherwise difficult to recognize, thus forming “scientific unit tests” for the computational model.

In Chapter 5, I discussed methods relating to the development of the features analysis tool, including considerations for performing kernel density estimation and using features analyses for statistical hypothesis testing.

I then introduced concepts for energy function-based computational models and the relationship with feature models. I discussed how the computational constraints of computational models—as opposed to pure statistical models—constrain the development of further models. I then introduced computational models for H-bonding, and in particular the H-bond model in Rosetta. Through three case studies considering deriving the functional form, parameter fitting, and interaction with other terms in the energy function, I showed how features analysis could be used to build and evaluate feature models in the context of the H-bond model in Rosetta (Chapter 6).

A final aspect of the computational analysis of Rosetta energy functions is the development of scientific benchmarks. Benchmarks allow researchers to quantify their improvements to the energy function. In chapter 7, I put features analysis in the context of community oriented scientific benchmarks and developed a complementary type of scientific benchmark called recovery benchmarks that directly test the predictive accuracy of a computational model. Using a spectrum of recovery benchmarks, I assessed the impact of the modifications the modeling of H-bonds developed in chapter 6 and conclude that the combined energy function improves the scientific benchmarks across the board. I then assessed the sensitivity of the results to the weight of the H-bond model.

8.2 Scientific Modeling as a Community Endeavor

A central goal of this dissertation was to develop tools to facilitate the practical development of computational modeling. I have had the good fortune of working closely with the vibrant Rosetta community to address critical modeling challenges. This has required participating in a large-scale library reorganization project, and co-leading the highly successful Rosetta Boot Camp to train new developers and increase diversity in the developer community. By working with researchers in the Rosetta community to develop and use the tools in this dissertation, I have made significant contributions to the ability to build and evaluate computational models. The concepts of features analysis and feature modeling are now standard language in the usage of Rosetta modeling endeavors.

My work in features analysis and recovery scientific benchmarks elevates the role of scientific benchmarking as a mature, consensus-driven process for guiding the improvement of central energy function. This work has been highly collaborative; for example, the paper detailing the core scientific benchmarks used to assess the Rosetta score function was written in collaboration with researchers at the University of Washington, Seattle, Scripps Research Institute, the University of California San Francisco, Johns Hopkins University, Duke University, the University of Washington St. Louis, and the University of North Carolina at Chapel Hill. In May of 2013, members of the Rosetta community convened a special two-day conference at the Talaris conference center in Seattle, Washington to evaluate and decide on recent improvements to the standard Rosetta energy function Score12. The work I described in this dissertation on improving the H-bond model was included as part of the new Rosetta standard energy function, Talaris2013, along with the disulfide model developed by Frank Dimaio, new bond geometry developed by Yifan Song, and a new a rotamer library by the Dunbrack lab. Following the meeting, in collaboration with Andrew Leaver-Fay, I have addressed the computational as well as social challenge of smoothly transitioning developers and users to use a new default energy

function. For example different users may have different views on what evidence is sufficient for deciding to try a new energy function in a production run. If the default is changed and they insist on using the previous default, backwards compatibility and making the choice of the energy function explicit rather than implicitly taking the default requires further code development.

Also presented at the Talaris meeting were progress reports on exciting improvements to the solvation model, backbone geometry, and new relax sampling protocols. The process of model assessment and scientific benchmarking will allow these efforts to progress rapidly.

8.3 Future Directions and Uses of the Features Analysis Tool

Two remaining issues with the H-bond modeling uncovered by my work in features analysis are modeling of salt-bridge geometry and relative H-bond strength. Salt bridges are an H-bonding motif in which sidechains form two parallel H-bonds. The feature geometry of salt bridges has been detailed in Donald (2011). Implementing the described feature definitions and performing features analysis, I observe that Rosetta simulations do not recapitulate native salt bridge geometry. Particularly striking is that aspartate and glutamate amino acids preferentially form bifurcated H-bonds (two donors to one acceptor) with arginine rather than forming salt-bridge bidentate H-bonds (two donors to two acceptors). Figure 8.3.1 shows the geometry and Figure 8.3.2 shows a relevant feature distribution. Shifting this balance may require creating a non-pairwise additive H-bond model, which will benefit from features analysis in deriving the functional form, parameterization, and evaluation of its utility.

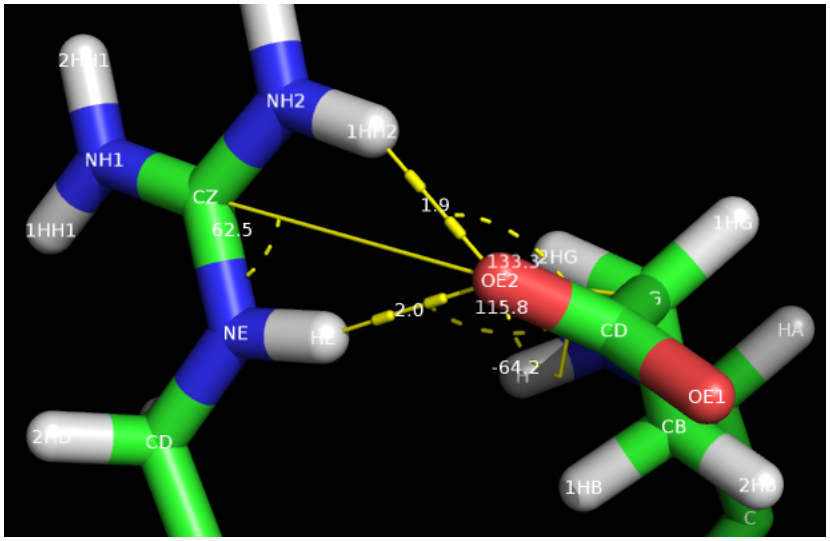


Figure 8.3.1 Bifurcated Salt Bridge: Arginine (left) forming bifurcated H-bonds with glutamate (right). In Rosetta predictions this interaction is favored more than bifurcated interactions where the two oxygen atoms of glutamate with two H-bonds with arginine to form a salt bridge interaction.

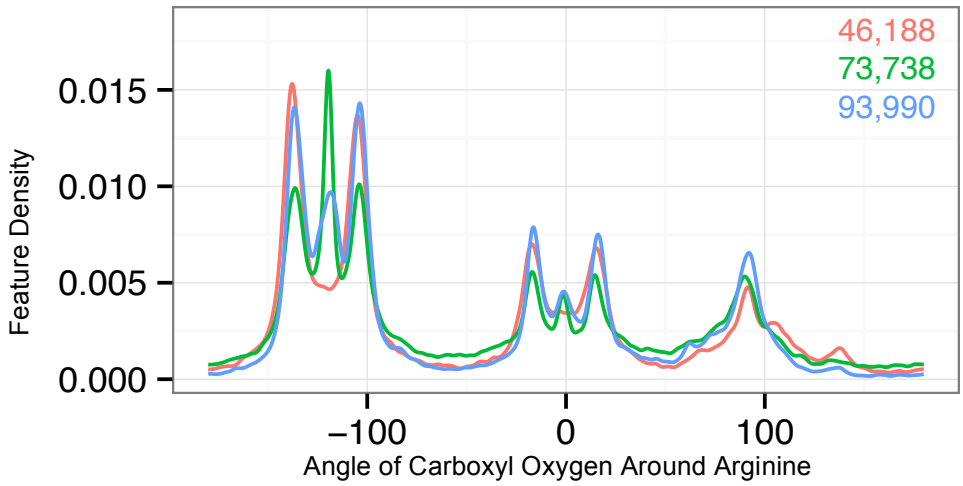


Figure 8.3.2 Angle of carboxyl oxygen atoms around the arginine functional group: Sample sources: Native (Red), Score12 (Green), Talaris2013 (Blue). The pair of peaks on the left and center corresponds to salt bridges with the peak the middle of the left pair corresponds to the bifurcated geometry shown in Figure 8.3.1. The bifurcated peak is non-existent in Natives, large in Score12 and moderate in Talaris2013, indicating that work still remains to recapitulate this feature distribution.

A second issue uncovered by the features analysis is that H-bonds are preferentially formed by interactions between certain donors and acceptors. This is consistent with the hypothesis that H-bond interaction strength is determined by the proton affinity at an interaction site (Gilli 2009). I define the interaction affinity based on the mutual information metric between a donor chemical type and an acceptor chemical type as the negative log of the relative probability of observing an interaction involving those types divided by the relatively probability of observing the donor type and the relative probability of observing the acceptor type,

$$I(d,a) = -\log\left(\frac{p(d,a)}{p(d)p(a)}\right)$$

Preliminary work shows that the strength can be modulated to recapitulate this feature.

8.4 Use of Discovered Features in Machine Learning Classifiers

The term ‘feature’ is often used in machine learning, where complex object are classified using feature vectors. A primary goal of my features analysis tool is to assist researchers in exploring

and identifying features that measure discrepancies between predictions and experimentally characterized conformations. A natural extension would be to include discovered features in automated classification algorithms. As an example, Qiu et al. (2008) fit parameters of a support vector machine with a fixed set of features to discriminate experimentally determined conformations from Rosetta predictions; it would be conceivable to use features developed using my tools to extend their model. “Feature engineering” is a valuable, though labor-intensive, task that feeds into computational learning algorithms (Bengio 2012). Although there has been significant recent progress in “deep learning” methods, which automatically learn high-level representations of data, there is still a pressing need for programs to aid—rather than replace—human intelligence (Brooks 1996). For example, the annual data mining and knowledge discovery competition KDD Cup (www.kde.org/kdd2013) attracts thousands of teams to build computational models for a specific data set. Recent winners (Yu 2010, Li 2013) report that they relied on feature engineering—using insight into the problem domain to build feature models that are combined to give overall predictions.

A A FEATURES REPORTER FRAMEWORK

The appendix gives more details on how to interact with the FeaturesReporter Framework, described in in Section 3.2 and 5.1. In The FeaturesReporter class interface is outlined in Section A.1, usage of the ReportToDB mover is described in Section A.2, and the support for extracting features in parallel is described in Section A.3. The steps necessary to for implementing a FeaturesReporter for the Features reporter framework described in Section 3.2 include:

1. Implement the FeaturesReporter class interface (see directly below).
2. Add FeaturesReporter to FeaturesReporterFactory.
3. Add the FeaturesReporter to the FeaturesReporterTests Unit Test.
4. Consider adding the FeaturesReporter to the features integration test.
5. Document the FeaturesReporter in the Features Database Schema page.
6. Add new types in FeaturesReporters organizational page

A.1 FeaturesReporter Class Interface

The FeaturesReporter base class interface has the following components, to be implemented by FeaturesReporters:

A.1.1 Required Methods

- **type_name**: Returns the a string for the type of the feature reporter
- **schema**: Return SQL statements that setup tables in the database to contain the features. To support all database backends, use the schema generation framework.
- **report_features**: Extract all features to the database

A.1.2 Optional Methods

- **features_reporter_dependencies**: Returns a vector of the names of the features reporters that this one depends on. (In the Rosetta Scripts, the ReportToDB mover enforces this dependency by requiring the FeaturesReporters listed here to be defined higher in the list.)
- **parse_my_tag**: How in rosetta scripts the <Feature name=(& type_name string)/> subtag to the ReportToDB mover is parsed.
- **load_into_pose**: If the data is used to initialize an aspect of a pose, put the logic here.
- **delete_records**: Delete all records from the database associated with a structure.

As an example consider the `PoseCommentsFeatures` feature reporter shown below. Arbitrary textual information may be associated with a pose in the form of *(key, val)* comments. The `PoseCommentsFeatures FeaturesReporter` extracts all defined comments to a table `pose_comments` using the `struct_id` and key as the primary key.
The `struct_id` references the structures table that identifies each of the structures in the database.

In the `report_features` function, `sessionOP` is an owning pointer to the database where the features should be written. See the database interface for how to obtain and interact with database sessions.

```
string
PoseCommentsFeatures::type_name() const {
  return "PoseCommentsFeatures";
}

string
PoseCommentsFeatures::schema() const {
  Return
    "CREATE TABLE IF NOT EXISTS pose_comments (\n"
    " struct_id INTEGER,\n"     " key TEXT,\n"
    " value TEXT,\n"
    " FOREIGN KEY (struct_id)\n"
    " REFERENCES structures (struct_id)\n"
    " DEFERRABLE INITIALLY DEFERRED,\n"
    " PRIMARY KEY(struct_id, key));";
}

Size
PoseCommentsFeatures::report_features(
  Pose const & pose,
  Size struct_id,
  sessionOP db_session
){
  typedef map< string, string >::value_type kv_pair;
  foreach(kv_pair const & kv, get_all_comments(pose)){
    statement stmt = safely_prepare_statement(
      "INSERT INTO pose_comments VALUES (?,?,?)",
      db_session)
    stmt.bind(1, struct_id);
    stmt.bind(2, kv.first);
    stmt.bind(3, kv.second);
    safely_write_to_database(stmt);
  }
  return 0;
}
```

A FeaturesReporter may optionally be constructed with a ScoreFunction. For example, see the RotamerRecoveryFeatures class.

A.2 ReportToDB

Use the ReportToDB mover with the Rosetta XML scripting to specify which features should be extracted to the features database.

- <ReportToDB> tag
  - **name**: mover identifier so it can be included in the PROTOCOLS block of the RosettaScripts
  - Database Connection Options: options concerning how to connect to the database
  - **sample_source**: short text description stored in the *sample_source* table
  - **protocol_id**: (optional) specifies the *protocol_id* in the *protocols* table rather than auto-incrementing it.
  - **cache_size**: he maximum amount of memory to use before writing to the database (currently SQLite3 only)
  - **task_operations**: restricts extracting features to a relevant subset of residues. Since task operations are designed as tasks for side-chain remodeling, residue features are reported when the residue is "packable". If a features reporter involves more than one residue, the convention is that it is reported only if each residue is specified.

- <feature> tag (subtag of <ReportToDB>)
  - **name**: specifies the FeaturesReporter to include data in the database

Here is an example RosettaScripts that extracts structure scores for each input structure using the score12_w_corrections score function. Note the StructureScoresFeatures depends on the ScoreTypes features and will give an error if ScoreTypeFeatures does not come first.

```
<ROSETTASCRIPTS>
  <SCOREFXNS>
    <s weights=score12_w_corrections/>
  </SCOREFXNS>
  <MOVERS>
    <ReportToDB name=features database_name=scores.db3>
      <feature name=ScoreTypeFeatures/>
      <feature name=StructureScoresFeatures scfxn=s/>
    </ReportToDB>
  </MOVERS>
  <PROTOCOLS>
    <Add mover_name=features/>
  </PROTOCOLS>
</ROSETTASCRIPTS>
```

Since ReportToDB is simply a mover, it can be included in any RosettaScripts protocol. For example, to extract the features from a set of PDB files listed in `structures.list`, and the above script saved in `parser_script.xml`, execute the following command:

```
  rosetta_scripts.linuxgccrelease
    -output:nooutput
    -l structures.list
    -parser:protocol parser_script.xmL
```

This will generate an SQLite3 database file `scores.db3` containing the features defined in each of the specified FeaturesReporters for each structure in `structures.list`. See the features integration test for a working example.

A.3 Extracting Features In Parallel

Currently the ReportToDB mover is compatible with MPI runs for both client server database architectures (e.g. with MySQL or PostgreSQL) and partitioning a features database into separate databases that used as shards or merged together (e.g. with SQLite3). See the `features_parallel` integration test for a working example for writing to separate databases. To be concrete, consider an example where there 1000 structures split into 4 batches then the scripts for the run processing the first batch would contain:

```
<ReportToDB
  name=features_reporter
  db="features.db3_01"
  batch_description="batch1"
  protocol_id=1
  first_struct_id=1>
  ...
</ReportToDB>
```

and the script for the run processing the second batch would contain:

```
<ReportToDB
  name=features_reporter
  db="features.db3_02"
  batch_description="batch2"
  protocol_id=2
  first_struct_id=26>
  ...
</ReportToDB>
```

After the runs are complete, locate the merge.sh script in rosetta_tests/features/sample_sources/ and run

```
merge.sh features.db3 features.db3_*
```

which will merge the features from each of the `features.db3_xx` database into `features.db3`.

- **TIP 1**: Merging feature databases should be done for batches of structures that conceptually come from the same sample source. It is best to keep structures from different sample sources in separate databases and only during the analysis use the SQLite3 ATTACH statement to bring them together.
- **TIP 2**: If you run PostgreSQL, merging is not necessary. If you use SQLite3, merging is needed.
- **WARNING**: Extracting many databases in parallel generates high data transfer rates. This can be taxing on a cluster with a shared file system.

B FEATURES REPORTER CLASES

This appendix documents the FeaturesReporters that are currently implemented in Rosetta. They follow the organization described in Figure 3.2.3: Meta (B.1), Chemical (B.2), One-Body (B.3), Two-Body (B.4), Multi-Body (B.5), Multi-Structure (B.6), Energy (B.7), and Experimental (B.8). The live wiki documentation is located wiki.rosettacommons.org/index.php/FeaturesReporters with up-to-date documentation.

For each FeaturesReporter the documentation describes a description, special usage (if needed), and the tables managed and for each table notable columns and the SQL schema definition.

B.1 Meta Features

A meta features reporter reports information about the batch of structures and the protocol that was used to generate it. These are built into the ReportToDB mover and not extensible.

B.1.1 ProtocolFeatures

A protocol is represented as all the information necessary to reproduce the results of the Rosetta application execution. The features associated of each application execution are ultimately linked with a single row in the protocols table through the `BatchFeatures` reporter.

protocols:

- *command_line*: The complete command line used to execute Rosetta
- *specified_options*: The non-default options specified in the option system
- *svn_url*: The url for the SVN repository used for the Rosetta source code
- *svn_version*: The SVN revision number of the svn repository
  script: The contents of the rosetta_scripts XML script if run via a RosettaScripts protocol

```
CREATE TABLE IF NOT EXISTS protocols (
  protocol_id INTEGER PRIMARY KEY AUTOINCREMENT,
  command_line TEXT,
  specified_options TEXT,
  svn_url TEXT,
  svn_version TEXT,
  script TEXT);
```

batches:

```
CREATE TABLE IF NOT EXISTS batches (
  batch_id INTEGER PRIMARY KEY AUTOINCREMENT,
  protocol_id INTEGER,
  name TEXT,
  description TEXT,
  FOREIGN KEY (protocol_id) REFERENCES
    protocols(protocol_id) DEFERRABLE INITIALLY DEFERRED));
```

B.1.2 JobDataFeatures

Store *string*, *string-string*, and *string-real* data associated with a job. As an example, the ligand docking code uses this with the DatabaseJobOutputter.

job_string_data

- *data_key*: Associate labeled keys with a structure

```
CREATE TABLE IF NOT EXISTS string_data (
  struct_id INTEGER,
  data_key TEXT,
  FOREIGN KEY (struct_id) REFERENCES
    structures(struct_id) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY (struct_id, data_key));
```

job_string_string_data:

- *data_key*, *data_value*: Associate labeled text strings with a structure

```
CREATE TABLE IF NOT EXISTS string_string_data (
  struct_id INTEGER,
  data_key TEXT,
  data_value TEXT,
  FOREIGN KEY (struct_id) REFERENCES
    structures(struct_id) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY (struct_id, data_key));
```

job_string_real_data:

- *data_key*, *data_value*: Associate labeled, real numbers with a structure

```
CREATE TABLE IF NOT EXISTS string_real_data (
  struct_id INTEGER,
  data_key TEXT,
  data_value REAL,
  FOREIGN KEY (struct_id) REFERENCES
    structures(struct_id) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY (struct_id, data_key));
```

B.1.3 PoseCommentsFeatures

Arbitrary textual information may be associated with a pose in the form of *(key, val)* comments. The PoseCommentsFeatures stores this information as a feature.

pose_comments:
All pose comments are extracted.

```
CREATE TABLE IF NOT EXISTS pose_comments (
  struct_id INTEGER,
  key TEXT,
  value TEXT,
  FOREIGN KEY (struct_id) REFERENCES
    structures (struct_id) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY(struct_id, key));
```

B.2 Chemical Features

B.2.1 AtomTypeFeatures

```
<feature name=AtomTypeFeatures/>
```

This FeaturesReporter stores the atom-level chemical information stored in the Rosetta AtomTypeSet. This includes base parameters for the Lennard-Jones van der Waals term and Lazaridis Karplus solvation model.

atom_types:
The atom type in the atom type set along with Lennard-Jones and Lazaridis Karplus parameters

- *atom_type_set_name*: The name of the atom type set. For atom type sets stored in the database, the parameters are the following directory *rosetta_database/chemical/atom_type_sets/<atom_type_set_name>*
- *name*: The name of the atom type
- *-_jones_{radius/well_depth}*: The base parameters for the Lennard Jones van der Waals term.
- *lazaridis_karplus_{lambda, degrees_of_freedom, volume}*: The base parameters for the Lazardis Karplus solvation model.

```
CREATE TABLE IF NOT EXISTS atom_types (
  atom_type_set_name TEXT,
  name TEXT,
  element TEXT,
  lennard_jones_radius REAL,
  lennard_jones_well_depth REAL,
  lazaridis_karplus_lambda REAL,
  lazaridis_karplus_degrees_of_freedom REAL,
  lazaridis_karplus_volume REAL,
  PRIMARY KEY(atom_type_set_name, name));
```

atom_type_property_values:
Enumerates the valid properties that an atom can have. Each property is either true or false except the hybridization properties which is either UNKNOWN (represented as not present) or at most one of hybridization types.

- *property*: Valid properties for use in the *atom_type_properties* table (see below)

```
CREATE TABLE IF NOT EXISTS atom_type_property_values (
  property TEXT,
  PRIMARY KEY(property));

INSERT INTO atom_type_property_values VALUES ( 'ACCEPTOR' );
INSERT INTO atom_type_property_values VALUES ( 'DONOR' );
INSERT INTO atom_type_property_values VALUES ( 'POLAR_HYDROGEN' );
INSERT INTO atom_type_property_values VALUES ( 'AROMATIC' );
INSERT INTO atom_type_property_values VALUES ( 'H2O' );
INSERT INTO atom_type_property_values VALUES ( 'ORBITALS' );
INSERT INTO atom_type_property_values VALUES ( 'VIRTUAL' );
INSERT INTO atom_type_property_values VALUES ( 'SP2_HYBRID' );
INSERT INTO atom_type_property_values VALUES ( 'SP3_HYBRID' );
INSERT INTO atom_type_property_values VALUES ( 'RING_HYBRID' );
```

atom_type_properties:
The properties of a atom

- *atom_type_set_name*, *atom_type*: How the atom type is identified
- *property*: A foreign key in to the *atom_type_property_values* table (see above)

```
CREATE TABLE IF NOT EXISTS atom_type_properties (
  atom_type_set_name TEXT,
  name TEXT,
  property TEXT,
  FOREIGN KEY(atom_type_set_name, name) REFERENCES
    atom_types (atom_type_set_name, name) DEFERRABLE INITIALLY DEFERRED,
  FOREIGN KEY(property) REFERENCES
    atom_type_property_values (property) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY(atom_type_set_name, name));
```

atom_type_extra_parameters:
Extra numerical parameters that can be associated with an atom type

- *atom_type_set_name*, *atom_type*: How the atom type is identified
- *parameter*: The name of the parameter associated with the atom type
- *value*: The value of the parameter associated with the atom type

```
CREATE TABLE IF NOT EXISTS atom_type_extra_parameters (
  atom_type_set_name TEXT,
  name TEXT,
  parameter TEXT,
  value REAL,
  FOREIGN KEY(atom_type_set_name, name) REFERENCES
    atom_types (atom_type_set_name, name) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY(atom_type_set_name, name));
```

B.2.2 ResidueTypesFeatures

ResidueTypes store information about the chemical nature of the residue. The information is read in from the from *rosetta_database/chemical/residue_type_sets/<residue_type_set_name>/-residue_types/*.

residue_type:

- *residue_type_set_name*: The name of the ResidueTypeSet. e.g. *fa_standard* or *centroid*
- *name*: The unique string that identifies a residue type in the ResidueTypeSet
- *name3*: Three letter abbreviation for the residue type
- *name1*: One letter abbreviation for the residue type
- *aa*: Three letter abbreviation for the residue type, where non-canonical residues are *UNK*.
- *lower_connect*, *upper_connect*: The atoms that connect the residue with the rest of the residues.
- *nbr_atom*: The atom used for neighbor calculations
- *nbr_radius*: A measure of the size of a residue for neighbor calculations.

```
CREATE TABLE IF NOT EXISTS residue_type (
  residue_type_set_name TEXT,
  name TEXT,
  version TEXT,
  name3 TEXT,
  name1 TEXT,
  aa TEXT,
```

```
  lower_connect INTEGER,
  upper_connect INTEGER,
  nbr_atom INTEGER,
  nbr_radius REAL,
  PRIMARY KEY(residue_type_set_name, name));
```

residue_type_atom:
Each atom in the residue type is identified.

- *residue_type_set_name*: The name of the ResidueTypeSet e.g. *fa_standard* or *centroid*.
- *atom_index*: The index of the atom in the residue.
- *atom_name*: The name of the atom in the residue following the PDB naming convention. e.g. in canonical amino acids, the C-beta atom is *CB1*.
- *atom_type_name*: The name of the atom type of the atom. e.g. in canonical amino acids, the C-beta atom is ' *CB* '. Note: all atom names are exactly 4 characters.
- *mm_atom_type_name*: The molecular mechanics name of the atom in the CHARMM naming scheme. e.g. in canonical amino acids, the C-beta atom is *CT2*.
- *charge*: The amount of charge associated with the atom.
- *is_backbone*: Is the atom part of the backbone?

```
CREATE TABLE IF NOT EXISTS residue_type_atom (
  residue_type_set_name TEXT,
  residue_type_name TEXT,
  atom_index INTEGER,
  atom_name TEXT,
  atom_type_name TEXT,
  mm_atom_type_name TEXT,
  charge REAL,
  is_backbone INTEGER,
  FOREIGN KEY(residue_type_set_name, residue_type_name) REFERENCES
    residue_type(residue_type_set_name, name) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY(residue_type_set_name, residue_type_name, atom_index));
```

residue_type_bond:
The covalent bonds in the residue type

- residue_type_set_name: The name of the ResidueTypeSet e.g. fa_standard or centroid.
- atom1, atom2: The atoms participating in the bond, where the atom index of atom1 is less than the atom index of atom2.
- bond_type: The type of chemical bond.

```
CREATE TABLE IF NOT EXISTS residue_type_bond (
  residue_type_set_name TEXT,
  residue_type_name TEXT,
  atom1 INTEGER,
  atom2 INTEGER,
  bond_type INTEGER,
  FOREIGN KEY(residue_type_set_name, residue_type_name) REFERENCES
    residue_type(residue_type_set_name, name) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY(residue_type_set_name, residue_type_name, atom1, atom2));
```

residue_type_cut_bond:
Covalent bonds that that form non-tree topologies, e.g. (CD-N) in proline and (CE1-CZ) in tyrosine.

- *residue_type_set_name*: The name of the ResidueTypeSet e.g. *fa_standard* or *centroid*.

- *atom1*, *atom2*: The atoms participating in the cut bond, where the atom index of *atom1* is less than the atom index of *atom2*.

```
CREATE TABLE IF NOT EXISTS residue_type_cut_bond (
  residue_type_set_name TEXT,
  residue_type_name TEXT,
  atom1 INTEGER,
  atom2 INTEGER,
  FOREIGN KEY(residue_type_set_name, residue_type_name) REFERENCES
    residue_type(residue_type_set_name, name) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY(residue_type_set_name, residue_type_name, atom1, atom2));
```

residue_type_chi:
The chi torsional degrees of freedom in the ResidueType

- *residue_type_set_name*: The name of the ResidueTypeSet e.g. *fa_standard* or *centroid*.
- *chino*: The index of the chi degree of freedom
- *atom1*, *atom2*, *atom3*, *atom3*, *atom4*: The atoms that define the torsional degree of freedom

```
CREATE TABLE IF NOT EXISTS residue_type_chi (
  residue_type_set_name TEXT,
  residue_type_name TEXT,
  atom1 TEXT,
  atom2 TEXT,
  atom3 TEXT,
  atom4 TEXT,
  chino INTEGER,
  FOREIGN KEY(residue_type_set_name, residue_type_name) REFERENCES
    residue_type(residue_type_set_name, name) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY(residue_type_set_name, residue_type_name, atom1, atom2));
```

residue_type_chi_rotamer:
Chi torsional degrees of freedom are binned into discrete rotamer conformations. Each row is a bin for a chi torisional degree of freedom.

- *residue_type_set_name*: The name of the ResidueTypeSet e.g. *fa_standard* or *centroid*.
- *chino*: The index of the chi torsional degree of freedom
- *mean*: The center of the angle bin
- *sdev*: The standard deviation of the bin about the *mean*

```
CREATE TABLE IF NOT EXISTS residue_type_chi_rotamer (
  residue_type_set_name TEXT,
  residue_type_name TEXT,
  chino INTEGER,
  mean REAL,
  sdev REAL,
  FOREIGN KEY(residue_type_set_name, residue_type_name) REFERENCES
    residue_type(residue_type_set_name, name) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY(residue_type_set_name, residue_type_name, chino, mean, sdev));
```

residue_type_proton_chi:
Chi torsional degrees of freedom controlling the coordinates of hydrogen atoms.

- *residue_type_set_name*: The name of the ResidueTypeSet e.g. *fa_standard* or *centroid*.
- *chino*: The index of the chi torsional degree of freedom
- *sample*: The sample angle plus or minus each extra sample angle. e.g. The hydroxyl hydrogen in tyrosine is controlled by the third chi torsional degree of freedom with two

torsional bins, trans at 180 degrees and cis at 0 degrees, both in the plane of the aromatic ring. To increase conformational sampling 8 extra rotamer bins +/- 1 degree and +/- 20 degrees for each sample bin.

- *extra*: Is this sample bin an extra rotamer bin?

```
CREATE TABLE IF NOT EXISTS residue_type_proton_chi (
  residue_type_set_name TEXT,
  residue_type_name TEXT,
  chino INTEGER,
  sample REAL,
  is_extra INTEGER,
  FOREIGN KEY(residue_type_set_name, residue_type_name) REFERENCES
    residue_type(residue_type_set_name, name) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY(residue_type_set_name, residue_type_name, chino, sample));
```

residue_type_property:

- *residue_type_set_name*: The name of the ResidueTypeSet e.g. *fa_standard* or *centroid*.
- *property*: Properties associated with the ResidueType e.g. PROTEIN, POLAR, or SC_ORBITALS

```
CREATE TABLE IF NOT EXISTS residue_type_property (
  residue_type_set_name TEXT,
  residue_type_name TEXT,
  property TEXT,
  FOREIGN KEY(residue_type_set_name, residue_type_name) REFERENCES
    residue_type(residue_type_set_name, name) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY(residue_type_set_name, residue_type_name, property));
```

residue_type_variant_type:

- *residue_type_set_name*: The name of the ResidueTypeSet e.g. *fa_standard* or *centroid*.
- *variant_type*: Variant types associated with the ResidueType, e.g. DEPROTONATED, DISULFIDE, or MODRE

```
CREATE TABLE IF NOT EXISTS residue_type_variant_type (
  residue_type_set_name TEXT,
  residue_type_name TEXT,
  variant_type TEXT,
  FOREIGN KEY(residue_type_set_name, residue_type_name) REFERENCES
    residue_type(residue_type_set_name, name)DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY(residue_type_set_name, residue_type_name, variant_type));
```

B.2.3 UnrecognizedAtomFeatures

```
  <feature
    name=UnrecognizedAtomFeatures
    neighbor_distance_cutoff=(&Real 12.0)/>
```

UnrecognizedAtom store information about unrecognized atoms. This information is stored in the PDBInfo and is usually populated when there is a residue in a PDB file that does not match any recognized Residue parameter files that is saved with the *-in:remember_unrecognized_res* flag.

unrecognized_residues:
Details about the unrecognized residues

- *residue_number*: The residue number of the unrecognized residue. NOTE: Unrecognized residues are not stored in the residues table.
- *name3*: Three letter abbreviation for the residue type.

- *max_temperature*: The highest B-factor for any atom in the unrecognized residue. The occupancy could also be added here as well.

```
CREATE TABLE unrecognized_residues(
  struct_id INTEGER AUTOINCREMENT,
  residue_number INTEGER,
  name3 TEXT,
  max_temperature REAL,
  FOREIGN KEY (struct_id) REFERENCES
    structures(struct_id) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY (struct_id, residue_number));
```

unrecognized_atoms:
Atomic details about unrecognized residues

- *atom_name*: e.g. the pdb atom name column
- *coord_{x,y,z}*: The atomic coordinates

```
CREATE TABLE unrecognized_atoms(
  struct_id INTEGER AUTOINCREMENT,
  residue_number INTEGER,
  atom_name TEXT,
  coord_x REAL,
  coord_y REAL,
  coord_z REAL,
  temperature REAL,
  FOREIGN KEY (struct_id) REFERENCES
    structures(struct_id) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY (struct_id, residue_number, atom_name));
```

unrecognized_neighbors:
Close contacts between residues and unrecognized residues, this can be used as a filter for gathering statistics in bulk from the protein databank without representing ligands etc appropriately.

- *closest_contact*: Distance between the ACTCOORD in the residue and the closest atom in each unrecognized residue. Only saves contacts that are within *neighbor_-distance_cutoff*, which defaults to 12A.

```
CREATE TABLE unrecognized_neighbors(
  struct_id INTEGER AUTOINCREMENT NOT NULL,
  residue_number INTEGER NOT NULL,
  unrecognized_residue_number REAL NOT NULL,
  closest_contact REAL NOT NULL,
  FOREIGN KEY (struct_id, residue_number) REFERENCES
    residues(struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY (struct_id, residue_number));
```

B.3 One-Body Features

B.3.1 ResidueFeatures

The ResidueFeatures stores information about each residue in a conformation.

residues:
This connects the sequence position with the name of the residue and the ResidueType of the residue.

- *name3*: The three letter amino acid code. If it is not a canonical amino acid it is *UNK*.
- *res_type*: The unique identifier for the ResidueType of the residue.

```
CREATE TABLE IF NOT EXISTS residues (
  struct_id INTEGER AUTOINCREMENT,
  resNum INTEGER,
  name3 TEXT,
  res_type TEXT,
  FOREIGN KEY (struct_id) REFERENCES
    structures (struct_id) DEFERRABLE INITIALLY DEFERRED,
  CONSTRAINT resNum_is_positive CHECK (resNum >= 1),
  PRIMARY KEY(struct_id, resNum));
```

B.3.2 ResidueConformationFeatures

Store the geometry of residues that have canonical backbones but possibly non-canonical sidechains. The geometry is broken into backbone torsional degrees of freedom, *nonprotein_residue_conformation*, sidechain degrees of freedom, *nonprotein_residue_angles*, and atomic coordinates, *residue_atom_coords*.

This differs from ProteinResidueConformationFeatures (B.3.3) in that the residue angles are stored as a *chinum -> chiangle* lookup and atomic xyz-coordinates, rather than a table with slots for 4 chi values. If you know you working only with protein residues, you can conserve space by using the ProteinResidueConformationFeatures.

nonprotein_residue_conformation:

- *phi*, *psi*, *omega*: Angles of backbone torsional degrees of freedom

```
CREATE TABLE IF NOT EXISTS nonprotein_residue_conformation (
  struct_id INTEGER AUTOINCREMENT,
  seqpos INTEGER,
  phi REAL,
  psi REAL,
  omega REAL,
  FOREIGN KEY (struct_id, seqpos) REFERENCES
    residues (struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY (struct_id, seqpos));
```

nonprotein_residue_angles:

- *chinum*: The index of the chi torsional degree of freedom in the sidechain of the residue
- *chiangle*: The angle of the chi torsional degree of freedom in the sidechain of the residue

```
CREATE TABLE IF NOT EXISTS nonprotein_residue_angles (
  struct_id INTEGER AUTOINCREMENT,
  seqpos INTEGER,
  chinum INTEGER,
```

```
  chiangle REAL,
  FOREIGN KEY (struct_id, seqpos) REFERENCES
    residues (struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY (struct_id, seqpos));
```

residue_atom_coords:

- *x*, *y*, *z*: Spatial coordinates of atom *atomno* in residue *seqpos* in the lab coordinate frame.

```
CREATE TABLE IF NOT EXISTS residue_atom_coords (
  struct_id INTEGER AUTOINCREMENT,
  seqpos INTEGER,
  atomno INTEGER,
  x REAL, y REAL,
  z REAL,
  FOREIGN KEY (struct_id, seqpos) REFERENCES
    residues (struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY (struct_id, seqpos, atomno));
```

B.3.3 ProteinResidueConformationFeatures

The conformation of protein residues is described by the coordinates of each atom. A reduced representation just specifies the values for each torsional angle degree of freedom, including the backbone and sidechain torsional angles. Since Proteins have only canonical amino acids, sidechains have at most 4 torsional angles.

protein_residue_conformation:
The degrees of freedom in each residue conformation.

- *secstruct*: The secondary structure of the residue. NOTE: this is not computed by DSSP but taken from fragments. See Pose::secstruct().
- *phi*, *psi*, *omega*: Backbone torsional angles
- *chi**: Sidechain torsional angles

```
CREATE TABLE IF NOT EXISTS protein_residue_conformation (
  struct_id INTEGER AUTOINCREMENT,
  seqpos INTEGER,
  secstruct STRING,
  phi REAL,
  psi REAL,
  omega REAL,
  chi1 REAL,
  chi2 REAL,
  chi3 REAL,
  chi4 REAL,
  FOREIGN KEY (struct_id, seqpos) REFERENCES
    residues (struct_id, resNum) DEFERRABLE INITIALLY DEFERRED);
```

residue_atom_coords:
The atomic coordinates for each residue. Note if all of the residues are ideal, then this table is not populated.

- *x*, *y*, *z*: Atomic coordinates in lab coordinate frame.

```
CREATE TABLE IF NOT EXISTS residue_atom_coords (
  struct_id INTEGER AUTOINCREMENT,
  seqpos INTEGER,
  atomno INTEGER,
  x REAL,
  y REAL,
```

```
  z REAL,
  FOREIGN KEY (struct_id, seqpos) REFERENCES
    residues (struct_id, resNum) DEFERRABLE INITIALLY DEFERRED);
```

B.3.4 ProteinBackboneTorsionAngleFeatures

The ProteinBackboneTorsionAngleFeatures stores the backbone torsion angle degrees of freedom needed represent proteins made with canonical backbones.

protein_backbone_torsion_angles:

- *phi*: The torsion angle defined by *C_(i-1)*, *N_i*, *Ca_i*, and *C_i* atoms (-180, 180)
- *psi*: The torsion angle defined by *N_i*, *Ca_i*, *C_i*, and *N_(i+1)* atoms (-180, 180)
- *omega*: The torsion angle defined by *Ca_i*, *C_i*, *N_(i+1)*, and *Ca_(i+1)* (-180, 180)

```
CREATE TABLE IF NOT EXISTS protein_backbone_torsion_angles (
  struct_id INTEGER AUTOINCREMENT,
  resNum INTEGER,
  phi REAL,
  psi REAL,
  omega REAL,
  FOREIGN KEY (struct_id, resNum) REFERENCES
    residues (struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY (struct_id, resNum));
```

B.3.5 ProteinBondGeometryFeatures

The ProteinBondGeometryFeatures reporter stores bond-angle, bond-length, and bond-torsion for canonical protein amino acids.

Let *i* be an atom number of given residue and let bonded_neighbors(*i*) be the set of bonded neighbors of atom *i*. Then if *j*, *k* are in bonded_neighbors(*i*) such that $j < k$, then (*j*, *i*, *k*) is a bond angle. In other words, there is a row in bond_intrares_angles such that outAtm1Num = *j*, cenAtmNum = *i*, and outAtm2Num = *k*.

bond_intrares_angles:

- *cenAtmNum*: atom number of the center atom defining the bond angle
- *outAtm{1,2}Num*: atom numbers of the outer atoms defining the bond angle
- *cenAtmName*: the atom name of the center atom
- *outAtm{1,2}Name*: the atom names of the outer atoms
- ideal: ideal angle defined by -scoring:bonded_params or by default in chemical/mm_atom_type_sets/fa_standard/par_all27_prot_na.prm
- *observed*: actual angle in structure
- *difference*: angle deviation from ideal angle
- *energy*: a harmonic potential away from the ideal angle with the spring constant defined by the residue type, and atom identities. - NOTE: THIS CODE IS NOT UP TO DATE WITH CURRENT CART_BONDED ENERGY CALCULATIONS, RESULTS WILL DIFFER

```
CREATE TABLE IF NOT EXISTS bond_intrares_angles (
  struct_id INTEGER AUTOINCREMENT,
  resNum INTEGER,
  cenAtmNum INTEGER,
```

```
  outAtm1Num INTEGER,
  outAtm2Num INTEGER,
  cenAtmName TEXT,
  outAtm1Name TEXT,
  outAtm2Name TEXT,
  ideal REAL,
  observed REAL,
  difference REAL,
  energy REAL,
  FOREIGN KEY (struct_id, resNum) REFERENCES
    residues (struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY (struct_id, resNum, cenAtmNum, outAtm1Num, outAtm2Num));
```

bond_interres_angles:

- cenResNum:
- connResNum:
- cenAtmNum:
- outAtm{Cen,Conn}Num:
- cenAtmName:
- outAtm{Cen,Conn}Name:
- ideal: ideal angle defined by -scoring:bonded_params or by default in chemical/mm_atom_type_sets/fa_standard/par_all27_prot_na.prm
- *observed*: actual angle in structure
- *difference*: angle deviation from ideal angle
- *energy*: a harmonic potential away from the ideal angle with the spring constant defined by the residue type, and atom identities. - NOTE: THIS CODE IS NOT UP TO DATE WITH CURRENT CART_BONDED ENERGY CALCULATIONS, RESULTS WILL DIFFER

```
CREATE TABLE IF NOT EXISTS bond_interres_angles (
  struct_id INTEGER AUTOINCREMENT,
  cenResNum INTEGER,
  connResNum INTEGER,
  cenAtmNum INTEGER,
  outAtmCenNum INTEGER,
  outAtmConnNum INTEGER,
  cenAtmName TEXT,
  outAtmCenName TEXT,
  outAtmConnName TEXT,
  ideal REAL,
  observed REAL,
  difference REAL,
  energy REAL,
  FOREIGN KEY (struct_id, cenResNum) REFERENCES
    residues (struct_id, cenResNum) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY (
    struct_id, cenResNum, connResNum,
    cenAtmNum, outAtmCenNum, outAtmConnNum));
```

bond_intrares_lengths:

- *atm{1,2}Num*: atom numbers of atoms that neighbors, usually because they are covalently bound
- *atm{1,2}Name*: the names of the atoms
- ideal: ideal length defined by -scoring:bonded_params or by default in chemical/mm_atom_type_sets/fa_standard/par_all27_prot_na.prm

- *observed*: actual length in structure
- *difference*: angle deviation from ideal length
- *energy*: a harmonic potential away from the ideal length with the spring constant defined by the residue type, and atom identities. - NOTE: THIS CODE IS NOT UP TO DATE WITH CURRENT CART_BONDED ENERGY CALCULATIONS, RESULTS WILL DIFFER

```
CREATE TABLE IF NOT EXISTS bond_intrares_lengths (
  struct_id INTEGER AUTOINCREMENT,
  resNum INTEGER,
  atm1Num INTEGER,
  atm2Num INTEGER,
  atm1Name TEXT,
  atm2Name TEXT,
  ideal REAL,
  observed REAL,
  difference REAL,
  energy REAL,
  FOREIGN KEY (struct_id, resNum) REFERENCES
    residues (struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY (struct_id, resNum, atm1Num, atm2Num));
```

bond_interres_lengths:

- *res{1,2}Num*: Two residues that have atoms that are bound together
- *atm{1,2}Num*: atom numbers of atoms that neighbors, one from each residue, usually because they are covalently bound
- *atm{1,2}Name*: the names of the atoms
- ideal: ideal length defined by -scoring:bonded_params or by default in chemical/mm_atom_type_sets/fa_standard/par_all27_prot_na.prm
- *observed*: actual length in structure
- *difference*: angle deviation from ideal length
- *energy*: a harmonic potential away from the ideal length with the spring constant defined by the residue type, and atom identities. - NOTE: THIS CODE IS NOT UP TO DATE WITH CURRENT CART_BONDED ENERGY CALCULATIONS, RESULTS WILL DIFFER

```
CREATE TABLE IF NOT EXISTS bond_interres_lengths (
  struct_id INTEGER AUTOINCREMENT,
  res1Num INTEGER,
  res2Num INTEGER,
  atm1Num INTEGER,
  atm2Num INTEGER,
  atm1Name TEXT,
  atm2Name TEXT,
  ideal REAL,
  observed REAL,
  difference REAL,
  energy REAL,
  FOREIGN KEY (struct_id, res1Num) REFERENCES
    residues (struct_id, res1Num) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY (struct_id, res1Num, atm1Num, atm2Num));
```

bond_intrares_torsions:

```
CREATE TABLE IF NOT EXISTS bond_intrares_torsions (
  struct_id INTEGER AUTOINCREMENT,
  resNum INTEGER,
  atm1Num INTEGER,
```

```
atm2Num INTEGER,
atm3Num INTEGER,
atm4Num INTEGER,
atm1Name TEXT,
atm2Name TEXT,
atm3Name TEXT,
atm4Name TEXT,
ideal REAL,
observed REAL,
difference REAL,
energy REAL,
FOREIGN KEY (struct_id, resNum) REFERENCES
  residues (struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
PRIMARY KEY (struct_id, resNum, atm1Num, atm2Num, atm3Num, atm4Num));
```

B.3.6 RotamerFeatures

The RotamerFeatures reporter stores the predicted sidechain conformation given the backbone torsion angles and the observed deviation away from the ideal torsion angles away for the rotamer. Currently this is restricted to canonical amino acids and use with the Dunbrack library.

The *dun08* and *dun10* libraries define semi-rotameric conformations

for *ASP*, *GLU*, *PHE*, *HIS*, *ASN*, *GLN*, *TRP*, and *TYR* where the last torsion angle is treated as a continuous variable.

For the *dun02* library see

- Dunbrack RL, Cohen FE. Bayesian statistical analysis of protein side-chain rotamer preferences. Protein science: a publication of the Protein Society. 1997;6(8):1661-81.
- Dunbrack RL. Rotamer libraries in the 21st century. Current opinion in structural biology. 2002; 12(4):431–440.

For the *dun10* library see

- Shapovalov MV, Dunbrack RL. A smoothed backbone-dependent rotamer library for proteins derived from adaptive kernel density estimates and regressions. Structure (London, England : 1993). 2011; 19(6):844-58.

residue_rotamers:

- *rotamer_bin*: The dunbrack library divides sidechain conformation space into discrete rotameric conformations. For fully rotameric conformations this is based on all the sidechain torsion angles. for semi-rotameric conformations this is based on all but the last sidechain torsion angles.
- *nchi*: The number of rotameric torsion angles in the residue. For example *nchi* for tyrosine is 1 in the *dun10* library.
- *semi_rotameric*: Boolean value, true if the sidechain is a semi-rotameric amino acid.
- *chi{1,2,3,4}_mean*: The expected value of the rotameric torsion angles given the backbone conformation and semi-rotameric torsion angles (if semi-rotameric). This is bilinear/trilinear interpolated data recorded on 10 degree bins.
- *chi{1,2,3,4}_standard_deviation*: The standard deviation of the rotameric torsion angles given the backbone conformation and semi-rotameric torsion angles (if semi-rotameric).
- *chi{1,2,3,4}_deviation*: The angle deviation away from the mean.
- *rotamer_bin_probability*: The probability of being in the rotamer bin.

```
CREATE TABLE residue_rotamers IF NOT EXISTS (
  struct_id INTEGER AUTOINCREMENT,
  residue_number INTEGER,
  rotamer_bin INTEGER,
  nchi INTEGER,
  semi_rotameric INTEGER,
  chi1_mean REAL,
  chi2_mean REAL,
  chi3_mean REAL,
  chi4_mean REAL,
  chi1_standard_deviation REAL,
  chi2_standard_deviation REAL,
  chi3_standard_deviation REAL,
  chi4_standard_deviation REAL,
  chi1_deviation REAL,
  chi2_deviation REAL,
  chi3_deviation REAL,
  chi4_deviation REAL,
  rotamer_bin_probability REAL,
  FOREIGN KEY (struct_id, resNum) REFERENCES
    residues (struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY (struct_id, residue_number));
```

B.3.7 ResidueBurialFeatures

Measures of burial are important for determining solvation and desolvation effects.

residue_burial:

- *ten_a_neighbors*: The number of residues within 10 Angstroms (not counting the residue itself. The distance is measured from the NBR_ATOM in the residue type parameter file, which is usually the C-beta atom.
- *twelve_a_neighbors*: The number of residues within 12 Angstroms.
- *neigh_vect_raw*: The length of the average displacement of neighboring residues. The region of inclusion is set by the options *score:NV_lbound* and *score:NV_ubound*, defaulting to 3.3 and 11.1 Angstroms. This follows the Durham E, et al. Solvent Accessible Surface Area Approximations for Protein Structure Prediction.
- *sasa_r100*, *sasa_r140*, *sasa_200*: The solvent accessible surface area with different sizes of probes (1.0Å, 1.4Å, 2.0Å).

```
CREATE TABLE IF NOT EXISTS residue_burial (
  struct_id INTEGER AUTOINCREMENT,
  resNum INTEGER,
  ten_a_neighbors INTEGER,
  twelve_a_neighbors INTEGER,
  neigh_vect_raw REAL,
  sasa_r100 REAL,
  sasa_r140 REAL,
  sasa_r200 REAL,
  FOREIGN KEY (struct_id, resNum) REFERENCES
    residues (struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY (struct_id, resNum));
```

B.3.8 ResidueSecondaryStructureFeatures

Secondary structure is a classification scheme for residues that participate in regular, multi-residue interactions.

residue_secondary_structure:

- *dssp*: The Dictionary of Secondary Structure classification scheme following Kabsch and Sander, Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. The coding is described http://swift.cmbi.ru.nl/gv/dssp.

```
CREATE TABLE IF NOT EXISTS residue_secondary_structure(
  struct_id INTEGER AUTOINCREMENT,
  resNum INTEGER,
  dssp TEXT,
  FOREIGN KEY(struct_id, resNum) REFERENCES
    residues(struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY(struct_id, resNum));

CREATE TABLE dssp_codes(
  code TEXT NOT NULL,
  label TEXT NOT NULL,
  PRIMARY KEY (code));

INSERT INTO "dssp_codes" VALUES('H','H: a-Helix');
INSERT INTO "dssp_codes" VALUES('E','E: b-Sheet');
INSERT INTO "dssp_codes" VALUES('T','T: HB Turn');
INSERT INTO "dssp_codes" VALUES('G','G: 3/10 Helix');
INSERT INTO "dssp_codes" VALUES('B','B: b-Bridge');
INSERT INTO "dssp_codes" VALUES('S','S: Bend');
INSERT INTO "dssp_codes" VALUES('I','I: pi-Helix');
INSERT INTO "dssp_codes" VALUES(' ','Irregular');
```

B.3.9 BetaTurnDetectionFeatures

```
<feature name=BetaTurnDetectionFeatures/>
```

This reporter scans all available windows of four residues and determines if a β-turn is present, determines the type of β-turn and then writes the starting residue number and turn type to a database.

beta_turns:

```
CREATE TABLE IF NOT EXISTS beta_turns (
  struct_id INTEGER AUTOINCREMENT,
  residue_begin INTEGER,
  turn_type TEXT,
  FOREIGN KEY (struct_id, residue_begin) REFERENCES
    residues (struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY(struct_id, residue_begin));
```

B.3.10 RotamerBoltzmannWeightFeatures

Measure how constrained each residue is, following Fleishman, Khare, Koga, & Baker, "Restricted sidechain plasticity in the structures of native proteins and complexes" (Fleishman 2011).

rotamer_boltzmann_weight:

- *boltzmann_weight*: Compute the energy e_i for each rotamer minimized in a fixed environment. If E is the energy of the whole structure and the temperature, $T = .8$, then $boltzmann\_weight = \frac{1}{\sum_i e^{\frac{E-e_i}{T}}}$.

```
CREATE TABLE IF NOT EXISTS rotamer_boltzmann_weight (
  struct_id INTEGER AUTOINCREMENT,
  resNum INTEGER,
  boltzmann_weight REAL,
  FOREIGN KEY (struct_id, resNum) REFERENCES
    residues (struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY (struct_id, resNum));
```

B.4 Two-Body Features

B.4.1 PairFeatures

The PairFeatures measures the distances between residues.

residue_pairs:
The information stored here follows 'pair' EnergyMethod. The functional form for the pair EnergyMethod is described in Simons, K.T., et al, Improved Recognition of Native-Like Protein Structures Using a Combination of Sequence-Dependent and Sequence-Independent Features of Proteins, (Proteins 1999).

- *resNum{1/2}*: the rosetta Residue indices of residues involved. Note, each pair is only recorded once and resNum1 < resNum2.
- *res{1/2}_10A_neighbors*: Number of neighbors for each residue, used as a proxy for burial. (These columns are going to be moved to the **residue_burial** table soon.)
- A *residue center* is represented by the actcoord, which is defined to be the average geometric center of of the ACT_COORD_ATOMS specified in the residue type params file for each residue type.
- *actcoord_dist*: The Cartesian distance between residue centers.
- *polymeric_sequence_dist*: The sequence distance between the residues. If either residue is not a *polymer* residue or if they are on different chains, this is -1.

```
CREATE TABLE IF NOT EXISTS residue_pairs (
  struct_id INTEGER,
  resNum1 INTEGER,
  resNum2 INTEGER,
  res1_10A_neighbors INTEGER,
  res2_10A_neighbors INTEGER,
  actcoord_dist REAL,
  polymeric_sequence_dist INTEGER,
  FOREIGN KEY (struct_id, resNum1) REFERENCES
    residues (struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  FOREIGN KEY (struct_id, resNum2) REFERENCES
    residues (struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  CONSTRAINT res1_10A_neighbors_greater_than CHECK
    (res1_10A_neighbors >= 1),
  CONSTRAINT res2_10A_neighbors_greater_than CHECK
    (res2_10A_neighbors >= 1));
```

B.4.2 AtomAtomPairFeatures

```
<feature
  name=AtomAtomPairFeatures
  min_dist=(&real 0)
  max_dist=(&real 10)
  nbins=(&integer 15)/>
```

The distances between pairs of atoms together form indicate the packing of a structure. Since there are a large number of atom pairs, here, the information is summarized by atom pair distributions for each pair of atom types (Rosetta AtomType -> element type). See AtomInResidueAtomInResiduePairFeatures for an alternative binning of atom-atom interactions.

atom_pairs:
Binned distribution of pairs of types of atoms

- *atom_type*: The AtomType of the central atom. This is a subset of the AtomTypes defined in the full-atom AtomTypeSet atom_properties.txt: *CAbb*, *CObb*, *OCbb*, *CNH2*, *COO*,*CH1*, *CH2*, *CH3*, *aroC*, *Nbb*, *Ntrp*, *Nhis*, *NH2O*, *Nlys*, *Narg*, *Npro*, *OH*, *ONH2*, *OOC*, *Oaro*, *Hpol*, *Hapo*, *Haro*, *HNbb*, *HOH*, and *S*.
- *element*: The element type of the second atom: *C*, *N*, *O*, and *H*
- *{lower/upper}_break*: The boundaries for the distance bin
- *count*: The number of atom-atom instances of the correct type that occur in the specific distance range in the structure.

```
CREATE TABLE IF NOT EXISTS atom_pairs (
  struct_id INTEGER,
  atom_type TEXT,
  element TEXT,
  lower_break REAL,
  upper_break REAL,
  count INTEGER,
  FOREIGN KEY (struct_id) REFERENCES
    structures (struct_id) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY (struct_id, atom_type, element, lower_break));
```

B.4.3 AtomInResidueAtomInResiduePairFeatures

The distances between pairs of atoms is an indicator of the packing of a structure. Since there are a large number of atom pairs, here, the information is summarized by atom pair distributions for each pair of atom types (residue type + atom number). This is very similar in spirit to (Lu 2001), however, they use different distance bins. Here, (0,1], ..., (9,10] are used because they are easy. It may make sense to come up with a better binning upon further analysis. The molar fraction of atom types can be computed by joining with the Residues table since the types are unique within each residue type. If this is turns out to be too cumbersome, it may need to be pre-computed. **WARNING**: Currently, this generates an inordinate amount of data!!! ~250M per structure.

atom_in_residue_pairs:
Binned distribution of pairs of types of atoms

- *residue_type1*, *atom_type1*: The ResidueType and atom number for the first atom type
- *residue_type2*, *atom_type2*: The ResidueType and atom number for the second atom type
- *distance_bin*: Group all atom pairs in the range (distance_bin-1, distance_bin]
- *count*: Number of atom pairs in the distance bin

```
CREATE TABLE IF NOT EXISTS atom_in_residue_pairs (
  struct_id INTEGER,
  residue_type1 TEXT,
  atom_type1 TEXT,
  residue_type2 TEXT,
  atom_type2 TEXT,
  distance_bin TEXT,
```

```
  count INTEGER,
  FOREIGN KEY (struct_id) REFERENCES
    structures (struct_id) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY (
    struct_id, residue_type1, atom_type1,
    residue_type2, atom_type2, distance_bin),
  CONSTRAINT count_greater_than CHECK (count >= 0));
```

B.4.4 ProteinBackboneAtomAtomPairFeatures

The ProteinBackboneAtomAtomPairFeatures reporter measures all the atom pair distances between backbone atoms in pairs residues where the action coordinate is within 10 Å. This follows the analysis done in *Song Y, Tyka M, Leaver-Fay A, Thompson J, Baker D. Structure guided forcefield optimization. Proteins: Structure, Function, and Bioinformatics. 2011*. There, they looked at these distances for pairs of residues that form secondary structure.

protein_backbone_atom_atom_pairs:

- *resNum{1,2}*: The indices of the protein residues. Note: resNum1 < resNum2.
- *{N,Ca,C,O,Ha}_{N,Ca,C,O,Ha}_dist*: The distance between the *N*, *Cα*, *C*, *O* or *Hα* atom on the first residue to the *N*, *Cα*, *C*, *O* or *Hα* atom on the second residue.

```
CREATE TABLE IF NOT EXISTS protein_backbone_atom_atom_pairs (
  struct_id TEXT,
  resNum1 INTEGER,
  resNum2 INTEGER,
  N_N_dist REAL,
  N_Ca_dist REAL,
  N_C_dist REAL,
  N_O_dist REAL,
  N_Ha_dist REAL,
  Ca_N_dist REAL,
  Ca_Ca_dist REAL,
  Ca_C_dist REAL,
  Ca_O_dist REAL,
  Ca_Ha_dist REAL,
  C_N_dist REAL,
  C_Ca_dist REAL,
  C_C_dist REAL,
  C_O_dist REAL,
  C_Ha_dist REAL,
  O_N_dist REAL,
  O_Ca_dist REAL,
  O_C_dist REAL,
  O_O_dist REAL,
  O_Ha_dist REAL,
  Ha_N_dist REAL,
  Ha_Ca_dist REAL,
  Ha_C_dist REAL,
  Ha_O_dist REAL,
  Ha_Ha_dist REAL,
  FOREIGN KEY (struct_id, resNum1) REFERENCES
    residues (struct_id, resNum1) DEFERRABLE INITIALLY DEFERRED,
  FOREIGN KEY (struct_id, resNum2) REFERENCES
    residues (struct_id, resNum2) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY (struct_id, resNum1, resNum2));
```

B.4.5 HBondFeatures

The HBondFeatures measures the geometry of hydrogen bonds. The most current reference is Kortemme, Morozov, Baker, An Orientation-dependent Hydrogen Bonding Potential Improves Prediction of Specificity and Structure for Proteins and Protein-Protein Complexes, (JMB 2003).

The HBondFeatures feature reporter takes the following options:

```
<feature
  name=HBondFeatures
  scorefxn=(&scorefxn)
  definition_type=["energy", "AHdist"]
  definition_threshold=(&real)/>
```

- *scorefxn*: Use a the parameters in a defined score function to evaluate the hydrogen bonds
- *definition_type*, *definition_threshold*: How should a hydrogen bond be defined? The default is a hydrogen bond is an interaction where the H-bond energy is < 0, i.e. `energy` with a `definition_threshold=0`.

hbond_sites:
Conceptually these are positively and negatively charged functional groups that can form hydrogen bonds.

- *atmNum*: For donor functional groups, atmNum is the atom number of the polar hydrogen. For acceptor functional groups, atmNum is the atom number of an acceptor atom.
- *HBChemType*: The HBChemType string corresponding to an HBAccChemType or HBDonChemType depending on if the site is a donor or acceptor.

```
CREATE TABLE hbond_sites (
  struct_id INTEGER,
  site_id INTEGER,
  resNum INTEGER,
  HBChemType TEXT,
  atmNum INTEGER,
  is_donor INTEGER,
  chain INTEGER,
  resType TEXT,
  atmType TEXT,
  FOREIGN KEY(struct_id, resNum) REFERENCES
    residues(struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY(struct_id, site_id));
```

hbond_site_atoms:
Each hydrogen bond site defines a portion of a frame by bonded atoms.

- Donor atoms:
  - *atm*: The polar *hydrogen* atom
  - *base*: The parent atom of the hydrogen atom. This is the *donor*.
- Acceptor atoms:
  - *atm*: The *acceptor* atom
  - *base*: The parent of the acceptor atom. This is the *base*.
  - *bbase*: The parent of the base atom.

- *base2*: The alternate second base atom of the acceptor. Note: The parent atom is defined by column 6 of the ICOOR_SECTION in each residue type params files. The base to acceptor unit vector is defined by the hybridization type of the acceptor atom and the above atoms.

```
CREATE TABLE IF NOT EXISTS hbond_site_atoms (
  struct_id INTEGER,
  site_id INTEGER,
  atm_x REAL,
  atm_y REAL,
  atm_z REAL,
  base_x REAL,
  base_y REAL,
  base_z REAL,
  bbase_x REAL,
  bbase_y REAL,
  bbase_z REAL,
  base2_x REAL,
  base2_y REAL,
  base2_z REAL,
  FOREIGN KEY(site_id, struct_id) REFERENCES
    hbond_sites(site_id, struct_id) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY(struct_id, site_id));
```

hbond_site_environment:
The energy and geometry of hydrogen bonds depends upon its structural context. The hbond_site_environment table collects measures of burial, secondary structure, and total hydrogen bonding.

- *sasa_r100*, *sasa_r140*, *sasa_r200*: The solvent accessible surface area of the heavy atom of the polar site with water probes of 1.0 Å, 1.4 Å and 2.0 Å.
- *hbond_energy*: Half of the total energy of all hbonds at the polar site.
- *num_hbonds*: The number of hbonds at the polar site.

```
CREATE TABLE IF NOT EXISTS hbond_site_environment (
  struct_id INTEGER,
  site_id INTEGER,
  sasa_r100 REAL,
  sasa_r140 REAL,
  sasa_r200 REAL,
  hbond_energy REAL,
  num_hbonds INTEGER,
  FOREIGN KEY(struct_id, site_id) REFERENCES
    hbond_sites(struct_id, site_id) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY(struct_id, site_id));
```

hbond_sites_pdb
The PDB file format stores identification, geometric and experimental information about each atom. Here, the information stored for the heavy atom of the polar site is stored.

- *chain*, *resNum*, *icode*: The PDB identifier of the for the heavy atom. NOTE, the *icode* is necessary to uniquely identify an atom. NOTE: The rosetta numbering and the PDB numbering may be different.
- *heavy_atom_temperature*: The temperature factor which measures the disorder of the heavy atom.
- *heavy_atom_occupancy*: The occupancy for the heavy atom.

```
CREATE TABLE IF NOT EXISTS hbond_sites_pdb (
```

```
  struct_id INTEGER,
  site_id INTEGER,
  chain TEXT,
  resNum INTEGER,
  iCode TEXT,
  heavy_atom_temperature REAL,
  heavy_atom_occupancy REAL,
  FOREIGN KEY(struct_id, site_id) REFERENCES
    hbond_sites(struct_id, site_id) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY(struct_id, site_id));
```

hbond_chem_types:

Text labels for the chemical type classifications

- *chem_type*: The *HBChemType* column in the *hbond_sites* table reference this.
- *label*: A label for the chemical type in the form <a|d><three-letter-chem-type-code>: <one-letter-amino-acid-types-where-this-type-is-found>. For example *hbdon_HXL* has*dHXL: s,t* as it is label.

```
CREATE TABLE IF NOT EXISTS hbond_chem_types (
  chem_type TEXT,
  label TEXT,
  PRIMARY KEY(chem_type));
```

hbonds:

A *hydrogen bond* is defined to be a donor *hbond_site* and acceptor *hbond_site* where bonding energy is negative.

- *HBEvalType*: The H-bond evaluation type encodes the chemical context of the hydrogen bond.
- *energy*: The H-bond energy is computed by evaluating the geometric parameters of the hydrogen bond.
- *envWeight*: If specified in the HBondOptions, the energy of a hydrogen bond can depend upon the solvent environment computed by the number of neighbors in the 10 Å neighbor graph.
- *score_weight*: The weight of this hydrogen bond in the provided score function. Each HBEvalType is associated with a HBondWeighType as a column in the HBEval.csv file in a H-bond parameter set. The HBondWeighType is then associated with a ScoreType via hb_eval_type_weight. To get the total energy multiply *energy * envWeight *score_weight*.
- *donRank*: The donRank is the rank of the HBond at the donor site. It is 0 if this is the only H-bond at donor site. Otherwise donRank is *i*, where this hbond is the *i*th strongest hbond at its donor, beginning with *i*=1.
- *accRank*: The accRank is the rank of the HBond at the acceptor site. It is 0 if this is the only H-bond at acceptor site. Otherwise accRank *i*, where this H-bond is the *i*th strongest H-bond at its acceptor, beginning with *i*=1.

```
CREATE TABLE IF NOT EXISTS hbonds (
  struct_id INTEGER,
  hbond_id INTEGER,
  don_id INTEGER,
  acc_id INTEGER,
  HBEvalType INTEGER,
  energy REAL,
```

```
  envWeight REAL,
  score_weight REAL,
  donRank INTEGER,
  accRank INTEGER,
  FOREIGN KEY (struct_id, don_id) REFERENCES
    hbond_sites (struct_id, site_id) DEFERRABLE INITIALLY DEFERRED,
  FOREIGN KEY (struct_id, acc_id) REFERENCES
    hbond_sites (struct_id, site_id) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY(struct_id, hbond_id));
```

hbond_geom_coords:
The geometric parameters of a hydrogen bond are used to evaluate the energy of the of interaction.

- *AHdist*: The distance between the *acceptor* and *hydrogen* atoms.
- *cosBAH*: The cosine of the angle defined by the *base*, *acceptor* and *hydrogen* atoms.
  - NOTE: The angle is the exterior angle: cosBAH=1 when linear and cosAHD=0 when perpendicular.
  - NOTE: If the `-corrections:score:hbond_measure_sp3acc_BAH_from_hvy` flag is set, then the base atom for Sp3 acceptors is the heavy atom, otherwise it is the hydrogen atom. (Historical aside, in Score12, the hydrogen atom was used as the base to enforce separation between the covalently bound hydrogen and the hydrogen bonding hydrogen.)
- *cosAHD*: The cosine of the angle defined by the *acceptor*, *hydrogen* and *donor* atoms.
  - NOTE: The angle is the exterior angle: cosAHD=1 when linear and cosAHD=0 when perpendicular
- *chi*: The torsional angle defined by the *abase2*, *base*, *acceptor* and *hydrogen* atoms. NOTE: The value is in radians, [-pi, pi].

```
CREATE TABLE IF NOT EXISTS hbond_geom_coords (
  struct_id INTEGER,
  hbond_id INTEGER,
  AHdist REAL,
  cosBAH REAL,
  cosAHD REAL,
  chi REAL,
  FOREIGN KEY(struct_id, hbond_id) REFERENCES
    hbonds(struct_id, hbond_id) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY(struct_id, hbond_id));
```

hbond_lennard_jones:
The Lennard-Jones attraction energy (*atrE*), repulsion energy (*repE*), and solvation energy (*solv*) are computed over pairs of atoms. Because of the large number of such atom pairs, reporting the geometry and energy for each instance is too costly. However, since the LJ terms may double count the interaction energy between hydrogen bonding atoms, the LJ interactions between just hydrogen bonding atoms are explicitly reported here.

- *don_acc*: LJ energy between the *donor* and the *acceptor* atoms
- *don_acc_base*: JL energy between the *donor* and the *acceptor base* atoms
- *h_acc*: LJ energy between the *hydrogen* and *acceptor* atoms
- *h_acc_base*: JL energy between the *hydrogen* and *acceptor base* atoms
  - Note: To compare these energies against hydrogen energies, they must be weighted by the ScoreFunction weight set.

```
CREATE TABLE IF NOT EXISTS hbond_lennard_jones (
```

```
  struct_id INTEGER,
  hbond_id INTEGER,
  don_acc_atrE REAL,
  don_acc_repE REAL,
  don_acc_solv REAL,
  don_acc_base_atrE REAL,
  don_acc_base_repE REAL,
  don_acc_base_solv REAL,
  h_acc_atrE REAL,
  h_acc_repE REAL,
  h_acc_solv REAL,
  h_acc_base_atrE REAL,
  h_acc_base_repE REAL,
  h_acc_base_solv REAL,
  FOREIGN KEY (struct_id, hbond_id) REFERENCES
    hbonds (struct_id, hbond_id) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY(struct_id, hbond_id));
```

B.4.6 OrbitalFeatures

The OrbitalFeatures stores information about chemical interactions involving orbitals. Orbitals are atomically localized electrons that can form weak, orientation dependent interactions with polar and aromatic functional groups and other orbitals. Orbital geometry are defined in the residue type sets in the database. Following the orbitals score term, orbitals are defines between residues where the action center is at most 11 Å apart.

HPOL_orbital:
Interactions between orbitals and polar hydrogens. Intra-residue interactions are excluded.

- polar hydrogens: Polar hydrogens are identified by res2.Hpos_polar_sc()
- orbName1: This is like LP10 and is the second column of the ORBITALS tag in the residue parameter files.
- dist: This is the distance between the orbital and the polar hydrogen
- angle: This is the cosine of the angle defined by the atom the orbital is attached to, the orbital and the polar hydrogen.

```
CREATE TABLE IF NOT EXISTS HPOL_orbital (
  struct_id TEXT,
  resNum1 INTEGER,
  orbName1 TEXT,
  resNum2 INTEGER,
  hpolNum2 INTEGER,
  resNum1 INTEGER,
  resName2 TEXT,
  htype2 TEXT,
  OrbHdist REAL,
  cosAOH REAL,
  cosDHO REAL,
  chiBAOH REAL,
  chiBDHO REAL,
  AOH_angle REAL,
  DHO_angle REAL,
  chiBAHD REAL,
  cosAHD REAL,
  FOREIGN KEY (struct_id, resNum1) REFERENCES
    residues (struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  FOREIGN KEY (struct_id, resNum2) REFERENCES
    residues (struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY(struct_id, resNum1, orbName1, resNum2, hpolNum2));
```

HARO_orbital:
Interactions between orbitals and aromatic hydrogens. Intra-residue interactions are excluded.

- *aromatic hydrogens*: Aromatic hydrogens are identified by *res2.Haro_index()*
- *orbName1*: This is like *LP10* and is the second column of the ORBITALS tag in the residue parameter files.
- *dist*: This is the distance between the orbital and the aromatic hydrogen
- *angle*: This is the cosine of the angle defined by the atom the orbital is attached to, the orbital and the aromatic hydrogen.

```
CREATE TABLE IF NOT EXISTS orbital_orbital_interactions (
  struct_id TEXT,
  resNum1 INTEGER,
  orbName1 TEXT,
  resNum2 INTEGER,
  haroNum2 INTEGER,
  resName1 TEXT,
  orbNum1 INTEGER,
  resName2 TEXT,
  htype2 TEXT,
  orbHdist REAL,
  cosAOH REAL,
  cosDHO REAL,
  chiBAOH REAL,
  chiBDHO REAL,
  AOH_angle REAL,
  DHO_angle REAL,
  chiBAHD REAL,
  cosAHD REAL,
  FOREIGN KEY (struct_id, resNum1) REFERENCES
    residues (struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  FOREIGN KEY (struct_id, resNum2) REFERENCES
    residues (struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY(struct_id, resNum1, orbName1, resNum2, haroNum2));
```

orbital_orbital:
Interactions between orbitals and polar hydrogens. Intra-residue interactions are excluded. To avoid double counting, *resNum1* < *resNum2*.

- *polar hydrogens*: Polar hydrogens are indexed from *1' to* res2.n_orbitals()
- *orbName1*: This is like *LP10* and is the second column of the ORBITALS tag in the residue parameter files.
- *dist*: This is the distance between the orbital and the second orbital
- *angle*: This is the cosine of the angle defined by the atom the orbital is attached to, the orbital and the second orbital.

```
CREATE TABLE IF NOT EXISTS orbital_orbital (
  struct_id TEXT,
  resNum1 INTEGER,
  orbName1 TEXT,
  resNum2 INTEGER,
  orbNum2 INTEGER,
  resName1 TEXT,
  orbNum1 INTEGER,
  resName2 TEXT,
  orbName2 TEXT,
  orbOrbdist REAL,
  cosAOO REAL,
  cosDOO REAL,
```

```
  chiBAOO REAL,
  chiBDOO REAL,
  AOO_angle REAL,
  DOO_angle REAL,
  chiBAHD REAL,
  cosAHD REAL,
  FOREIGN KEY (struct_id, resNum1) REFERENCES
    residues (struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  FOREIGN KEY (struct_id, resNum2) REFERENCES
    residues (struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY(struct_id, resNum1, orbName1, resNum2, orbNum2));
```

B.4.7 SaltBridgeFeatures

The SaltBridgeFeatures represent salt bridges and related interactions following the definition in: Donald JE, Kulp DW, DeGrado WF. Salt bridges: Geometrically specific, designable interactions. Proteins: Structure, Function, and Bioinformatics. 2010.

salt_bridges:
A row represents the center of an oxygen of the acceptor group (*ASN*, *ASN*, *GLN*, *GLU*) being within 6 Å of the center of the donor group (*HIS*, *LYS*, *ARG*). Note: the center of *HIS* is the midpoint between the ring nitrogen atoms.

- *psi*: The angle of the oxygen around the donor group [-180, 180)
- *theta*: Angle out of the oxygen of donor group plane
- *rho*: Distance between the center of the donor group to the oxygen.
- *orbital*: *syn* or *anti*, the orbital of acceptor oxygen (based on the torsional angle about the acceptor-acceptor base bond).

```
CREATE TABLE IF NOT EXISTS salt_bridges (
  struct_id INTEGER,
  don_resNum INTEGER,
  acc_id INTEGER,
  psi REAL,
  theta REAL,
  rho REAL,
  orbital TEXT,
  FOREIGN KEY (struct_id, don_resNum) REFERENCES
    residues (struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  FOREIGN KEY (struct_id, acc_id) REFERENCES
    hbond_sites (struct_id, site_id) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY(struct_id, don_resNum, acc_id));
```

B.4.8 ChargeChargeFeatures

The ChargeChargeFeatures represent interactions between charged groups in molecular conformations. The primary interaction is through the Coulomb potential, which is proportional to $q_1 q_2 / R^2$. However, because the charge is not always centered at the atom, and the groups can shield the interaction, it is important to measure the angles as well. For each charged site, there are three atoms defined based on the hbond_site_atoms table: q_1 = atm, B_1 = base, C_1 = bbase, and similarly for q_2.

charge_charge_pair:
A row represents two charged polar group: (*ASP*, *GLU*, *LYS*, *ARG*, *HIS*) within 8 Å.

- *q{1,2}_charge*: If it is a donor, then 1, if it is an acceptor then -1
- *B1q1q2_angle*: The bond angle at q_1 formed by B_1 and q_2

- *B2q2q1_angle*: The bond angle at q2 formed by B2 and q1
- *q1q2_distance*: The distance between the q_1 and q_2 atoms
- *B1q1_torsion*: The torsion angle defined by $C_1 - B_1 - q_1 - q_2$
- *B2q2_torsion*: The torsion angle defined by $C_2 - B_2 - q_2 - q_1$

```
CREATE TABLE IF NOT EXISTS charge_charge_pairs (
  struct_id INTEGER,
  q1_site_id INTEGER,
  q2_site_id INTEGER,
  B1q1q2_angle REAL,
  B2q2q1_angle REAL,
  q1q2_distance REAL,
  B1q1_torsion REAL,
  B2q2_torsion REAL,
  FOREIGN KEY (struct_id, q1_site_id) REFERENCES
    hbond_sites (struct_id, site_id) DEFERRABLE INITIALLY DEFERRED,
  FOREIGN KEY (struct_id, q2_site_id) REFERENCES
    hbond_sites (struct_id, site_id) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY(struct_id, q1_site_id, q2_site_id));
```

B.4.9 LoopAnchorFeatures

```
<feature
  name=LoopAnchorFeatures
  min_loop_length=5
  max_loop_length=7/>
```

This reporter scans all available windows of a specified number of residues and calculates the translation and rotation to optimally superimpose the landing onto the takeoff of the loop. The translation and rotation data can then be used to compare different "classes" of loop anchors.

loop_anchors:

- *min_loop_length*: The minimum span of residues upon which to compute the translation and rotation.
- *max_loop_length*: The maximum span of residues upon which to compute the translation and rotation.

```
CREATE TABLE IF NOT EXISTS loop_anchors (
  struct_id INTEGER,
  residue_begin INTEGER,
  residue_end INTEGER,
  FOREIGN KEY (struct_id, residue_begin) REFERENCES
    residues (struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  FOREIGN KEY (struct_id, residue_end) REFERENCES
    residues (struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY(struct_id, residue_begin, residue_end));

CREATE TABLE IF NOT EXISTS loop_anchor_transforms (
  struct_id INTEGER,
  residue_begin INTEGER,
  residue_end INTEGER,
  x REAL,
  y REAL,
  z REAL,
  phi REAL,
  psi REAL,
  theta REAL,
  FOREIGN KEY (struct_id, residue_begin, residue_end) REFERENCES
```

```
    loop_anchors (struct_id, residue_begin, residue_end) DEFERRABLE INITIALLY
DEFERRED,
  PRIMARY KEY(struct_id, residue_begin, residue_end));
```

B.5 Multi-Body Features

B.5.1 StructureFeatures

A structure is a group of spatially organized residues. The definition corresponds with a Pose in Rosetta. Unfortunately in Rosetta there is not a well-defined way to identify a Pose. For the purposes of the features database, each structure is assigned a unique struct_id. To facilitate connecting structures in the database with structures in structures Rosetta, the tag field is unique.

structures:
Identify the structures in the features database

- *tag*: The tag identifies the structure in Rosetta. The following locations are searched in order.
  - pose.pdb_info()->name()
  - pose.data().get(JOBDIST_OUTPUT_TAG)
  - JobDistributor::get_instance()->current_job()->input_tag()

```
CREATE TABLE IF NOT EXISTS structures (
  struct_id INTEGER PRIMARY KEY AUTOINCREMENT,
  protocol_id INTEGER,
  tag TEXT,
  UNIQUE (protocol_id, tag),
  FOREIGN KEY (protocol_id) REFERENCES
 protocols (protocol_id) DEFERRABLE INITIALLY DEFERRED);
```

B.5.2 PoseConformationFeatures

The PoseConformationFeatures measures the conformation level information in a Pose. Together with the ProteinResidueConformationFeatures (B.3.3), the atomic coordinates can be reconstructed. To facilitate creating poses from conformation structure data stored in the features database, PoseConformationFeatures has a *load_into_pose* method.

pose_conformations:
This table stores information about sequence of residues in the conformation.

- *annotated_sequence*: The annotated sequence string of residue types that make up the conformation
- *total_residue*: The number of residues in the conformation
- *fullatom*: The ResidueTypeSet is *FA_STANDARD* if true, and *CENTROID* if false.

```
CREATE TABLE IF NOT EXISTS pose_conformations (
  struct_id INTEGER AUTOINCREMENT PRIMARY KEY,
  annotated_sequence TEXT,
  total_residue INTEGER,
  fullatom BOOLEAN,
  FOREIGN KEY (struct_id) REFERENCES
    structures (struct_id) DEFERRABLE INITIALLY DEFERRED);
```

fold_trees:
The fold tree specifies a graph of how the residues are attached together. If the residues are thought of as vertices, each row in the *fold_trees* table specifies a directed edge.

- *(start_res, start_atom)*: The initial residue and the attachment atom within the residue
- *(end_res, end_atom)*: The terminal residue and the attachment atom with the residue
- *label*: -2 if it is a *CHEMICAL* edge, -1 if is a *PEPTIDE* edge, and 1, 2, ... is a *JUMP* attachment. See here for details. The geometry of the *JUMP* attachments is stored in the *jumps* table.
- *keep_stub_in_residue*: For completeness, the option to keep stub in residue is stored.

```
CREATE TABLE IF NOT EXISTS fold_trees (
  struct_id INTEGER AUTOINCREMENT,
  start_res INTEGER,
  start_atom TEXT,
  stop_res INTEGER,
  stop_atom TEXT,
  label INTEGER,
  keep_stub_in_residue BOOLEAN,
  FOREIGN KEY (struct_id) REFERENCES
    structures (struct_id) DEFERRABLE INITIALLY DEFERRED);
```

jumps:

Each *JUMP* edge in the fold tree is specified by a coordinate transformation, which is encoded in the *jumps* table.

- *jump_id*: The canonical ordering of jumps in a conformation.
- {x,y,z}{x,y,z}: coordinates of the rotation matrix
- {x,y,z}: coordinates of the translation vector

```
CREATE TABLE IF NOT EXISTS jumps (
  struct_id INTEGER AUTOINCREMENT,
  jump_id INTEGER,
  xx REAL,
  xy REAL,
  xz REAL,
  yx REAL,
  yy REAL,
  yz REAL,
  zx REAL,
  zy REAL,
  zz REAL,
  x REAL,
  y REAL,
  z REAL,
  FOREIGN KEY (struct_id) REFERENCES
    structures (struct_id) DEFERRABLE INITIALLY DEFERRED);
```

chain_endings:

The conformation is broken into chemically bonded chains, which are identified by the chain endings. Note: If there are *n* chains, then there are *n-1* chain_endings.

- *end_pos*: The last sequence position in the conformation of a chain.

```
CREATE TABLE IF NOT EXISTS chain_endings (
  struct_id INTEGER AUTOINCREMENT,
  end_pos INTEGER,
  FOREIGN KEY (struct_id) REFERENCES
    structures (struct_id) DEFERRABLE INITIALLY DEFERRED);
```

B.5.3 GeometricSolvationFeatures

geometric_solvation:
The exact geometric solvation score which is computed by integrating the H-bond energy not occupied by other atoms.

- *hbond_site_id*: A hydrogen bonding donor or acceptor
- *geometric_solvation_exact*: The non-pairwise decomposable version of the geometric solvation score.

```
CREATE TABLE IF NOT EXISTS geometric_solvation (
  struct_id INTEGER AUTOINCREMENT,
  hbond_site_id TEXT,
  geometric_solvation_exact REAL,
  FOREIGN KEY (struct_id, hbond_site_id) REFERENCES
    hbond_sites(struct_id, site_id) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY(struct_id, hbond_site_id));
```

B.5.4 RadiusOfGyrationFeatures

Measure the radius of gyration for each structure. The radius of gyration measure of how compact a structure is in O(n). It is the expected displacement of mass from the center of mass. The Wikipedia page is has some information. Also see, Lobanov (2008).

radius_of_gyration:

- *radius_of_gyration*: Let *C* be the center of mass and r_i be the position of residue *i*'th of *n* residues, then the radius of gyration is defined to be $Rg = \sqrt{\sum_{r_i} \frac{(r_i - c)^2}{n-1}}$. Note: the normalizing factor is $n - i$ to be consistent with r++ (the previous version of Rosetta). Atoms with variant type "REPLONLY" are ignored. See the RG_Energy_Fast class for more details.

```
CREATE TABLE IF NOT EXISTS radius_of_gyration (
  struct_id INTEGER AUTOINCREMENT,
  radius_of_gyration REAL,
  FOREIGN KEY(struct_id) REFERENCES
    structures(struct_id) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY(struct_id));
```

B.5.5 SandwichFeatures

Extract β-sandwiches conservatively so that it correctly excludes α-helix that is identified as beta-sandwiche by SCOP and excludes beta-barrel that is identified as beta-sandwiches by CATH. To dump into pdb files, use the format_converter application. Analyze β-sandwiches such as (ϕ, ψ) angles in core/edge strand each assign one β-sheet between two β-sheets that constitute one β-sandwich as additional chain so that InterfaceAnalyzer can be used.

sw_can_by_components:

```
CREATE TABLE sw_can_by_components(
  struct_id INTEGER AUTOINCREMENT NOT NULL,
  sw_can_by_components_PK_id INTEGER NOT NULL,
  tag TEXT NOT NULL,
  sw_can_by_sh_id INTEGER NOT NULL,
  sheet_id INTEGER,
```

```
  sheet_antiparallel INTEGER,
  sw_can_by_components_bs_id INTEGER,
  sw_can_by_components_bs_edge INTEGER,
  intra_sheet_con_id INTEGER,
  inter_sheet_con_id INTEGER,
  residue_begin INTEGER NOT NULL,
  residue_end INTEGER NOT NULL,
  FOREIGN KEY (struct_id) REFERENCES
    structures(struct_id) DEFERRABLE INITIALLY DEFERRED,
  FOREIGN KEY (struct_id, residue_begin) REFERENCES
    residues(struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  FOREIGN KEY (struct_id, residue_end) REFERENCES
    residues(struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY (struct_id, sw_can_by_components_PK_id));
```

B.5.6 SecondaryStructureSegmentFeatures

Report continuous segments of secondary structure. DSSP is used to define secondary structure, but simplified to be simply H, E, and L (all DSSP codes other than H and E). Due to this simplification of DSSP codes, the dssp column is NOT a foreign key to the dssp_codes table.

secondary_structure_segments:

```
CREATE TABLE IF NOT EXISTS secondary_structure_segments (
  struct_id INTEGER AUTOINCREMENT NOT NULL,
  segement_id INTEGER NOT NULL,
  residue_begin INTEGER,
  residue_end INTEGER,
  dssp TEXT NOT NULL,
  FOREIGN KEY (struct_id) REFERENCES
    structures(struct_id) DEFERRABLE INITIALLY DEFERRED,
  FOREIGN KEY (struct_id, residue_begin) REFERENCES
    residues(struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  FOREIGN KEY (struct_id, residue_end) REFERENCES
    residues(struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY (struct_id, segment_id));
```

B.5.7 SmotifFeatures

Record a set of geometric parameters defined by two pieces of adjacent secondary structure (Fernandez-Fuentes 2010).

smotifs:

```
CREATE TABLE smotifs(
  struct_id INTEGER AUTOINCREMENT NOT NULL,
  smotif_id INTEGER NOT NULL,
  secondary_struct_segment_id_1 INTEGER NOT NULL,
  secondary_struct_segment_id_2 INTEGER NOT NULL,
  loop_segment_id INTEGER NOT NULL,
  distance REAL NOT NULL,
  hoist REAL NOT NULL,
  packing REAL NOT NULL,
  meridian REAL NOT NULL,
  FOREIGN KEY (struct_id) REFERENCES
    structures(struct_id) DEFERRABLE INITIALLY DEFERRED,
  FOREIGN KEY (struct_id, secondary_struct_segment_id_1) REFERENCES
    secondary_structure_segments(struct_id, segment_id) DEFERRABLE INITIALLY
DEFERRED,
  FOREIGN KEY (struct_id, secondary_struct_segment_id_2) REFERENCES
```

```
    secondary_structure_segments(struct_id, segment_id) DEFERRABLE INITIALLY
DEFERRED,
  FOREIGN KEY (struct_id, loop_segment_id) REFERENCES
    secondary_structure_segments(struct_id, segment_id) DEFERRABLE INITIALLY
DEFERRED,
        PRIMARY KEY (struct_id, smotif_id));
```

B.5.8 StrandBundleFeatures

Function summary: Find all strands -> Leave all pair of strands -> Leave all pair of sheets
Function detail: It generates smallest unit of beta-sandwiches that are input files of Tim's SEWING protocol. After finding all beta strands in PDF files, leave all pair of beta strands (either parallel or anti-parallel) among them. Then leave all pair of beta sheets (which are constituted with 4 beta strands each). As it finds strands/sheets, it finds only those that meet criteria specified in option. 'strand_pairs' table and 'sandwich' table are created in a same schema respectively.

strand_pairs:

```
CREATE TABLE strand_pairs(
  struct_id INTEGER AUTOINCREMENT NOT NULL,
  strand_pairs_id INTEGER NOT NULL,
  bool_parallel INTEGER NOT NULL,
  beta_select_id_i INTEGER NOT NULL,
  beta_select_id_j INTEGER NOT NULL,
  FOREIGN KEY (struct_id) REFERENCES
    structures(struct_id) DEFERRABLE INITIALLY DEFERRED,
  FOREIGN KEY (struct_id, beta_select_id_i) REFERENCES
  beta_selected_segments(struct_id, beta_selected_segments_id) DEFERRABLE
INITIALLY DEFERRED,
  FOREIGN KEY (struct_id, beta_select_id_j) REFERENCES
    beta_selected_segments(struct_id, beta_selected_segments_id) DEFERRABLE
INITIALLY DEFERRED,
  PRIMARY KEY (struct_id, strand_pairs_id));
```

sandwich:

```
CREATE TABLE sandwich(
  struct_id INTEGER AUTOINCREMENT NOT NULL,
  sandwich_id INTEGER NOT NULL,
  sp_id_1 INTEGER NOT NULL,
  sp_id_2 INTEGER NOT NULL,
  shortest_sc_dis REAL NOT NULL,
  FOREIGN KEY (struct_id) REFERENCES
    structures(struct_id) DEFERRABLE INITIALLY DEFERRED,
  FOREIGN KEY (struct_id, sp_id_1) REFERENCES
    strand_pairs(struct_id, strand_pairs_id) DEFERRABLE INITIALLY DEFERRED,
  FOREIGN KEY (struct_id, sp_id_2) REFERENCES
    strand_pairs(struct_id, strand_pairs_id) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY (struct_id, sandwich_id));
```

B.6 Multi-Structure Features

B.6.1 ProteinRMSDFeatures

Compute the atom-wise root mean squared deviation between the conformation being reported and a previously saved conformation. There are several ways of specifying the reference structure, which are considered in the following order

Using the SavePoseMover:

```
<ROSETTASCRIPTS>
  <MOVERS>
    <SavePoseMover
      name=spm_init_struct
      reference_name=init_struct/>
    <ReportToDB
      name=features_reporter
      db="features_SAMPLE_SOURCE_ID.db3"
      sample_source="SAMPLE_SOURCE_DESCRIPTION">
      <feature
        name=ProteinRMSDFeatures
        reference_name=init_struct/>
    </ReportToDB>
  </MOVERS>
  <PROTOCOLS>
    <Add mover_name=spm_init_struct/>
    <Add mover_name=features_reporter/>
  </PROTOCOLS>
</ROSETTASCRIPTS>
```

protein_rmsd:

- *reference_tag*: The tag of the structure this structure is compared against
- *protein_CA*: the C-alpha atoms of protein residues are considered
- *protein_CA_or_CB*: the C-alpha and C-beta atoms of protein residues are considered
- *protein_backbone*: the backbone atoms (N, C-alpha, C) of protein residues are considered
- *protein_backbone_including_O*: the backbone atoms and the carbonyl oxygen atoms of protein residues are considered
- *protein_backbone_sidechain_heavy_atom*: the non-hydrogen atoms of protein residues are considered
- *heavyatom*: all non-hydrogen atoms are considered
- *nbr_atom*: the neighbor atoms are considered (the C-beta atom for canonical proteins) see the NBR_ATOM tag in the residue topology files
- *all_atom*: all atoms are considered

```
CREATE TABLE IF NOT EXISTS protein_rmsd (
  struct_id INTEGER AUTOINCREMENT,
  reference_tag TEXT,
  protein_CA REAL,
  protein_CA_or_CB REAL,
  protein_backbone REAL,
  protein_backbone_including_O REAL,
  protein_backbone_sidechain_heavyatom REAL,
  heavyatom REAL,
  nbr_atom REAL,
  all_atom REAL,
  FOREIGN KEY (struct_id) REFERENCES
    structures (struct_id) DEFERRABLE INITIALLY DEFERRED,
```

```
  PRIMARY KEY (struct_id, reference_tag));
```

B.6.2 RotamerRecoveryFeatures

The RotamerRecoverFeatures is a wrapper for the rotamer_recovery scientific benchmark so it can be included as a feature.

```
<feature
  name=RotamerRecovery
  scfxn=(&string)
  protocol=(&string)
  comparer=(&string)
  mover=(&string)/>
```

See the above link for explanations of the parameters.

rotamer_recovery:
The rotamer_recovery of a feature is how similar Rosetta's optimal conformation is compared to the input conformation when Rosetta's optimal conformation is biased to the input conformation.

- **struct_id**, **resNum**: These are the primary keys for the *residues* table in the ResidueFeatures reporter
- **divergence**: This is the *score* that the *RRProtocol* returns.

```
CREATE TABLE IF NOT EXISTS nchi (
  name3 TEXT,
  nchi INTEGER,
  PRIMARY KEY (name3));

INSERT OR IGNORE INTO nchi VALUES('ARG', 4);
INSERT OR IGNORE INTO nchi VALUES('LYS', 4);
INSERT OR IGNORE INTO nchi VALUES('MET', 3);
INSERT OR IGNORE INTO nchi VALUES('GLN', 3);
INSERT OR IGNORE INTO nchi VALUES('GLU', 3);
INSERT OR IGNORE INTO nchi VALUES('TYR', 2);
INSERT OR IGNORE INTO nchi VALUES('ILE', 2);
INSERT OR IGNORE INTO nchi VALUES('ASP', 2);
INSERT OR IGNORE INTO nchi VALUES('TRP', 2);
INSERT OR IGNORE INTO nchi VALUES('PHE', 2);
INSERT OR IGNORE INTO nchi VALUES('HIS', 2);
INSERT OR IGNORE INTO nchi VALUES('ASN', 2);
INSERT OR IGNORE INTO nchi VALUES('THR', 1);
INSERT OR IGNORE INTO nchi VALUES('SER', 1);
INSERT OR IGNORE INTO nchi VALUES('PRO', 1);
INSERT OR IGNORE INTO nchi VALUES('CYS', 1);
INSERT OR IGNORE INTO nchi VALUES('VAL', 1);
INSERT OR IGNORE INTO nchi VALUES('LEU', 1);
INSERT OR IGNORE INTO nchi VALUES('ALA', 0);
INSERT OR IGNORE INTO nchi VALUES('GLY', 0);

CREATE TABLE IF NOT EXISTS rotamer_recovery (
  struct_id INTEGER AUTOINCREMENT,
  resNum INTEGER,
  divergence REAL,
  recovered INTEGER,
  PRIMARY KEY(struct_id, resNum));
```

B.7 Energy Features

B.7.1 ScoreTypeFeatures

The ScoreTypeFeatures store the score types for as for all EnergyMethods.

```
<feature name=ScoreTypeFeatures scorefxn=(default_scorefxn &string)/>
```

score_types:
Store information about the EnergyMethod associated with each score type.

- *batch_id*: The score types are reference the *batches* table to allow for score types to change over time.
- score_type_id: The core::scoring::ScoreType enum values.
- *score_type_name*: The string version of the *core::scoring::ScoreType* enum values.

```
CREATE TABLE IF NOT EXISTS score_types (
  batch_id INTEGER,
  score_type_id INTEGER,
  score_type_name TEXT,
  FOREIGN KEY (batch_id) REFERENCES
    batches (batch_id) DEFERRABLE INITIALLY DEFERRED),
  PRIMARY KEY (batch_id, score_type_id);
```

B.7.2 ScoreFunctionFeatures

The ScoreFunctionFeatures store the weights and energy method options for the score function.

```
<feature
  name=ScoreFunctionFeatures
  scorefxn=(default_scorefxn &string)/>
```

score_function_weights:
Store the non-zero score term weights.

- *score_function_name*: The tag name of the score function specified in the <SCOREFUNCTIONS/> block
- score_type_id: The core::scoring::ScoreType enum values
- *weight*: The score term weight

```
CREATE TABLE IF NOT EXISTS score_function_weights (
  batch_id INTEGER,
  score_function_name INTEGER,
  score_type_id INTEGER,
  weight REAL,
  FOREIGN KEY (batch_id, score_type_id) REFERENCES
    score_types (batch_id, score_type_id) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY (batch_id, score_type_id));
```

score_function_method_options:
Store the method options for the score function.

- *score_function_name*: The tag name of the score function specified in the <SCOREFUNCTIONS/> block
- *option_key*: The method option
- *option_value*: The value of the method option

```
CREATE TABLE IF NOT EXISTS score_function_method_options (
```

```
  batch_id INTEGER,
  score_function_name INTEGER,
  option_key TEXT,
  option_value TEXT,
  FOREIGN KEY (batch_id) REFERENCES
    batches (batch_id) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY (batch_id, score_function_name, option_key);
```

B.7.3 StructureScoresFeatures

```
<feature name=StructureScoresFeatures scorefxn=(&scorefxn)/>
```

The StructureScoresFeatures stores the overall score information for all enabled EnergyMethods. Depends on ScoreTypesFeatures.

structure_scores:

- *score_value*: The weighted score value for the given type.

```
CREATE TABLE IF NOT EXISTS structure_scores (
  batch_id INTEGER  struct_id INTEGER AUTOINCREMENT,
  score_type_id INTEGER,
  score_value INTEGER,
  FOREIGN KEY (struct_id) REFERENCES
    structures (struct_id) DEFERRABLE INITIALLY DEFERRED,
  FOREIGN KEY (batch_id, score_type_id) REFERENCES
    score_types (batch_id, score_type_id) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY (batch_id, struct_id, score_type_id));
```

B.7.4 ResidueScoresFeatures

```
<feature name=ResidueScoresFeatures scorefxn=(&scorefxn)/>
```

The ResidueScoresFeatures stores the score of a structure at the residue level. Terms that evaluate a single residue are stored in *residue_scores_1b*. Terms that evaluate pairs of residues are stored in *residue_scores_2b*. Terms that depend on the whole structure are stored via the StructureScoresFeatures.

residue_scores_1b:
The one body scores for each residue in the structure.

- *score_type*: The score type as a string
- *score_value*: The score value
- *context_dependent*: 0 if the score type is context-independent and 1 if it is context-dependent

```
CREATE TABLE IF NOT EXISTS residue_scores_1b (
  batch_id INTEGER,
  struct_id INTEGER AUTOINCREMENT,
  resNum INTEGER,
  score_type_id INTEGER,
  score_value REAL,
  context_dependent INTEGER,
  FOREIGN KEY (struct_id, resNum) REFERENCES
    residues (struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  FOREIGN KEY (batch_id, score_type_id) REFERENCES
    score_types (batch_id, score_type_id) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY(batch_id, struct_id, resNum, score_type));
```

residue_scores_2b:
The two-body scores for each pair of residues in the structure. Note: Intra-residue two body terms are stored in this table with resNum1 == resNum2.

- *resNum1*, *resNum2*: The rosetta residue numbering for the participating residues. Note: *resNum1* <= *resNum2*
- *score_type*: The score type as a string
- *score_value*: The score value
- *context_dependent*: 0 if the score type is context-independent and 1 if it is context-dependent

```
CREATE TABLE IF NOT EXISTS residue_scores_2b (
  batch_id INTEGER,
  struct_id INTEGER AUTOINCREMENT,
  resNum1 INTEGER,
  resNum2 INTEGER,
  score_type_id INTEGER,
  score_value REAL,
  context_dependent  INTEGER,
  FOREIGN KEY (struct_id, resNum1) REFERENCES
    residues (struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  FOREIGN KEY (struct_id, resNum2) REFERENCES
    residues (struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  FOREIGN KEY (batch_id, score_type_id) REFERENCES
    score_types (batch_id, score_type_id) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY(batch_id, struct_id, resNum1, resNum2, score_type));
```

B.7.5 ResidueTotalScoresFeatures

```
<feature name=ResidueTotalScoresFeatures scorefxn=(&scorefxn)/>
```

The ResidueTotalScoresFeatures stores for each residue the total score for the one body terms for that residue and half the total score for the two body terms involving that residue. Note that terms that depend on the whole structure are stored via the StructureScoresFeatures. In order to include hydrogen bonding in the totals, the score function must specify the *decompose_bb_hb_into_pair_energies* flag.

residue_total_scores:
The one body scores for each residue in the structure.

- *score_value*: SUM(1b) + SUM(2b)/2 energies

```
CREATE TABLE IF NOT EXISTS residue_total_scores (
  struct_id INTEGER AUTOINCREMENT,
  resNum INTEGER,
  score_value REAL,
  FOREIGN KEY (struct_id, resNum) REFERENCES
    residues (struct_id, resNum)  DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY(struct_id, resNum));
```

B.7.6 HBondParameterFeatures

The parameters for the hydrogen bond potential are specified in the minirosetta_database as parameter sets. Each parameter set specifies polynomials, fade functions, and which are applied to which hydrogen bond chemical types. To indicate parameter set, either use *-hbond_params <database_tag>* set on the command line, or set*score_function.energy_method_*

options().hbond_options()->params_database_tag(<database_tag>). See the HBondDatabase class for more information.

hbond_fade_interval:
Limited interaction between geometric dimensions are controlled by simple fading functions of the form __/----__. See the FadeInterval class for more information.

- *database_tag*: The hydrogen bond parameter set
- *name*: The name of the fade interval referenced in the *hbond_evaluation_types* table
- *junction_type*: The junction type indicates how the function between the knots should be interpolated. Currently the options are *piecewise_linear*, and *smooth* which uses a cubic spline with zero derivative at the knots.
- *min0*, *fmin*, *fmax*, *max0*: The x-coordinates of the knots

```
CREATE TABLE IF NOT EXISTS hbond_fade_interval(
  database_tag TEXT,
  name TEXT,
  junction_type TEXT,
  min0 REAL,
  fmin REAL,
  fmax REAL,
  max0 REAL,
  PRIMARY KEY(database_tag, name));
```

hbond_polynomial_1d:
One dimensional polynomials for each geometric dimension used to compute the hydrogen bond energy. See the Polynomial_1d class for more information.

- *database_tag*: The hydrogen bond parameter set
- *name*: The name of the polynomial referenced in the *hbond_evaluation_types* table.
- *dimension*: The geometric dimension with which the polynomial should be used.
- *xmin*, *xmax*: The polynomial is truncated beyond the *xmin* and *xmax* values.
- *root1*, *root2*: The values where the polynomial equals 0.
- *degree*: The number of coefficients in the polynomial. For example 10x^2 - 3x + 1 would have degree 3.
- *c_**: The coefficients of the polynomial, ordered from highest power to the lowest power.

```
CREATE TABLE IF NOT EXISTS hbond_polynomial_1d (
  database_tag TEXT,
  name TEXT,
  dimension TEXT,
  xmin REAL,
  xmax REAL,
  root1 REAL,
  root2 REAL,
  degree INTEGER,
  c_a REAL,
  c_b REAL,
  c_c REAL,
  c_d REAL,
  c_e REAL,
  c_f REAL,
  c_g REAL,
  c_h REAL,
  c_i REAL,
  c_j REAL,
  c_k REAL,
```

```
  PRIMARY KEY(database_tag, name));
```

hbond_evaluation_types:
Associate to (donor chemical type, acceptor chemical type, sequence separation) hydrogen bond types, the which fade intervals, polynomials and weight types to be used in evaluating and assigning hydrogen bond energy. See hbonds_geom.cc for where they are actually used.

- *database_tag*: The hydrogen bond parameter set
- *don_chem_type*: The donor chemical type component of the hydrogen bond type. This is the value used in the hbond_site.HBChemType column when the site is a hydrogen bond donor.
- *acc_chem_type*: The acceptor chemical type component of the hydrogen bond type. This is the value used in the hbond_site.HBChemType column when the site is a hydrogen bond acceptor.
- *separation*: The sequence separation type component of the hydrogen bond type. This is used as a proxy for participation in local sequence motifs like intra-helix hydrogen bonding. NOTE: Separation is defined as *acc_resNum* - *don_resNum* when both residues are polymers and on the same chain and infinity otherwise.
- *AHdist_{short/long}_fade*: The fading functions to be applied to the AHdist polynomial evaluations. The short/long distinction allows for different angle dependence for hydrogen bonds that have different bond lengths. The distinction follows in spirit the behavior originally described in Kortemme 2003.
- *{cosBAH/cosAHD}_fade*: The fading functions to be applied to the cosBAH/cosAHD polynomial evaluations.
- *AHdist*: Polynomial to be used for the evaluation of the Acceptor -- Hydrogen distance geometric degree of freedom.
- *cosBAH_{short/long}*: Polynomials to be used for the evaluation of the cosine of the Acceptor Base -- Acceptor -- Hydrogen geometric degree of freedom.
- *cosAHD_{short/long}*: Polynomials to be used for the evaluation of the cosine of the Acceptor -- Hydrogen -- Donor geometric degree of freedom.
- *weight_type*: Which slot in the score vector the energy of the hydrogen bond should be accumulated into. See the WeightType for allowable types.

```
CREATE TABLE IF NOT EXISTS hbond_evaluation_types (
  database_tag TEXT,
  don_chem_type TEXT,
  acc_chem_type TEXT,
  separation TEXT,
  AHdist_short_fade TEXT,
  AHdist_long_fade TEXT,
  cosBAH_fade TEXT,
  cosAHD_fade TEXT,
  AHdist TEXT,
  cosBAH_short TEXT,
  cosBAH_long TEXT,
  cosAHD_short TEXT,
  cosAHD_long TEXT,
  weight_type TEXT,
  FOREIGN KEY(database_tag, AHdist_short_fade) REFERENCES
    hbond_fade_interval(database_tag, name) DEFERRABLE INITIALLY DEFERRED,
  FOREIGN KEY(database_tag, AHdist_long_fade) REFERENCES
    hbond_fade_interval(database_tag, name) DEFERRABLE INITIALLY DEFERRED,
  FOREIGN KEY(database_tag, cosBAH_fade) REFERENCES
    hbond_fade_interval(database_tag, name) DEFERRABLE INITIALLY DEFERRED,
  FOREIGN KEY(database_tag, cosAHD_fade) REFERENCES
    hbond_fade_interval(database_tag, name) DEFERRABLE INITIALLY DEFERRED,
```

```
  FOREIGN KEY(database_tag, AHdist) REFERENCES
    hbond_polynomial_1d(database_tag, name) DEFERRABLE INITIALLY DEFERRED,
  FOREIGN KEY(database_tag, cosBAH_short) REFERENCES
    hbond_polynomial_1d(database_tag, name) DEFERRABLE INITIALLY DEFERRED,
  FOREIGN KEY(database_tag, cosBAH_long) REFERENCES
    hbond_polynomial_1d(database_tag, name) DEFERRABLE INITIALLY DEFERRED,
  FOREIGN KEY(database_tag, cosAHD_short) REFERENCES
    hbond_polynomial_1d(database_tag, name) DEFERRABLE INITIALLY DEFERRED,
  FOREIGN KEY(database_tag, cosAHD_long) REFERENCES
    hbond_polynomial_1d(database_tag, name) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY (database_tag, don_chem_type, acc_chem_type, separation));
```

B.7.7 ScreeningFeatures

ScreeningFeatures is used to consolidate scoring information related to a ligand into a table for later processing. It is most useful when you would otherwise need to join multiple very large tables, and was originally intended for use in a High Throughput Screening Pipeline. This mover must be used in conjunction with the screening_job_inputter.

```
<feature name=(name &string) chain=(chain &string)>
  <descriptor type=(descriptor_name &string)/>
  <descriptor type=(descriptor_name &string)/>
</feature>
```

In the XML definition above, chain is a ligand chain and the descriptor type is the string key in a string_real or string_string pair added to the job. This mover currently assumes that ligands consist of one residue.

screening_features:

- *struct_id* - The struct_id
- *residue_number* - The ligand residue number
- *chain_id* - The ligand chain id
- *name3* - The 3 letter name of the ligand
- *group_name* - The group_name specified by the screening_job_inputter
- *descriptor_data* - The data associated with the descriptors defined in the xml script. The data is formatted as a JSON dictionary.

screening_features:

```
CREATE TABLE screening_features (
  struct_id bigint(20) NOT NULL,
  residue_number int(11) NOT NULL,
  chain_id varchar(1) NOT NULL,
  name3 text NOT NULL,
  group_name text NOT NULL,
  descriptor_data text NOT NULL,
  PRIMARY KEY (struct_id,residue_number),
  CONSTRAINT screening_features_ibfk_1 FOREIGN KEY (struct_id) REFERENCES
    structures (struct_id));
```

B.8 Experimental Data Features

B.8.1 PdbDataFeatures

The PdbDataFeatures records information that is stored in the protein databank structure format.

```
<feature name=PdbDataFeatures/>
```

residue_pdb_identification:
Identify residues using the PDB nomenclature. Note, this numbering has biological relevance and therefore may be negative, skip numbers, etc. When using the *DatabaseInputer* or *Database-Outputter* with the Rosetta job distributor, this table is mapped to the PDBInfo object.

- *struct_id*, *residue_number*: References the primary key in the residues table
- *chain_id*: ATOM record columns 21
- *insertion_code*: ATOM record column 26
- *pdb_residue_number*: PDB identification 22-25

```
CREATE TABLE IF NOT EXISTS residue_pdb_identification (
  struct_id INTEGER AUTOINCREMENT,
  residue_number INTEGER,
  chain_id TEXT,
  insertion_code TEXT,
  pdb_residue_number INTEGER,
  FOREIGN KEY (struct_id, residue_number) REFERENCES
    residues (struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY (struct_id, residue_number));
```

residue_pdb_confidence:
Summarize atom level confidence measures (B-factors and occupancy) to the residue level with the intention that they will be used to filter out residues.

- *max_*_temperature*: The maximum temperature (ATOM record columns 60-65) over different atom subsets: all, backbone, sidechain.
- *min_*_occupancy*: The minimum occupancy (ATOM record columns 54-59) over different atom subsets: all, backbone, sidechain.

```
CREATE TABLE IF NOT EXISTS residue_pdb_confidence (
  struct_id INTEGER, residue_number INTEGER,
  max_temperature REAL,
  max_bb_temperature REAL,
  max_sc_temperature REAL,
  min_occupancy REAL,
  min_bb_occupancy REAL,
  min_sc_occupancy REAL,
  FOREIGN KEY (struct_id, residue_number) REFERENCES
    residues (struct_id, resNum) DEFERRABLE INITIALLY DEFERRED,
  PRIMARY KEY (struct_id, residue_number)
```

REFERECES

Anfinsen, C.B. “Principles That Govern the Folding of Protein Chains.” *Science* 181, no. 4096 (1973): 223–230.

Arunan, Elangannan, Gautam R. Desiraju, Roger a. Klein, Joanna Sadlej, Steve Scheiner, Ibon Alkorta, David C. Clary, et al. “Definition of the Hydrogen Bond (IUPAC Recommendations 2011).” *Pure and Applied Chemistry* 83, no. 8 (2011).

Ashworth, Justin, Gregory K Taylor, James J Havranek, S Arshiya Quadri, Barry L Stoddard, and David Baker. “Computational Reprogramming of Homing Endonuclease Specificity at Multiple Adjacent Base Pairs.” *Nucleic Acids Research* 38, no. 16 (2010).

Baker, E. N., and R. E. Hubbard. “Hydrogen Bonding in Globular Proteins.” *Prog. Biophys. Molec. Biol.* 44 (1984): 97–179.

Baxter, Rodney. *Exactly Solved Models in Statistical Mechanics.* Dover Publications, (2008).

Beauchamp, Kyle A, Yu-shan Lin, Rhiju Das, and Vijay S Pande. “Are Protein Force Fields Getting Better? A Systematic Benchmark on 524 Diverse NMR Measurements.” *Journal of Chemical Theory and Computation* 8 (2012): 1409-1414.

Beck, Kent. *Test-Driven Development by Example*. Addison-Wesley Professional, (2002).

Bellis, A. “The CppDB Library.” 2010.

Bengio, Yoshua, Aaron Courville, and Pascal Vincent. “Representation Learning : A Review and New Perspectives” no. 1993 (2012): 1–34.

Benkert, Pascal, Silvio C E Tosatto, and Dietmar Schomburg. “QMEAN: A Comprehensive Scoring Function for Model Quality Assessment.” *Proteins* 71, no. 1 (2008): 261–77.

Benson, Dennis A., Ilene Karsch-Mizrachi, David J. Lipman, James Ostell, and David L. Wheeler. “GenBank.” *Nucleic Acids Research* 33 (2005).

Berger, James O. “Statistical Decision Theory and Bayesian Analysis.” Springer-Verlag New York, Inc., (1985).

Berjanskii, Mark, Yongjie Liang, Jianjun Zhou, Peter Tang, Paul Stothard, You Zhou, Joseph Cruz, et al. “PROSESS: a Protein Structure Evaluation Suite and Server.” *Nucleic Acids Research* 38 (2010): W633–40.

Berkholz, Donald S, Peter B Krenesky, John R Davidson, and P Andrew Karplus. “Protein Geometry Database: a Flexible Engine to Explore Backbone Conformations and Their Relationships to Covalent Geometry.” *Nucleic Acids Research* 38 (2010): D320-5.

Berkholz, Donald S., Maxim V. Shapovalov, Roland L. Dunbrack, and P. Andrew Karplus. “Conformation Dependence of Backbone Geometry in Proteins.” *Structure* 17, no. 10 (1993).

Berman, H M, J Westbrook, Z Feng, G Gilliland, T N Bhat, H Weissig, I N Shindyalov, and P E Bourne. “The Protein Data Bank.” *Nucleic Acids Research* 28, no. 1 (2000): 235–42.

Bernstein, Frances C., Thomas F. Koetzle, Graheme J.B. Williams, Edgar F. Meyer Jr., Michael D. Brice, John R. Rodgers, Olga Kennard, Takehiko Shimanouchi, and Mitsuo Tasumi. “The Protein Data Bank: a Computer-based Archival File for Macromolecular Structures.” *Journal of Molecular Biology* (1977): 535–542.

Betancourt, Marcos R. “Efficient Monte Carlo Trial Moves for Polypeptide Simulations Efficient Monte Carlo Trial Moves for Polypeptide Simulations.” *Journal of Chemical Physics* 174905 (2005).

Betancourt, Marcos R, and Jeffrey Skolnick. “Local Propensities and Statistical Potentials of Backbone Dihedral Angles in Proteins.” *Journal of Molecular Biology* 342, no. 2 (2004): 635–49.

Boobbyer, David N A, Peter J Goodford, Peter M McWhinnie, and Rebecca C Wade. “New Hydrogen-bond Potentials for Use in Determining Energetically Favorable Binding Sites on Molecules of Known Structure.” *Journal of Medicinal Chemistry* 32, no. 5 (1989): 1083–1094.

Boomsma, Wouter, Kanti V Mardia, Charles C Taylor, Jesper Ferkinghoff-Borg, Anders Krogh, and Thomas Hamelryck. “A Generative, Probabilistic Model of Local Protein Structure.” *Proceedings of the National Academy of Sciences of the United States of America* 105, no. 26 (2008).

Bower, Michael J, Fred E Cohen, and Roland L Dunbrack Jr. “Prediction of Protein Side-chain Rotamers from a Backbone-dependent Rotamer Library: a New Homology Modeling Tool.” *Journal of Molecular Biology* 267, no. 5 (1997): 1268–1282.

Box, George E P. “Sampling and Bayes’ Inference in Scientific Modelling and Robustness.” *Journal of the Royal Statistical Society. Series A* 143, no. 4 (1980): 383–430.

Boyd, Stephen, Lin Xiao, and Almir Mutapcic. “Subgradient Methods” (2003): 1–21.

Bradley, Philip, Kira M S Misura, and David Baker. “Toward High-resolution de Novo Structure Prediction for Small Proteins.” *Science* 309, no. 5742 (2005): 1868–71.

Branden, Carl, and John Tooze. “Introduction to Protein Structure.” Garland Science, (1999).

Brooks, Bernard R, Robert E Bruccoleri, Barry D Olafson, David J States, S Swaminathan, and Martin Karplus. “CHARMM: A Program for Macromolecular Energy, Minimization, and Dynamics Calculations.” *Journal of Computational Chemistry* 4, no. 2 (1983): 187–217.

Brooks Jr., Fred. “Toolsmith II.” *Communications of the ACM* 39, no. 3 (1996): 61–68.

Buja, Andreas, Dianne Cook, Heike Hofmann, Michael Lawrence, Eun-Kyung Lee, Deborah F Swayne, and Hadley Wickham. “Statistical Inference for Exploratory Data Analysis and Model Diagnostics.” *Philosophical Transactions. Series A, Mathematical, Physical, and Engineering Sciences* 367, no. 1906 (2009).

Chaudhuri, Probal, and JS Marron. “SiZer for Exploration of Structures in Curves.” *Journal of the American Statistical …* 94, no. 447 (1999): 807–823.

———. “Scale Space View of Curve Estimation.” *Annals of Statistics*, 28, no. 2 (2000.): 408–428.

Chen, Vincent B., W. Bryan Arendall, Jeffrey J. Headd, Daniel A Keedy, Robert M. Immormino, Gary J. Kapral, Laura W. Murray, Jane S. Richardson, and David C. Richardson. “MolProbity: All-atom Structure Validation for Macromolecular Crystallography.” *Acta Crystallographica. Section D, Biological Crystallography* 66 (2010).

Choi, Hwanho, Hongsuk Kang, and Hwangseo Park. “New Angle-Dependent Potential Energy Function for Backbone – Backbone Hydrogen Bond in Protein – Protein Interactions.” *Journal of Computational Chemistry* (2009).

Choi, Hwanho, Hongsuk Kang, and Hwangseo Park. “Extended Morse Function Model for Angle-dependent Hydrogen Bond in Protein-protein Interactions.” *The Journal of Physical Chemistry. B* 114, no. 8 (2010).

Codd, E F. “A Relational Model of Data for Large Shared Data Banks. 1970.” *M.D. Computing: Computers in Medical Practice* 15, no. 3 (1970): 162–6.

Cornell, Wendy D, Piotr Cieplak, Christopher I Bayly, Ian R Gould, Kenneth M Merz, David M Ferguson, David C Spellmeyer, Thomas Fox, James W Caldwell, and Peter A Kollman. “A Second Generation Force Field for the Simulation of Proteins, Nucleic Acids , and Organic Molecules” no. 6 (1995): 5179–5197.

Cybulski, Slawomir M, and Steve Scheiner. “Hydrogen Bonding and Proton Transfers Involving the Carboxylate Group.” *Journal of the American Chemical Society* 111, no. 1 (1989): 23–31.

Dannenberg, J. J., Laury Haskamp, and Artëm Masunov. “Are Hydrogen Bonds Covalent or Electrostatic? A Molecular Orbital Comparison of Molecules in Electric Fields and H-Bonding Environments.” *The Journal of Physical Chemistry A* 103, no. 35 (1999).

Dantas, Gautam, Colin Corrent, Steve L Reichow, James J Havranek, Ziad M Eletr, Nancy G Isern, Brian Kuhlman, Gabriele Varani, Ethan a Merritt, and David Baker. “High-resolution Structural and Thermodynamic Analysis of Extreme Stabilization of Human Procarboxypeptidase by Computational Protein Design.” *Journal of Molecular Biology* 366, no. 4 (2007): 1209–21.

Davis, Ian W, W Bryan Arendall, David C Richardson, and Jane S Richardson. “The Backrub Motion: How Protein Backbone Shrugs When a Sidechain Dances.” *Structure* 14, no. 2 (2006).

De Finetti, Bruno. *Theory of Probability*. Wiley, 1974.

Ding, Feng, and Nikolay V Dokholyan. “Emergence of Protein Fold Families through Rational Design.” *PLoS Computational Biology* 2, no. 7 (2006): e85.

DiMaio, Echols, Headd, Terwilliger, Adamas, and Baker, 2013. “Improved low-resolution crystallographic refinement with Phenix and Rosetta.” *Nature Methods*, 10 (2013): 1102-1104.

Dobson, Neil, Gautam Dantas, David Baker, and Gabriele Varani. “High-resolution Structural Validation of the Computational Redesign of Human U1A Protein.” *Structure* 14, no. 5 (2006): 847–56.

Donald, Jason E., Daniel W. Kulp, and William F. DeGrado. “Salt Bridges: Geometrically Specific, Designable Interactions.” *Proteins: Structure, Function, and Bioinformatics* 79 (2011): 898-915.

Engh, R., and R. Huber. “Accurate Bond and Angle Parameters for X-ray Protein Structure Refinement.” *Acta Crystallographica Section A Foundations of Crystallography* 47, no. 4 (1991): 392–400.

Fabiola, Felcy, and Richard Bertram. “An Improved Hydrogen Bond Potential: Impact on Medium Resolution Protein Structures.” *Protein Science* (2002): 1415–1423.

Fernandez-Fuentes, N., Dybas, J.M. and Fiser, A., “Structural characteristics of novel protein folds.” *PLoS computational biology*, 6(4), (2010): p.e1000750.

Fersht, A.R., J.P. Shi, J. Knill-Jones, D.M. Lowe, A.J. Wilkinson, D.M. Blow, P. Brick, Paul Carter, M.M.Y. Waye, and Greg Winter. “Hydrogen Bonding and Biological Specificity Analysed by Protein Engineering.” *Nature* 314, no. 6008 (1985): 235–238.

Fewster, Mark, and Dorothy Graham. *Software Test Automation*. Addison-Wesley Professional, 1999.

Fisher, Ronald A. *Design of Experiments*. 7th ed. New York: Hafner Press, 1971.

Fleishman, Sarel J, Sagar D Khare, Nobuyasu Koga, and David Baker. “Restricted Sidechain Plasticity in the Structures of Native Proteins and Complexes.” *Protein Science* 20 (2011).

Fleishman, Sarel J, Andrew Leaver-Fay, Jacob E Corn, Eva-Maria Strauch, Sagar D Khare, Nobuyasu Koga, Justin Ashworth, et al. “RosettaScripts: a Scripting Language Interface to the Rosetta Macromolecular Modeling Suite.” *PloS One* 6, no. 6 (January 2011): e20161.

Fleishman, Sarel J, Timothy a Whitehead, Damian C Ekiert, Cyrille Dreyfus, Jacob E Corn, Eva-Maria Strauch, Ian a Wilson, and David Baker. “Computational Design of Proteins Targeting the Conserved Stem Region of Influenza Hemagglutinin.” *Science (New York, N.Y.)* 332, no. 6031.

Fleming, P.J., and G.D. Rose. “Do All Backbone Polar Groups in Proteins Form Hydrogen Bonds?” *Protein Science: A Publication of the Protein Society* 14, no. 7 (2005): 1911.

Fodje, M N, and Salam Al-Karadaghi. “Occurrence, Conformational Features and Amino Acid Propensities for the Pi-helix.” *Protein Engineering* 15, no. 5 (2002): 353–8.

Fowler, Martin, Kent Beck, John Brant, and William Opdyke. “Refactoring: Improving the Design of Existing Code” (2002).

Fraser, James S, Henry van den Bedem, Avi J Samelson, P Therese Lang, James M Holton, Nathaniel Echols, and Tom Alber. “Accessing Protein Conformational Ensembles Using Room-temperature X-ray Crystallography.” *Proceedings of the National Academy of Sciences of the United States of America* 108, no. 39 (2011).

Friedland, Gregory D, Anthony J Linares, Colin a Smith, and Tanja Kortemme. “A Simple Model of Backbone Flexibility Improves Modeling of Side-chain Conformational Variability.” *Journal of Molecular Biology* 380, no. 4 (2008).

Frishman, Dmitrij, and Patrick Argos. “Knowledge-based Protein Secondary Structure Assignment.” *Proteins: Structure, Function, and Bioinformatics* 23, no. 4 (1995): 566–579.

Gavezzotti, A, and C Filippini. “Geometry of the Intermolecular XH...Y (X, Y= N, O) Hydrogen Bond and the Calibration of Empirical Hydrogen-Bond Potentials.” *The Journal of Physical Chemistry* (1994): 4831–4837.

Gelman, Andrew. “A Bayesian Formulation of Exploratory Data Analysis and Goodness-of-fit Testing.” *International Statistical Review* 71, no. 2 (2003): 369–382.

———. “Exploratory Data Analysis for Complex Models.” *Journal of Computational and Graphical Statistics* 13, no. 4 (2004): 755–779.

Gelman, Andrew, XL Meng, and Hal Stern. “Posterior Predictive Assessment of Model Fitness via Realized Discrepancies.” *Statistica Sinica* 6 (1996): 733–807.

Gilis, Dimitri, and Marianne Rooman. “Predicting Protein Stability Changes Upon Mutation Using Database-derived Potentials: Solvent Accessibility Determines the Importance of Local Versus Non-local Interactions Along the Sequence.” *Journal of Molecular Biology* 272, no. 2 (1997): 276–290.

Gilli, Paola, Loretta Pretto, Valerio Bertolasi, and Gastone Gilli. “Predicting Hydrogen-Bond Strengths from Acid-Base Molecular Properties. The pK a Slide Rule: Toward the Solution of a Long-Lasting Problem.” *Accounts of Chemical Research* 42, no. 1 (2009): 33–44.

Ginzinger, Simon W, Christian X Weichenberger, and Manfred J Sippl. “Detection of Unrealistic Molecular Environments in Protein Structures Based on Expected Electron Densities.” *Journal of Biomolecular NMR* 47, no. 1 (2010).

Gnanadesikan, R, and Murray Hiu. “Probability Plotting Methods for the Analysis of Data” *Biometrika* 55 (1968): 1-17.

Goffin, Jean-louis. “Subgradient Optimization in Nonsmooth Optimization” I, no. x (2012): 277–290.

Good, Irving John. *Probability and the Weighing of Evidence*. 1st ed. Griffin, 1950.

Gretton, Arthur, and K Borgwardt. “A Kernel Method for the Two-sample Problem.” *NIPS* (2006).

Grishaev, Alexander, and Ad Bax. “An Empirical Backbone-backbone Hydrogen-bonding Potential in Proteins and Its Applications to NMR Structure Refinement and Validation.” *Journal of the American Chemical Society* 126, no. 23 (2004): 7281–92.

Grzybowski, Bartosz a., Alexey V. Ishchenko, Robert S. DeWitte, George M. Whitesides, and Eugene I. Shakhnovich. “Development of a Knowledge-Based Potential for Crystals of Small

Organic Molecules: Calculation of Energy Surfaces for C=0···H−N Hydrogen Bonds." *The Journal of Physical Chemistry B* 104, no. 31 (2000): 7293–7298.

Hamelryck, Thomas, Mikael Borg, Martin Paluszewski, Jonas Paulsen, Jes Frellsen, Christian Andreetta, Wouter Boomsma, Sandro Bottaro, and Jesper Ferkinghoff-Borg. "Potentials of Mean Force for Protein Structure Prediction Vindicated, Formalized and Generalized." *PloS One* 5, no. 11 (2010): e13714.

Hingerty, B E, R H Ritchie, T L Ferrell, and J E Turner. "Dielectric Effects in Biopolymers: The Theory of Ionic Saturation Revisited." *Biopolymers* 24, no. 3 (1985): 427–439.

Hipp, D.R. "SQLite3" (2013).

Hoag, David. "Apollo Guidance and Navigation: Considerations of Apollo IMU Gimbal Lock." *Canbridge: MIT Instrumentation Laboratory* (1963): 1–64.

Hobza, Pavel. "Calculations on Noncovalent Interactions and Databases of Benchmark Interaction Energies." *Accounts of Chemical Research* 45, no. 4 (2012): 663–72.

Hollingsworth, Scott a, Matthew C Lewis, Donald S Berkholz, Weng-Keen Wong, and P Andrew Karplus. "$(\Phi,\Psi)_2$ Motifs: a Purely Conformation-Based Fine-Grained Enumeration of Protein Parts At the Two-Residue Level." *Journal of Molecular Biology* 416, no. 1 (2012): 78–93.

Hooft, R W, C Sander, and G Vriend. "Positioning Hydrogen Atoms by Optimizing Hydrogen-bond Networks in Protein Structures." *Proteins* 26, no. 4 (1996): 363–76.

Huler, E, and S Lifson. "Energy Functions for Peptides and Proteins. I. Derivation of a Consistent Force Field Including the Hydrogen Bond from Amide Crystals" *Journal of the American Chemical Society* 70, no. 17 (1974): 5319–5327.

Jacak, Ron, Andrew Leaver-Fay, and Brian Kuhlman. "Computational Protein Design with Explicit Consideration of Surface Hydrophobic Patches." *Proteins* 80, no. 3 (2012): 825–38.

———. "Computational Protein Design with Explicit Consideration of Surface Hydrophobic Patches." *Proteins* 80, no. 3 (2012): 825–38.

Jacobs, Donald J, A J Rader, Leslie A Kuhn, and M F Thorpe. "Protein Flexibility Predictions Using Graph Theory 1" 44 (2001): 150–165.

Jacobson, Matthew P, George A Kaminski, Richard A Friesner, and Chaya S Rapp. "Force Field Validation Using Protein Side Chain Prediction." *The Journal of Physical Chemistry B* 106, no. 44 (2002).

Jain, N. "Scoring Noncovalent Protein-ligand Interactions: a Continuous Differentiable Function Tuned to Compute Binding Affinities." *Journal of Computer-aided Molecular Design* 10, no. 5 (1996): 427–40.

Jaynes, Edwin T. *Probability Theory: The Logic of Science*. Cambridge: Cambridge University Press, 2003.

Jeffreys. *Theory of Probability*. Oxford University Press, 1961.

Jones, J. E. “On the Determination of Molecular Fields. II. From the Equation of State of a Gas.” *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character* 106, no. 738 (1924): 463–477.

Jones, M.C., J.S. Marron, and S.J. Sheather. “A Brief Survey of Bandwidth Selection for Density Estimation.” *Journal of the American Statistical Association* 91, no. 433 (1996): 401–407.

Jorgensen, William L, David S Maxwell, and Julian Tirado-Rives. “Development and Testing of the OPLS All-Atom Force Field on Conformational Energetics and Properties of Organic Liquids.” *Journal of the American Chemical Society* 118, no. 45 (1996): 11225–11236.

Kabsch, Wolfgang, and Christian Sander. “Dictionary of Protein Secondary Structure: Pattern Recognition of Hydrogen-bonded and Geometrical Features.” *Biopolymers* 22, no. 12 (1983): 2577–2637.

Karatzoglou, Alexandros, Alex Smola, and Kurt Hornik. “Kernlab: Kernel-based Machine Learning Lab,” 2013.

Kay, Lewis E. “Protein Dynamics from NMR.” *Nature Structural Biology* (1998): 513–517.

Keedy, Daniel a, Christopher J Williams, Jeffrey J Headd, W Bryan Arendall, Vincent B Chen, Gary J Kapral, Robert a Gillespie, et al. “The Other 90% of the Protein: Assessment Beyond the Calphas for CASP8 Template-based and High-accuracy Models.” *Proteins* 77 (2009): 29–49.

Khare, Sagar D, Yakov Kipnis, Per Jr Greisen, Ryo Takeuchi, Yacov Ashani, Moshe Goldsmith, Yifan Song, et al. “Computational Redesign of a Mononuclear Zinc Metalloenzyme for Organophosphate Hydrolysis.” *Nat Chem Biol* 8, no. 3 (2012): 294–300.

Khatib, Firas, Seth Cooper, Michael D Tyka, Kefan Xu, Ilya Makedon, Zoran Popovic, David Baker, and Foldit Players. “Algorithm Discovery by Protein Folding Game Players.” *Proceedings of the National Academy of Sciences of the United States of America* 108, no. 47 (011): 18949–53.

Koekemoer, Gerhard, and Jan W.H Swanepoel. “Transformation Kernel Density Estimation With Applications.” *Journal of Computational and Graphical Statistics* 17, no. 3 (September 2008): 750–769.

Koga, Nobuyasu, Rie Tatsumi-Koga, Gaohua Liu, Rong Xiao, Thomas B Acton, Gaetano T Montelione, and David Baker. “Principles for Designing Ideal Protein Structures.” *Nature* 491, no. 7423 (November 08, 2012): 222–227.

Koller, Daphne, Friedman, Nir. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press (2009).

Kortemme, Tanja, Alexandre V. Morozov, and David Baker. “An Orientation-dependent Hydrogen Bonding Potential Improves Prediction of Specificity and Structure for Proteins and Protein–Protein Complexes.” *Journal of Molecular Biology* 326, no. 4 (2003): 1239–1259.

Krieger, Elmar, Tom Darden, Sander B Nabuurs, Alexei Finkelstein, and Gert Vriend. “Making Optimal Use of Empirical Energy Functions: Force-field Parameterization in Crystal Space.” *Proteins* 57, no. 4 (2004): 678–83.

Krieger, Elmar, G. Koraimann, and Gert Vriend. “Increasing the Precision of Comparative Models with YASARA NOVA—a Self-parameterizing Force Field.” *Proteins: Structure, Function, and Bioinformatics* 47, no. 3 (2002): 393–402.

Kubelka, Jan, Thang K Chiu, David R Davies, William a Eaton, and James Hofrichter. “Sub-microsecond Protein Folding.” *Journal of Molecular Biology* 359, no. 3 (2006): 546–53.

Kuhlman, B, and D Baker. “Native Protein Sequences Are Close to Optimal for Their Structures.” *PNAS* 97, no. 19 (2000): 10383–10388.

Kuhlman, B, G Dantas, GC Ireton, and G Varani. “Design of a Novel Globular Protein Fold with Atomic-level Accuracy.” *Science* (2003).

Laganowsky, Arthur, Justin L P Benesch, Meytal Landau, Linlin Ding, Michael R Sawaya, Duilio Cascio, Qingling Huang, Carol V Robinson, Joseph Horwitz, and David Eisenberg. “Crystal Structures of Truncated alphaA and alphaB Crystallins Reveal Structural Mechanisms of Polydispersity Important for Eye Lens Function.” *Protein Science* 19 (2010): 1031–1043.

Lane, Thomas, J., Schwantes, Christian R., Beauchamp, Kyle A., Pande, Vijay S.. “Probing the origins of two state folding.” *Journal of Chemical Physics*, 139 (2013).

Langley, Charles H., and Norman L. Allinger. “Molecular Mechanics (MM4) and Ab Initio Study of Amide−Amide and Amide−Water Dimers.” *The Journal of Physical Chemistry A* 107, no. 26 (July 2003): 5208–5216.

Laskowski, R. a., M. W. MacArthur, D. S. Moss, and J. M. Thornton. “PROCHECK: a Program to Check the Stereochemical Quality of Protein Structures.” *Journal of Applied Crystallography* 26, no. 2 (1993): 283–291.

Lazaridis, Themis, and Martin Karplus. “Effective Energy Function for Proteins in Solution.” *Proteins Structure Function and Genetics* 35, no. 2 (April 1999): 139–45.

Leaver-Fay, Andrew, Ron Jacak, P Benjamin Stranges, and Brian Kuhlman. “A Generic Program for Multistate Protein Design.” *PloS One* 6, no. 7 (2011): e20937.

Leaver-Fay, Andrew, Matthew J. O’Meara, Mike Tyka, Ron Jacak, Yifan Song, Elizabeth H. Kellogg, James Thompson, et al. “Scientific Benchmarks for Guiding Macromolecular Energy Function Improvement.” *Methods in Enzymology* 523, no. 6 (2013).

Leaver-Fay, Andrew, Michael Tyka, Steven M Lewis, Oliver F Lange, James Thompson, Ron Jacak, Kristian Kaufman, et al. “ROSETTA3: An Object-oriented Software Suite for the Simulation and Design of Macromolecules.” *Methods in Enzymology* 487, no. 11 (2011): 545–74.

Lee, B, and F. M. Richards. “The Interpretation of Protein Structures: Estimation of Static Accessibility.” *Journal of Molecular Biology* 55 (1971): 379–400.

Li, DW, and R Brüschweiler. “Iterative Optimization of Molecular Mechanics Force Fields from NMR Data of Full-length Proteins.” *Journal of Chemical Theory and Computation* (2011): 1773–1782.

Li, Hong-min, Da-cheng Wang, Zong-hao Zeng, Lei Jin, and Ren-Qiu Hu. “Crystal Structure of an Acidic Neurotoxin from Scorpion Buthus Martensii Karsch at 1.85 A Resolution.” *Journal of Molecular Biology* 261 (1996): 415–431.

Liang, Shide, and N.V. Grishin. “Side-chain Modeling with an Optimized Scoring Function.” *Protein Science* 11, no. 2 (2002): 322–331.

———. “Side-chain Modeling with an Optimized Scoring Function.” *Protein Science* 11, no. 2 (2002): 322–331.

Lii, Jenn-Huei, and Norman L Allinger. “The Important Role of Lone-pairs in Force Field (MM4) Calculations on Hydrogen Bonding in Alcohols.” *The Journal of Physical Chemistry. A* 112, no. 46 (2008): 11903–13.

Lii, JH, and NL Allinger. “Directional Hydrogen Bonding in the MM3 Force Field II.” *Journal of Computational Chemistry* 19, no. 9 (1998): 1001–1016.

———. “Directional Hydrogen Bonding in the MM3 Force Field. I.” *Journal of Physical Organic Chemistry* 7 (1994): 591–609.

Lippincott, E.R., and R. Schroeder. “One-Dimensional Model of the Hydrogen Bond.” *The Journal of Chemical Physics* 23 (1955): 1099.

Liu, Cui, Dong-Xia Zhao, and Zhong-Zhi Yang. “Direct Evaluation of Individual Hydrogen Bond Energy in Situ in Intra- and Intermolecular Multiple Hydrogen Bonds System.” *Journal of Computational Chemistry* 33, no. 4 (2012): 379–90.

Liu, Zhiguo, Guitao Wang, Zhanting Li, and Renxiao Wang. “Geometrical Preferences of the Hydrogen Bonds on Protein−Ligand Binding Interface Derived from Statistical Surveys and Quantum Mechanics Calculations.” *Journal of Chemical Theory and Computation* 4, no. 11 (2008): 1959–1973.

Loader, CR. “Bandwidth Selection : Classical or Plug-In?” *Annals of Statistics* 27, no. 2 (1999): 415–438.

Lobanov MY, Bogatyreva NS, Galzitskaya OV. “Radius of gyration as an indicator of protein structure compactness.” *Molecular Biology*. 42, no. 4 (2008): 623-628

Lovell, S C, J M Word, J S Richardson, and D C Richardson. “The Penultimate Rotamer Library.” *Proteins* 40 (2000): 389–408.

Lu, H., Skolnick, J. “A distance-dependent atomic knowledge-based potential for improved protein structure selection.” *Proteins* 44, no. 3 (2001): 223–32.

Mackerell, A, D Bashford, M Bellott, R Dunbrack, J Evanseck, M Field, S Fischer, J Gao, H Guo, and S Ha. “All-atom Empirical Potential for Molecular Modeling and Dynamics Studies of Proteins.” *Journal of Physical Chemistry* 102 (1998): 3586–3616.

MacKerell, AD, Nilesh Banavali, and Nicolas Foloppe. “Development and Current Status of the CHARMM Force Field for Nucleic Acids.” *Biopolymers* (2000): 257–265.

Mardia, K.V., and P.E. Jupp. *Directional Statistics*. 2nd ed. Wiley, 1999.

Mardia, Kanti V, Jes Frellsen, Mikael Borg, Jesper Ferkinghoff-borg, and Thomas Hamelryck. “A Statistical View on the Reference Ratio Method” (2010): 55–61.

Marin, Jean-Michel, Pierre Pudlo, Christian P. Robert, and Robin J. Ryder. “Approximate Bayesian Computational Methods.” *Statistics and Computing* 22, no. 6 (2011): 1167–1180.

Marron, J S, and D Ruppert. “Transformations to Reduce Boundary Bias in Kernel Density Estimation” 56, no. 4 (1994): 653–671.

Martin, Juliette, Guillaume Letellier, Antoine Marin, Jean-François Taly, Alexandre G de Brevern, and Jean-François Gibrat. “Protein Secondary Structure Assignment Revisited: a Detailed Analysis of Different Assignment Methods.” *BMC Structural Biology* 5 (2005).

Meszaros, Gerard. *xUnit Test Patterns*. Upper Saddle River, NJ: Addison-Wesley, 2007.

Morozov, A V, T Kortemme, and D Baker. “Evaluation of Models of Electrostatic Interactions in Proteins.” *Journal of Physical Chemistry B* 107, no. 9 (2003): 2075-2090.

Müller-Dethlefs, K, and P Hobza. “Noncovalent Interactions: a Challenge for Experiment and Theory.” *Chemical Reviews* 100, no. 1 (2000): 143–68.

Murray-Rust, Peter, and Jenny P Glusker. “Directional Hydrogen Bonding to Sp2-and Sp3-hybridized Oxygen Atoms and Its Relevance to Ligand-macromolecule Interactions.” *Journal of the American Chemical Society* no. 2 (1984): 1018–1025.

Nelder, J. A., and R. Mead. “A Simplex Method for Function Minimization.” *Computer Journal* 7, no. 4 (1961): 308–313.

Neyman, J. “Outline of a Theory of Statistical Estimation Based on the Classical Theory of Probability.” *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 236, no. 767 (1937): 333–380.

Nocedal, J., and S. J. Wright. *Numerical Optimization*. 2nd ed. Springer, 2006.

Novotný, Jiří, Robert Bruccoleri, and Martin Karplus. “An Analysis of Incorrectly Folded Protein Models: Implications for Structure Predictions.” *Journal of Molecular Biology* 177, no. 4 (1984): 787–818.

Park, B H, E S Huang, and M Levitt. “Factors Affecting the Ability of Energy Functions to Discriminate Correct from Incorrect Folds.” *Journal of Molecular Biology* 266, no. 4 (1997): 831–46.

Parzen, E. “On Estimation of a Probability Density Function and Mode.” *The Annals of Mathematical Statistics* (1962).

Paulsen, Jonas, Martin Paluszewski, and KV Mardia. “A Probabilistic Model of Hydrogen Bond Geometry in Proteins.” *Amsta.leeds.ac.uk* (2008): 61–64.

Pearson, Karl. “The Fundamental Problem of Practical Statistics.” *Biometrika* 13, no. 1 (1920): 1–16.

Petrella, R J, T Lazaridis, and M Karplus. “Protein Sidechain Conformer Prediction: a Test of the Energy Function.” *Folding & Design* 3, no. 5 (January 1998): 353–77.

Piana, Stefano, Kresten Lindorff-Larsen, and David E Shaw. “How Robust Are Protein Folding Simulations with Respect to Force Field Parameterization?” *Biophysical Journal* 100, no. 9 (2011): L47–9.

Ponder, Jay W, and Frederic M Richards. “Tertiary Templates for Proteins Use of Packing Criteria in the Enumeration of Allowed Sequences for Different Structural Classes” (1987): 775–791.

Potapov, Vladimir, Mati Cohen, and Gideon Schreiber. “Assessing Computational Methods for Predicting Protein Stability Upon Mutation: Good on Average but Not in the Details.” *Protein Engineering Design and Selection* 22, no. 9 (2009): 553–560.

Pugalenthi, G, K Shameer, N Srinivasan, and R Sowdhamini. “HARMONY: a Server for the Assessment of Protein Structures.” *Nucleic Acids Research* 34 (2006): W231–4.

Qiu, Jian, Will Sheffler, David Baker, and William Stafford Noble. “Ranking Predicted Protein Structures with Support Vector Regression.” *Proteins* 71, no. 3 (2008): 1175–82.

Ramachandran, G.N., and V Sasisekharan. “Conformation of Polypeptides and Proteins.” *Advances in Protein Chemistry* 23 (1968): 283–437.

Reid, C. “Semiempirical Treatment of the Hydrogen Bond.” *The Journal of Chemical Physics* 30, no. 1 (1959): 182.

Řezáč, Jan, and P Hobza. “Advanced Corrections of Hydrogen Bonding and Dispersion for Semiempirical Quantum Mechanical Methods.” *Journal of Chemical Theory and Computation* (2011): 141–151.

Řezáč, Jan, and Pavel Hobza. “Describing Noncovalent Interactions Beyond the Common Approximations: How Accurate Is the ‘Gold Standard,’ CCSD(T) at the Complete Basis Set Limit?” *Journal of Chemical Theory and Computation* 9, no. 5 (2013): 2151–2155.

Richardson, JS, and DC Richardson. “Amino Acid Preferences for Specific Locations at the Ends of Alpha Helices.” *Science* 240, no. 4859 (1988): 1648–1652.

Richter, Leaver-Fay, Khare, Bjelic, and Baker. “De Novo Enzyme Design Using Rosetta3.” *PloS One* 6 (5) 2011.

Rohl, CA, CEM Strauss, K Misura, and D Baker. “Protein Structure Prediction Using Rosetta.” *Methods in Enzymology* 383 (2004): 66–93.

Rosenblatt, M. “Remarks on Some Nonparametric Estimates of a Density Function.” *The Annals of Mathematical Statistics* (1956).

Rubin, DB. “Bayesianly Justifiable and Relevant Frequency Calculations for the Applied Statistician.” *The Annals of Statistics* 12, no. 4 (1984): 1151–1172.

Saccenti, Edoardo, and Antonio Rosato. “The War of Tools: How Can NMR Spectroscopists Detect Errors in Their Structures?” *Journal of Biomolecular NMR* 40, no. 4 (2008): 251–61.

Shapovalov, Maxim V, and Roland L Dunbrack. “A Smoothed Backbone-dependent Rotamer Library for Proteins Derived from Adaptive Kernel Density Estimates and Regressions.” *Structure* 19, no. 6 (2011): 844–58.

Sharabi, O Z, Chen Yanover, Ayelet Dekel, and Julia M Shifman. “Optimizing Energy Functions for Protein – Protein Interface Design.” *Journal of Computational Chemistry* (2010).

Sharabi, Oz, Ayelet Dekel, and Julia M Shifman. “Triathlon for Energy Functions: Who Is the Winner for Design of Protein-protein Interactions?” *Proteins* 79, no. 5 (2011): 1487–98.

Shaw, David E, Ron O Dror, John K Salmon, J P Grossman, Kenneth M Mackenzie, Joseph A Bank, Cliff Young, et al. “Millisecond-Scale Molecular Dynamics Simulations on Anton” (2009): 1–11.

Sheather, S.J., and M.C. Jones. “A Reliable Data-based Bandwidth Selection Method for Kernel Density Estimation” *Journal of the Royal Statistical Society. Series B* 53, no. 3 (1991): 683–690.

Sheffler, Will, and David Baker. “RosettaHoles: Rapid Assessment of Protein Core Packing for Structure Prediction, Refinement, Design, and Validation.” *Protein Science : a Publication of the Protein Society* 18, no. 1 (2009): 229–39.

Shor, Naum Zuselevich. *Minimization Methods for Non-Differentiable Functions*. Springer-Verlag, 1985.

Siegel, J. B., a. Zanghellini, H. M. Lovick, G. Kiss, a. R. Lambert, J. L. St.Clair, J. L. Gallaher, et al. “Computational Design of an Enzyme Catalyst for a Stereoselective Bimolecular Diels-Alder Reaction.” *Science* 329, no. 5989 (2010): 309–313.

Silverman, BW. *Density Estimation for Statistics and Data Analysis*, 1986.

Simons, K T, I Ruczinski, C Kooperberg, B a Fox, C Bystroff, and D Baker. “Improved Recognition of Native-like Protein Structures Using a Combination of Sequence-dependent and Sequence-independent Features of Proteins.” *Proteins* 34, no. 1 (1999): 82–95.

Smith, Colin, and Tanja Kortemme. “Backrub-like Backbone Simulation Recapitulates Natural Protein Conformational Variability and Improves Mutant Side-chain Prediction.” *Journal of Molecular Biology* 380, no. 4 (2008): 742–56.

Song, Yifan, Mike Tyka, Andrew Leaver-Fay, James Thompson, and David Baker. “Structure Guided Forcefield Optimization.” *Proteins: Structure, Function, and Bioinformatics* (2010).

Stickle, Douglas F, Leonard G Presta, Ken A Dill, and George D Rose. “Hydrogen Bonding in Globular Proteins.” *J. Mol. Biol.* no. 226 (1992): 1143–1159.

Taylor, Charles C. “Automatic Bandwidth Selection for Circular Density Estimation.” *Computational Statistics & Data Analysis* 52, no. 7 (2008): 3493–3500.

Tosatto, Silvio C E. “The victor/FRST Function for Model Quality Estimation.” *Journal of Computational Biology : a Journal of Computational Molecular Cell Biology* 12, no. 10 (2005): 1316–27.

Tukey, John. *Exploratory Data Analysis*. Addison-Wesley, 1977.

Tyka, Michael D, Kenneth Jung, and David Baker. “Efficient Sampling of Protein Conformational Space Using Fast Loop Building and Batch Minimization on Highly Parallel Computers.” *Journal of Computational Chemistry* 33, no. 31 (2012): 2483–91.

Tyka, Michael D, Daniel a Keedy, Ingemar André, Frank Dimaio, Yifan Song, David C Richardson, Jane S Richardson, and David Baker. “Alternate States of Proteins Revealed by Detailed Energy Landscape Mapping.” *Journal of Molecular Biology* 405, no. 2 (2010): 607–618.

Vanhee, Peter, Erik Verschueren, Lies Baeten, Francois Stricher, Luis Serrano, Frederic Rousseau, and Joost Schymkowitz. “BriX: a Database of Protein Building Blocks for Structural Analysis, Modeling and Design.” *Nucleic Acids Research* 39 (2011): D435–42.

Vedani, Angelo. “YETI: An Interactive Molecular Mechanics Program for Small-molecule Protein Complexes.” *Journal of Computational Chemistry* 9, no. 3 (1988): 269–280. doi:10.1002/jcc.540090310.

Vinogrado, S.N., and R.H. Linnell. “Spectroscopic Manifestations of Hydrogen Bonding.” Van Nostrand Reinhold: *Hydrogen Bonding*, Chapter 3, 1971.

Vreven, Thom, Howook Hwang, Brian G Pierce, and Zhiping Weng. “Prediction of Protein-protein Binding Free Energies.” *Protein Science* 21, no. 3 (2012): 396–404.

Vriend, G. “WHAT IF : A Molecular Modeling and Drug Design Program.” *Journal of Molecular Graphics* 8, no. 3 (1990): 52–56.

Vymětal, J, and J Vondrášek. “Critical Assessment of Current Force Fields. Short Peptide Test Case.” *Journal of Chemical Theory and Computation* 9 (2012).

Wand, M.P, J.S. Marron, and D. Ruppert. “Transformations in Density Estimation” 86, no. 414 (1991): 343–353.

Wand, M.P., and M.C. Jones. *Kernel Smoothing*. London: Chapman & Hall, 1995.

Wang, C, O Schueler-Furman, and D Baker. “Improved Side-chain Modeling for Protein–protein Docking.” *Protein Science* 14, no. 5 (2005): 1328–1339.

Wang, Lee-Ping, Jiahao Chen, and Troy Van Voorhis. “Systematic Parametrization of Polarizable Force Fields from Quantum Chemistry Data.” *Journal of Chemical Theory and Computation* 9, no. 1 (2013): 452–460.

Wang, Lee-Ping, Teresa Head-Gordon, Jay W Ponder, Pengyu Ren, John D Chodera, Peter K Eastman, Todd J Martinez, and Vijay S Pande. "Systematic Improvement of a Classical Molecular Model of Water." *The Journal of Physical Chemistry. B* (2013).

Wang, Xueyi, Gary Kapral, Laura Murray, D. Richardson, Jane Richardson, and J. Snoeyink. "RNABC: Forward Kinematics to Reduce All-atom Steric Clashes in RNA Backbone." *Journal of Mathematical Biology* 56, no. 1 (2008): 253–278.

Warshel, A, and M Levitt. "Theoretical Studies of Enzymic Reactions: Dielectric, Electrostatic and Steric Stabilization of the Carbonium Ion in the Reaction of Lysozyme." *Journal of Molecular Biology* (1976): 227–249.

Weiner, SJ, and PA Kollman. "A New Force Field for Molecular Mechanical Simulation of Nucleic Acids and Proteins." *Journal of the American Chemical Society* no. 17 (1984): 765–784.

Wickham, H.A. "Practical Tools for Exploring Data and Models." Doctoral Dissertation, Iowa State University, Ames. (2008).

Wickham, Hadley. "A Layered Grammar of Graphics." *Journal of Computational and Graphical Statistics* 19, no. 1 (2010): 3–28.

———. "The Split-Apply-Combine Strategy for Data Analysis." *Journal of Statistical Software* 40, no. 1 (2011).

Wilkinson, Leland. *The Grammar of Graphics*. 2nd ed. Springer, 2005.

Willard, Leigh, Anuj Ranjan, Haiyan Zhang, Hassan Monzavi, Robert F. Boyko, Brian D. Sykes, and Wishart David S. "VADAR: a Web Server for Quantitative Evaluation of Protein Structure Quality." *Nucleic Acids Research* 31, no. 13 (2003): 3316–3319.

Wishart, David S, David Arndt, Mark Berjanskii, An Chi Guo, Yi Shi, Savita Shrivastava, Jianjun Zhou, You Zhou, and Guohui Lin. "PPT-DB: The Protein Property Prediction and Testing Database." *Nucleic Acids Research* 36, no. Database issue (January 2008): D222–9.

Woodroofe, M. "On Choosing a Delta-Sequence." *The Annals of Mathematical Statistics* (1970).

Word, J M, S C Lovell, J S Richardson, and D C Richardson. "Asparagine and Glutamine: Using Hydrogen Atom Contacts in the Choice of Side-chain Amide Orientation." *Journal of Molecular Biology* 285, no. 4 (January 1999): 1735–47.

Word, J.M., S.C. Lovell, T.H. LaBean, H.C. Taylor, M.E. Zalis, B.K. Presley, J.S. Richardson, and D.C. Richardson. "Visualizing and Quantifying Molecular Goodness-of-fit: Small-probe Contact Dots with Explicit Hydrogen Atoms." *Journal of Molecular Biology* 285, no. 4 (1999): 1711–1733.

wwPDB. "Protein Data Bank Contents Guide : Atomic Coordinate Entry Format Description" (2013).

Yanover, Chen, Ora Schueler-Furman, and Yair Weiss. "Minimizing and Learning Energy Functions for Side-chain Prediction." *Journal of Computational Biology : a Journal of Computational Molecular Cell Biology* 15, no. 7 (2008): 899–911.

Yu, J, SVN Vishwanathan, Simon Gunter, and Nicol N. Schraudolf. “A quasi-Newton Approach to Nonsmooth Convex Optimization Problems in Machine Learning.” *The Journal of Machine Learning Research* 11 (2010): 1–5.