# Protein folding: not just another optimization problem

Kevin Karplus

karplus@soe.ucsc.edu

Biomolecular Engineering Department

Undergraduate and Graduate Director, Bioinformatics

University of California, Santa Cruz

# Outline of Talk

- What is Bioinformatics?
- What is a protein?
- The folding problem and variants on it:
  - Local structure prediction
  - Fold recognition with HMMs
    - What is a null model?
    - Why use the reverse-sequence null?
    - Two approaches to statistical significance.
    - What distribution do we expect for scores?
    - Fitting the distribution.
  - Comparative modeling
  - “Ab initio” methods
  - Contact prediction

# What is Bioinformatics?

Bioinformatics: using computers and statistics to make sense out of the mountains of data produced by high-throughput experiments.

- Genomics: finding important sequences in the genome and annotating them.
- Phylogenetics: “tree of life”.
- Systems biology: piecing together various control networks.
- DNA microarrays: what genes are turned on under what conditions.
- Proteomics: what proteins are present in a mixture.
- Protein structure prediction.

# What is a protein?

- There are many abstractions of a protein: a band on a gel, a string of letters, a mass spectrum, a set of 3D coordinates of atoms, a point in an interaction graph, . . . .
- For us, a protein is a long skinny molecule (like a string of letter beads) that folds up consistently into a particular intricate shape.
- The individual “beads” are amino acids, which have 6 atoms the same in each “bead” (the *backbone* atoms: N, H, CA, HA, C, O).
- The final shape is different for different proteins and is essential to the function.
- The protein shapes are important, but are expensive to determine experimentally.

# Folding Problem

The *Folding Problem*:
If we are given a sequence of amino acids (the letters on a string of beads), can we predict how it folds up in 3-space?

---

```
MTMSRRNTDA ITIHSILDWI EDNLESPLSL EKVSERSGYS KWHLQRMFKK
ETGHSLGQYI RSRKMTEIAQ KLKESNEPIL YLAERYGFES QQTLTRTFKN
YFDVPPHKYR MTNMQGESRF LHPLNHYNS
```

$\downarrow$

Too hard!

# Fold-recognition problem

The *Fold-recognition Problem*:
Given a sequence of amino acids $A$ (the *target* sequence) and a library of proteins with known 3-D structures (the *template* library),
figure out which templates $A$ match best, and align the target to the templates.

- The backbone for the target sequence is predicted to be very similar to the backbone of the chosen template.
- Progress has been made on this problem, but we can usefully simplify further.

# Remote-homology Problem

The *Homology Problem*:
Given a target sequence of amino acids
and a library of protein *sequences*,
figure out which sequences $A$ is similar to and align them to $A$.

- No structure information is used, just sequence information. This makes the problem easier, but the results aren't as good.
- This problem is fairly easy for recently diverged, very similar sequences, but difficult for more remote relationships.

## New-fold prediction

- What if there is *no* template we can use?
- We can try to generate many conformations of the protein backbone and try to recognize the most protein-like of them.
- Search space is huge, so we need a good conformation generator and a cheap cost function to evaluate conformations.

# Secondary structure Prediction

- Instead of predicting the entire structure, we can predict local properties of the structure.
- What local properties do we choose?
- We want properties that are well-conserved through evolution, easily predicted, and useful for finding and aligning templates.
- One popular choice is a 3-valued helix/strand/other alphabet—we have investigated many others. Typically, predictors get about 80% accuracy on 3-state prediction.
- Many machine-learning methods have been applied to this problem, but the most successful is neural networks.

# CASP ~~Competition~~ Experiment

- Everything published in literature “works”
- CASP set up as true blind test of prediction methods.
- Sequences of proteins about to be solved released to prediction community.
- Predictions registered with organizers.
- Experimental structures compared with solution by assessors.
- “Winners” get papers in *Proteins: Structure, Function, and Bioinformatics*.

# Predicting Local Structure

- Want to predict some local property at each residue.
- Local property can be emergent property of chain (such as being buried or being in a beta sheet).
- Property should be conserved through evolution (at least as well as amino acid identity).
- Property should be somewhat predictable (we gain information by predicting it).
- Predicted property should aid in fold-recognition and alignment.
- For ease of prediction and comparison, we look only at discrete properties (alphabets of properties).

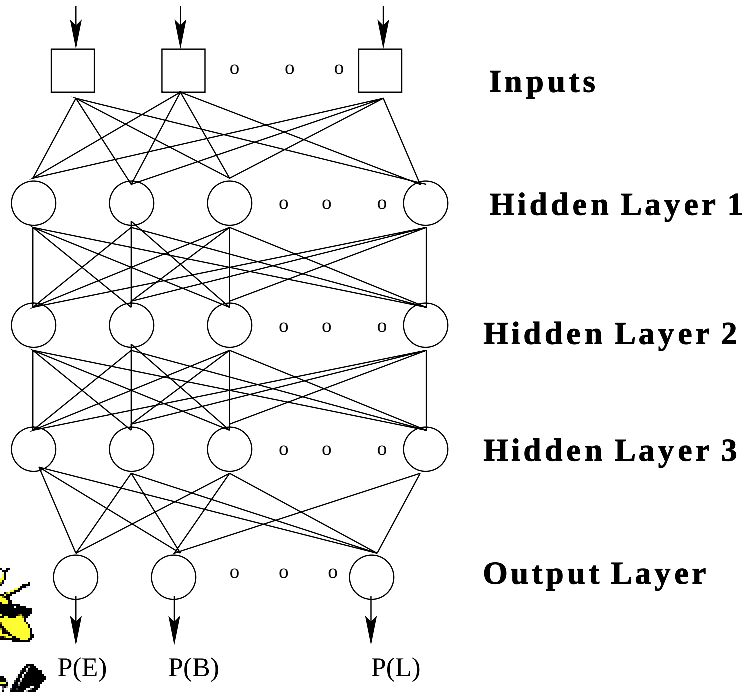
# Using Neural Net

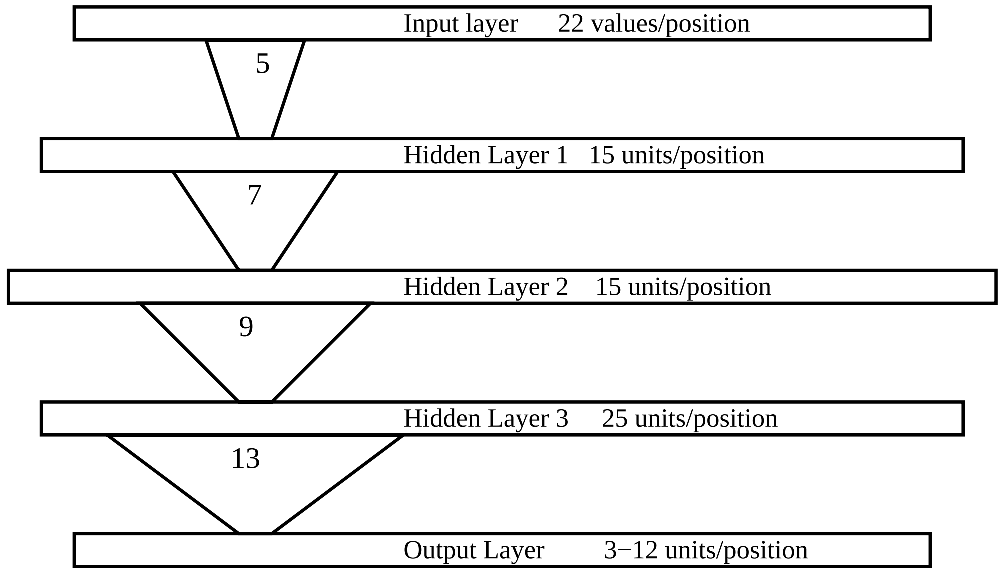
- We use neural nets to predict local properties.
- Input is profile with probabilities of amino acids at each position of target chain, plus insertion and deletion probabilities.
- Output is probability vector for local structure alphabet at each position.
- Each layer takes as input windows of the chain in the previous layer and provides a probability vector in each position for its output.
- We train neural net to maximize $\sum \log(P(\text{correct output}))$.

# Neural Net

Typical net has 4 layers and 6471 weight parameters:

| input/pos | window | output/pos | weights |
|---:|---:|---:|---:|
| 22 | 5 | 15 | 1665 |
| 15 | 7 | 15 | 1590 |
| 15 | 9 | 15 | 2040 |
| 15 | 13 | 6 | 1176 |





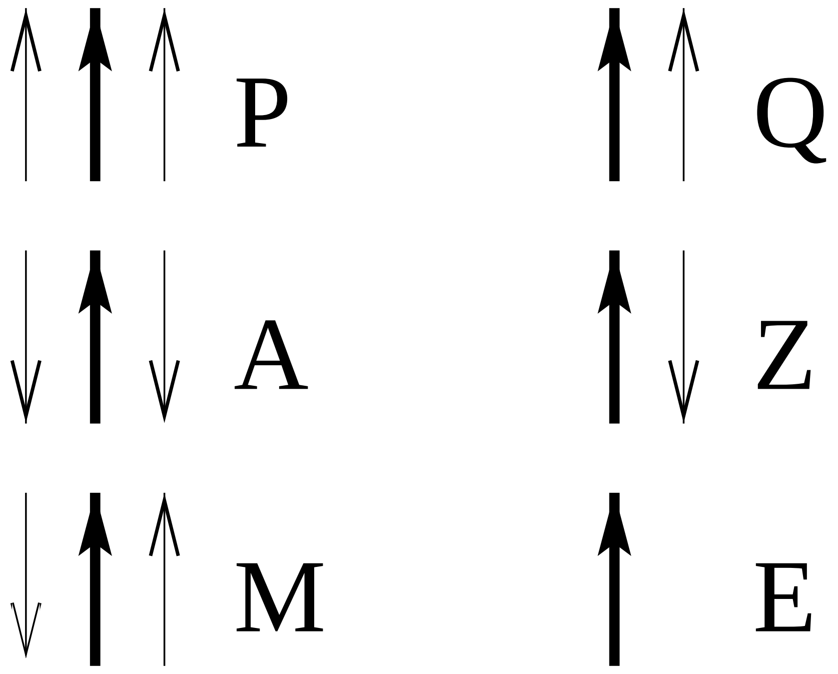
Input layer 22 values/position
5
Hidden Layer 1 15 units/position
7
Hidden Layer 2 15 units/position
9
Hidden Layer 3 25 units/position
13
Output Layer 3−12 units/position

# DSSP

- DSSP is a popular program to define secondary structure.
- 7-letter alphabet: EBGHSTL
  - E = $\beta$ strand
  - B = $\beta$ bridge
  - G = $3_{10}$ helix
  - H = $\alpha$ helix
  - I = $\pi$ helix (very rare, so we lump in with H)
  - S = bend
  - T = turn
  - L = everything else (DSSP uses space for L)

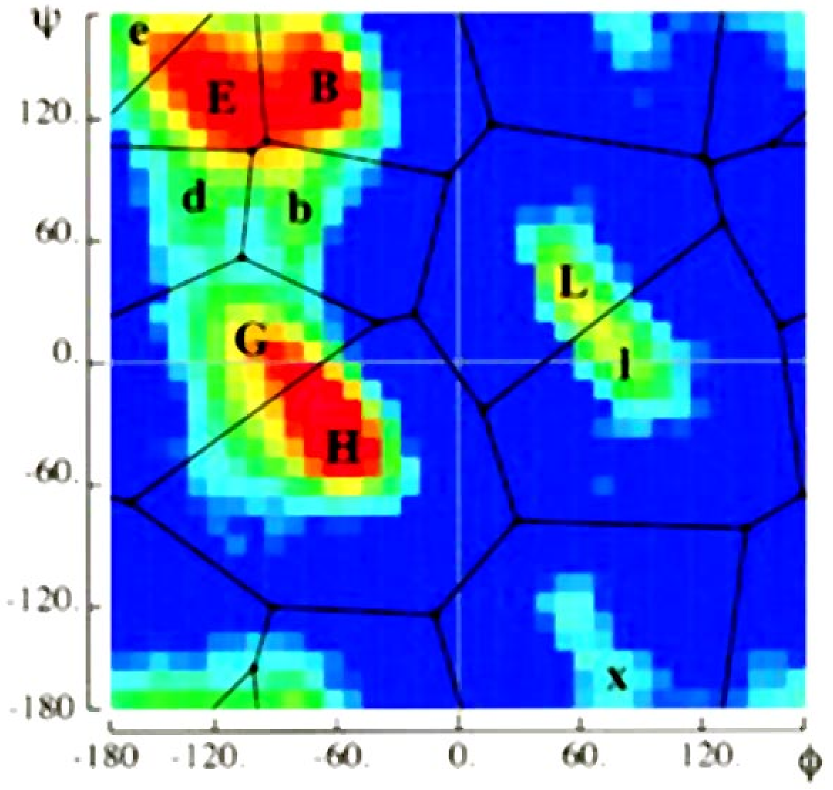
# STR: Extension to DSSP

- Yael Mandel-Gutfreund noticed that parallel and anti-parallel strands had different hydrophobicity patterns, implying that parallel/antiparallel can be predicted from sequence.
- We created a new alphabet, splitting DSSP's E into 6 letters:



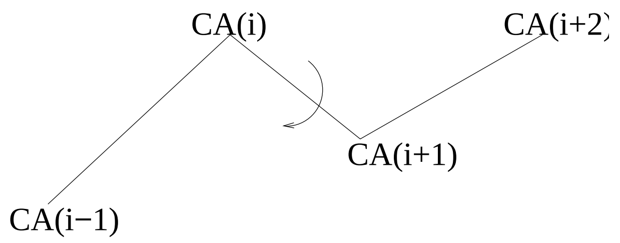
# HMMSTR $\phi$-$\psi$ alphabet

- For HMMSTER, Bystroff did k-means classification of $\phi$-$\psi$ angle pairs into 10 classes (plus one class for cis peptides).
- We used just the 10 classes, ignoring the $\omega$ angle.

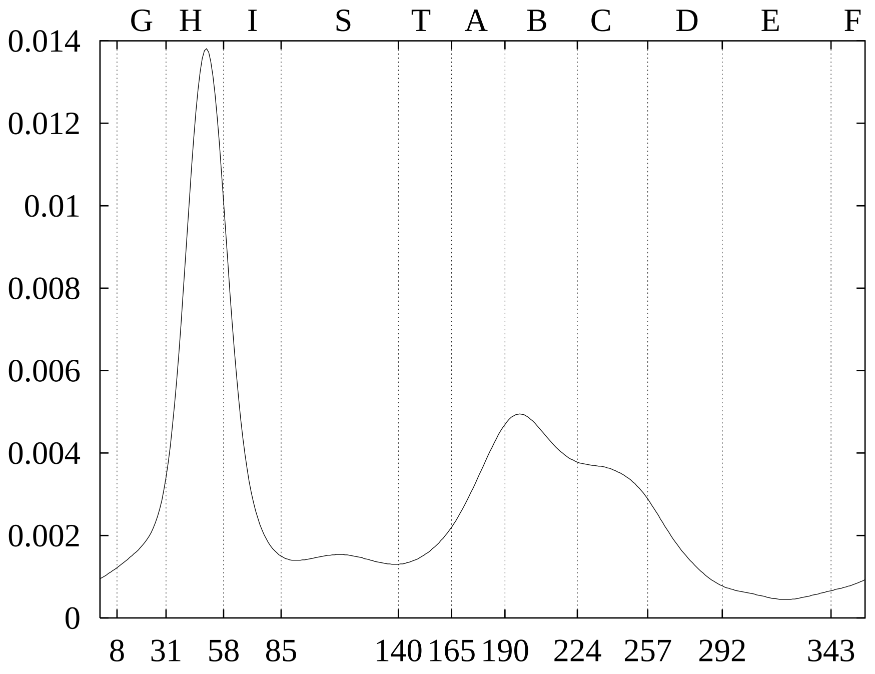


# ALPHA11: $\alpha$ angle

- Backbone geometry can be mostly summarized with one angle per residue:

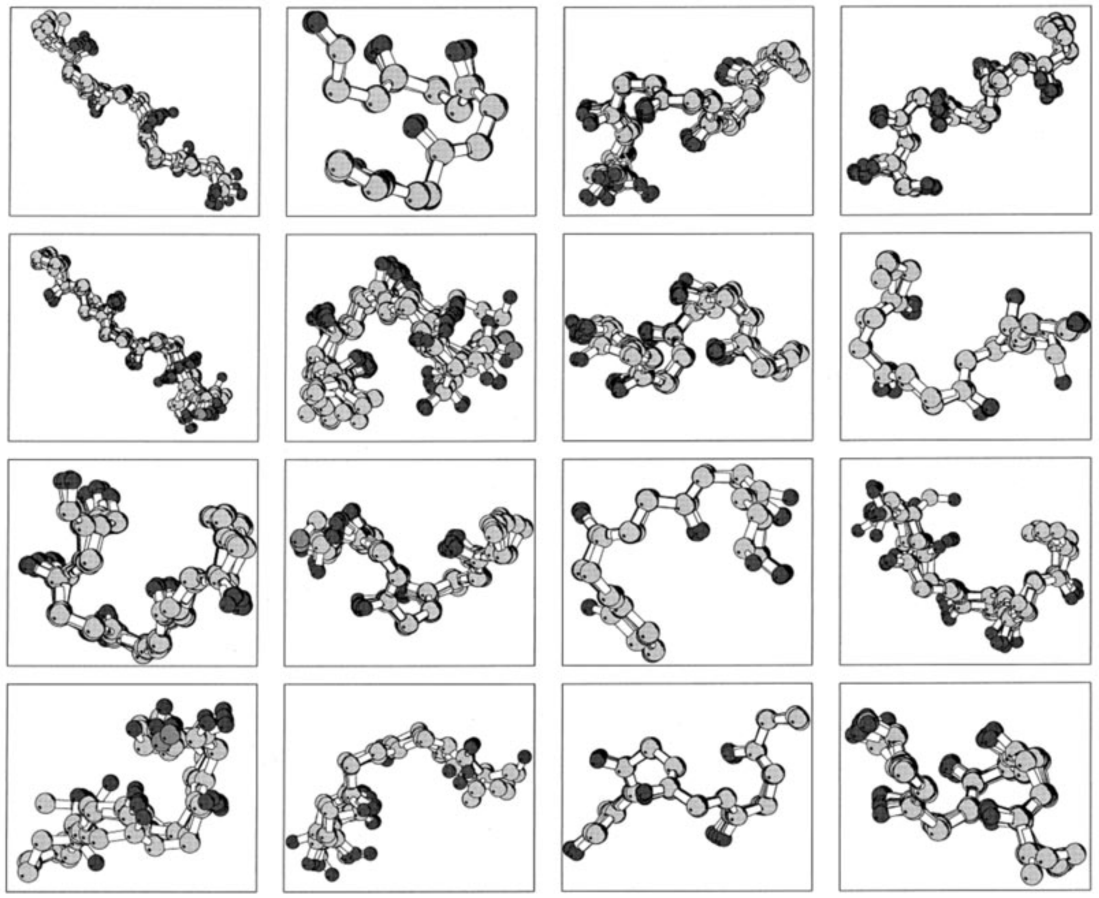


- We discretize into 11 classes:



# de Brevern's Protein Blocks

Clustered on 5-residue window of $\phi$-$\psi$ angles:



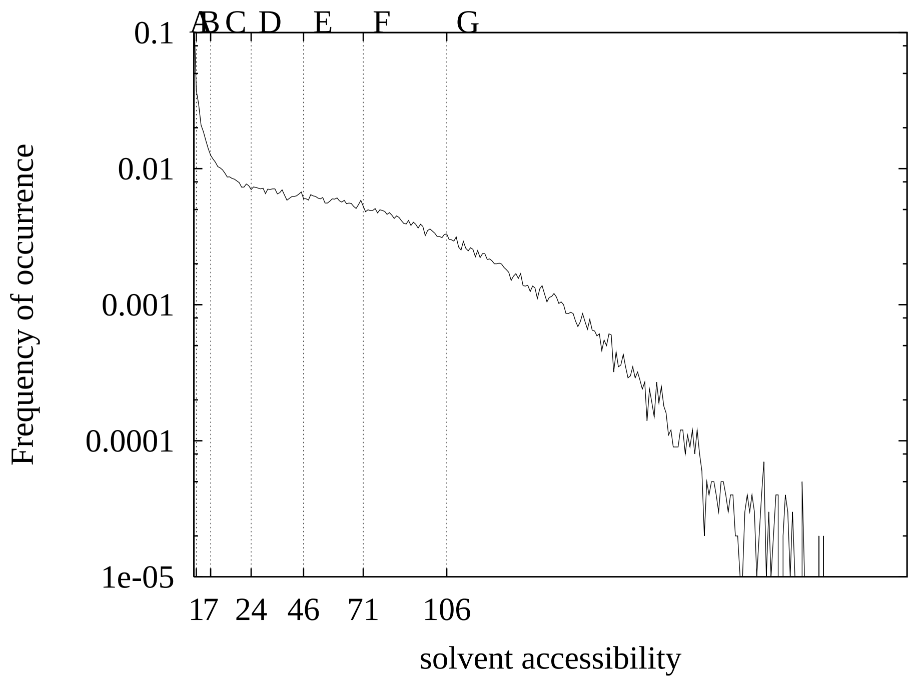
# Burial alphabets

Our second set of investigations was for a sampling of the many burial alphabets, which are discretizations of various accessibility or burial measures:

- solvent accessible surface area
- relative solvent accessible surface area
- neighborhood-count burial measures

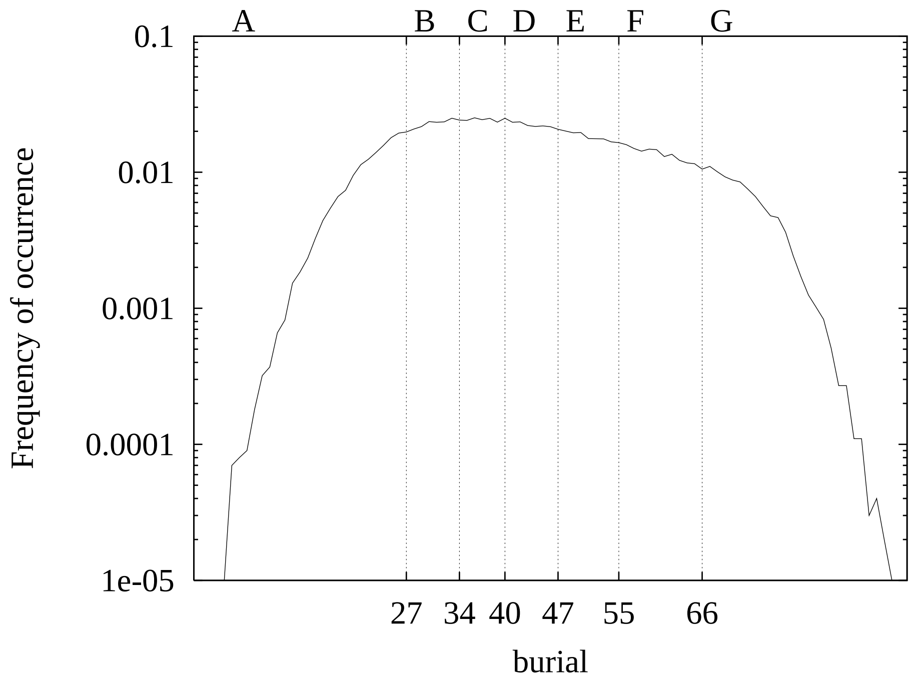
# Solvent Accessibility

- Absolute SA: area in square Ångstroms accessible to a water molecule, computed by DSSP.
- Relative SA: Absolute SA/ max SA for residue type (using Rost's table for max SA).



# Burial

- Define a sphere for each residue.
- Count the number of atoms or of residues within that sphere.
- Example: center= $C_\beta$, radius=14Å, count= $C_\beta$, quantize in 7 equi-probable bins.



# Mutual Information

- Mutual information between two random variables (letters of alphabet):

$$MI(X,Y) = \sum_{i,j} P(i,j) \log \frac{P(i,j)}{P(i)P(j)} \ ,$$

- We look at mutual information between different alphabets at same position in protein. (redundancy)
- We look at mutual information with one alphabet between corresponding positions on alignments of sequences.

# Information Gain

- Information gain is how much more we know about a variable after making a prediction.

$$I(X) = \text{average} \log \frac{\hat{P}_i(X_i)}{P_0(X_i)}$$

- $\hat{P}_i$ is predicted probability vector for position $i$
- $X_i$ is actual observation at position $i$
- $P_0$ is background probability vector

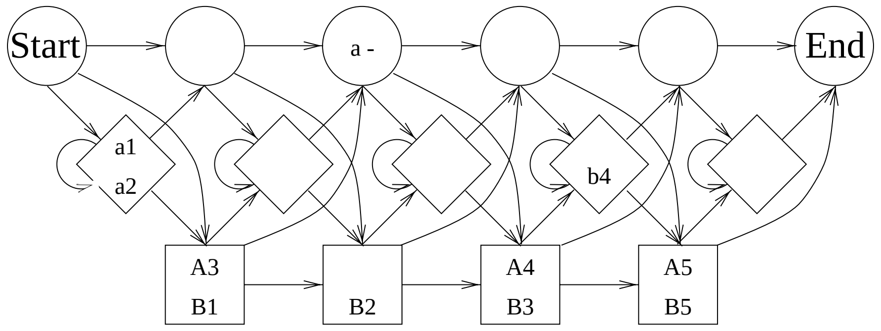
# Conservation and Predictability

| | | | | conservation | predictability | |
|---|---|---|---|---|---|---|
| | alphabet | | MI | | info gain | |
| Name | size | entropy | with AA | mutual info | per residue | $Q_{\|A\|}$ |
| str | 13 | 2.842 | 0.103 | **1.107** | 1.009 | 0.561 |
| protein blocks | 16 | 3.233 | 0.162 | 0.980 | **1.259** | 0.579 |
| stride | 6 | 2.182 | 0.088 | 0.904 | 0.863 | 0.663 |
| DSSP | 7 | 2.397 | 0.092 | 0.893 | 0.913 | 0.633 |
| stride-EHL | 3 | 1.546 | 0.075 | 0.861 | 0.736 | 0.769 |
| DSSP-EHL | 3 | 1.545 | 0.079 | 0.831 | 0.717 | 0.763 |
| CB-16 | 7 | 2.783 | 0.089 | **0.682** | 0.502 | |
| CB-14 | 7 | 2.786 | 0.106 | 0.667 | **0.525** | |
| CB-12 | 7 | 2.769 | 0.124 | 0.640 | 0.519 | |
| rel SA | 7 | 2.806 | 0.183 | 0.402 | 0.461 | |
| abs SA | 7 | 2.804 | 0.250 | 0.382 | 0.447 | |

# Hidden Markov Models

- *Hidden Markov Models* (HMMs) are a very successful way to capture the variability possible in a family of proteins.
- An HMM is a stochastic model—that is, it assigns a probability to every possible sequence.
- An HMM is a finite-state machine with a probability for emitting each letter in each state, and with probabilities for making each transition between states.
- Probabilities of letters sum to one for each state.
- Probabilities of transitions out of each state sum to one for that state.
- We also include *null states* that emit no letters, but have transition probabilities on their out-edges.

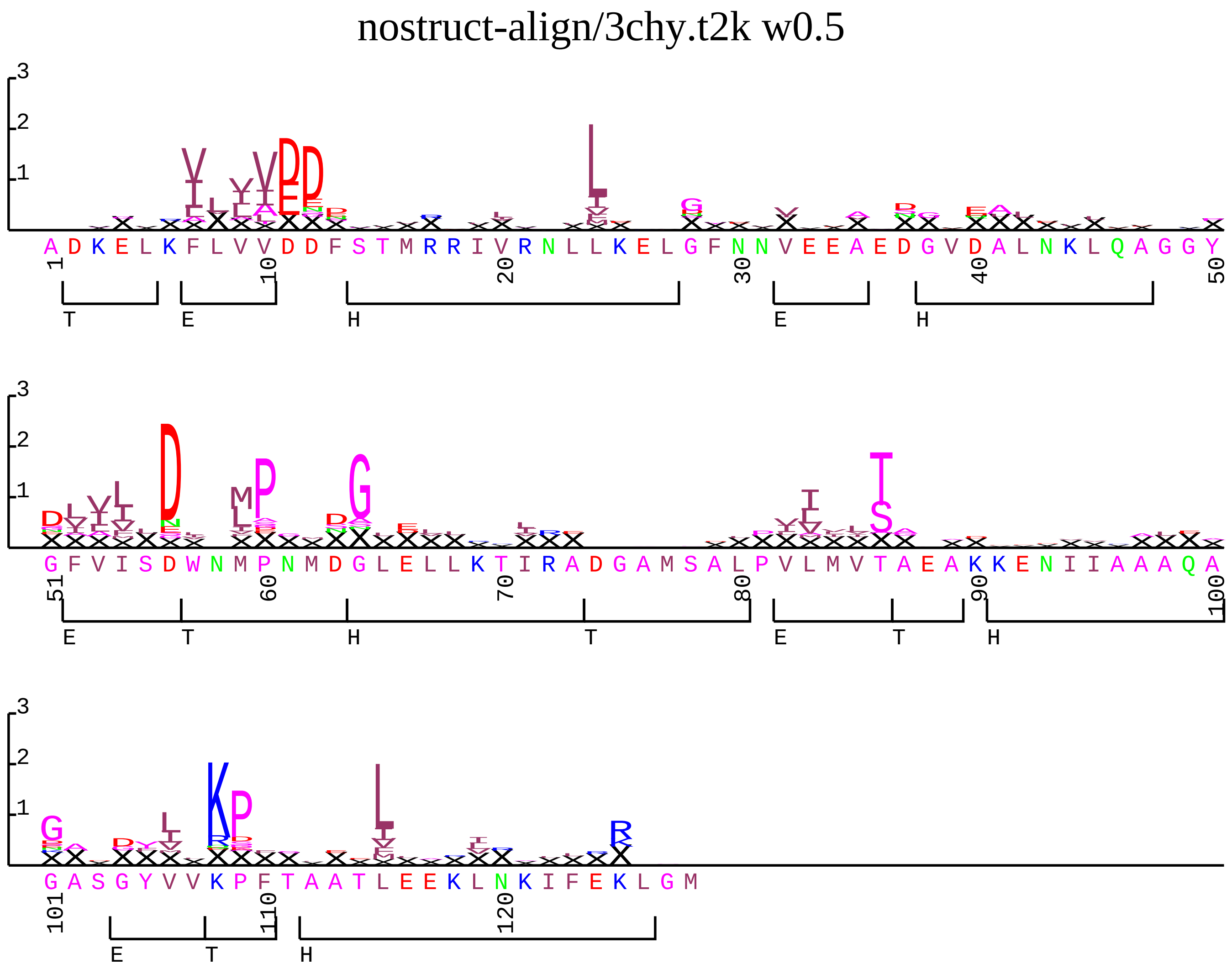
# Profile Hidden Markov Model



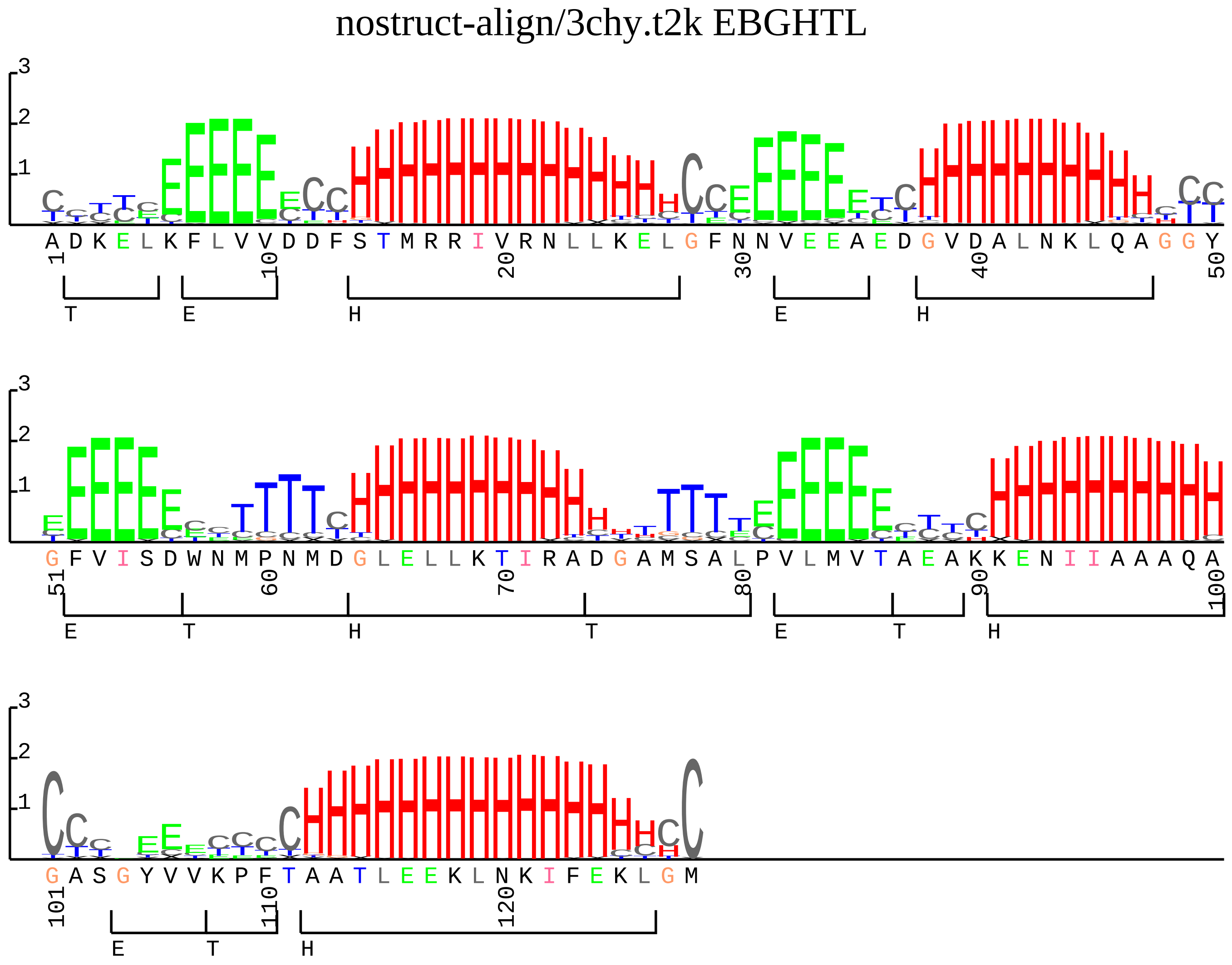
```
a1 a2 A3    -    A4  .  A5
 .  .  B1   B2   B3  b4 B5
```

- Circles are null states.
- Squares are *match states*, each of which is paired with a null *delete state*. We call the match-delete pair a *fat state*.
- Each fat state is visited exactly once on every path from Start to End.
- Diamonds are *insert states*, and are used to represent possible extra amino acids that are not found in most of the sequences in the family being modeled.

# What is single-track HMM looking for?



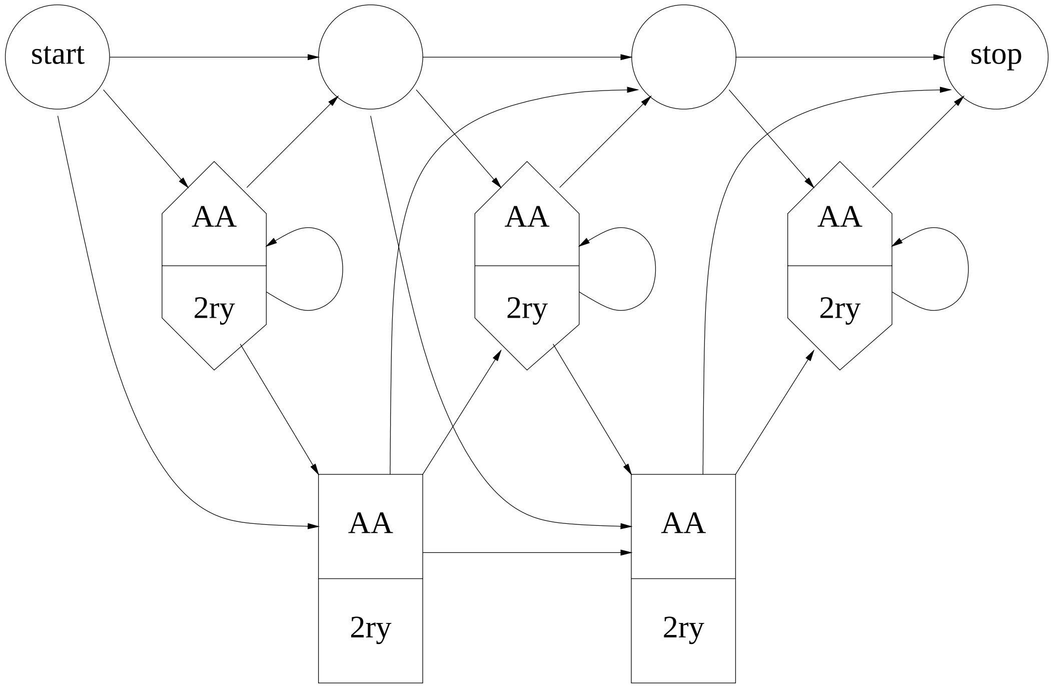
# What is second track looking for?



# Multi-track HMMS

We can also use alignments to build a two- or three-track target HMM:

- Amino-acid track (created from the multiple alignment).
- Local-structure track(s) with probabilities from neural net.
- Can align template (AA+local) to target model.



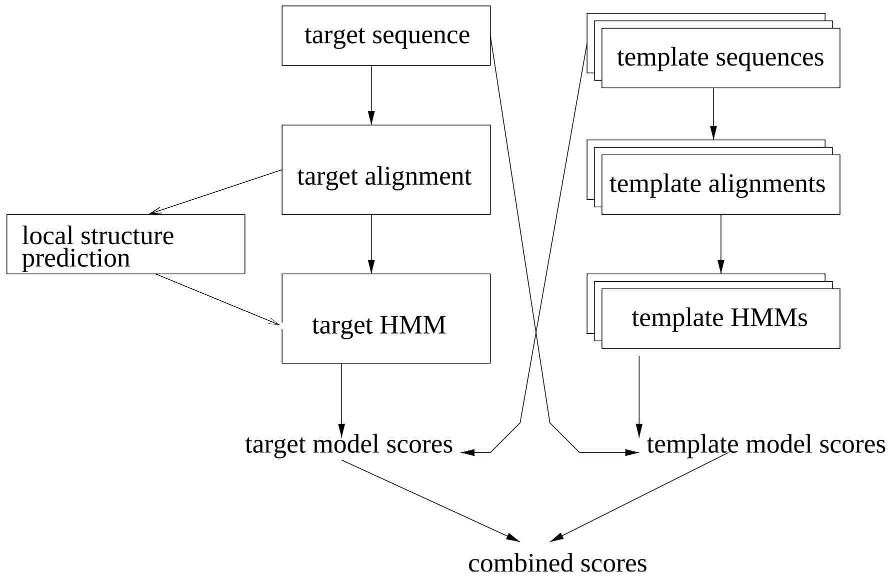
# Target-model Fold Recognition

- Find probable homologs of target sequence and make multiple alignment.
- Make secondary structure probability predictions based on multiple alignment.
- Build an HMM based on the multiple alignment and predicted 2ry structure (or just on multiple alignment).
- Score sequences and secondary structure sequences for proteins that have known structure (all sequences for AA-only, 8,000-11,000 representatives for multi-track).
- Select the best-scoring sequence(s) to use as templates.

# Template-library Fold Recognition

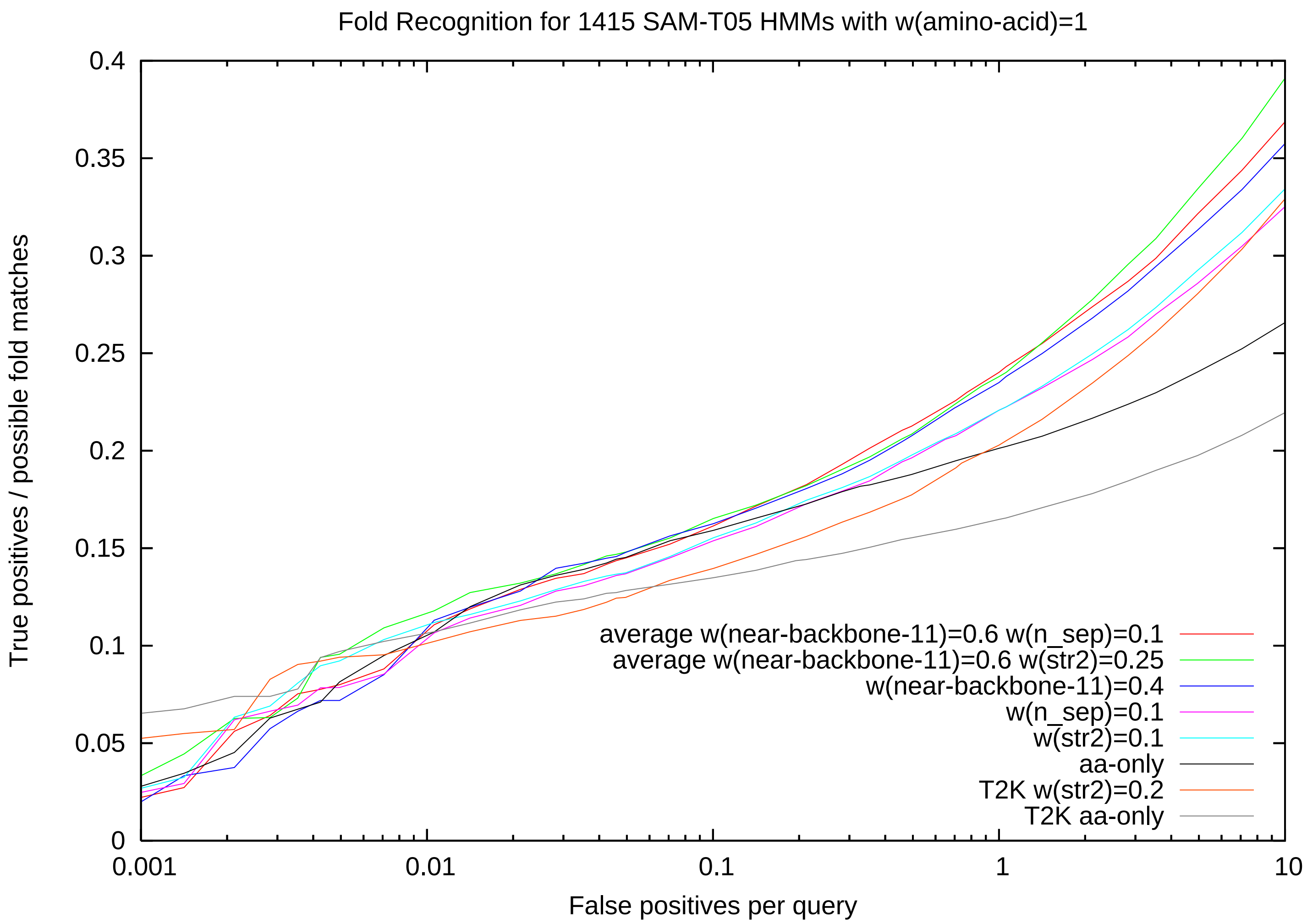
- Build an HMM for each protein in the template library, based on the template sequence (and any homologs you can find).
- The T2K library has over 11,000 templates from PDB.
- For the fold-recognition problem, structure information can be used in building these models (though we currently don't).
- Score target sequence with all models in the library.
- Select the best-scoring model(s) to use as templates.

# Combined SAM-T02 method



- Combine the costs from the template library search and the target library searches using different local structure alphabets.
- Choose one of the many alignments of the target and template (whatever method gets best results in testing).
- `http://www.soe.ucsc.edu/research/compbio/HMM-apps/T02-query.html`

# Fold recognition results



# Scoring HMMS and Bayes Rule

- The *model* $M$ is a computable function that assigns a probability Prob $(A \mid M)$ to each string $A$.
- When given a string $A$, we want to know how likely the model is. That is, we want to compute something like Prob $(M \mid A)$.
- Bayes Rule:

$$\text{Prob}\left(M \,\middle|\, A\right) = \text{Prob}\left(A \,\middle|\, M\right) \frac{\text{Prob}(M)}{\text{Prob}(A)} .$$

- Problem: Prob$(A)$ and Prob$(M)$ are inherently unknowable.

# Null models

- Standard solution: ask how much more likely $M$ is than some *null hypothesis* (represented by a *null model*).

$$\frac{\text{Prob}\,(M \mid A)}{\text{Prob}\,(N \mid A)} = \frac{\text{Prob}\,(A \mid M)}{\text{Prob}\,(A \mid N)}\,\frac{\text{Prob}(M)}{\text{Prob}(N)}\ .$$

- $\frac{\text{Prob}(M)}{\text{Prob}(N)}$ is the *prior odds ratio*, and represents our belief in the likelihood of the model before seeing any data.
- $\frac{\text{Prob}\left(M|A\right)}{\text{Prob}\left(N|A\right)}$ is the *posterior odds ratio*, and represents our belief in the likelihood of the model after seeing the data.

# Standard Null Model

- Null model is an i.i.d (independent, identically distributed) model.

$$\text{Prob}\left(A \,\middle|\, N, \text{len}\,(A)\right) = \prod_{i=1}^{\text{len}(A)} \text{Prob}(A_i)\ .$$

$$\begin{aligned}\text{Prob}\left(A \,\middle|\, N\right) &= \text{Prob}(\text{string of length len}\,(A)) \\ &\quad \prod_{i=1}^{\text{len}(A)} \text{Prob}(A_i)\ .\end{aligned}$$

- The length modeling is often omitted, but one must be careful then to normalize the probabilities correctly.

## Problems with standard null

- When using the standard null model, certain sequences and HMMs have anomalous behavior. Many of the problems are due to unusual composition—a large number of some usually rare amino acid.
- For example, metallothionein, with 24 cysteines in only 61 total amino acids, scores well on any model with multiple highly conserved cysteines.

# Reversed model for null

- We avoid composition bias (and several other problems) by using a reversed model $M^r$ as the null model.
- The probability of a sequence in $M^r$ is exactly the same as the probability of the reversal of the sequence given $M$.
- If we assume that $M$ and $M^r$ have equal prior likelihood, then

$$\frac{\text{Prob}\,(M \mid S)}{\text{Prob}\,(M^r \mid S)} = \frac{\text{Prob}\,(S \mid M)}{\text{Prob}\,(S \mid M^r)}\ .$$

- This method corrects for composition biases, length biases, and several subtler biases.

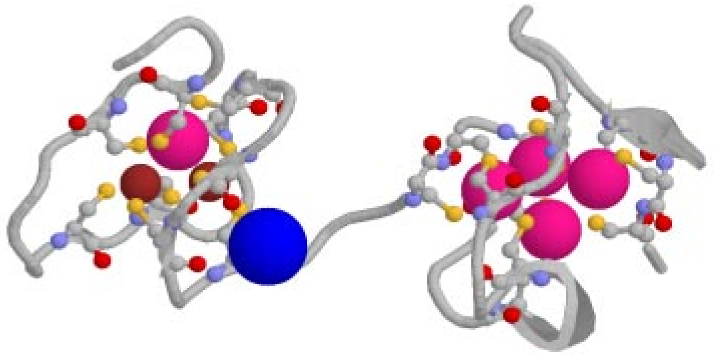
# Composition as source of error

A cysteine-rich protein, such as metallothionein, can match any HMM that has several highly-conserved cysteines, even if they have quite different structures:

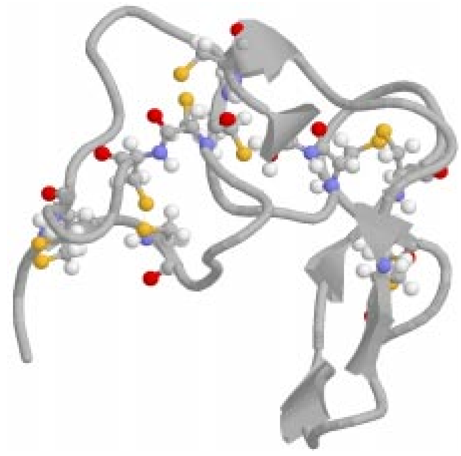
| | | cost in nats | |
|---|---|---|---|
| | | model – | model – |
| HMM | sequence | standard null | reversed-model |
| 1kst | 4mt2 | -21.15 | 0.01 |
| 1kst | 1tabI | -15.04 | -0.93 |
| 4mt2 | 1kst | -15.14 | -0.10 |
| 4mt2 | 1tabI | -21.44 | -1.44 |
| 1tabI | 1kst | -17.79 | -7.72 |
| 1tabI | 4mt2 | -19.63 | -1.79 |

# Composition examples

Metallothionein Isoform II (4mt2)

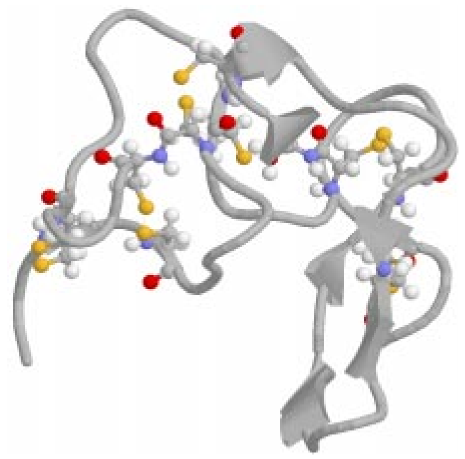


Kistrin (1kst)

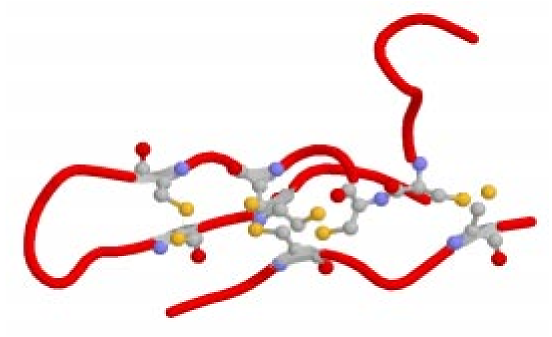


# Composition examples

Kistrin (1kst)

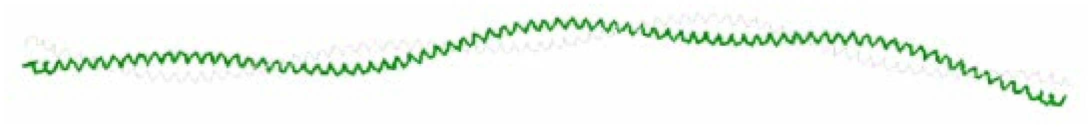


Trypsin-binding domain of Bowman-Birk Inhibitor (1tabI)



# Helix examples

Tropomyosin (2tmaA)

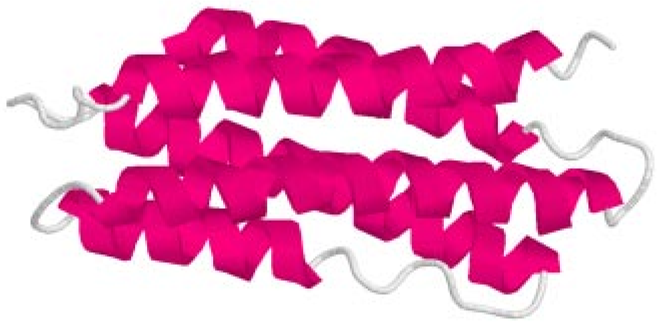


Colicin Ia (1cii)

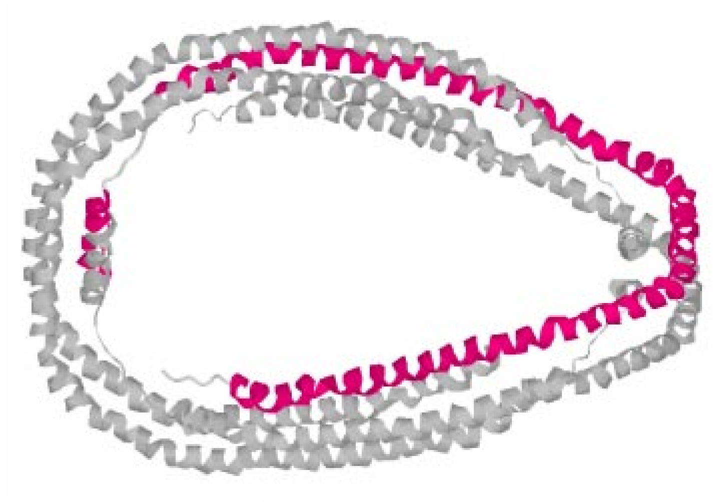
Flavodoxin mutant (1vsgA)

# Helix examples

Apolipophorin III (1aep)



Apolipoprotein A-I (1av1A)



# What is Statistical Significance?

- The statistical significance of a hit, $P_1$, is the probability of getting a score as good as the hit “by chance,” when scoring a single “random” sequence.
- When searching a database of $N$ sequences, the significance is best reported as an E-value—the expected number of sequences that would score that well by chance: $E = P_1 N$.
- Some people prefer the p-value: $P_N = 1 - (1 - P_1)^N$, For large $N$ and small $E$, $P_N \approx 1 - e^{-E} \approx E$.
- I prefer E-values, because our best scores are often not significant, and it is easier to distinguish between E-values of 10, 100, and 1000 than between p-values of 0.999955, 1.0 – 4E-44, and 1.0 – 5E-435

# Approaches to Statistical Significance

- (Markov's inequality) For any scoring scheme that uses

$$\ln \frac{\text{Prob}\,(\text{seq} \mid M_1)}{\text{Prob}\,(\text{seq} \mid M_2)}$$

  the probability of a score better than $T$ is less than $e^{-T}$ for sequences distributed according to $M_2$. This method is independent of the actual probability distributions.

- (Classical parameter fitting) If the "random" sequences are not drawn from the distribution $M_2$, but from some other distribution, then we can try to fit some parameterized family of distributions to scores from a random sample, and use the parameters to compute $P_1$ and $E$ values for scores of real sequences.

# Our Assumptions

**Bad assumption 1:** The sequence and reversed sequence come from the same underlying distribution.

**Bad assumption 2:** The scores with a standard null model are distributed according to an extreme-value distribution:

$$P\left(\ln \text{Prob}\left(\text{seq} \,\middle|\, M\right) > T\right) \approx G_{k,\lambda}(T) = 1 - \exp(-ke^{\lambda T}) \ .$$

**Bad assumption 3:** The scores with the model and the reverse-model are independent of each other.

**Result:** The scores using a reverse-sequence null model are distributed according to a sigmoidal function:

$$P(\text{score} > T) = (1 - e^{\lambda T})^{-1} \ .$$

# Derivation of sigmoidal distribution

(Derivation for *costs*, not *scores*, so more negative is better.)

$$\begin{aligned}
P(\text{cost} < T) &= \int_{-\infty}^{\infty} P(c_M = x) \int_{x-T}^{\infty} P(c_{M'} = y)\,dy\,dx \\
&= \int_{-\infty}^{\infty} P(c_M = x) P(c_{M'} > x - T)\,dx \\
&= \int_{-\infty}^{\infty} k\lambda \exp(-ke^{\lambda x}) e^{\lambda x} \exp(-ke^{\lambda(x-T)})\,dx \\
&= \int_{-\infty}^{\infty} k\lambda e^{\lambda x} \exp(-k(1 + e^{-\lambda T}) e^{\lambda x})\,dx
\end{aligned}$$

# Derivation of sigmoid (cont.)

If we introduce a temporary variable to simplify the formulas: $K_T = k(1 + \exp(-\lambda T))$, then

$$\begin{aligned}
P(\text{cost} < T) &= \int_{-\infty}^{\infty} (1 + e^{-\lambda T})^{-1} K_T \lambda e^{\lambda x} \exp(-K_T e^{\lambda x}) dx \\
&= (1 + e^{-\lambda T})^{-1} \int_{-\infty}^{\infty} K_T \lambda e^{\lambda x} \exp(-K_T e^{\lambda x}) dx \\
&= (1 + e^{-\lambda T})^{-1} \int_{-\infty}^{\infty} g_{K_T,\lambda}(x) dx \\
&= (1 + e^{-\lambda T})^{-1}
\end{aligned}$$

# Fitting $\lambda$

- The $\lambda$ parameter simply scales the scores (or costs) before the sigmoidal distribution, so $\lambda$ can be set by matching the observed variance to the theoretically expected variance.
- The mean is theoretically (and experimentally) zero.
- The variance is easily computed, though derivation is messy:

$$E(c^2) = (\pi^2/3)\lambda^{-2} \ .$$

- $\lambda$ is easily fit by matching the variance:

$$\lambda \approx \pi \sqrt{N/(3 \sum_{i=0}^{N-1} c_i^2)} \ .$$

# Two-parameter family

- We made three dangerous assumptions: reversibility, extreme-value, and independence.
- To give ourselves some room to compensate for deviations from the extreme-value assumption, we can add another parameter to the family.
- We can replace $-\lambda T$ with any strictly decreasing odd function.
- Somewhat arbitrarily, we chose

$$-\operatorname{sign}(T)|\lambda T|^{\tau}$$

so that we could match a “stretched exponential” tail.

# Fitting a two-parameter family

For two-parameter symmetric distribution, we can fit using 2nd and 4th moments:

$$\begin{aligned} E(c^2) &= \lambda^{-2/\tau} K_{2/\tau} \\ E(c^4) &= \lambda^{-4/\tau} K_{4/\tau} \end{aligned}$$

where $K_x$ is a constant:

$$\begin{aligned} K_x &= \int_{-\infty}^{\infty} y^x (1+e^y)^{-1} (1+e^{-y})^{-1} dy \\ &= -\Gamma(x+1) \sum_{k=1}^{\infty} (-1)^k / k^x \ . \end{aligned}$$

# Fitting a two-parameter family (cont.)

- The ratio $E(c^4)/(E(c^2))^2 = K_{4/\tau}/K^2_{2/tau}$ is independent of $\lambda$ and monotonic in $\tau$, so we can fit $\tau$ by binary search.
- Once $\tau$ is chosen we can fit $\lambda$ using $E(c^2) = \lambda^{-2/\tau}K_{2/\tau}$.

## Student’s t-distribution

- On the advice of statistician David Draper, we tried maximum-likelihood fits of Student’s t-distribution to our heavy-tailed symmetric data.
- We couldn’t do moment matching, because the degrees of freedom parameter for the best fits turned out to be less than 4, where the 4th moment of Student’s t is infinite.
- The maximum-likelihood fit of Student’s t seemed to produce too heavy a tail for our data.
- We plan to investigate other heavy-tailed distributions.

# Use database, not random sequences

- Calibration with random sequences works ok for 1-track, but not 2-track HMMs.
- “Random” secondary structure sequences (i.i.d. model) are **not** representative of real sequences.
- Fixes:
  - Better secondary structure decoy generator
  - Use real database, but avoid problems with contamination by true positives by taking only costs $> 0$ to get estimate of $E(\text{cost}^2)$ and $E(\text{cost}^4)$.

# What went wrong with Protein Blocks?

- de Brevern's protein blocks provided one of our most predictable local structure alphabets.
- The 2-track HMMs using de Brevern's protein blocks did **much** worse than AA-only HMMs. Why?
- The protein blocks alphabet strongly violates reversibility assumption.
- Encoding cost in bits for secondary structure strings using Markov chains:

| alphabet | 0-order | 1st-order | reverse-forward |
|---|---|---|---|
| amino acid | 4.1896 | 4.1759 | 0.0153 |
| stride | 2.3330 | 1.0455 | 0.0042 |
| dssp | 2.5494 | 1.3387 | 0.0590 |
| pb | 3.3935 | 1.4876 | 3.0551 |

## Undertaker

- Undertaker is UCSC's attempt at a fragment-packing program.
- Named because it optimizes burial.
- Representation is 3D coordinates of all heavy atoms (not hydrogens).
- Can replace backbone fragments (a la Rosetta) or full alignments—chain need not remain contiguous.
- Conformations can borrow heavily from fold-recognition alignments, without having to lock in a particular alignment.
- Use genetic algorithm with many conformation-change operators to do stochastic search.

# Fragfinder

Fragments are provided to undertaker from 3 sources:

- Generic fragments (2-4 residues, exact sequence match) are obtained by reading in 500–1000 PDB files, and indexing all fragments.
- Long specific fragments (and full alignments) are obtained from the various target and template alignments generated during fold recognition.
- Medium-length fragments (9–12 residues long) for every position are generated from the HMMs with `fragfinder`, a new tool in the SAM suite.

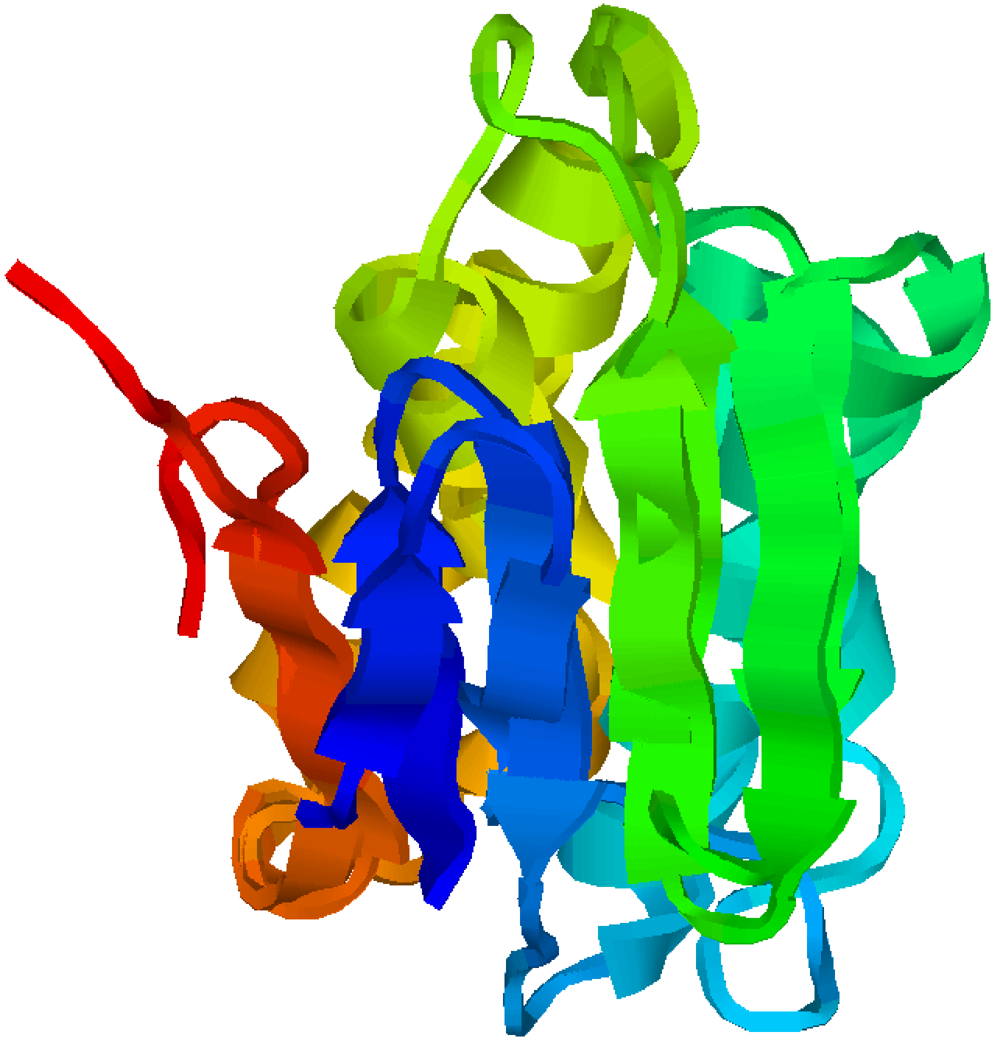
## Cost function

- Cost function is modularly designed—easy to add or remove terms.
- Cost function can include predictions of local properties by neural nets.
- Clashes and hydrogen bonds are important components.
- There are over 40 cost function components available: burial functions, disulfides, contact order, rotamer preference, radius of gyration, constraints, ...

# Target T0201 (NF)

- We tried forcing various sheet topologies and selected 4 by hand.
- Model 1 has right topology (5.912Å all-atom, 5.219Å $C_\alpha$).
- Unconstrained cost function not good at choosing topology (two strands curled into helices).
- Helices were too short.

# Target T0201 (NF)



## Contact prediction

- Use mutual information between columns.
- Thin alignments aggressively (30%, 35%, 40%, 50%, 62%).
- Compute e-value for mutual info (correcting for small-sample effects).
- Compute rank of log(e-value) within protein.
- Feed log(e-values), log rank, contact potential, joint entropy, and separation along chain for pair, and amino-acid profile, predicted burial, and predicted secondary structure for each residue of pair into a neural net.

# Open problem

Given a contingency table for a small sample of pairs of independent discrete random variables, what is the distribution of the mutual information statistic:

$$MI(X,Y) = \sum_{i,j} P(i,j) \log \frac{P(i,j)}{P(i)P(j)} \ ,$$

where the probabilities are the maximum-likelihood estimates from the observed sample.
Asymptotic results ($\chi^2$ distribution) are known, but neither the shape of the distribution nor how to fit its parameters have been established theoretically (we have good empirical fits).

# Evaluating contact prediction

Two measures of contact prediction:

- Accuracy:

$$\frac{\sum \chi(i,j)}{\sum 1}$$

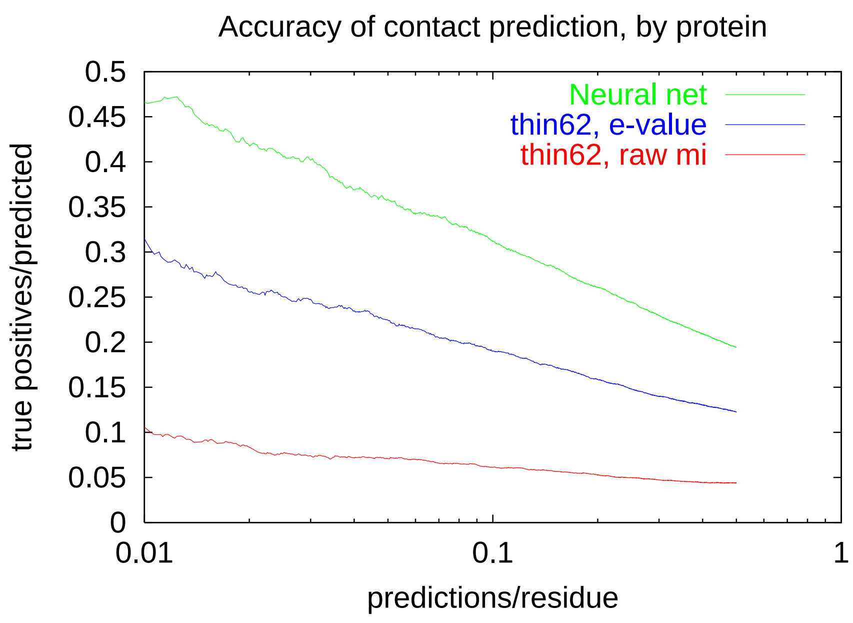
(favors short-range predictions, where contact probability is higher)

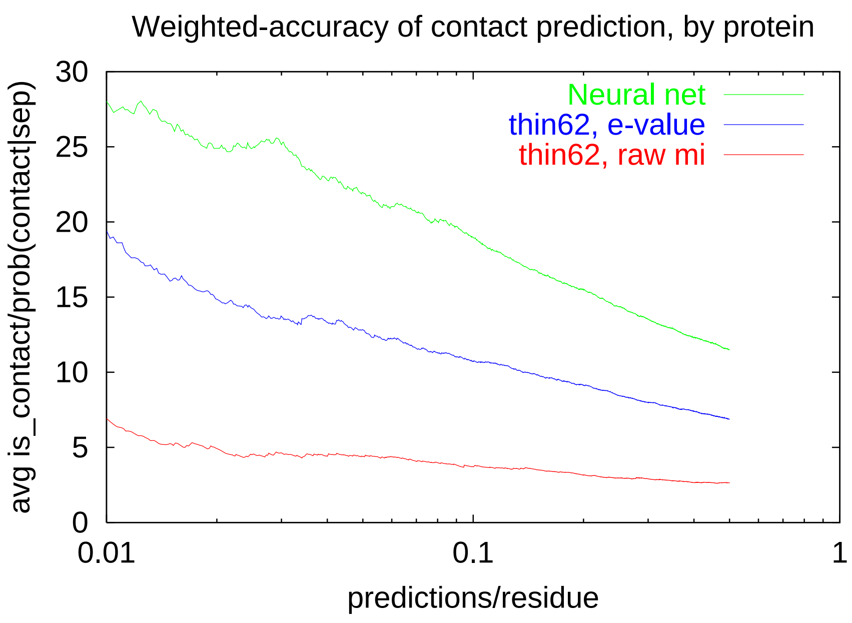
- Weighted accuracy:

$$\frac{\sum \frac{\chi(i,j)}{\text{Prob}\left(\text{contact}|\text{separation}_{=|i-j|}\right)}}{\sum 1}$$

(1 if predictions no better than chance based on separation).

# Contact prediction results

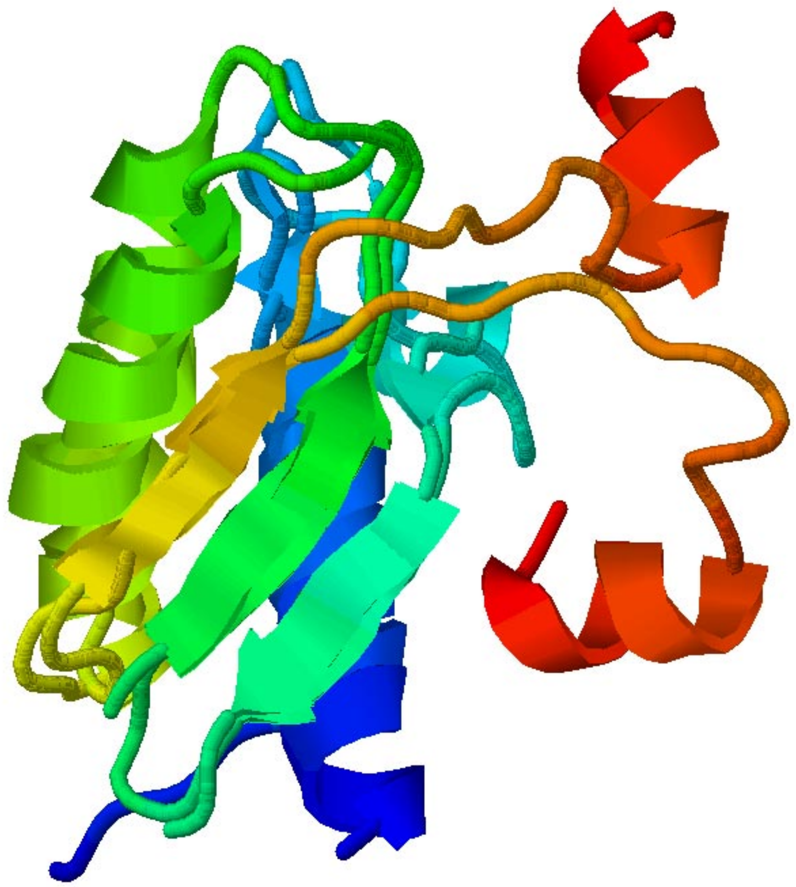




## Target T0230 (FR/A)

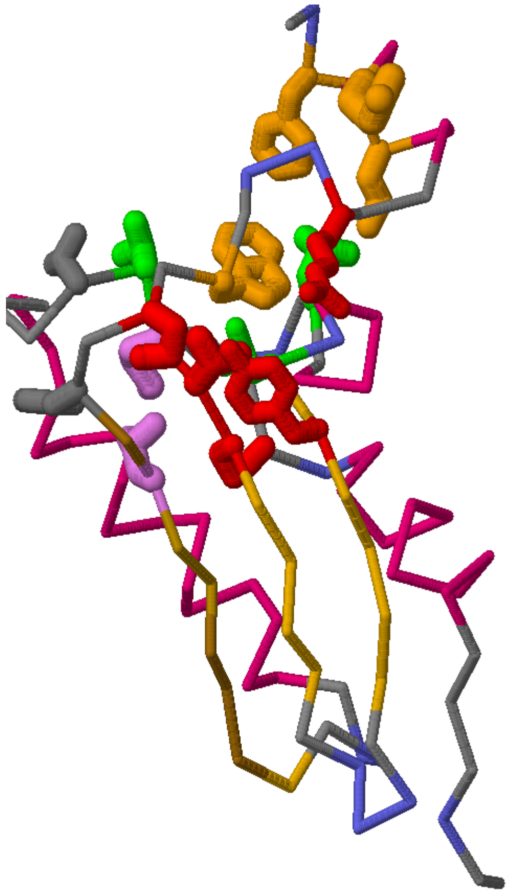
- Good except for C-terminal loop and helix flopped wrong way.
- We have secondary structure right, including phase of beta strands.
- Contact prediction helped, but we put too much weight on it—decoys fit predictions better than real structure does.

# Target T0230 (FR/A)



# Target T0230 (FR/A)

Real structure with contact predictions:



## Web sites

**These slides:**

`http://www.soe.ucsc.edu/~karplus/papers/not-just-opt-may-2006.pdf`

**SAM-T06 prediction server:**

`http://www.soe.ucsc.edu/research/compbio/SAM_T06/T06-query.html`

**CASP6 all our results and working notes:**

`http://www.soe.ucsc.edu/~karplus/casp6/`

**Predictions for all yeast proteins:**

`http://www.soe.ucsc.edu/~karplus/yeast/`

**UCSC bioinformatics (research and degree programs) info:**

`http://www.soe.ucsc.edu/research/compbio/`

**SAM tool suite info:**

`http://www.soe.ucsc.edu/research/compbio/sam.html`