Supplementary notes for "*Statistical inference of protein structural alignments using information and compression*"

James H. Collier[1], Lloyd Allison[1], Arthur M. Lesk[2], Peter J. Stuckey [3], Maria Garcia de la Banda[1] & Arun S. Konagurthu[1*]

[1]*Faculty of Information Technology, Monash University, Clayton, VIC 3800 Australia.*

[2]*Department of Biochemistry and Molecular Biology, Pennsylvania State University, University Park, PA 16802 USA.*

[3]*Department of Computing and Information Systems, University of Melbourne, Parkville, VIC 3010 Australia.*

S1 Bayesian framework of `I-value` and its statistical properties

The principles and main features of the Minimum Message Length (MML) framework are described in the main text. To summarise, given a structural alignment as a one-to-one correspondence (denoted as $\mathcal{A}$) between coordinate data of a pair of protein structures (denoted by $\langle S, T \rangle$), `I-value` estimates the Shannon *information* content[1] (denoted as $I(\mathcal{A} \,\&\, \langle S, T \rangle)$) as the negative logarithm of the joint probability of the alignment hypothesis $\mathcal{A}$ and the data $\langle S, T \rangle$:

$$I\text{-value} = I(\mathcal{A} \,\&\, \langle S, T \rangle) = -\log_2(\Pr(\mathcal{A} \,\&\, \langle S, T \rangle)).$$

*to whom correspondence should be addressed

From product rule of probabilites over the events $\mathcal{A}$, S and T, we have

$$\Pr(\mathcal{A} \,\&\, \langle S,T\rangle) = \Pr(\mathcal{A})\Pr(\langle S,T\rangle\,|\mathcal{A}) = \Pr(\langle S,T\rangle)\Pr(\mathcal{A}|\,\langle S,T\rangle)$$

where $\Pr(\mathcal{A}, \langle S,T\rangle)$ is the joint probability of alignment $\mathcal{A}$ and the structural coordinates in S and T, $\Pr(\mathcal{A})$ is the prior probability of the alignment $\Pr(\langle S,T\rangle\,,\mathcal{A})$ is the likelihood, $\Pr(\langle S,T\rangle)$ is the prior probability of the structural coordinates in S and T, and $\Pr(\mathcal{A}|\,\langle S,T\rangle)$ is the posterior probability of the alignment.

This product rule can be restated in Shannon's information[1] terms by applying negative logarithm on both sides:

$$\underbrace{-\log\left(\Pr(\mathcal{A} \,\&\, \langle S,T\rangle)\right)}_{I(\mathcal{A}\,\&\,\langle S,T\rangle)\text{ or }\texttt{I-value}} = \underbrace{-\log(\Pr(\mathcal{A}))}_{I(\mathcal{A})} - \underbrace{\log(\Pr(\langle S,T\rangle\,|\mathcal{A}))}_{I(\langle S,T\rangle\,|\mathcal{A})} = \underbrace{-\log(\Pr(\langle S,T\rangle))}_{I(\langle S,T\rangle)} - \underbrace{\log(\Pr(\mathcal{A}|\,\langle S,T\rangle))}_{I(\mathcal{A}|\,\langle S,T\rangle)}$$

As mentioned in the main text, $I(\mathcal{A}\,\&\,\langle S,T\rangle)$ or `I-value` is computed as a length of a two-part message:

$$\begin{aligned}
\underbrace{I(\mathcal{A}\,\&\,\langle S,T\rangle)}_{I\text{-value}} &= \underbrace{I(\mathcal{A})}_{\text{First part}} + \underbrace{I(\langle S,T\rangle\,|\mathcal{A})}_{\text{Second part}} \\
&= I(\mathcal{A}) + I(S|\mathcal{A}) + I(T|S\&\mathcal{A}) \\
&= \underbrace{I(\mathcal{A})}_{\text{First part}} + \underbrace{I_{\text{null}}(S) + I(T|S\&\mathcal{A})}_{\text{Second part}} \qquad \text{bits.} \qquad (1)
\end{aligned}$$

The `I-value` measure has several desirable statistical properties:

1. The `I-value` varies according to the posterior probability of $\mathcal{A}$: $I(\mathcal{A}\,\&\,\langle S,T\rangle) \propto -\log(\Pr(\mathcal{A}|\,\langle S,T\rangle))$.

2. The difference between the `I-values` of any two competing alignments, say $\mathcal{A}_1$ and $\mathcal{A}_2$, gives the log-odds posterior ratio:

$$\begin{aligned} I(\mathcal{A}_1 \& \ \langle S,T\rangle) - I(\mathcal{A}_2 \& \ \langle S,T\rangle) &= -\log(\Pr(\mathcal{A}_1 \& \langle S,T\rangle)) + \log(\Pr(\mathcal{A}_2 \& \langle S,T\rangle)) \\ &= \log\left(\frac{\Pr(\mathcal{A}_2 |\, \langle S,T\rangle)}{\Pr(\mathcal{A}_1 |\, \langle S,T\rangle)}\right). \end{aligned}$$

This property makes the comparison and selection of competing alignments statistically robust.

3. This framework provides a natural null hypothesis test. If the `I-value` of an alignment $\mathcal{A}$ is worse (longer) than that of the null model encoding of the structural coordinates, then $\mathcal{A}$ must be rejected. That is, reject $\mathcal{A}$ if $I(\mathcal{A} \,\&\, \langle S,T\rangle) \geq I_{\text{null}}(S) + I_{\text{null}}(T)$. See Section S2 for details.

The computation of `I-value` involves message length terms such as $I(\mathcal{A})$, $I(\langle S,T\rangle \,|\mathcal{A})$, $I_{\text{null}}(S)$, $I_{\text{null}}(T)$ *et cetera* rely on probabilistic models of encoding, described below.

S2 Computation of Null model message length: $I_{\text{null}}(S)$ and $I_{\text{null}}(T)$

Kasarapu and Allison (2015)[2,3] recently proposed an MML-based unsupervised learning method to infer a probabilistic mixture model using directional probability density functions. In particular, they looked at three-dimensional (3D) von Mises-Fisher and Kent probability distributions wrapped on a 3D unit sphere [2,3]. These mixture models were inferred on the empirically observed

directions† of C_α coordinate data from wwPDB. (See `http://lcb.infotech.monash.edu.au/kent-mixture-modelling/`.)

A mixture model is a probability density function of the form:

$$\mathcal{M}(\hat{x};\vec{\Psi}) = \sum_{k=1}^{|\mathcal{M}|} w_k f_k(\hat{x};\vec{\vartheta}_k) \tag{2}$$

where, $\hat{x}$ is the random variable denoting a direction, $\vec{\Psi} = \{|\mathcal{M}|, \{(w_k, \vec{\vartheta}_k)_{\forall 1 \leq k \leq |\mathcal{M}|}\}\}$ gives a vector of parameters of the mixture model containing $|\mathcal{M}|$ component directional probability density functions, f_k, $(\forall 1 \leq k \leq |\mathcal{M}|)$, with their respective component probabilities w_k (note, $\sum_{k=0}^{|\mathcal{M}|} w_k = 1$) and component parameters $\vec{\vartheta}_k$.

This work employs the Kent mixture model inferred by Kasarapu (2015)[3] on protein coordinate data. As a demonstration of the fidelity of their inferred mixture model, Fig. S1 shows the empirical distribution of C_α directions (left frame) along with the distribution of randomly sampled directions from the mixture model of Kasarapu (2015)[3] (right frame). The similarity between the two distributions clearly suggests that the mixture model faithfully characterises, into a parametric form, the empirical distribution of C_α directions.

To describe the computation of the null model message length terms using the 23-component Kent mixture model[3], $I_{\text{null}}(S)$ and $I_{\text{null}}(T)$, consider an arbitrary chain C containing n successive C_α coordinates $C = \{\vec{c}_1, \vec{c}_2, \cdots \vec{c}_n\}$. The length of the null model message we use to transmit this

†Directions are given by the spherical coordinates of C_α atoms, denoting the zenith angle (or co-latitude) and azimuth angle (or longitude), measured in a canonical frame of reference.

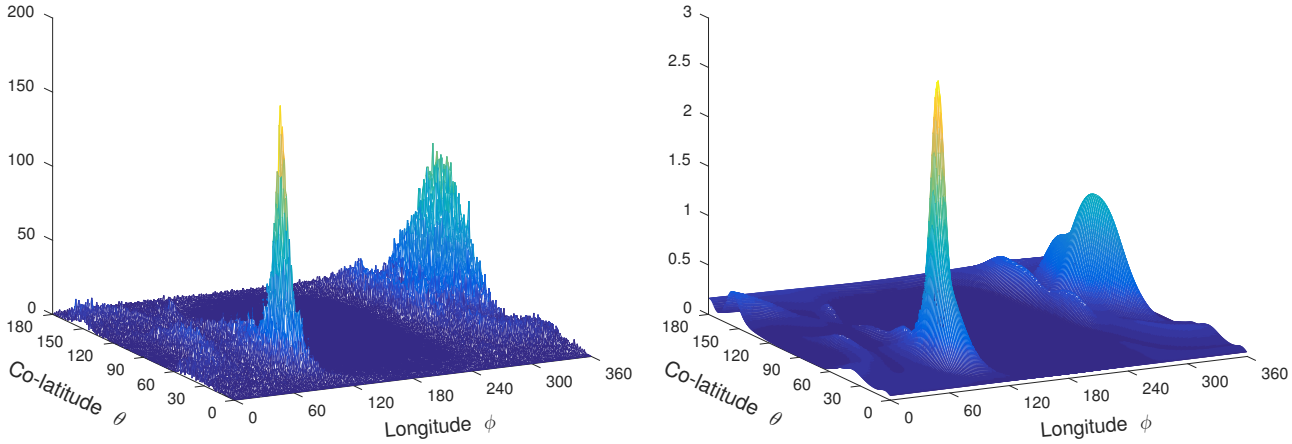

Figure S1: Fidelity of the 23-component Kent mixture model of Kasarapu (2015).[3] Left frame: The empirical distribution of canonical directions of C_α atoms from a set of 1,802 SCOP domains. Each observed direction is represented here using the spherical coordinates (θ, ϕ) denoting the spherical coordinates (also co-latitude and longitude. Right frame: The distribution of randomly sampled directions drawn from the probability distribution defined by the 23-component Kent mixture model.

chain is given by:

$$I_{\text{null}}(C) = I_{\log^*}(n) + \sum_{j=1}^{n} I_{\text{null}}(\vec{c}_j) \tag{3}$$

where term $I_{\log^*}(n)$ represents the number of bits needed to transmit number n over a $\log^*$ integer code.[4] The term $I_{\text{null}}(\vec{c}_j)$ represents the number of bits needed to transmit a single $\vec{c}_j$ using the 23-component Kent mixture model whose computation is described below.

Let r_j and $\hat{x}_j$ be the spherical coordinates, radius and direction (co-latitude and longitude), respectively of any coordinate $\vec{c}_j \in C$. The null model message length to transmit $\vec{c}_j$ is given by $I_{\text{null}}(\vec{c}_j) = I_{\text{radius}}(r_j) + I_{\text{direction}}(\hat{x}_j)$ bits, that is, by the sum of the code lengths required to transmit the radius and the direction respectively. The transmission of the radius relies on the observation that the partial double bond character of the peptide bond in proteins imposes a strict constraint on the distances between successive C_α atoms. Hence, each r_j is the distance between C_α atom

$\vec{c}_{j-1}$ and $\vec{c}_j$, and can be encoded efficiently using a normal distribution $\mathcal{N}$ with the parameters $\mu = 3.8$Å and standard deviation of $\sigma = 0.2$ Å.[4] This code length is computed as $I_{\text{radius}}(r_j) = -\log_2(\epsilon \cdot \mathcal{N}(r_j; \mu, \sigma))$.

The receipt of r_j by the receiver reduces their uncertainty of the position of $\vec{c}_j$ to lie anywhere on the surface of a sphere with radius r_j centered at $\vec{c}_{j-1}$. However, the direction $\hat{x}_j$ of $\vec{c}_j$ is still unknown to the receiver and must be transmitted so that the receiver can decode $\vec{c}_j$ without loss of information. Each $\hat{x}_j \equiv \langle \theta_j, \phi_j \rangle$ pair is encoded and transmitted in the context of the preceding three transmitted coordinates[‡] using the 23-component Kent mixture model, denoted as $\mathcal{M}$ as shown in Equation 2. The code length to encode the direction is computed as $I_{\text{direction}}(\hat{x}_j) = -\log_2(\epsilon^2 \cdot \mathcal{M}(\hat{x}_j; \vec{\Psi})) = -\log_2\left(\epsilon^2 \cdot \sum_{k=1}^{|\mathcal{M}|} w_k f_k(\hat{x}_j; \vec{\vartheta}_k)\right)$.

Time Complexity of computing the null model message length The computation of the null model message length for any chain of C_α coordinates with n atoms requires $O(n)$ effort. This is easy to see: the computation of each $I_{\text{direction}}(\hat{x}_j)$ requires computing the likelihood of $\hat{x}_j$ given each component density function in the null mixture. The mixture model[3] contains a constant ($|\mathcal{M}| = 23$) number of component Kent probability density functions. So the likelihood of each $\hat{x}_j$ can be computed in constant time. Similarly, the computation of $I_{\text{radius}}(r_j)$ of each C_α atom is also a constant time operation. Therefore, over all n atoms, the computation of the null model message length of a given chain is $O(n)$.

[‡]Since this encoding requires a context of three preceding residues, as a boundary case, the first three C_αs are transmitted using the null model.

S3 Computing relationship model messagelength: $I(\mathcal{A}\ \&\ \langle S,T\rangle)$

As described in the main text, the relationship model message is encoded over two-parts. In the first part the alignment $\mathcal{A}$ is transmitted as a string over `match (m)`,`insert (i)`,`delete (d)` states, followed by the coordinate data of the structures, $\langle S,T\rangle$, given this alignment relationship. The length of this two-part message is shown in Equation 1. The details of encoding required for the two parts are explained under their respective subheadings.

Computing the first part message length, $I(\mathcal{A})$: A three-state probabilistic model for alignment encoding is used here, described originally by Allison et al. (1992)[5] to encode alignments of macromolecules, specifically over the match (m), insert (i), and delete (d) states. The communication of an alignment $\mathcal{A}$ from transmitter to receiver can be achieved by first sending its length, $|\mathcal{A}|$, over the $\log^*$ integer code, followed by sending each state symbol over a probability distribution modeled using a first-order Markov model. Such a model permits 9 possible state transitions: `mm, im, dm, mi, ii, di, md, id, dd`. Associated with each possible state transition is a transition probability: $\Pr(\texttt{mm})$, $\Pr(\texttt{im})$, $\Pr(\texttt{dm})$, and so on. These probabilities are subject to the total probability and symmetry constraints: $\Pr(\texttt{mm})+\Pr(\texttt{mi})+\Pr(\texttt{md})=1$, $\Pr(\texttt{im})+\Pr(\texttt{ii})+\Pr(\texttt{id})=1$, $\Pr(\texttt{dm})+\Pr(\texttt{di})+\Pr(\texttt{dd})=1$, $\Pr(\texttt{mi})=\Pr(\texttt{im})=\Pr(\texttt{md})=\Pr(\texttt{dm})$, $\Pr(\texttt{dd})=\Pr(\texttt{ii})$, and $\Pr(\texttt{id})=\Pr(\texttt{di})$.

While these probabilities can be computed for any given alignment string on the transmitter's side, the receiver however will need to know these values in advance to decode the alignment string. An adaptive code similar to the one described by Wallace and Boulton[6] can be used to construct a

decodable message over this first-order Markov model. This approach requires maintaining only 4 counters (`cntr1,...,cntr4`), one for each *distinct* probability term. These four counters are initialized to 1. The first state symbol (eg: '`m`') is transmitted with a uniform probability of $1/3$. Subsequently, each state symbol (eg: '`i`') is transmitted over the probability that can be computed by dividing the counter pertaining to the current state transition (eg: '`mi`') with the sum total of all counters starting with the previous state symbol (eg: '`mm`' or '`mi`' or '`md`'). Note that both the transmitter and receiver can encode and decode (respectively) the state symbols over such a transmission. Once the current state symbol is transmitted, the counter pertaining to this state transition is incremented by one.

The code length to encode a state transition is the negative logarithm of its estimated probability. Summing up over all state transitions in the alignment string, and adding to it the code length required to transmit the size of the alignment over an integer code results in the estimation of $I(\mathcal{A})$ used in this work. (The symmetry constraints make the formulation of $I(\mathcal{A})$ into a closed-form formula rather inconvenient. Nevertheless, the computation of $I(\mathcal{A})$ is computationally efficient as discussed below.)

Time Complexity of computing $I(\mathcal{A})$ Assume $|S| = m$ and $|T| = n$. Without loss of generality, assume also that $m \leq n$. The maximum string length of any possible alignment $\mathcal{A}$ between S and T is $m + n \leq 2n = O(n)$. Using the adaptive code the computation of $I(\mathcal{A})$ is efficient as it requires maintaining a fixed set of counters and, for each symbol in $\mathcal{A}$, dividing two numbers and performing a negative logarithm of that ratio. Therefore, the computation of $I(\mathcal{A})$ is $O(n)$.

Computing the second part message length, $I(\langle S, T\rangle \,|\mathcal{A})$: The transmission of the coordinates of S and T using the alignment information forms the second part of the message. To achieve this, the coordinates in S are transmitted over the null model approach described in Section S2 over a message that is $I_{\text{null}}(S)$ bits long. After decoding this information the receiver knows both $\mathcal{A}$ (from receiving the first part) and the coordinates of S. The goal now is to transmit the coordinates of T based on this available information.

Given $\mathcal{A}$ and S, each C_α in T is either aligned with some C_α in S or remains unaligned. Therefore, T can be partitioned into successive monotonic blocks alternating between aligned and unaligned C_α coordinates. By monotonic block, we mean any run of successive C_αs that are all either aligned or unaligned in T. For example, the alignment `iiidmmmmmmmim` can be partitioned in 5 blocks (delimited by '`|`' symbols) as `iii|d|mmmmmm|i|m`, where the blocks of `i`'s represent the unaligned coordinates of T while the blocks of `m`'s represent the coordinates in T that are aligned with corresponding coordinates in S. (Note that the blocks containing `d`'s can be ignored since they represent the unaligned coordinates in S, which have already been transmitted, and for this part of the transmission we are only interested in the coordinates of T.)

The unaligned coordinates of T (i.e., each chain of successive coordinates that forms any block of `i`'s in the alignment) are transmitted using the null model method described in Section S2. Thus, these unaligned coordinates offer no compression with respect to the null model transmission. Let us denote the code length to transmit these unaligned coordinates in T as $I_{\text{unaligned}}(T|S, \mathcal{A})$.

The source of compression in this scheme potentially comes from encoding the aligned coordinates in T, building on the information of their corresponding coordinates in S. Below, we propose an efficient encoding method for such coordinates.

In this work, the estimation of the code length term, $I_{\text{aligned}}(T|S, \mathcal{A})$, to transmit the aligned coordinates of T given S and $\mathcal{A}$, is achieved using a mixture of two encoding schemes. One takes into account the *global* spatial similarity of *positions* of aligned coordinates between S and T, while the other takes into account the *local* similarity in the *directions* of the coordinates. Appropriately, we call these two models the global and local models of encoding, respectively.

Encoding the aligned coordinates of T using the global model: To compute the code length of stating any $\vec{t_j} \in T$ that is aligned with a $\vec{s_i} \in S$ over the global model, the following procedure is employed. The two structures are first superposed based on their coordinates defined by alignment $\mathcal{A}$. Subsequently, the set of residuals (norm of the error vectors) $\{\delta_{ij}\}$'s between each s_i and t_j is transmitted over a chi-squared distribution (with three degrees of freedom) using the MML approach of [7]. Further, the set of distances $\{r_j\}$'s between successive t_{j-1} and t_j is transmitted using the method described in Section S2 (see Fig. S2).

The information of δ_{ij} and r_j permits the construction of two spheres centered at $\vec{t}_{j-1}$ and at $\vec{s_i}$. These two spheres will intersect in a circle[§], reducing the uncertainty of t_j to lie on the circle of intersection. Given this, the coordinates of $\vec{t_j}$ can be transmitted over a uniform distribution on

[§]ignoring the pathological case when $t_{j-1} = s_i$ or in the best case when $s_i = t_j$

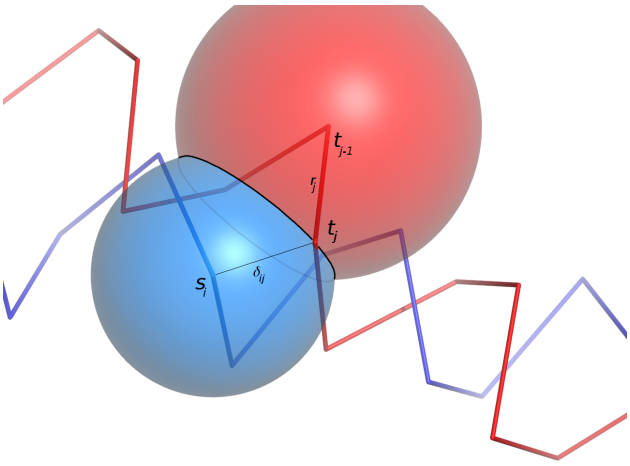

Figure S2: An illustration of the global alignment relationship model to transmit the coordinate of t_j as a deviation with respect to s_i. The receiver is sent δ_{ij} and r_j terms (see main text) which reduces the uncertainty of t_j to be on a circle derived by intersecting spheres centered at s_i and t_{j-1}.

the circle (see Fig. S2).

Encoding the aligned coordinates of T using the local model: The first three coordinates in every aligned block of T are encoded using the global model described above. For the remaining of matches in each mathced block, the following local encoding method is employed.

Recall from Equation 2 in Section S2, that the mixture model is of the form $\mathcal{M} = \sum_{k=1}^{|\mathcal{M}|} w_k \cdot f_k(\vec{\vartheta}_k)$, where w_k is the associated probability of the kth component (which in turn is a Kent directional probability distribution.) If no (extra) information were available about $\vec{t}_j$, then the transmitter would have no choice but to encode it using the null model. However, extra information is available in the form of the correspondence of $\vec{t}_j$ with $\vec{s}_i$. Thus, the probabilities of each component can be updated based on this relationship. Specifically, the null component probabili-

ties (w_k's) can be systematically updated given the expectation provided by the alignment that $\vec{t}_j$ is near $\vec{s}_i$.

Bayes theorem allows updating all w_k's given the knowledge of $\vec{s}_i$. Consider the computation of the posterior probability of the kth component given the direction $\hat{s}_i$, $\Pr(\text{Component}_k|\hat{s}_i)$:

$$\overbrace{\Pr(\text{Component}_k|\hat{s}_i)}^{w'_k \text{ (unnormalized)}} = \frac{\overbrace{\Pr(\text{Component}_k)}^{w_k}\overbrace{\Pr(\hat{s}_i|\text{Component}_k)}^{\text{likelihood}}}{\underbrace{\Pr(\hat{s}_i)}_{\text{constant over } k \text{ terms of the mixture model}}}$$

Note that $\Pr(\text{Component}_k)$ is equal to w_k of the null model, and $\Pr(\hat{s}_i|\text{Component}_k)$ is the likelihood of the direction $\hat{s}_i$ given the component probability density function, $f_k(\hat{s}_i|\vec{\vartheta}_k)$. While these two can be computed trivially using the null mixture model $\mathcal{M}$, $\Pr(\hat{s}_i)$ is unknown. However, each $\Pr(\text{Component}_k|\hat{s}_i)$ needs to be normalised (so that their sum over all components adds to 1) by dividing the numerator by $\sum_{k=1}^{|\mathcal{M}|}\Pr(\text{Component}_k|\hat{s}_i)$, the term disappears. Call these normalised posterior weights w'_k. The old w_k's are now replaced with the new w'_k's in the null mixture model $\mathcal{M}$ and t_j can now be encoded using the updated mixture model.

Time Complexity of computing $I(T|S,\mathcal{A})$ Assuming again that $|S| = m$ and $|T| = n$ and $m \leq n$, the global model of encoding requires the superposition of the aligned coordinates between S and T. The maximum number of aligned coordinates in any $\mathcal{A}$ is bounded by m. The computational effort to find the best superposition of two corresponding vector sets containing m vectors is $O(m)$.

For the global model the estimation of code length for δ_{ij} and r_j, as well as the encoding over the circle of intersection (between spheres) takes constant time. Over all aligned coordinates, the time complexity to estimate the message length over the global model takes worst-case $O(m)$ time as the number of matches is bounded by m. This is the same for the local encoding as each aligned $\vec{t_j} \in T$ requires the computation of the re-weighted probabilities (w'_k's) that take a constant $O(|\mathcal{M}| = 23)$ effort.

On the other hand, the encoding of unaligned coordinates of T (encoded using the null model) is $O(|T|) = O(n)$ in the worst-case. Therefore, the estimation of `I-value` given the coordinates of S and T and any alignment between them, $\mathcal{A}$, takes $O(n)$ time overall.

S4 `MMLigner` search heuristic

`MMLigner`'s search method is carried out in two phases. In the first phase, seed structural alignments are quickly generated using a deterministic approach that efficiently clusters and consistently assembles well-superposable fragment pairs for the two given structures S and T. These seed alignments are refined in the second phase using the `I-value` criterion over a heuristic search.

Phase 1: Generating alignment seeds *Identification of a library of maximal fragment pairs (MFPs).* Given a pair of structures S and T, we first identify all well-superposable, maximally-extended fragment pairs (MFPs) that fit within a threshold of RMSD (= 2.5 Å). This results in a library of MFPs for the given pair of structures being aligned.

Filtering the library of MFPs. The library of MFPs is then filtered to contain only those MFPs that can be jointly superposed with at least two other MFPs in the set. Specifically, every pair of non-overlapping MFPs are jointly superposed under the threshold of RMSD of (= 3 Å). Any MFP that does not superpose within this threshold will be discarded since it shows no evidence of being spatially consistent with other MFPs in the set. This eliminates MFPs that are locally but not globally meaningful.

For each pair that jointly superposes within the threshold of RMSD, we look to extend the superposition using yet another (non-overlapping) MFP. Any pair of MFPs that is not extendible and whose combined length is < 15 residues is again discarded. This further prunes the original library of MFPs.

We note that, if carried out naively, this filtering step can be computationally expensive. However, we are able to achieve an exhaustive and extremely efficient joint superpositions over all pairs of MFPs (and their further extensions to triples) by benefiting from our recent work that has identified the sufficient statistics for orthogonal least-squares superposition of vector sets [8]. To do this, when the original library of MFPs is computed, we store the sufficient statistics of the superposition in each MFP. The RMSD of joint superposition of pairs of MFPs as well as their sufficient statistics can be computed as a constant time update using the sufficient statistics of individual MFPs.[8] Similarly, extensions of pairs of MFPs to triples can also be updated in constant time. As a result, the identification of MFPs and the pruning can be carried out exhaustively in seconds.

Clustering the filtered set of MFPs. The aim of this step is to partition the filtered set of MFPs into groups (or clusters) of related MFPs so that a seed alignment can be explored within each cluster.

Our iterative clustering heuristic proceeds as follows. First, the filtered set of MFPs is sorted in decreasing order of length (in terms of number of residue pairs in each MFP) and the longest MFP in the filtered set is assigned as the initial singleton cluster. The iterative process of clustering then starts by traversing the remaining sorted list of filtered MFPs, starting from the longest. For each MFP in the list, we attempt to add it to any of the existing clusters. Otherwise, a new cluster is created with this MFP as its singleton member. In general, the MFP can be added to a cluster if that MFP jointly superposes with at least 40% of the cluster's existing MFPs.

At the end of this procedure, the clusters whose combined length is less than 18 residues are deleted and the remaining ones are used in the next step to identify seed alignments.

Finding a seed alignment using the clustered MFPs. For the MFPs represented in each cluster, a scoring matrix is computed. Each cell in this matrix represents a score for aligning a specific residue-pair between the two structures being aligned. For each triplet of MFPs in the set that superpose within the threshold of RMSD (3 Å), scores (computed from the combined lengths of MFPs involved and their RMSD) are populated in the weight matrix for the residue-pairs involved in those MFPs. A rough seed alignment using a dynamic programming algorithm described in the pairwise step of `MUSTANG`.[9]

Phase 2: Iterative refinement of the seed alignment using `I-value` criterion Using each seed alignment identified in the previous phase as the starting point, a series of perturbations to these alignments are carried out to identify the final alignment(s) that `MMLigner` reports. The fitness of each perturbed alignment is evaluated using the `I-value` measure based on the amount of compression achieved compared to the null model.

This approach that is similar to the Expectation-Maximization (EM) technique common in statistical learning. Each alignment is efficiently represented as a vector of blocks (where each match block stores the start indices in S and T of that match block, followed by the length of the block). Until the method converges (or reaches a maximum of iterations), the current *best* alignment at each iteration is operated upon by performing the following primitive permutation operations, on each of the blocks.

`ExtendMatchBlock(`$i, l,$ **`direction):`** This search primitive tries to extend the i-th block in an alignment by l residues either on the left side or on the right side (depending on the given `direction`) of the match block. That is, it tries to create new matches out of the deletes and inserts (if any) flanking the specified block in the specified direction. The number of new matches is limited by $\min(\|inserts\|, \|deletes\|)$.

`ShrinkMatchBlock(`$i, l,$ **`direction)`** This primitive tries to shrink the i-th match block by l residues either on the left or right of the match block. That is, it tries to remove macthes by creating new insertions and deletions (*indels*) flanking the current block in the specified direction. The number of new indels is limited by $\|matchblock\|$.

ShiftMatches($i, l,$ direction): This primitive tries to shift l aligned residues from the start of the current block to the end of the previous one, or vice versa. This can only be done when the gap separating two successive match blocks is monotonous (that is, the gap is either entirely inserts, or entirely deletes). Upon this perturbation, the gap length between the successive blocks remains unchanged, but the size the specified match block either increases or decreases by l. The shift size, in turn, is limited by the size of the previous or current block, depending on the direction of shift.

SlideMatchBlock($i, l,$ direction): This primitive tries to change the residue-residue correspondences of a block by moving (or *sliding*) l residues in S left or right relative to T. Note that the number of correspondences in a block remains constant. The size of the shift is limited by the size of the block being operated upon plus the number of gaps available in the direction of the shift.

RealignClosest(i): This primitive perturbs the current alignment by realigning the (subsets of) residues in S and T around a specified match block. Before this perturbation is explored, T is superposed on S based on the current (full) state of the alignment. For any i-th match block, consider the subsets of residues in S and (transformed) T covering the specified match block, and its left and right flanking gap region. A Euclidean distance matrix is computed for each residue-residue pairs within this subsets. The residue-pairs within these subsets that exceed the maximum match distance observed with the current match block are ignored from the realignment. For the permissible residue pairs, a Needleman-Wunsch[10] dynamic programming algorithm is used to realign the residues around this specified region.

DeleteMatchBlock(i): This primitive perturbs the current alignment by deleting the entire match

Algorithm 1: `MMLigner`'s Expectation-Maximization search heuristic

input : An initial (seed) alignment, $\mathcal{A}$, a reference structure, S, and a target structure, T
output: A `MMLigner` alignment, $\mathcal{A}^{\text{best}}$

$nIters \leftarrow 1$; $\mathcal{A}^{\text{best}} \leftarrow \mathcal{A}$;
while $nIters \leq 50$ **do**
  $best\_ivalur \leftarrow I(\mathcal{A}, \langle S, T \rangle)$;
  $perturbed[\,] \leftarrow [\,]$; $perturbed\_ivalue[\,] \leftarrow [\,]$;
  for $i \leftarrow 1$ **to** $\|match\_blocks\|$ **do**
    for $l \leftarrow 1$ **to** 3 **do**
      for $perturbType$ = {`ExtendMatchBlock`, `shrinkMatchBlock`, `shiftMatches` `slideMatchBlock`} **do**
        $\mathcal{A}_{tmp} \leftarrow$ `perturbType`$(i, l,$ left$)$;
        $perturbed\_ivalue$.append$(I(\mathcal{A}_{tmp}, \langle S, T \rangle))$; $perturbed$.append$(\mathcal{A}_{tmp})$;
        $\mathcal{A}_{tmp} \leftarrow$ `perturbType`$(i, l,$ right$)$;
        $perturbed\_ivalue$.append$(I(\mathcal{A}_{tmp}, \langle S, T \rangle))$; $perturbed$.append$(\mathcal{A}_{tmp})$;
    $\mathcal{A}_{tmp} \leftarrow$ `RealignClosest`(i);
    $perturbed\_ivalue$.append$(I(\mathcal{A}_{tmp}, \langle S, T \rangle))$; $perturbed$.append$(\mathcal{A}_{tmp})$;
    $\mathcal{A}_{tmp} \leftarrow$ `DeleteMatchBlock`(i);
    $perturbed\_ivalue$.append$(I(\mathcal{A}_{tmp}, \langle S, T \rangle))$; $perturbed$.append$(\mathcal{A}_{tmp})$;
  $\mathcal{A}_{tmp} \leftarrow$ best$(perturbed_ivalue[\,], perturbed[\,])$;
  if $perturbed\_ivalue[\mathcal{A}_{tmp}] < best\_ivalue$ **then**
    $\mathcal{A}^{\text{best}} \leftarrow \mathcal{A}_{tmp}$;
    $best\_ivalue \leftarrow perturbed\_ivalue[\mathcal{A}_{tmp}]$;
    $nIters \leftarrow nIters + 1$;
  else
    break;
return $\mathcal{A}^{\text{best}}$;

block and replacing it with appropriate inserts and deletes.

Search using the above primitives: Any given seed alignment is refined using the above primitives over an Expectation-Maximization algorithm defined in Algorithm **??**. At each iteration, the algorithm attempts to exhaustively perturb each match block using the above perturbation primitives and greedily chooses the best perturbation using the `I-value` criterion. This continues until either the alignment converges (it always will but might take long) or reaches the maximum number of iterations. This behaviour is intended to ensure the algorithm explores the local

space thoroughly around the (reasonably good) starting point provided by the seed alignment of `MMLigner`.

S5 Selecting the SCOP domain dataset used for software benchmarking

In the results section of the main text, we benchmark various alignment program on a dataset containing 2500 structural domain pairs selected randomly from SCOP domain database. [11,12] No two domains in our dataset share more than 40% sequence identity.

The dataset is identified using the following procedure. ASTRAL SCOP 40[12] domains are used and separated out into buckets depending on their sizes (number of residues). Two domains in the same bucket differ no more than 50 residues in their lengths. The SCOP hierarchy for all the domains within each bucket is recorded. A pivot domain is randomly chosen from the entire ASTRAL SCOP 40 collection. Assume that this pivot domain falls with the i-th bucket. Using the SCOP hierarchy in this bucket, we select:

- one domain (randomly) that belongs to the same SCOP Family as the pivot
- one domain (randomly) that belongs to the same SCOP Superfamily as the pivot, but not the Family
- one domain (randomly) that belongs to the same SCOP Fold the pivot, but not the Family or Superfamily.
- one domain (randomly) that belongs to the same SCOP Class the pivot, but not the Family,

Superfamily or Fold.

- one domain (randomly) that belongs to a different class.

This selection process is repeated until we have 500 distinct pivots and their respective five domains. This results in 2500 distinct structural domain pairs. (Dataset is available from `http://lcb.infotech.monash.edu.au/mmligner`.)

S6 Supplementary figure supporting Fig 2. in the main text

`MMLigner` can be run with an option to filter out alignments for structural pairs composed of matches involving standard supersecondary structures, which nevertheless yield positive compression over their respective. These are reported by `MMLigner` in its default run (see Fig 2. in the main text). Fig. S3 shows the runs with the filter option turned on, over the same dataset used in the main text.

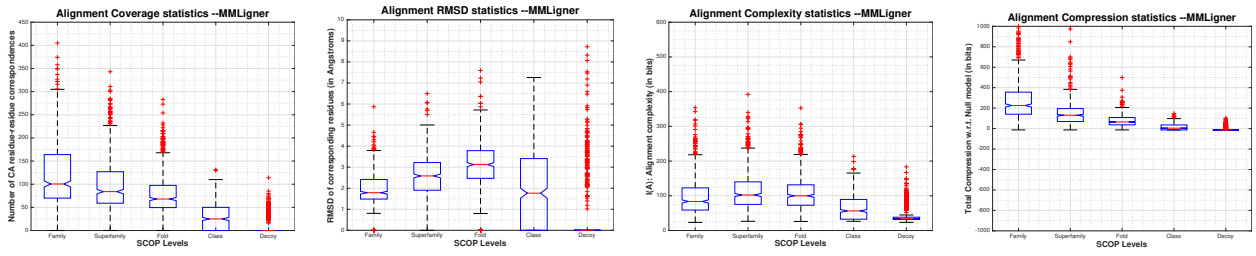

Figure S3: Supplementary figure to Fig 2. in the main text. Notched Box-Whisker plots displaying the distribution of values from various criteria across $500 \times 5 = 2500$ alignments generated by `MMLigner` with a filter option turned on (default is off; see results in Fig 2. in the main text). The filter option ignores any alignment(s) returned by `MMLigner` that, although yielding positive compression, is composed of matches involving common supersecondary structures.

S7 Benchmarking the runtimes of alignment programs

Using the SCOP dataset containing 2500 structural pairs (see Section S6) we compare the runtime performance of `MMLigner`and 5 other widely used structural alignment programs (see Section 3 in the main text). The runtime statistics (measured in seconds of CPU time) are summarised in the table below,.

Table S1: Average runtime statistics (measured in seconds of CPU time) of the structural alignment programs `MMLigner`, `CE`, `DALI`, `TM-Align`, `FATCAT`, and `LGA`

Alignment Program	Family	Superfamily	Fold	Class	Decoy	Average	Statistic
MMLigner	8.74	14.93	12.23	7.85	4.37	8.88	3rd quartile
	11.06	17.59	17.57	8.74	4.83	11.95	Mean
	1.89	3.21	3.60	2.67	1.76	2.51	Median
	0.75	1.12	1.32	1.02	0.66	0.92	1st quartile
DALI	2.66	2.50	2.64	2.65	1.85	2.51	3rd quartile
	2.60	2.68	2.72	2.21	1.72	2.39	Mean
	0.75	0.74	0.77	0.77	0.58	0.71	Median
	0.38	0.40	0.38	0.39	0.34	0.38	1st quartile
TM-Align	0.15	0.16	0.16	0.16	0.16	0.16	3rd quartile
	0.15	0.16	0.16	0.17	0.18	0.16	Mean
	0.07	0.08	0.08	0.07	0.07	0.07	Median
	0.04	0.04	0.04	0.04	0.03	0.04	1st quartile
LGA	3.88	2.81	1.44	1.39	1.23	2.20	3rd quartile
	6.03	5.08	4.45	4.51	4.24	4.86	Mean
	0.66	0.26	0.05	0.09	0.08	0.18	Median
	0.00	0.00	0.00	0.00	0.00	0.00	1st quartile
CE	2.47	2.58	2.66	2.74	2.74	2.66	3rd quartile
	2.27	2.48	2.53	2.68	2.67	2.52	Mean
	1.82	1.97	2.01	2.10	2.07	1.99	Median
	1.56	1.61	1.64	1.71	1.70	1.64	1st quartile
FATCAT	5.23	5.08	5.12	4.76	4.19	4.90	3rd quartile
	4.71	4.62	4.54	4.46	4.03	4.47	Mean
	3.42	3.30	3.32	3.36	3.07	3.27	Median
	2.70	2.69	2.66	2.66	2.52	2.64	1st quartile

References

1. Shannon, C. E. A mathematical theory of communication. *Bell Syst. Technical Jrnl.* **27**, 379–423 (1948).

2. Kasarapu, P. & Allison, L. Minimum message length estimation of mixtures of multivariate gaussian and von mises-fisher distributions. *Machine Learning* 1–46 (2015). URL `http://dx.doi.org/10.1007/s10994-015-5493-0`.

3. Kasarapu, P. Modelling of directional data using kent distributions. *arXiv preprint arXiv:1506.08105* (2015).

4. Collier, J. H., Allison, L., Lesk, A. M., de la Banda, M. G. & Konagurthu, A. S. A new statistical framework to assess structural alignment quality using information compression. *Bioinformatics* **30**, i512–i518 (2014).

5. Allison, L., Wallace, C. & Yee, C. Finite-state models in the alignment of macromolecules. *J. Mol. Evol.* **35**, 77–89 (1992).

6. Wallace, C. S. & Boulton, D. M. The information content of a multistate distribution. *J. of Theor. Biol.* **23**, 269–278 (1969).

7. Wallace, C. S. *Statistical and Inductive Inference using Minimum Message Length*. Information Science and Statistics (SpringerVerlag, 2005).

8. Konagurthu, A. S., Kasarapu, P., Allison, L., Collier, J. H. & Lesk, A. On sufficient statistics of least-squares superposition of vector sets. In *RECOMB*, vol. LNCS/LNBI 8394, 144–159 (2014).

9. Konagurthu, A. S., Whisstock, J. C., Stuckey, P. J. & Lesk, A. M. MUSTANG: a multiple structural alignment algorithm. *Proteins* **64**, 559–574 (2006).

10. Needleman, S. B. & Wunsch, C. D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology* **48**, 443–453 (1970).

11. Lo Conte, L. *et al.* SCOP: a structural classification of proteins database. *Nucleic Acids Res.* **28**, 257–259 (2000).

12. Chandonia, J. M. *et al.* The astral compendium in 2004. *Nucleic Acids Res.* **32** (2004).

Acknowledgements Authors acknowledge Parthan Kasarapu for assistance on using his mixture models.

Funding This research is funded by Australian Research Council (ARC) Discovery Project grant (DP150100894). JHC is supported by Australian Postgraduate Award (APA) and NICTA PhD scholarship. NICTA is funded by the Australian Government through the Department of Communications and the ARC through the ICT Centre of Excellence Program.

Competing Interests The authors declare that they have no competing financial interests.

Website `MMLigner` software and all material related to this work is found at `http://lcb.infotech.monash.edu.au/mmligner`.

Correspondence Correspondence and requests for materials should be addressed to Arun Konagurthu. (email: arun.konagurthu@monash.edu).