Probabilistic Programming for Protein Super-positioning

William Paul Bullock (Candidate)

Thomas Wim Hamelryck (Supervisor)

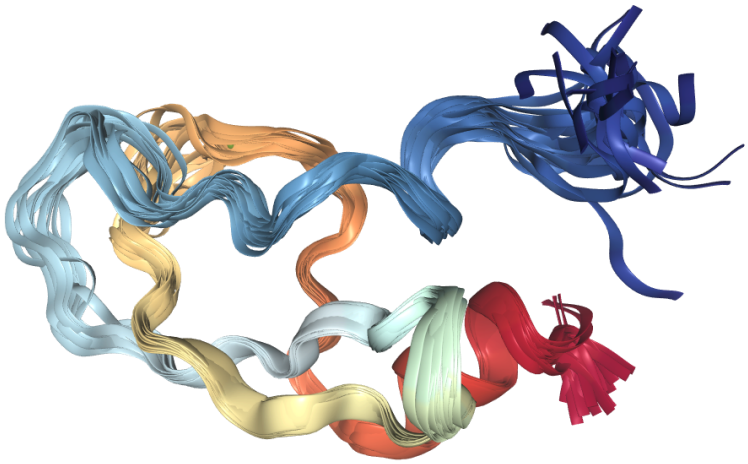


Image: NMR Ensemble of all superimposed 1IRH protein structures.

Abstract

Protein structural alignment is the comparison of the 3D conformations of two or more protein structures and is commonly used to infer established potential homologies between the structures.

In this investigation, an automated, Bayesian probabilistic method of protein super-positioning short polypeptide structures has been developed. Showing proof-of-concept; the model can be used to successfully infer a mean structure (the optimal superposition) for two short simulated polypeptide structures; however various issues are introduced when scaling the model to longer simulated polypeptides or full-length real protein structures.

Further fine-tuning of the method and its parameters will likely allow it's future use and application to multiple, full length, protein sequences.

Introduction

Similarities and differences between the 3D conformation of two or more protein structures can be used to infer established potential homologies between the structures.

This is commonly known as structural alignment. A chief technique in this discipline is determining a structural superposition, an optimal alignment of the different structures over the same co-ordinates.

This is traditionally calculated using a simple least squares fitting algorithm, in which the optimal rotations and translations are found by minimizing the sum of the squared distances among all structures in the superposition thus calculating a 'Mean structure' which is a compromise between all structures being compared. (McLachlan and IUCr, 1982)

More recently, there has been increasing interest in the development of Bayesian or Maximum-Likelihood models.

This investigation builds upon techniques developed by (Theobald and Steindel, 2012), Where they present an expectation maximization(EM) solution to a Gaussian statistical approach of the super-positioning problem.

This investigation adopts a similarly Gaussian based method though it moves away from an expectation-maximisation(EM) maximum-likelihood (ML) approach with a multivariate log-likelihood function.

Instead, we investigate the use of a Bayesian inference approach utilizing with a multivariate normal log-posterior probability density function, numerically optimised using maximum a priori methods.

The chief benefit of this approach is the ability to establish priors, which inject into the estimation pre-existing connotations about the nature of the parameter being optimised, which when well selected provide a starting-point that can be more rapidly and accurately optimised into the an estimate true parameter. Potentially providing quicker and more accurate super-positioning of any multitude of similar protein structures.

This project is focused on structural biology (proteins) and was intended as a learning exercise focused on developing mathematics and programming skills; particularly within Bayesian statistics and python's pyMC3 package.

Materials

Software:

Programming elements were completed in Python 3.6, chiefly using the package PyMC3 for the probabilistic programming functionalities which is built upon Theano architecture. (Salvatier, Wiecki and Fonnesbeck, 2015)

Other packages of use were:

- Biopython
- NumPy
- Matplotlib
- RMSD (Coutsias, Seok and Dill, 2004)

Data:

The initial testing stages of this project utilized 'bootstrapped data' randomly generated via NumPy functions.

Later protein structures (1ENH & 1IRH) from RCSB The Protein Data Bank (Berman *et al.*, 2000), were used to test the efficacy of the method with real data.

Methods

The Gaussian perturbation model

Our statistical model assumes that X_i protein structure forms will, due to the central limit theorem (CLT) tend towards a Gaussian probability density. Under this assumption a Gaussan perturbation model can be adopted, where by any observed structure X_i is a descendant of some unknown mean structure which has undergone an unknown arbitrary translation, rotation and the addition of statistical noise, such that:

$$X_i = (\mu \cdot R_i) + T_i + \varepsilon$$

Where:

X_i is any and all possible protein structure in the distribution.

μ is the mean structure of all X_i forms.

R_i is a rotation in 3D space

T_i is a translation 3D space

ε is statistical noise

This model allows the representation of each protein structure Xi as N-by-3 Matrix, where N is the number of (x,y,z) co-ordinate vectors in he matrix, each vector specifying the 3D position of the central carbon atom (henceforth Cα) in each residue of a polypeptide chain.

By necessity then, R_i is an orthogonal 3-by-3 rotation matrix of determinant = 1, T_i is a 1-by-3 column vector, and ε is a N-by-3 matrix of randomly sampled Gaussian error of mean 0 and $\sigma = 0.1$.

Under this model, the unknown mean structure can be inferred from any observed structure, if the true values of the R, T parameters are known, and interfering statistical noise is accounted for. Thus the unknown optimal superposition (μ) of X_i structures can be determined by optimizing the aforementioned parameters for all X_i structures in a distribution. In this investigation this was achieved via a Bayesian inference based implementation in pyMC3.

Bayesian Inference Overview

Bayesian inference is a method of statistical inference which updates the probability of a hypothetical event as more evidence becomes possible.

Bayesian inference methodology is based around the re-iteration of Bayes' theorem, until convergence. The formula can be explained generally, as below:

$$P(H \mid e) = \frac{P(e \mid H)\, P(H)}{P(e)}$$

Where:

H is indicative of our hypothesis

e is the evidence for the hypothesis

P(H|e) is the posterior distribution, and is the probability of given the observed evidence, this is typically what we are trying to infer.

P(H) is the prior distribution, the probability of H being true before evidence is taken into account.

P(e|H) is the likelihood, the probability of the observed evidence given the hypothesis is true.

P(e) is marginal probability, the probability of the evidence regardless of the state of the hypothesis. (Often false positives + true positives).

The re-iteration of this model allows for evidence to 'update' knowledge about the prior, (the probability of H being true) which can then be used to generate new evidence which then can once again be used to update H, until convergence.

The process can be detailed as:

1. One defines the prior distribution p(H) as appropriate for the model.
2. Data is gathered regarding the probability of an event P(e)
3. this data is used to update the prior distribution using Bayes' theorem

   p(H|e) = p(e|H) p(H) / p(e)

   to infer a posterior distribution.
4. The process can be repeated if desired, taking the posterior distribution obtained to be the new prior.

5. this is repeated until an ultimate posterior distribution is reached, which can no longer be updated (convergence)

This model is utilized under an iterative optimization algorithm such as the expectation-maximisation (EM) method; in which the E step creates a function for the expectation for the log-liklihood using current estimates of of the parameters, then an M step which computes new parameters which maximise this expectation. It should be noted however, that in this investigation we actually maximise the log-posterior, this is explained later.

In the context of this investigation, the observed X_i protein structures are the 'evidence', and our priors are over several different parameters; translation, rotation, statistical noise, and mean structure. Below I discuss how these are each implemented in each of the models we created.

Our general superposition posterior model

The full joint probability density function(PDF) of our Gaussian perturbation model is well represented by a matrix normal log-posterior function, though for the sake of mathematical simplicity, we 'flattened' all input matrices into one dimensional vectors and utilized the multivariate normal distribution log-posterior function.

A matrix normal distribution is a generalization of the multivariate normal distribution, to be applicable when X is a matrix of values; this in turn is itself a generalization of the univariate normal distribution in instances when X is a (typically column) vector. This choice of distribution suits our model, where X_i is a perturbation of an unknown mean structure within a normal distribution and is a matrix of values.

The probability of the two structures we wish to superimpose, can be shown as:

$$P(X_1 | (\mu \cdot R_1) + T_1, \Sigma UV)$$

$$P(X_2 | (\mu \cdot R_2) + T_2, \Sigma UV)$$

Where ΣUV represents the variance-covariance matrix resultant of the Kronecker product of the U & V covariance matrices of the two structures, where U contains the among-row variance (covariance between atoms), and V contains the among-column variance (variance of each atom). The formation of this construct is detailed below.

Priors used

Prior over the Translations

Translations have the simplest prior, being a 1-by-3 column vector consisting of variables each taken randomly from univariate normal probability distribution of $\mu = 0$ and $\sigma = 1$

A simple normal distribution was chosen here because we waned to simulate random movement in any direction of the center-of-mass of the protein away from the origin in euclidean space. A random translation vector was created for both protein structures and added to the X_1 and X_2 after rotation.

Prior over the Rotation(s)

For each rotation matrix utilized by the model:

Three values were pulled from a 3D uniform distribution(lower bound 0, upper 1) and transformed into a unit quaternion, which are complex numbers with a corresponding 3D rotation. This was achieved via the method in (Perez-Sala *et al.*, 2013)

Each quaternion was then able to be 'unpacked' into the corresponding rotation matrix via the method specified in: (Mred, 1998)

A uniform prior was used as it is uninformative, and rotations are equally likely in any and all directions.

Prior over the Covariance Matrix

The construction of the prior for the covariance matrix of a multivariate normal distribution was very similar to as recommended in the PyMC3 documentation by (The PyMC Development Team, 2017).

It is worth noting that the use of a wishart distribution is more orthodox for the formulation of this prior. But has reportedly been found to be poorly suited to Bayesian computational methods. (Various, 2014)

As such, the Lewdanowski-Kurowicka-Joe (LKJ) distribution is used in this project instead. (Lewandowski, Kurowicka and Joe, 2009)

The LKJ distribution provides a prior on the correlation matrix, which can then be combined with priors on the standard deviation of each component to induce a prior on the covariance matrix.

In this model the standard deviations have half-Cauchy distribution priors, ranging from 0 to 0.5. The half-Cauchy distribution was chosen because it can only yield positive values, (standard deviation cannot be negative), and because of its computational stability.

The Cholesky decomposition of the variance-covariance matrix is yielded from this operation, (for computational efficiency), this is then expanded from its lower-triangular state and dot-multiplied with its own transpose to yield the full first variance-covariance matrix, U.

The second variance-covariance matrix, V, is the dot product of a random positive scalar, (from a half-normal distribution) and an identity matrix.

The Kronecker product of U & V forms the final variance-covariance matrix. The combination of the two allows for the model to optimize the covariance of the atoms as a parameter (U), as well as accounting for statistical noise (V).

Prior over the Mean structure

The prior over the unknown mean structure is also a from a uniform distribution.

A number of vectors equal to that in the observed structures are created from points taken uniformly within a 3d sphere, using the method outlined by (Simon Cory, 2017), an additional constraint of a distance of 3.8Å between each value is added, (initializes as a diameter line through the centre, demonstrated in *fig1.*), and these vectors are vertically stacked into a matrix, resembling the dimensions of matrices of the input structures.

This is intended to initialize the prior as a neutral chain which can be optimized to loosely mimic the 3D conformation of the trace of Cα atoms comprising the center of the residues of a polypeptide chain.

This is the prior that we are chiefly interested in optimizing, by estimating this prior correctly for two or more known structures, we will have found the mean structure between them, which in our statistical model is synonymous with the optimal super-position.

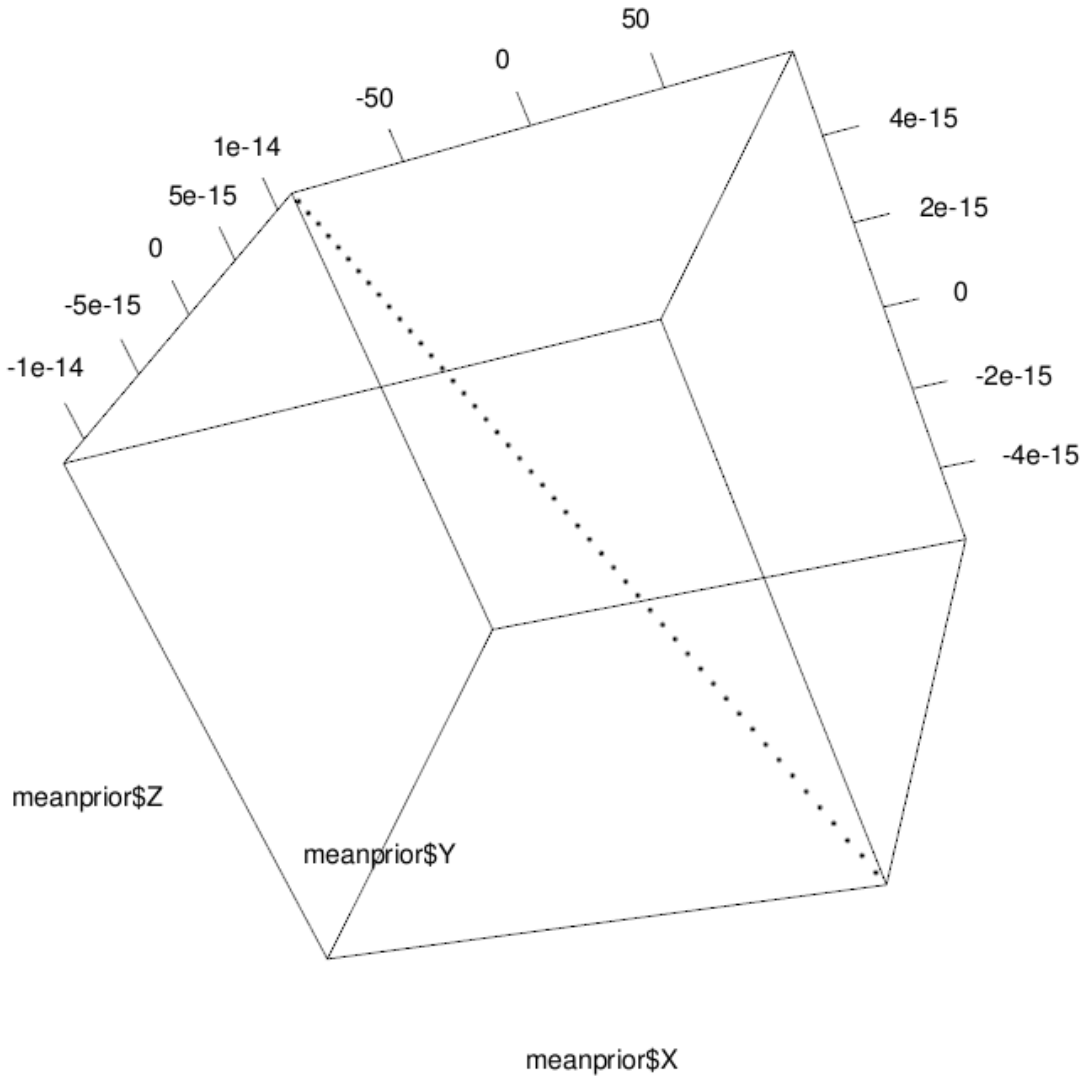


Figure 1. Demonstration of the initialization of the mean prior, with N uniformly sampled point from a 3D Sphere, each 3.8Å apart.

Overview of how pyMC3 operates

When a model is fully established and is then called by the function 'find_MAP', pyMC3 (by default) uses the optimization algorithm to find the maximum(mode) of the log-posterior (henceforth MAP), for our full joint PDF (the log space is used as it is strictly increasing and computationally easier to determine a maximum) with respect to each of the unknown variables' and their various priors.

Because we are operating in vector calculus space, all partial differential information of our multivariate log-posterior function in use are represented in the Jacobian matrix of that function, this is then used to find all the second order partial derivatives of each variable, approximating the Hessian matrix of the function.

PyMC3 can then use the Hessian to compute the local gradient of the function, at some point, with respect to each specific variable, and use this knowledge to alter that variable so it 'climbs' the curve, and find the find the MAP(mode) for the prior parameter associated with that variable, P(prior|variable).

This new estimation of the prior is then substituted with the initial prior, updating the function, new observed data is drawn, and the process is re-iterated again.

Often we have two log-postieror functions with multiple variables to optimise. If some of these variables are shared (chiefly covariance matrix and mean structure) their parameters will be optimised with respect to the chain rule. That is, the derivative will be found, following the above process for the point-wise product of the two functions.

The process cycles until 10.000 iterations have taken place, or all the parameters have reached convergence, where the posterior remains equal to the prior after a MAP and cannot be improved upon, for all variables' priors.

This process will have optimised the rotations, translations, covariance matrix finally mean structure(superposition) for the two protein structures input into the model.

Note: Root-mean-square deviation (RMSD)

In one of our models our estimate of the super-position is compared with the RMSD solution, this was calculated using the Biopython.PDB modules' inbuilt method.

At other times for the RMSD calculation used as a measure of closeness of our estimated structure to a true struture was clacualted using the Kabsch algorithm via the python package RMSD.

The RMSD is simply the measure of the average distance between the Cα atoms of aligned structures, typically one structure is rotated and translated onto another in a way which minimises the RMSD value.

The Models

(raw code for each has been submitted as additional documentation)

The first model: with a fixed mean, no translations, and two unknown rotations

This model was a simplified form implemented to test the viability of our Gaussian perturbation model.

This model omits the priors over the unknown mean structure and the translation. Focusing instead on ascertaining the feasibility of optimizing the rotation and covariance priors.

This model also uses randomly bootstrapped data in the likeness of a protein, rather than actual polypeptide chains.

A random mean structure of size N-by-3 (default N=10), the comprising values of which are drawn from a normal distribution of $\mu = 0$ and $\sigma = 1$, and scaled y a factor of 100 (so as to minimise the effect of noise, added later).

Two random rotation matrices, R_1 & R_2 are formed. Each of these is formed from six 2D random rotation vectors uniformly sampled from a 2-dimensional sphere, which are then concatenated into a 3x3 orthogonal rotation matrix. (Perez-Sala *et al.*, 2013)

Two input structures X_1, X_2, which are perturbations of the mean, are formed by finding the dot product of the random mean structure and their respective rotation matrix.

Two N-by-3 noise matrices are created from a normal distribution of $\mu = 0$ and $\sigma = 1$ and one is added to each of the input structures X_1, X_2. Such that:

$$X_1 = (\mu \cdot R_1) + \varepsilon_1$$

And:

$$X_2 = (\mu \cdot R_2) + \varepsilon_2$$

The full joint PDF of this simplified model is adapted from our generalization earlier can be expressed as.

$$P(X_i|(\mu \cdot R_i), \Sigma UV)$$

The mean structure and the two derived bootstrapped protein structures are input into the model as known true values for variables. The model is then called (as in the aforementioned section) to optimize the parameters for the covariance matrix and two rotations.

The second model: with an unknown mean, two unknown translations, and one unknown rotation

This model utilizes a single structure of the the 1ENH protein structure from the RCSB PDB. (Clarke *et al.*, 1994) parsed via Biopython.

This model is an extension of the first to now have an unknown mean, and a translation, both of which are variables with parameters to optimize.

One form of the protein structure of 1ENH is used as the true mean structure (μ), truncated to the first N residues.

The first protein structure X_1 is initialized to be the same as the mean, and has no rotation applied.

One random rotation is created (in the same manner as the previous model), and applied to a second separate instances of the same mean structure, to generate X_2.

Having two rotations results in having too many possible conformations of the estimated mean structure, meaning the model has difficulty reaching convergence.

It has been found to be much better to initialize with the assumption that one of the structures is close to the true mean, then perform a Gaussian walk to alter the estimation of the mean structure into an optimal conformation existing in-between the two, through Bayesian updates.

(This is reminiscent of classical RMSD methods where one model is fixed and the other is rotated to superimpose it.)

These two structures(X1, X2) and the μ are all translated to the origin.

A random translation and random noise are added to each of X_1 & X_2 so that:

$$X_1 = \mu + T_1 + \varepsilon_1$$

$$X_2 = (\mu \cdot R_2) + T_2 + \varepsilon_2$$

This prompts the utilization of two slightly different derivatives of our Joint PDF:

$$P(X_1|\mu + T_1, \Sigma UV)$$

$$P(X_2|(\mu \cdot R_2) + T_2, \Sigma UV)$$

PyMC3's MAP method is then called, and all the variables of both functions are optimized.

Afterwards, the optimal configuration determined for the mean-structure was compared with the true mean the input structures were derived from, visually and by RMSD values.

The third model: two models of the same protein structure

This model utilizes a two different structures of the the 1IRH protein (models 1 & 2) from the RCSB PDB. (Mine *et al.*, 2002) parsed via Biopython.

There is no known mean-structure this time, the two structures are parsed and translated to the origin, then input directly into the model at sull size (or truncated to arbitrary length N residues) and the MAP optimizes the parameters in the same manner as the second of our models, thus giving the estimation of the mean-structure as one of them.

A RMSD solution is also computed using Biopython.PDB, this rotates the second protein structure(X_2) so that it is super-positioned over the first structure (X_1).

The RMSD solution of X_2 and the estimated-mean strucutre from our pyMC3 model are compared visually.

Results

Model 1:

```
True R1
[[-0.93146599  0.34408968  0.11820914]
 [-0.2128628  -0.25190827 -0.94405066]
 [-0.29506023 -0.90451341  0.30788788]]
MAP r1
[[-0.93088866  0.34120832  0.13047297]
 [-0.22456534 -0.25280071 -0.94109628]
 [-0.28812622 -0.90535556  0.31195287]]
The determinant is...
1.0
and orthogonality is...
True
```

```
True R2
[[ 0.1545133  -0.97572814  0.1551781 ]
 [-0.94389429 -0.19218473 -0.26856767]
 [ 0.29187189 -0.10497445 -0.95067932]]
MAP r2
[[ 0.15221597 -0.97566456  0.15782576]
 [-0.94458909 -0.19059905 -0.26725167]
 [ 0.29082942 -0.10840052 -0.95061432]]
The determinant is...
1.0
and orthogonality is...
True
```

Figure 2. Demonstration of MAP optimized rotations generated by the first model. Output taken directly from Python console. Left hand side shows the true rotation on X_1 at the top, and below that the MAP estimate of the rotation on X_1. Adjacent, the same is shown for X_2.

Our first model,with the fixed mean and no translations was able to consistently and with $<$ ~0.01 variance, successfully predict the rotation priors for the joint full PDF. For N = 10 simulated residues in a bootstrapped polypeptide chain. This can be seen in figure 2.

Model 2:

```
 Comparing rotations
Real r2
[[ 0.09770129  0.23135399  0.96795133]
 [ 0.68018612  0.69447046 -0.23464359]
 [-0.72649934  0.68131205 -0.08951311]]
Estimated r2
[[-0.09350827  0.12768885  0.98739646]
 [ 0.70882947  0.70497017 -0.02403838]
 [-0.69915448  0.69764792 -0.15643016]]
```

```
 Comparing translations
Real t1
[[ 0.58640743  0.16141172  2.07616836]]
Estimated t1
[ 0.46544284 -0.0794687   2.10598495]
Real t2
[[-0.07535146  1.07558708  0.40005284]]
estimated t2
[-0.00735371  1.11250184  0.38159316]
```

Figure 3. Demonstration of MAP optimized rotations and translations as generated by the second model. Output taken directly from Python console.

The second model, with an unknown mean and added translations, was similarly able to make good estimations of the rotations consistently, while N=10 or lower.

These estimations were noticeably less accurate than in the first model though.

The estimation of the translations was inadequate, but non-random and seemed close to correct.

Despite the inaccuracy of the translations, the approximation of the mean-structure was often very close to the true structure, as can be seen from figure 4, and table 1.

The model lacked robustness, and as N (number of residues in the model (polypeptide chain length)) increased the estimation of the mean structure became less accurate, less consistent, and the mean RMSD between the two increased. This can be seen in figures 5, 6 & 7 as well as table 1.

Number of residues (N)	Mean RMSD between Tμ & Eμ
10	0.8371804766
20	2.3728946792
30	7.9109600912

Table 1. Mean RMSD (of 10 repeats) between the true mean-structure and estimated mean-structure, in model 2, for E1NH polypeptide chains truncated to length N.

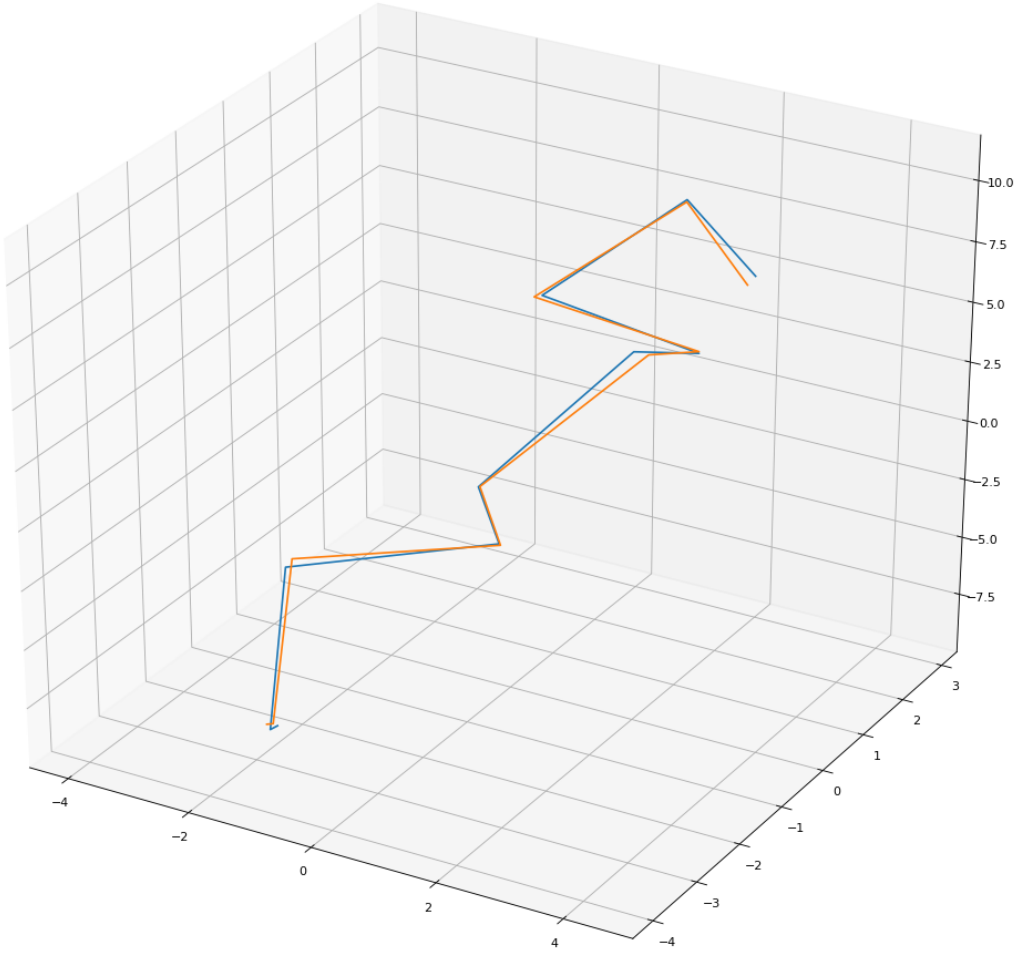


Figure 4. 3D conformational structures of the true mean structure (blue) and the estimated mean-structure(orange) in euclidean space, for N= 10, with arbitrary distances on the X,Y,Z axis.

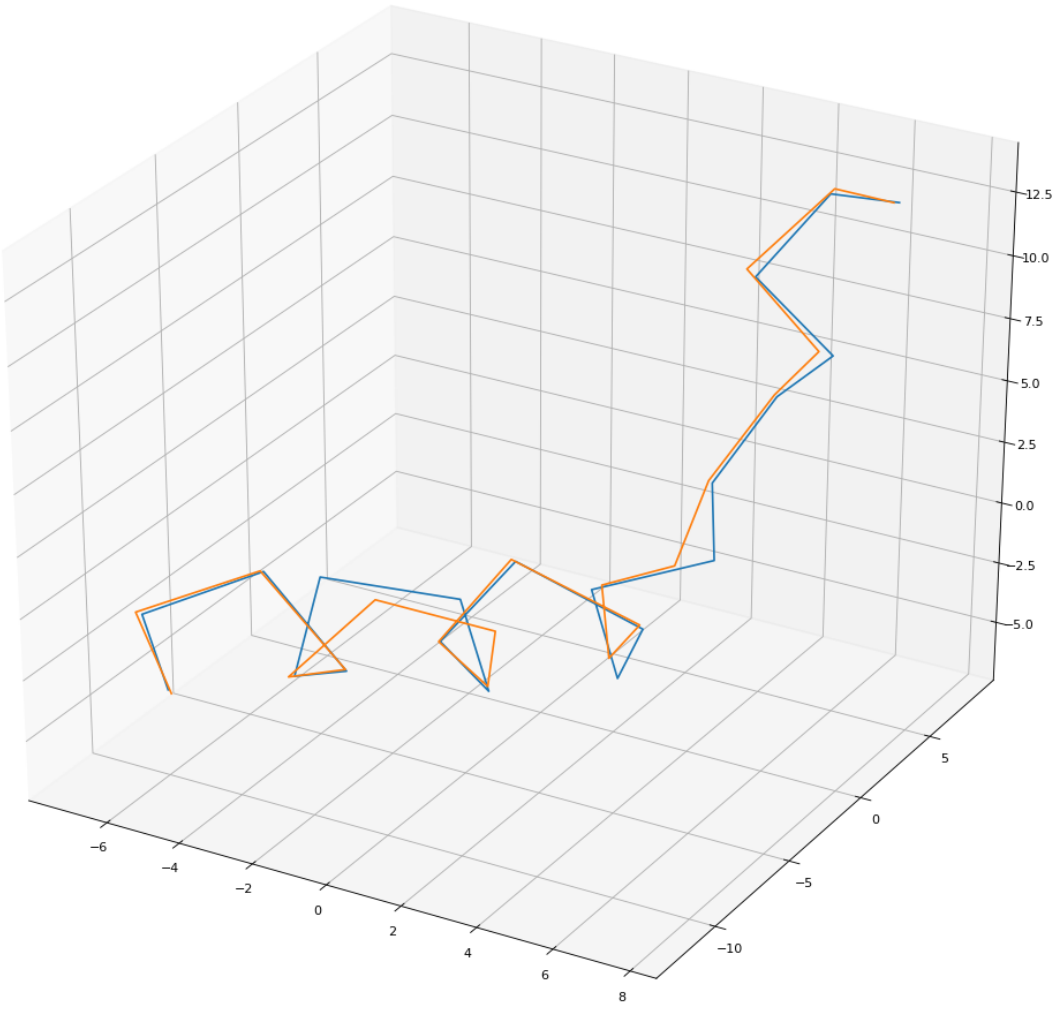
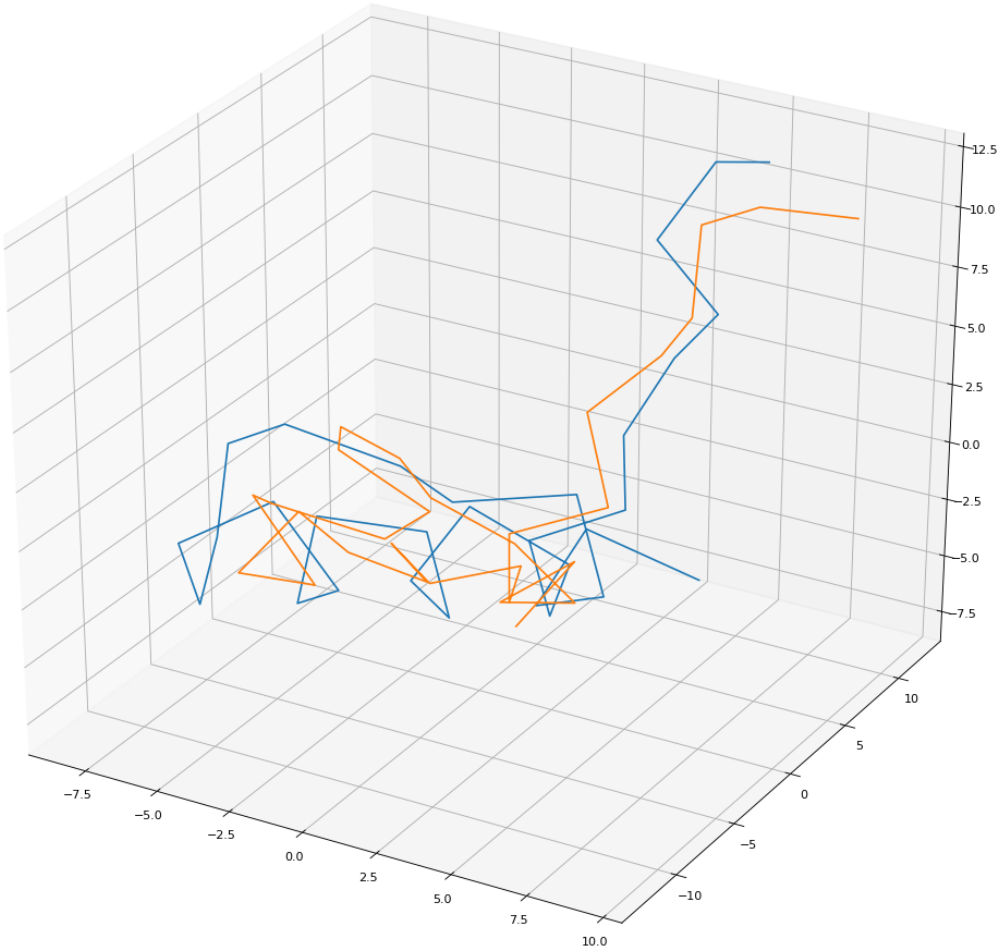
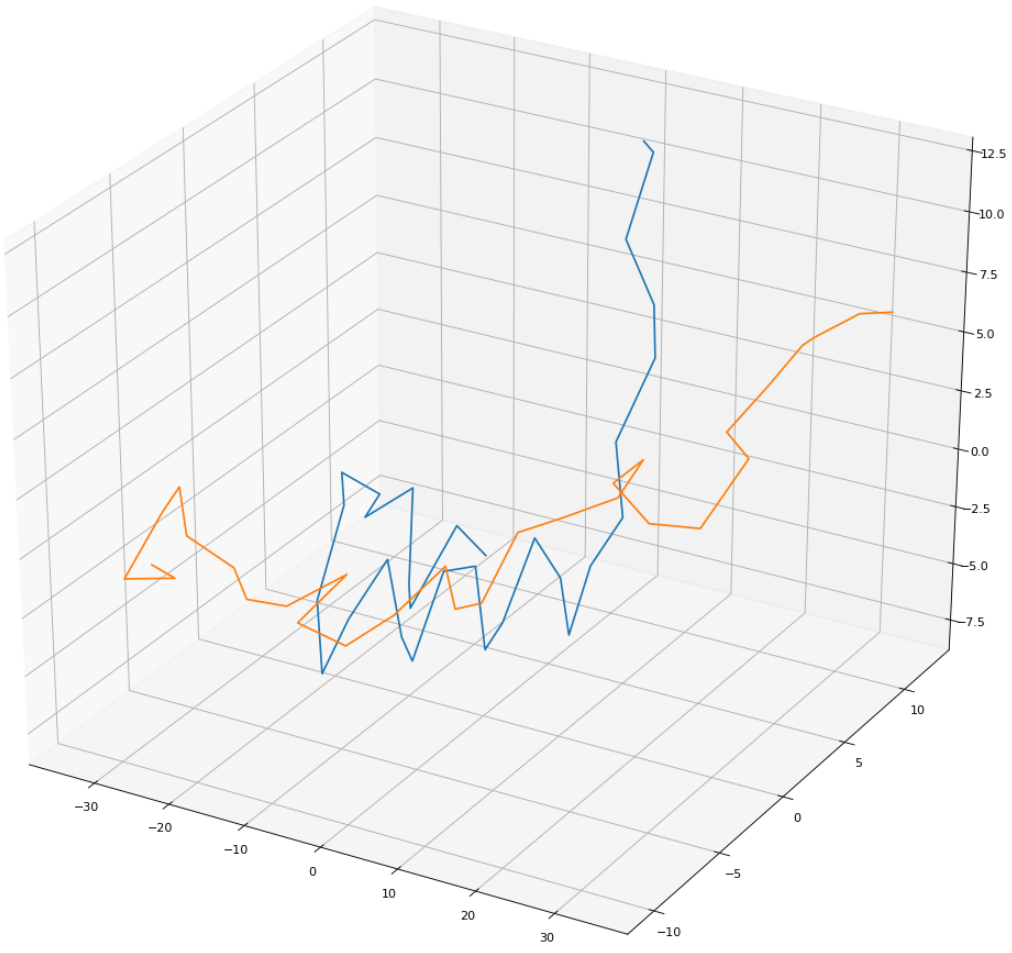


Figure 5. 3D conformational structures of the true mean structure (blue) and the estimated mean-structure(orange) in euclidean space, for N= 20, with arbitrary distances on the X,Y,Z axis.





Figured 6 & 7. 3D conformational structures of the true mean structure (blue) and the estimated mean-structure(orange) in euclidean space, for N= 30, with arbitrary distances on the X,Y,Z axis. From two separate repeats of the model.

Model 3:

This model was largely unsuccessful, no matter the length of two input structures.

The MAP estimation of the rotation and translation applied to X_2 is highly dissimilar to the optimal rotation and translation chosen for model 2 by the RMSD method.

Furthermore estimated mean structure poorly emulates the the solution found via the RSMD method this can be seen from Figures 8 & 9. The reasoning for these differences are speculated in the discussion.

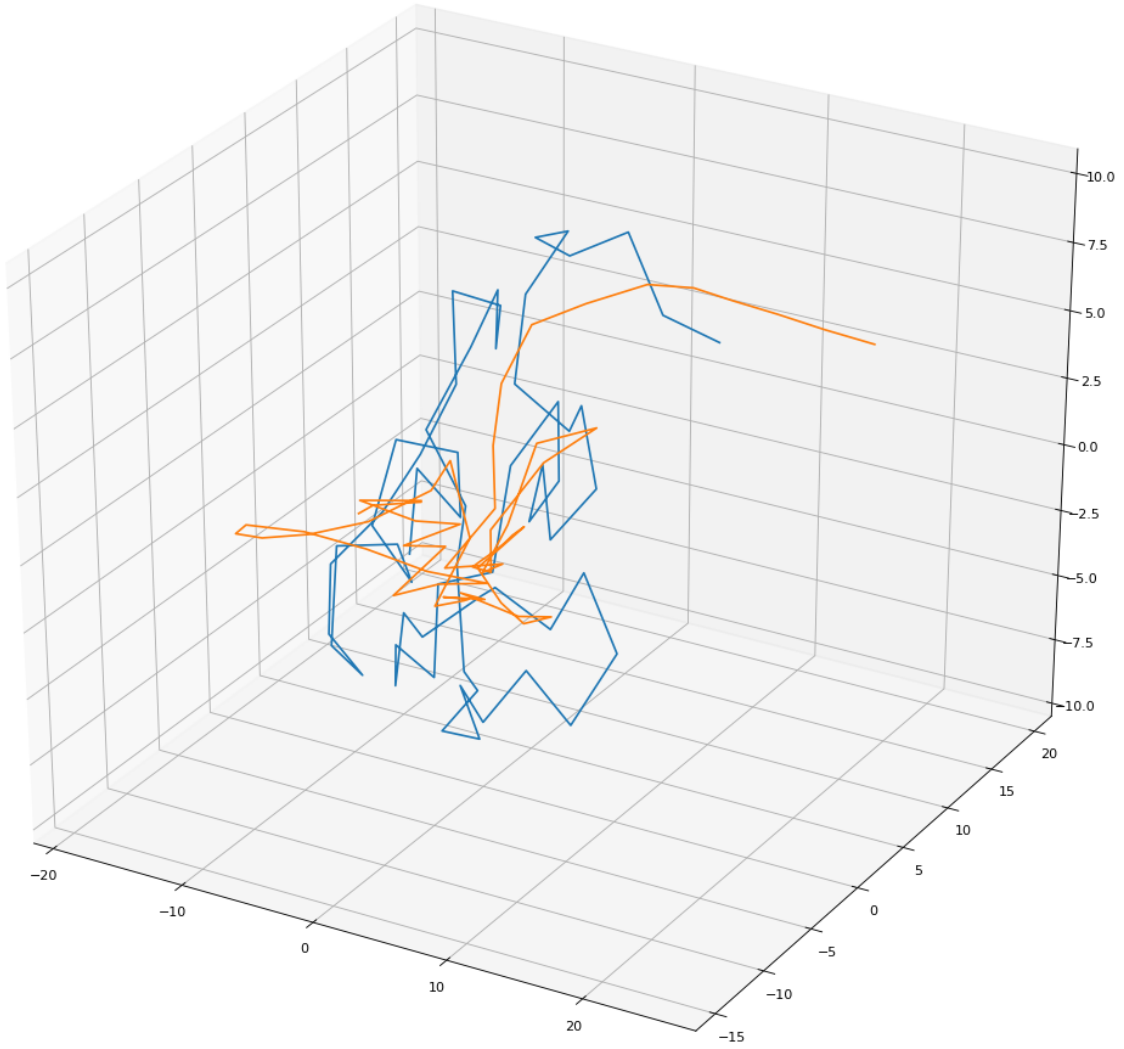


Figure 8. 3D conformational structures of the RMSD solution (blue) and the estimated mean-structure(orange) in euclidean space, for the whole residue chain (N = 61), with arbitrary distances on the X,Y,Z axis.

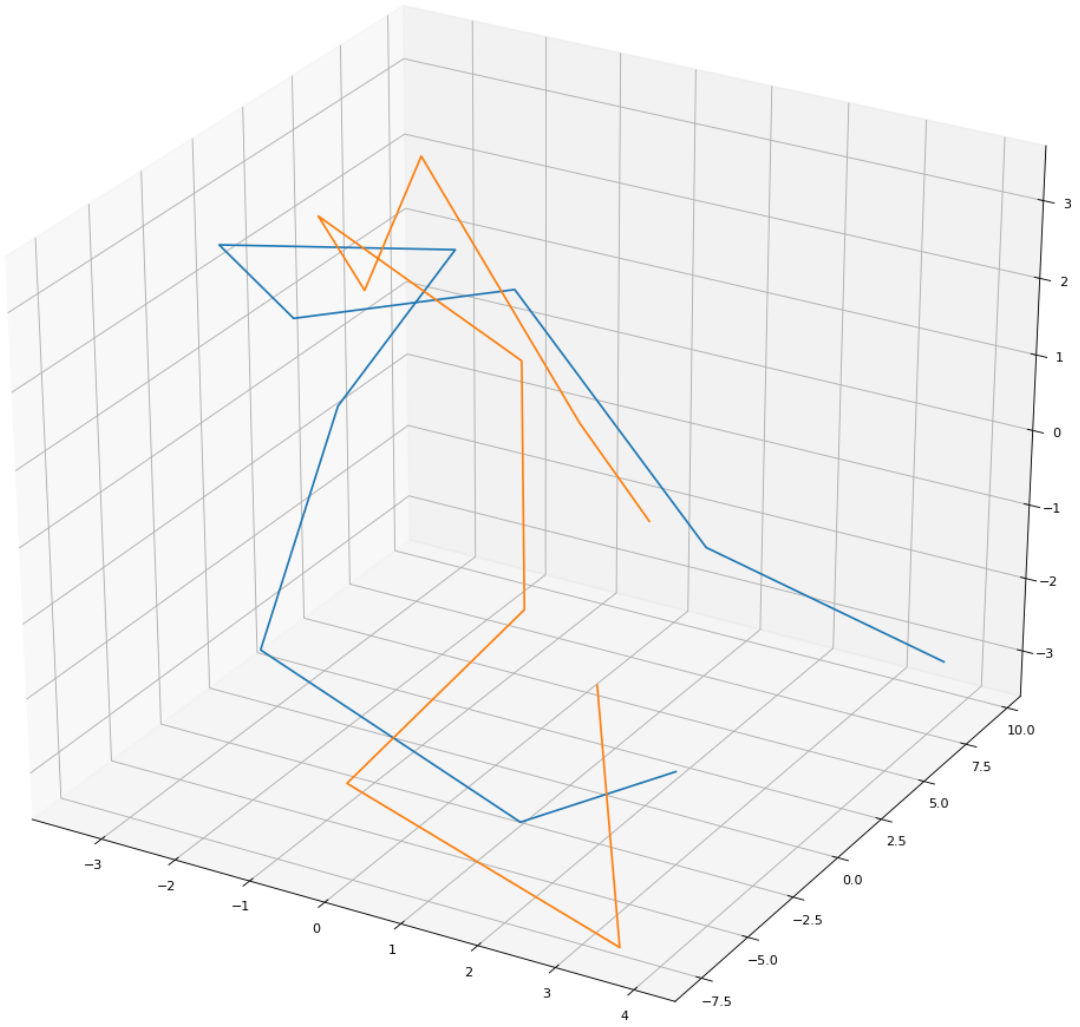


Figure 9. 3D conformational structures of the RMSD solution (blue) and the estimated mean-structure(orange) in euclidean space, N = 10, with arbitrary distances on the X,Y,Z axis.

Discussion

The initial model with a fixed-mean provided an accurate and robust method of predicting the two rotations, this was taken to be proof-of-concept, and lead us to expand into the following models.

Upon expanding this model to have an unknown mean structure, and account for spatial translations as well as rotations and noise, (model 2) several difficulties were encountered and many re-implementations of the model were made, most notably the prior over the unknown mean-structure has be modified several times before it's current state.

At this point the second model now successfully provides consistently accurate predictions of the rotation, and the mean structure with small polypeptides (10 residues or fewer), and reasonably good approximations of the spatial translations undertaken.

Regrettably the larger a polypeptide becomes (especially ~30 residues or greater), the less accurate and consistent the predictions become.

Removing the added translations and noise, which further perturbed the structures did not improve the accuracy of the model in circumstances with higher N values. This would seem to suggest that the problem is most likely with the prior over the mean-structure, though it could also be due to or the covariance matrix, and

There are several potential reasons for this, and potential avenues of further investigation into how the model my be improved, these are speculated to be:

- The prior over the mean initializes inadequately.
    - The model is becoming 'stuck' in a false minimum, because the mean structure initializes too far away from the true solution.
- The prior over the mean is still not informative enough.
    - The distance between Cα atoms is well-defined, but the model allows for theta/tau angle pairs between Cα atoms which a protein does not permit. The prior should be more restrictive. This especially apparent in figure 8, where the estimated structure (orange) doubles back on itself a number of times.

- Having only unfixed translations allow the optimal mean-structure too much freedom.
    - Fixing one of the translations in place would provide a definite starting point the initialization of the mean structure could Gaussian walk form towards the other structure(s)(similar to not having a rotation on X_1). It may currently be sliding freely until becoming stuck in local minimums, or have multiple competing optimal positions.
- The orientation of the covariance matrix is not taken into account.
    - Whilst the two structures are freely-moving the covariance drawn from them is not updating with regards to their orientation. This may be causing unnecessary constraint on the other optimization of the other variables. This could be counteracted by adding a rotation parameter to for the covariance matrix, or by holding all structures in the same orientation in euclidean space.
- While mathematically convenient, flattening the input matrices and using a multivariate normal PDF may have had unforeseen consequences.
    - Try remove flattening and re-implement the log-postirior function as a matrix normal PDF

The third model was unsuccessful even with short residue chains (N=10), it is my belief that this is not necessarily due to a problem with our model (other than those aforementioned) but is more likely mis-use of the Biopython toolset, which I suspect because the RMSD solution doesn't look like a protein structure either. Otherwise this could potentially be due to an error in the methodology (perhaps the two can't be realistically compared)?

This was an ambitions project to undertake in an 8 week block, it is my firm belief that this model shows proof-of-concept, and has made good progress towards an automated method of quickly and accurately predicting the optimal superposition of potentially hundreds of full-sized proteins, however truly reaching this goal will likely take many more weeks optimizing the priors of the model, and investigating the aforementioned.

With more time, I've no doubt it would have been possible to investigate and find solutions to all the above problems, and potentially extend the model to be able to superposition large numbers of full-sized protein structures.

Conclusions

In this investigation, an automated, Bayesian probabilistic method of super-positioning short polypeptide structures has been developed. Showing proof-of-concept; the model can be used to successfully infer a mean structure (the optimal superposition) for two short simulated polypeptide structures; however various issues are introduced when scaling the model to longer simulated polypeptides or full-length real protein structures.

Further fine-tuning of the method and its parameters will likely allow it's future use and application to multiple, full length, protein sequences.

This small initial investigation also suggests there are potentially far-reaching opportunities for the implementation of Bayesian probabilistic programming methods within other aspects of structural bioinformatics and extending to the discipline of bioinformatics as a whole.

Acknowledgements

Special thanks to Prof. Thomas Wim Hamelryck, for constructing many of the priors used, and his continued enthusiastic tutor-ledge through the many concepts involved in this project.

Citations

Berman, H. M. *et al.* (2000) 'The Protein Data Bank', *Nucleic Acids Research*. Oxford University Press, 28(1), pp. 235–242. doi: 10.1093/nar/28.1.235.

Clarke, N. D. *et al.* (1994) 'Structural studies of the engrailed homeodomain.', *Protein Sci.*, 3, pp. 1779–1787. doi: 10.2210/PDB1ENH/PDB.

Coutsias, E. A., Seok, C. and Dill, K. A. (2004) 'Using quaternions to calculate RMSD', *Journal of Computational Chemistry*, 25(15), pp. 1849–1857. doi: 10.1002/jcc.20110.

Lewandowski, D., Kurowicka, D. and Joe, H. (2009) 'Generating random correlation matrices based on vines and extended onion method', *Journal of Multivariate Analysis*. Academic Press, 100(9), pp. 1989–2001. doi: 10.1016/J.JMVA.2009.04.008.

McLachlan, A. D. and IUCr (1982) 'Rapid comparison of protein structures', *Acta Crystallographica Section A*. International Union of Crystallography, 38(6), pp. 871–873. doi: 10.1107/S0567739482001806.

Mine, S. *et al.* (2002) 'Structural mechanism for heparin-binding of the third Kunitz domain of human tissue factor pathway inhibitor.', *Biochemistry*, 41, pp. 78–85. doi: 10.2210/PDB1IRH/PDB.

Mred, S. (1998) *Matrix and Quaternion FAQ*. Available at: https://www.flipcode.com/documents/matrfaq.html#Q54 (Accessed: 25 January 2018).

Perez-Sala, X. *et al.* (2013) 'Uniform Sampling of Rotations for Discrete and Continuous Learning of 2D Shape Models', *Robotic Vision*, pp. 23–42. doi: 10.4018/978-1-4666-2672-0.

Salvatier, J., Wiecki, T. and Fonnesbeck, C. (2015) 'Probabilistic Programming in Python using PyMC', pp. 1–24. doi: 10.7717/peerj-cs.55.

Simon Cory (2017) *Generating uniformly distributed numbers on a sphere – Mathemathinking*. Available at: https://corysimon.github.io/articles/uniformdistn-on-sphere/ (Accessed: 25 January 2018).

The PyMC Development Team (2017) *LKJ Cholesky Covariance Priors for Multivariate Normal Models — PyMC3 3.3 documentation*. Available at: http://docs.pymc.io/notebooks/LKJ.html (Accessed: 25 January 2018).

Theobald, D. L. and Steindel, P. A. (2012) 'Optimal simultaneous superpositioning of multiple structures with missing data', *Bioinformatics*, 28(15), pp. 1972–1979. doi: 10.1093/bioinformatics/bts243.

Various (2014) *Wishart fails with valid arguments · Issue #538 · pymc-devs/pymc3, GitHUb*. Available at: https://github.com/pymc-devs/pymc3/issues/538#issuecomment-94153586 (Accessed: 25 January 2018).