PhD Thesis Report

Statistical Shape Analysis of Large Molecular Data Sets

Anthony Hennessey

Principal supervisor:

Dr. Christopher Fallaize

Supervisors:

Prof. Ian L. Dryden

Prof. Huiling Le

School of Mathematical Sciences

November 2017

Abstract

Protein classification databases are widely used in the prediction of protein structure and function, and amongst these databases the manually-curated Structural Classification of Proteins database (SCOP) is considered to be a gold standard. In SCOP, functional relationships are described by hyperfamily and superfamily categories and structural relationships are described by family, species and protein categories. We present a method to calculate a difference measure between pairs of proteins that can be used to reproduce SCOP2 structural relationship classifications, and that can also be used to reproduce a subset of functional relationship classifications at the superfamily level.

Calculating the difference measure requires first finding the best correspondence between atoms in two protein configurations. The problem of finding the best correspondence is known as the unlabelled, partial matching problem. We consider the unlabelled, partial matching problem through a detailed analysis of the approach presented in Green and Mardia (2006). Using this analysis, and applying domain-specific constraints, we develop a new algorithm called GProtA for protein structure alignment. The proposed difference measure is constructed from the root mean squared deviation of the aligned protein structures and a binary similarity measure, where the binary similarity measure takes into account the proportions of atoms matching from each configuration.

The GProtA algorithm and difference measure are applied to protein structure data taken from the Protein Data Bank. The difference measure is shown to correctly classify 62 of a set of 72 proteins into the correct SCOP family categories when clustered. Of the remaining 9 proteins, 2 are assigned incorrectly and 7 are considered indeterminate. In addition, a method for deriving characteristic signatures for categories is proposed. The signatures offer a mechanism by which a single comparison can be made to judge similarity to a particular category. Comparison using characteristic signatures is shown to correctly delineate proteins at the family level, including the identification of both families for a subset of proteins described by two family level categories.

Contents

List of Figures

List of Tables

Acknowledgements

My sincerest thanks:

To my supervisors Dr. Christopher Fallaize, Prof. Ian Dryden and Prof. Huiling Le for their generosity of knowledge and their guidance.

To Dr Christopher Tench for many valuable and insightful discussions.

To David Parkin for being an excellent and resourceful system administrator.

And to the many other inhabitants of the maths department that have freely given their help and friendship throughout a long and rewarding stay.

1

Introduction

1.1 Motivation and research question

Structure comparison and structure determination are primary goals of protein science (Tramontano 2005; Lesk 2013) and the comparison of protein structures requires the quantification of the difference between these protein structures. Most existing protein comparison techniques quantify structure difference through the application of heuristic algorithms. The primary goal of this work is to develop a method to quantify protein difference that is based on probabilistic modelling.

A significant role in protein comparison is often played by structure classifications. Structure classifications are themselves widely used for purposes such as the prediction of protein function, understanding evolutionary relationships and constructing data sets of representative protein structures. There currently exist many structure classification databases (Lesk 2013, chapter 6; Al-Lazikani et al. 2008; Pavlopoulou and Michalopoulos 2011), amongst these the SCOP, *Structural Classification of Proteins* (Hubbard et al. 1999) and *CATH Protein Structure Classification* (Sillitoe et al. 2015) databases are widely viewed as the gold standard (Csaba et al. 2009).[1] The SCOP database is the only fully manually curated database;

[1] An up-to-date list of databases containing either partially automated or fully automated classifications can be found in the PDB's Structure Classification links: `http://www.rcsb.org/pdb/static.do?p=general_information/web_`

CATH has a mixture of manually and automatically curated content. With SCOP being fully manually curated it offers us an ideal reference to gauge the ability of any proposed quantification of protein difference to capture differences that are considered important by experts. The importance of the SCOP database within the protein science community is illustrated by the following statistics: SCOP 1.75 was released in 2009 and contained classifications for 1,195 proteins (this was the last release before SCOP2 (Andreeva et al. 2014) which was released containing classifications for 995 proteins). From 2009–2011 (inclusive) 22,993 new protein structures were added to an existing 54,512 structures in the *Protein Data Bank*[2] and yet even with the relatively limited content in SCOP Fox et al. (2015) identified 439 articles published in 2012–2013 that use SCOP data.

A principle element of protein comparison is the optimal alignment of the structures being compared. There are many published protein structure alignment techniques – important examples of these are the following five which are made available on RCSB PDB[3]: FATCAT (Ye and Godzik 2003), CE (Shindyalov and Bourne 1998), Mammoth (Lupyan et al. 2005), TM-Align (Zhang and Skolnick 2005), TopMatch (Sippl 2008). There are also many measures of protein difference, for example Hasegawa and Holm (2009) list 26 structural similarity measures that are in common use. Despite the importance of the fully manually curated SCOP classifications and despite these many alignment and comparison techniques, the author was only able to find a single systematic attempt to reproduce either SCOP or SCOP2 classifications (Harder et al. 2012). Harder et al. (2012) builds on a technique to produce protein descriptors that represent protein structures as smoothed curves in 3D space (Røgen 2005). Harder et al. (2012) reports good agreement for proteins that have high level classification differences but is less successful for proteins that are closely related at lower classification levels, particularly when they are of differing sizes. The value of a comparison with SCOP as a validation of any proposed methods is clear, and as such in the final chapter of this work we use our developed protein alignment method and proposed difference measure to reproduce the classifications for a set of proteins included in SCOP2.

The remainder of this chapter is divided into four sections. Section 1.2 describes the *statistical theory of shape*. Protein science uses the idea of a *protein structure space* and

links/structure_classification.html.

[2] https://www.rcsb.org/pdb/static.do?p=general_information/pdb_statistics/index.html retrieved 2017-09-10.

[3] http://www.rcsb.org

a *protein function space*. Proteins that are spatially similar are described as being close together in protein structure space, and proteins of similar function are described as being close together in, the less well defined, protein function space. The statistical theory of shape fits well with the concept of protein structure space and has already been used to investigate the structural alignment of proteins (Hamelryck et al. 2012, Part IV); a review of these existing Bayesian structural alignment methods is conducted in Chapter 2. The statistical theory of shape is also the foundation for our proposed protein alignment algorithm which we develop in Chapter 3. Section 1.3 offers an introduction to protein chemistry and the current measures used in protein science for protein comparison. This section concentrates on the aspects of protein chemistry that we will exploit for the algorithm described in Chapter 3. In Section 1.4 we give a short overview on the reasoning behind the choice of software and programming languages used to reproduce published work, and for simulations and the implementation of our work. Finally in Section 1.5 we briefly introduce the structure of the remainder of the thesis.

1.2 The Statistical Theory of Shape

Shape theory simplifies the representation of an object to a collection of *labelled landmarks*. In the case of a geometric object, the landmarks would typically be the intersections of its edges and the simplification involves disregarding information relating to the faces and edges themselves (the edges being the intersections of the faces). Defining a protein structure by a series of landmarks - the derived atomic locations - is a natural choice and so the study of protein structure is particularly suited to shape theoretic treatment.

The foundation of shape theory is a mechanism to assign a measure of difference between two shapes. A very high level description of the approach taken is to represent the shape as a single point in a high dimensional space so that the distance between two points, representing two shapes, in that space can be measured. In shape theory the shape of an object must only include the information about the object that is invariant under *rotation*, *translation* and *scaling* (Kendall 1984); the procedures used to remove location, scale and rotation information will be referred to as *modding out*, i.e. if $\boldsymbol{A}'$ is $\boldsymbol{A}$ *modulo location* then $\boldsymbol{A}'$ is a representation of $\boldsymbol{A}$ once all information relating to the location of $\boldsymbol{A}$ in Euclidean space has been removed.

1.2.1 Labelling

Labelling is used to capture the correspondence of landmarks between objects, and the ability to assign specific properties to particular landmarks significantly broadens the potential applications of shape theory. Two possible examples of properties described by the labels are: a specific label indicates a specific feature in a facial recognition task, and the labels indicate the order in which landmarks were created. Figure 1.1 is an example in which labels describe the order in which landmarks were created allowing a path to be inferred. Four landmarks are arranged in two dimensional space at the corners of a square and here are $4!$ possible labelling schemes, each of which correspond to a different shape. Objects A and B have landmarks in the same spatial arrangement but the additional information encoded by the labels allows the interpretation of two clearly distinct objects; specifically in this case the simple representations of the letters N and C.

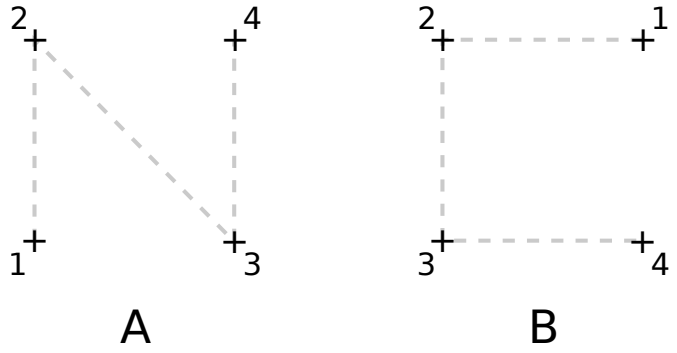


Figure 1.1: An example of where the addition of labelling information to landmarks offers a clear distinction in interpretation compared to the unlabelled spatial arrangement.

1.2.2 A matrix representation of labelled landmarks

A configuration of k landmarks in $\mathbb{R}^d$ is represented by a $k \times d$ matrix, $\boldsymbol{A} = (\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_k)^\mathsf{T}$, of Euclidean co-ordinates; where the subscript, i, of each set of Euclidean co-ordinates, $\boldsymbol{a}_i$, is the label of the specific landmark.

1.2.3 Modulo location

The location of a configuration can be described by its centroid, the mean of the Euclidean co-ordinates

$$\bar{\boldsymbol{a}} = \frac{1}{k}\sum_{i=1}^{k} \boldsymbol{a}_i \ .$$

Location can then be removed by subtracting the centroid, $\bar{\boldsymbol{a}}$, from the configuration; intuitively the configuration is translated so that it is centred at the origin

$$\overset{(k\times d)}{\overset{\mathsf{c}}{\boldsymbol{A}}} = \overset{(k\times d)}{\boldsymbol{A}} - \overset{(k\times 1)}{\mathbf{1}_k}\overset{(1\times d)}{\bar{\boldsymbol{a}}^{\mathsf{T}}} \ . \tag{1.1}$$

Equation 1.1 can be rewritten

$$\begin{aligned}\overset{\mathsf{c}}{\boldsymbol{A}} &= \mathbf{I}_k\boldsymbol{A} - \frac{1}{k}\overset{(k\times k)}{\mathbf{1}_k\mathbf{1}_k^{\mathsf{T}}}\boldsymbol{A} \\ &= \left(\mathbf{I}_k - \frac{1}{k}\mathbf{1}_k\mathbf{1}_k^{\mathsf{T}}\right)\boldsymbol{A} \\ &= \boldsymbol{Q}\boldsymbol{A} \ . \end{aligned} \tag{1.2}$$

where $\boldsymbol{Q}$ is the $k \times k$, rank $k-1$, matrix

$$\boldsymbol{Q} = \mathbf{I}_k - \frac{1}{k}\mathbf{1}_k\mathbf{1}_k^{\mathsf{T}} \ . \tag{1.3}$$

Note that $\boldsymbol{Q}$ depends only on k.

It is clear that $\overset{\mathsf{c}}{\boldsymbol{A}}$ (and by association $\boldsymbol{Q}\boldsymbol{A}$) are not full rank as knowing the location of the centroid the location of any individual landmark can be derived from the remaining $k-1$ landmarks. We want to avoid rank deficiency as our intention is ultimately to align sets of shapes by minimising squared distances between labelled landmarks, and rank deficiency will result in the being an ill-conditioned problem. Rank deficiency can be resolved by removing any row of $\overset{\mathsf{c}}{\boldsymbol{A}}$, however as a result of the arbitrary nature of the labelling this will present problems for the consistent comparison of objects. Consider Figure 1.2: the removal of a row relating to the location of any of landmarks 1, 3 or 4 from the centred configurations of $\boldsymbol{A}$ and $\boldsymbol{B}$ will result in a smaller perceived object difference for any measure based on distances of the remaining landmarks from the origin than the removal of the row relating to landmark 2.

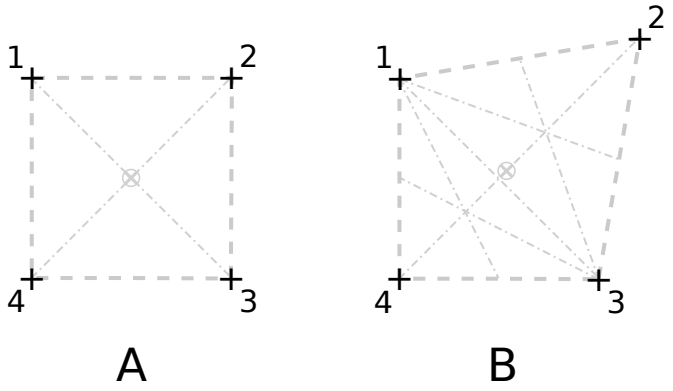


Figure 1.2: Two objects represented by four landmarks; the objects differ only in the location of landmark 2. The objects are shown to demonstrate that the removal of information relating to one landmark can lead to incorrect inference about their shape.

1.2.3.1 The Helmert matrix

In formulations of shape by D. G. Kendall[4] location is removed using a matrix closely related to $\boldsymbol{Q}$ called a *Helmert matrix*.

Recall the definition of the *special orthogonal group* of rotations

$$\mathbf{SO}(n) = \left\{ \boldsymbol{X} \in \mathbb{R}^{n \times n} \mid \boldsymbol{X}\boldsymbol{X}^{\mathsf{T}} = \boldsymbol{X}^{\mathsf{T}}\boldsymbol{X} = \mathbf{I}_n \text{ and } \det(\boldsymbol{X}) = +1 \right\} .$$

The order k Helmert matrix, $\boldsymbol{H}^{(k)}$, is a member of $\mathbf{SO}(k)$ and is of the form

$$\overset{k\times k}{\boldsymbol{H}^{(k)}} = \begin{pmatrix} \frac{1}{\sqrt{k}} & \frac{1}{\sqrt{k}} & \frac{1}{\sqrt{k}} & \frac{1}{\sqrt{k}} & \cdots & \frac{1}{\sqrt{k}} & \frac{1}{\sqrt{k}} & \frac{1}{\sqrt{k}} \\ \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & \cdots & 0 & 0 & 0 \\ \frac{-1}{\sqrt{6}} & \frac{-1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \frac{-1}{\sqrt{(k-2)(k-3)}} & \frac{-1}{\sqrt{(k-2)(k-3)}} & \frac{-1}{\sqrt{(k-2)(k-3)}} & \frac{-1}{\sqrt{(k-2)(k-3)}} & \cdots & \frac{k-3}{\sqrt{(k-2)(k-3)}} & 0 & 0 \\ \frac{-1}{\sqrt{(k-1)(k-2)}} & \frac{-1}{\sqrt{(k-1)(k-2)}} & \frac{-1}{\sqrt{(k-1)(k-2)}} & \frac{-1}{\sqrt{(k-1)(k-2)}} & \cdots & \frac{-1}{\sqrt{(k-1)(k-2)}} & \frac{k-2}{\sqrt{(k-1)(k-2)}} & 0 \\ \frac{-1}{\sqrt{k(k-1)}} & \frac{-1}{\sqrt{k(k-1)}} & \frac{-1}{\sqrt{k(k-1)}} & \frac{-1}{\sqrt{k(k-1)}} & \cdots & \frac{-1}{\sqrt{k(k-1)}} & \frac{-1}{\sqrt{k(k-1)}} & \frac{k-1}{\sqrt{k(k-1)}} \end{pmatrix} .$$

It will also be useful to note the following properties of the Helmert matrix which will be referred to later - using the notation $\boldsymbol{h}_i$ as the ith row of the *Helmert matrix* and $\boldsymbol{h}_{(i\ldots j)}$ is

[4] Kendall 1984; Kendall et al. 1999.

a matrix of the ith to jth rows of the *Helmert matrix*:

$$\boldsymbol{h}_1^\mathsf{T}\boldsymbol{h}_1 = 1 \tag{1.4}$$

$$\boldsymbol{h}_{(2\ldots k)}^\mathsf{T}\boldsymbol{h}_{(2\ldots k)} = \mathbf{I}_k - \frac{1}{k}\mathbf{1}_k\mathbf{1}_k^\mathsf{T} \tag{1.5}$$

$$\boldsymbol{h}_1\boldsymbol{h}_1^\mathsf{T} = \frac{1}{k}\mathbf{1}_k\mathbf{1}_k^\mathsf{T} \tag{1.6}$$

$$\boldsymbol{h}_{(2\ldots k)}\boldsymbol{h}_{(2\ldots k)}^\mathsf{T} = \mathbf{I}_{k-1} \tag{1.7}$$

It can be seen that the RHS of equation 1.5 is equal to the RHS of equation 1.3 i.e. that $\boldsymbol{h}_{(2\ldots k)}^\mathsf{T}\boldsymbol{h}_{(2\ldots k)}$ is a factorisation of $\boldsymbol{Q}$, and that equation 1.6 implies that any configuration matrix $\underset{k\times m}{\boldsymbol{A}}$ pre-multiplied by $\boldsymbol{H}^{(k)}$ will result in a matrix where the first row is proportional to the centroid of $\boldsymbol{A}$. Going forward $\boldsymbol{h}_{(2\ldots k)}$ will be referred to as the *order k Helmert sub-matrix* using the notation $\boldsymbol{H}_\mathsf{s}^{(k)}$ (the order may not be given if it is clear from the context).

To get an intuitive feel for the Helmert matrix consider the action of the Helmert matrix on the co-ordinates of an object with four landmarks ($k = 4$) in two dimensions ($d = 2$)

$$\begin{aligned}\boldsymbol{H}^{(4)}\boldsymbol{A} &= \begin{pmatrix} \frac{1}{\sqrt{4}} & \frac{1}{\sqrt{4}} & \frac{1}{\sqrt{4}} & \frac{1}{\sqrt{4}} \\ \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{-1}{\sqrt{6}} & \frac{-1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & 0 \\ \frac{-1}{\sqrt{12}} & \frac{-1}{\sqrt{12}} & \frac{-1}{\sqrt{12}} & \frac{3}{\sqrt{12}} \end{pmatrix} \begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{pmatrix} \\ &= \begin{pmatrix} \sqrt{4} & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{1\times 2}} & 0 & 0 \\ 0 & 0 & \frac{2}{\sqrt{2\times 3}} & 0 \\ 0 & 0 & 0 & \frac{3}{\sqrt{3\times 4}} \end{pmatrix} \begin{pmatrix} \frac{x_1+x_2+x_3+x_4}{4} & \frac{y_1+y_2+y_3+y_4}{4} \\ x_2 - \frac{x_1}{1} & y_2 - \frac{y_1}{1} \\ x_3 - \frac{x_1+x_2}{2} & y_3 - \frac{y_1+y_2}{2} \\ x_4 - \frac{x_1+x_2+x_3}{3} & y_4 - \frac{y_1+y_2+y_3}{3} \end{pmatrix}. \end{aligned} \tag{1.8}$$

When the configuration $\boldsymbol{A}$ is pre-multiplied by the Helmert matrix the first row of the resulting configuration is a multiple of the centroid which implies that the product of the Helmert matrix and a centred configuration will be of the form

$$\boldsymbol{H}\overset{\mathsf{c}}{\boldsymbol{A}} = \begin{pmatrix} \mathbf{0} & \boldsymbol{a}_1' & \boldsymbol{a}_2' & \ldots & \boldsymbol{a}_{k-1}' \end{pmatrix}^\mathsf{T}. \tag{1.9}$$

So for any centred configurations the first row of the product will always be zero. However, the result of pre-multiplying the centred configuration by the Helmert sub-matrix and pre-

multiplying the original configuration by the Helmert sub-matrix is always the same

$$\begin{aligned} \boldsymbol{H}_s\boldsymbol{Q}\boldsymbol{A} &= \boldsymbol{H}_s\boldsymbol{H}_s^{\mathsf{T}}\boldsymbol{H}_s\boldsymbol{A} \qquad \text{by equation 1.5} \\ &= \boldsymbol{H}_s\boldsymbol{A} \qquad \text{by equation 1.7 .} \end{aligned} \tag{1.10}$$

This result shows that pre-multiplication by the Helmert sub-matrix is strictly only removing location information and also implies that the resulting $\boldsymbol{H}_s\boldsymbol{A}$ matrix is of full rank.

It is worth emphasising that $\boldsymbol{H}^{(k)} \in \mathbf{SO}(k)$ and so there is a change in basis set implying that the geometric relation between the landmarks will be modified; this is illustrated in figure 1.3 which plots the result of pre-multiplying a set of landmarks by the Helmert sub-matrix. Pre-multiplication by $\boldsymbol{H}^{(k)}$ can be thought of as a rotation of co-ordinates in landmark space rather than a rotation of landmarks in co-ordinate space.

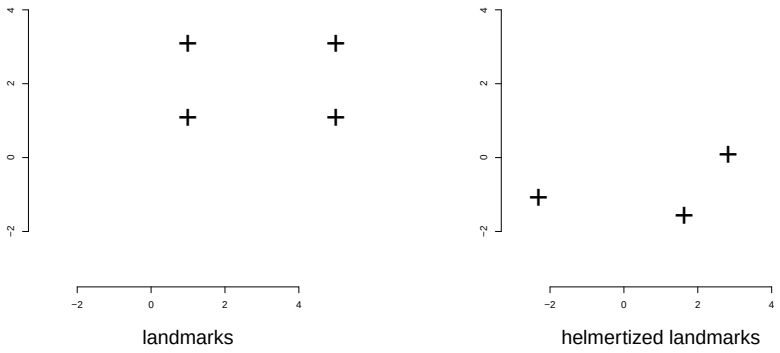


Figure 1.3: Comparison of the the original four landmarks of an object configuration to the result of pre-multiplication by a Helmert sub-matrix. This demonstrates that the geometric relationship between the landmarks is modified.

To simplify presentation Helmertized configurations will often be represented by the addition of an apostrophe such that

$$\boldsymbol{A}' = \boldsymbol{H}_s\boldsymbol{A}$$

1.2.4 Modulo scaling

To remove size it is possible to normalise using a quadratic measure of size for the centred configuration

$$\|\overset{c}{\boldsymbol{A}}\|_{\mathsf{F}} = \sqrt{\|\boldsymbol{a}_1 - \bar{\boldsymbol{a}}\|^2 + \|\boldsymbol{a}_2 - \bar{\boldsymbol{a}}\|^2 + \cdots + \|\boldsymbol{a}_k - \bar{\boldsymbol{a}}\|^2}\ .$$

This measure of size is the square root of the sum of the squared Euclidean distances between each landmark and the centroid of the configuration. Since $\overset{c}{\boldsymbol{A}} = \boldsymbol{Q}\boldsymbol{A}$ (equation 1.2) and $\boldsymbol{H}_s\boldsymbol{A} = \boldsymbol{H}_s\overset{c}{\boldsymbol{A}}$ (using equation 1.10) the orthogonality of $\boldsymbol{H}_s$ implies that $\|\overset{c}{\boldsymbol{A}}\|_F = \|\boldsymbol{H}_s\overset{c}{\boldsymbol{A}}\|_F$, hence we remove scale using the convenient form

$$\boldsymbol{A}^* = \frac{\boldsymbol{H}_s\boldsymbol{A}}{\|\boldsymbol{H}_s\boldsymbol{A}\|_F}$$

and so $\boldsymbol{A}^*$ is a set of $k-1$ normalised, Helmertized landmarks in d dimensional Euclidean space, $\mathbb{R}^d$, and we assume throughout that the landmarks are not all coincident, i.e that $\|\boldsymbol{H}_s\boldsymbol{A}\|_F > 0$. An alternative interpretation of $\boldsymbol{A}^*$ is as defining a unit vector in $m(k-1)$ dimensional Euclidean space, $\mathbb{R}^{d(k-1)}$, then the set of all possible $\boldsymbol{A}^*$s would map to the surface of a hypersphere in $\mathbb{R}^{d(k-1)}$; this interpretation of $\boldsymbol{A}^*$ is known as the *pre-shape* of the original configuration and the space of all possible pre-shapes for particular pair of k and d is known as the Kendall $\mathcal{S}_d^k$ *pre-shape sphere*.

Using a similar interpretation for the Helmertized landmarks, $\boldsymbol{A}'$, these can be thought of as forming a set of concentric spheres in $\mathbb{R}^{d(k-1)}$ equivalent to a cone in $\mathbb{R}^{dk}$ with its apex at the origin; this is the *pre-size-and-shape* space.

1.2.5 Shape distance

The shape of a configuration is the pre-shape, $\boldsymbol{A}^*$, modulo rotation. The shape space of k landmarks in m dimensions is written as Σ_d^k i.e. Σ_d^k is the quotient of $\mathcal{S}_d^k$ by $\mathbf{SO}(d)$. This concept is somewhat abstract and the descriptions of the topology and geometry of Σ_d^k are highly complex; however, what is of concern to this work are measures of differences between shapes and for this we can work in the much more intuitive pre-shape space.

The set of locations on the pre-shape sphere, $\mathcal{S}_d^k$, resulting from the action of $\mathbf{SO}(d)$ on a pre-shape configuration is a continuous, open set.[5] In essence this means that the rotations trace a continuous path or *fibre* over the the surface of the pre-shape sphere and that the distance between points on distinct fibres is measurable. The minimum length geodesic, in this case a great circle, between any two fibres on $\mathcal{S}_d^k$ is a measure of distance between the shapes in Σ_d^k; this is called the *Procrustes distance*. The minimum distance between the fibres calculated in $\mathbb{R}^{d(k-1)}$ in which the $\mathcal{S}_d^k$ is embedded will be referred to as the *chordal*

[5] Kendall et al. 1999, Section 6.1.

distance (often termed the *partial Procrustes distance* in the literature). The two distance measures have a simple functional relationship which is described in Dryden and Mardia (2016, Section 4.2) along with their relationship to other commonly used shape distances.

Next, a description is given of the method used to optimise the rotation of the pre-shape configurations which can be used to find the minimum chordal distance and hence the chordal shape distance. Consideration of the optimisation of the rotation suggests a simplified route to the chordal distance.

1.2.5.1 Optimising rotation

Optimisation of the rotation of two configurations to minimise the squared distance between associated landmarks is a constrained version of the *orthogonal Procrustes problem*; the constraint being that the solution is restricted to rotation matrices i.e. orthogonal matrices with determinant one. A general *singular value decomposition* (SVD) based solution was first given in Schönemann (1966).[6] It is noted both that as Helmertized configurations are being used no consideration of translation is needed, and that the following holds equally for the rotation optimisation when calculating shape distance or size-and-shape distance.

Minimising the chordal distance is equivalent to minimising the Frobenius norm under rotation i.e.

$$\inf_{\boldsymbol{R}\in\mathbf{SO}(d)} \|\boldsymbol{A}^* - \boldsymbol{B}^*\boldsymbol{R}\|_{\mathsf{F}} \ . \tag{1.11}$$

Recall that the *trace* of a square matrix is the sum of its diagonal elements

$$\mathrm{Tr}\left\{\boldsymbol{X}\right\} = \sum_{i=1}^{n} x_{(i,i)} \qquad \text{where} \quad \boldsymbol{X} \in \mathbb{R}^{n\times n}$$

and that

$$\mathrm{Tr}\left\{\boldsymbol{X}\boldsymbol{Y}\right\} = \mathrm{Tr}\left\{\boldsymbol{Y}\boldsymbol{X}\right\} \ .$$

Minimising the Frobenius norm with respect to the rotation, $\boldsymbol{R}$, is equivalent to maximising

[6] Details pertinent to shape are discussed in Kendall et al. (1999, Section 6.4), with a more algorithmic treatment suggested in Golub (2013, Section 6.4.1).

$\mathrm{Tr}\{\boldsymbol{A}^{*\mathsf{T}}\boldsymbol{B}^{*}\boldsymbol{R}\}$, since

$$\begin{aligned}
\|\boldsymbol{X}-\boldsymbol{Y}\boldsymbol{R}\|_{\mathsf{F}}^{2} &= \sum_{i=1}^{d}\|\boldsymbol{X}_{i}-\boldsymbol{Y}_{i}\Gamma\|_{2}^{2} \\
&= \mathrm{Tr}\left\{(\boldsymbol{X}-\boldsymbol{Y}\boldsymbol{R})^{\mathsf{T}}(\boldsymbol{X}-\boldsymbol{Y}\boldsymbol{R})\right\} \\
&= \mathrm{Tr}\left\{\boldsymbol{X}^{\mathsf{T}}\boldsymbol{X}-\boldsymbol{X}^{\mathsf{T}}\boldsymbol{Y}\boldsymbol{R}-\boldsymbol{R}^{\mathsf{T}}\boldsymbol{Y}^{\mathsf{T}}\boldsymbol{X}+\boldsymbol{R}^{\mathsf{T}}\boldsymbol{Y}^{\mathsf{T}}\boldsymbol{Y}\boldsymbol{R}\right\} \\
&= \mathrm{Tr}\left\{\boldsymbol{X}^{\mathsf{T}}\boldsymbol{X}+\boldsymbol{Y}^{\mathsf{T}}\boldsymbol{Y}-2\boldsymbol{X}^{\mathsf{T}}\boldsymbol{Y}\boldsymbol{R}\right\} \\
&= \mathrm{Tr}\left\{\boldsymbol{X}^{\mathsf{T}}\boldsymbol{X}\right\}+\mathrm{Tr}\left\{\boldsymbol{Y}^{\mathsf{T}}\boldsymbol{Y}\right\}-2\,\mathrm{Tr}\left\{\boldsymbol{X}^{\mathsf{T}}\boldsymbol{Y}\boldsymbol{R}\right\} \\
&= \|\boldsymbol{X}\|_{\mathsf{F}}^{2}+\|\boldsymbol{Y}\|_{\mathsf{F}}^{2}-2\,\mathrm{Tr}\left\{\boldsymbol{X}^{\mathsf{T}}\boldsymbol{Y}\boldsymbol{R}\right\}\ . \qquad (1.12)
\end{aligned}$$

If $\boldsymbol{A}^{*\mathsf{T}}\boldsymbol{B}^{*}$ is factorised using SVD (Golub 2013, Section 2.4), then

$$\boldsymbol{A}^{*\mathsf{T}}\boldsymbol{B}^{*}=\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^{\mathsf{T}}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrices, $\boldsymbol{U},\boldsymbol{V}\in\mathbf{O}(d)$, and $\boldsymbol{S}$ is a diagonal matrix, $\boldsymbol{S}=\mathrm{diag}(s_1,s_2,\ldots,s_d)$, where $s_1\geq s_2\geq\cdots\geq s_d\geq 0$.

Then

$$\begin{aligned}
\mathrm{Tr}\left\{\boldsymbol{A}^{*\mathsf{T}}\boldsymbol{B}^{*}\boldsymbol{R}\right\} &= \mathrm{Tr}\left\{\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^{\mathsf{T}}\boldsymbol{R}\right\} \\
&= \mathrm{Tr}\left\{\boldsymbol{S}\boldsymbol{U}\boldsymbol{V}^{\mathsf{T}}\boldsymbol{R}\right\} \\
&= \sum_{i=1}^{d}s_i\left(\boldsymbol{U}\boldsymbol{V}^{\mathsf{T}}\boldsymbol{R}\right)_{(i,i)} \qquad (1.13)
\end{aligned}$$

since $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{R}$ are all orthogonal and hence the maximum of their product is $\mathbf{I}_d$. Thus equation 1.13 is maximised when

$$\begin{aligned}
\mathbf{I} &= \boldsymbol{U}\boldsymbol{V}^{\mathsf{T}}\boldsymbol{R} \\
\boldsymbol{R}^{\mathsf{T}}\boldsymbol{R} &= \boldsymbol{U}\boldsymbol{V}^{\mathsf{T}}\boldsymbol{R} \\
\boldsymbol{R}^{\mathsf{T}} &= \boldsymbol{U}\boldsymbol{V}^{\mathsf{T}} \\
\boldsymbol{R} &= \boldsymbol{V}\boldsymbol{U}^{\mathsf{T}} \qquad (1.14)
\end{aligned}$$

At this point $\boldsymbol{V}\boldsymbol{U}^{\mathsf{T}}$ is only guaranteed to be a member of $\mathbf{O}(d)$ i.e. $\det(\boldsymbol{V}\boldsymbol{U}^{\mathsf{T}})\in\{-1,1\}$. For the case where $\det(\boldsymbol{V}\boldsymbol{U}^{\mathsf{T}})=-1$ the global optima includes a reflection and a local

optima transformation that is only a rotation is required.

Equation 1.13 is a linear function of the $(\boldsymbol{U}\boldsymbol{V}^{\mathsf{T}}\boldsymbol{R})_{(i,i)}$ defined on the space $[-1,1]^d$. The linearity implies that optima are reached at the boundaries of the space and as described the global optima is achieved at $(1,1,\ldots,1)$. The property of $\boldsymbol{S}$ that the singular values are ordered implies that the next greatest optima is achieved at $(1,1,\ldots,-1)$ and clearly this will have a determinant which is the negative of that of the global maxima, hence equation 1.14 can be modified to give the optima not allowing reflections

$$\boldsymbol{R} = \boldsymbol{V}\begin{pmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \ldots & \det(\boldsymbol{V}\boldsymbol{U}^{\mathsf{T}}) \end{pmatrix}\boldsymbol{U}^{\mathsf{T}} \,. \tag{1.15}$$

It will be useful to note later that a rotation can be parameterised by an axis of rotation and an angle of rotation around the axis. Using this parameterisation the angle of rotation, θ, can be derived from the rotation matrix using

$$\mathrm{Tr}(\boldsymbol{R}) = (d-2) + 2\cos\theta \,. \tag{1.16}$$

1.2.5.2 Chordal shape distance

The chordal shape distance is given by

$$\mathrm{d}\left(\boldsymbol{A},\,\boldsymbol{B}\right) = \inf_{\boldsymbol{R}\in\mathbf{SO}(d)} \|\boldsymbol{A}^* - \boldsymbol{B}^*\boldsymbol{R}\|_{\mathsf{F}} \,. \tag{1.17}$$

Using details from section 1.2.5.1

$$\begin{aligned} \mathrm{d}\left(\boldsymbol{A},\,\boldsymbol{B}\right)^2 &= \inf_{\boldsymbol{R}\in\mathbf{SO}(d)} \|\boldsymbol{A}^* - \boldsymbol{B}^*\boldsymbol{R}\|_{\mathsf{F}}^2 \\ &= \inf_{\boldsymbol{R}\in\mathbf{SO}(d)} \|\boldsymbol{A}^*\|_{\mathsf{F}}^2 + \|\boldsymbol{B}^*\|_{\mathsf{F}}^2 - 2\,\mathrm{Tr}\left\{\boldsymbol{A}^{*\mathsf{T}}\boldsymbol{B}^*\boldsymbol{R}\right\} \\ &= \inf_{\boldsymbol{R}\in\mathbf{SO}(d)} 1 + 1 - 2\,\mathrm{Tr}\left\{\boldsymbol{A}^{*\mathsf{T}}\boldsymbol{B}^*\boldsymbol{R}\right\} \\ &= 2 - 2\begin{cases} \sum_{i=1}^{d} s_i & \text{if } \det(\boldsymbol{V}\boldsymbol{U}^{\mathsf{T}}) = +1 \\ \left(\sum_{i=1}^{d-1} s_i\right) - s_d & \text{if } \det(\boldsymbol{V}\boldsymbol{U}^{\mathsf{T}}) = -1 \end{cases} . \end{aligned} \tag{1.18}$$

To achieve 1.18 requires factorising $\boldsymbol{A}^{*\mathsf{T}}\boldsymbol{B}^{*}$ to get $\det(\boldsymbol{V}\boldsymbol{U}^{\mathsf{T}})$; this is computationally expensive. However this can be avoided as follows:

The singular values from the SVD factorisation of $\boldsymbol{A}^{*\mathsf{T}}\boldsymbol{B}^{*}$ are the positive square roots of the eigenvalues of the square symmetric matrix $(\boldsymbol{A}^{*\mathsf{T}}\boldsymbol{B}^{*})^{\mathsf{T}}(\boldsymbol{A}^{*\mathsf{T}}\boldsymbol{B}^{*})$ (Horn and Johnson 2012, Theorem 2.6.3). The determinant of $\boldsymbol{V}\boldsymbol{U}^{\mathsf{T}}$ must equal the determinant of $\boldsymbol{A}^{*\mathsf{T}}\boldsymbol{B}^{*}$ since

- determinants are multiplicative i.e. $\det(\boldsymbol{X}\boldsymbol{Y}) = \det(\boldsymbol{X})\det(\boldsymbol{Y})$
- $\det(\boldsymbol{X}) = \det(\boldsymbol{X}^{\mathsf{T}})$
- the s_i are positive
- $\boldsymbol{A}^{*\mathsf{T}}\boldsymbol{B}^{*} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^{\mathsf{T}}$

Therefore 1.18 is equivalent to

$$\mathrm{d}\left(\boldsymbol{A},\,\boldsymbol{B}\right)^{2} \;=\; 2 - 2\begin{cases} \sum_{i=1}^{d}\lambda_i & \text{if } \det(\boldsymbol{A}^{*\mathsf{T}}\boldsymbol{B}^{*}) = +1 \\ \left(\sum_{i=1}^{d-1}\lambda_i\right) - \lambda_d & \text{if } \det(\boldsymbol{A}^{*\mathsf{T}}\boldsymbol{B}^{*}) = -1 \end{cases} \tag{1.19}$$

where the λ_i are the decreasingly ordered, positive square roots of the eigenvalues of $(\boldsymbol{A}^{*\mathsf{T}}\boldsymbol{B}^{*})^{\mathsf{T}}(\boldsymbol{A}^{*\mathsf{T}}\boldsymbol{B}^{*})$.

1.2.5.3 Size-and-shape distance

Consider the geometric interpretation of the pre-size-and-shape space. The pre-shape sphere is a projection of the pre-size-and-shape onto a unit hyper-sphere in $\mathbb{R}^{d(k-1)}$. The pre-size-and-shape can be thought of as a set of concentric spheres which represent the surface of a hyper-cone in $\mathbb{R}^{dk}$ with its apex at the origin. It can be seen that rotations of a pre-size-and-shape would trace out a fibre with a path equidistant from the cone axis, as the centroid size does not change. This implies that the minimum chordal distance will always be traced along the cone surface and so be equal to the minimum length geodesic; hence for size-and-shape the Procrustes distance and chordal distances are equal.

$$\rho_{\mathsf{ss}}\left(\boldsymbol{A},\,\boldsymbol{B}\right) = \mathrm{d}_{\mathsf{ss}}\left(\boldsymbol{A},\,\boldsymbol{B}\right) = \inf_{\boldsymbol{R}\in\mathbf{SO}(m)} \|\boldsymbol{A}' - \boldsymbol{B}'\boldsymbol{R}\|_{\mathsf{F}}\,. \tag{1.20}$$

It will also be useful later to note that the chordal size-and-shape distance is equal to the root of the optimised squared distances for the original configurations, when the original configurations have been centred. This can be seen from

$$\begin{aligned}
\|\boldsymbol{H}_{\mathsf{s}}\overset{\mathsf{c}}{\boldsymbol{A}} - \boldsymbol{H}_{\mathsf{s}}\overset{\mathsf{c}}{\boldsymbol{B}}\boldsymbol{R}\|_{\mathsf{F}}^2 &= \mathrm{Tr}\left\{\left(\boldsymbol{H}_{\mathsf{s}}\overset{\mathsf{c}}{\boldsymbol{A}} - \boldsymbol{H}_{\mathsf{s}}\overset{\mathsf{c}}{\boldsymbol{B}}\boldsymbol{R}\right)^{\mathsf{T}}\left(\boldsymbol{H}_{\mathsf{s}}\overset{\mathsf{c}}{\boldsymbol{A}} - \boldsymbol{H}_{\mathsf{s}}\overset{\mathsf{c}}{\boldsymbol{B}}\boldsymbol{R}\right)\right\} \\
&= \mathrm{Tr}\left\{\left(\overset{\mathsf{c}}{\boldsymbol{A}} - \overset{\mathsf{c}}{\boldsymbol{B}}\boldsymbol{R}\right)^{\mathsf{T}}\boldsymbol{H}_{\mathsf{s}}^{\mathsf{T}}\boldsymbol{H}_{\mathsf{s}}\left(\overset{\mathsf{c}}{\boldsymbol{A}} - \overset{\mathsf{c}}{\boldsymbol{B}}\boldsymbol{R}\right)\right\} \\
&= \mathrm{Tr}\left\{\left(\overset{\mathsf{c}}{\boldsymbol{A}} - \overset{\mathsf{c}}{\boldsymbol{B}}\boldsymbol{R}\right)^{\mathsf{T}}\boldsymbol{Q}\left(\overset{\mathsf{c}}{\boldsymbol{A}} - \overset{\mathsf{c}}{\boldsymbol{B}}\boldsymbol{R}\right)\right\} \\
&= \mathrm{Tr}\left\{\left(\overset{\mathsf{c}}{\boldsymbol{A}} - \overset{\mathsf{c}}{\boldsymbol{B}}\boldsymbol{R}\right)^{\mathsf{T}}\left(\overset{\mathsf{c}}{\boldsymbol{A}} - \overset{\mathsf{c}}{\boldsymbol{B}}\boldsymbol{R}\right)\right\} \quad \text{since } \overset{\mathsf{c}}{\boldsymbol{A}} = \boldsymbol{Q}\boldsymbol{A} \text{and } \boldsymbol{Q} \text{ is idempotent} \\
&= \|\overset{\mathsf{c}}{\boldsymbol{A}} - \overset{\mathsf{c}}{\boldsymbol{B}}\boldsymbol{R}\|_{\mathsf{F}}^2 \,. \qquad (1.21)
\end{aligned}$$

Using similar reasoning it can be seen that $\|\boldsymbol{H}_{\mathsf{s}}\overset{\mathsf{c}}{\boldsymbol{X}}\|_{\mathsf{F}}^2 = \|\overset{\mathsf{c}}{\boldsymbol{X}}\|_{\mathsf{F}}^2$ which implies a similar result to 1.21 for the chordal shape distance i.e. that

$$\begin{aligned}
\mathrm{d}\left(\overset{\mathsf{c}}{\boldsymbol{A}},\, \overset{\mathsf{c}}{\boldsymbol{B}}\right)^2 &= \inf_{\boldsymbol{R}\in\mathbf{SO}(d)}\left\|\frac{\boldsymbol{H}_{\mathsf{s}}\overset{\mathsf{c}}{\boldsymbol{A}}}{\|\boldsymbol{H}_{\mathsf{s}}\overset{\mathsf{c}}{\boldsymbol{A}}\|_{\mathsf{F}}} - \frac{\boldsymbol{H}_{\mathsf{s}}\overset{\mathsf{c}}{\boldsymbol{B}}}{\|\boldsymbol{H}_{\mathsf{s}}\overset{\mathsf{c}}{\boldsymbol{B}}\|_{\mathsf{F}}}\boldsymbol{R}\right\|_{\mathsf{F}}^2 \\
&= \inf_{\boldsymbol{R}\in\mathbf{SO}(d)}\left\|\frac{\overset{\mathsf{c}}{\boldsymbol{A}}}{\|\overset{\mathsf{c}}{\boldsymbol{A}}\|_{\mathsf{F}}} - \frac{\overset{\mathsf{c}}{\boldsymbol{B}}}{\|\overset{\mathsf{c}}{\boldsymbol{B}}\|_{\mathsf{F}}}\boldsymbol{R}\right\|_{\mathsf{F}}^2 . \qquad (1.22)
\end{aligned}$$

1.2.5.4 Isotropic error assumption

Implicit in the above shape distance calculations is the assumption of an isotropic error model when optimising the rotation (Section 1.2.5.1). Theobald and Wuttke (2006) points out that this commonly applied technique of minimising the sum of squared distances is not always robust when "errors have heterogeneous variances or the errors are correlated" and offers an alternative iterative algorithm to compute a more robust maximum likelihood estimation. In this work we have chosen to stay with the method of minimising the sum of squared distances. Initially this decision was made based on parsimony and the additional cost of calculating the maximum likelihood estimate. Ultimately the performance of the method has meant that we have not revisited this decision.

1.3 Proteins

1.3.1 Chemistry

A protein is a *linear polymer* made up of one or more *amino acid*s chained together. There are a set of twenty naturally occurring amino acid types from which all natural proteins are constructed. All amino acid types have the same backbone structure but are differentiated by their *sidechains*, and hence there are twenty distinct sidechain configurations that distinguish the naturally occurring amino acids. Structures of the 20 amino acids are illustrated in Figure 1.4 (Papachristodoulou et al. 2014, Chapter 4).

Protein amino acids always join together via their backbone and the amino acid sidechain is always connected to the backbone's central *alpha carbon*, C_α. Sidechain atoms are labelled with their chemical symbol and subscripted with Greek letters e.g. C_β, C_γ, S_δ, C_ϵ. The protein structure minus the sidechains is called the *polypeptide backbone*, the bonds between amino acids in the chain are known as *peptide bonds* and the amino acids making up the chain are referred to as the *residues*. Figure 1.5 illustrates the construction of the *polypeptide backbone*.

The polypeptide backbone forms an inscribed curve in three dimensional Euclidean space; the form of this curve is known as the protein's *folding pattern* or *fold*. Under standardised conditions of temperature and solvent a protein molecule with the same amino acid sequence will always achieve the same fold, known as the *native fold*; this is the lowest energy equilibrium structure for the standardised conditions. From here forward in the text *fold* and *native fold* will be used interchangeably. A good introduction to current understanding of the physical processes that lead to the achievement of the native fold can be found in Dill and Bromberg (2010).

The differing spatial configurations of amino acids constituting the protein are called *conformations*. The majority of the positional freedom of atoms in the polypeptide backbone comes from the rotational freedom along the inter atomic bond axes of the bonds connecting the backbone atoms; these rotation angles are called *conformation angles*. The conformation angles of the $\mathrm{N}-\mathrm{C}_\alpha$ and $\mathrm{C}_\alpha-\mathrm{C}$ bonds are not restricted by the bond structure and have complete freedom within *steric* limitations. The term steric refers to

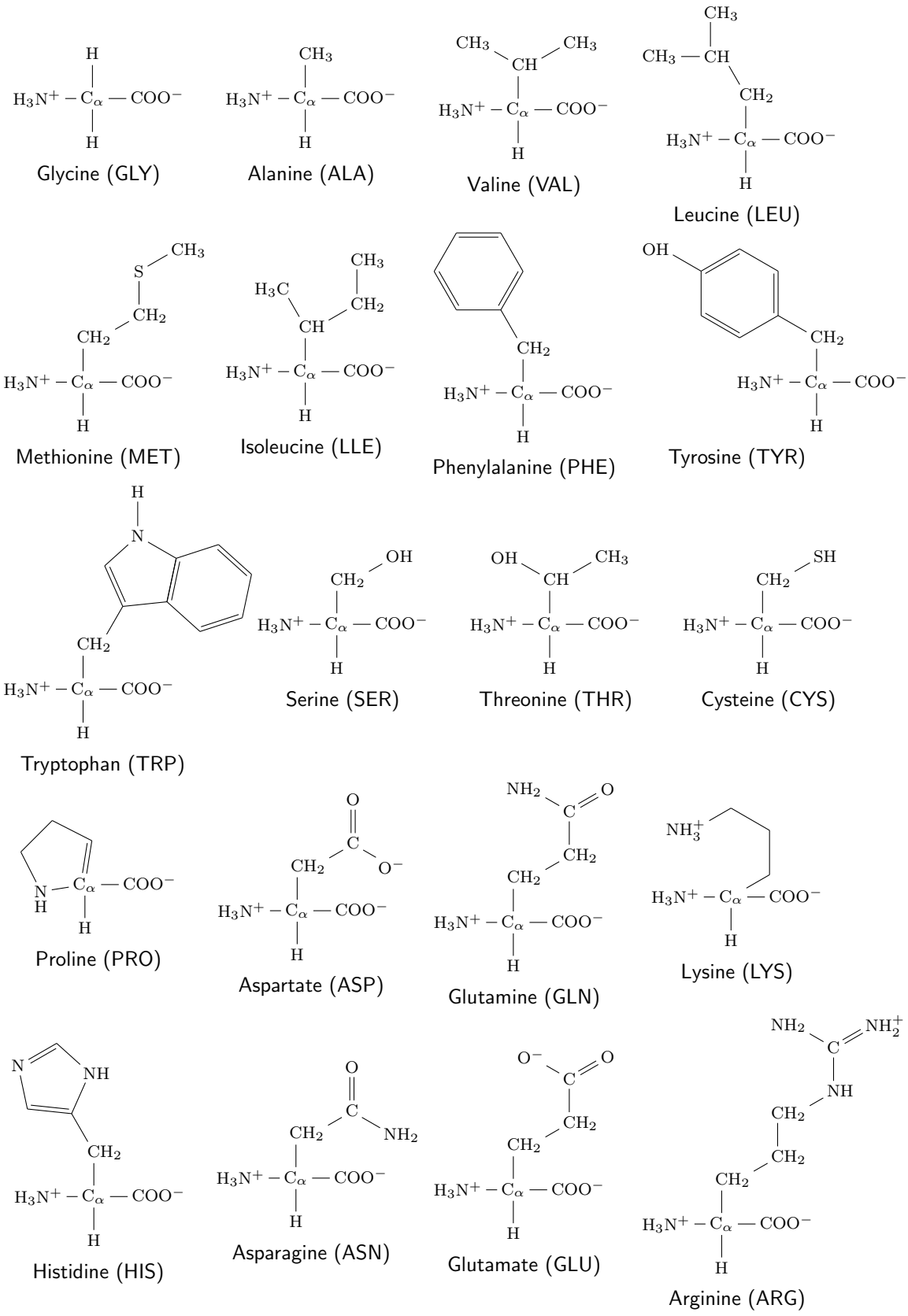


Figure 1.4: The 20 amino acids.

the energy cost associated with atoms being placed too close to each other; atoms can can be thought of as occupying a volume in space and so conformation angles that result in

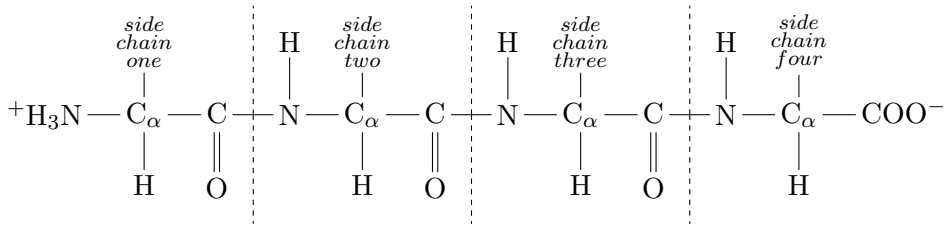


Figure 1.5: The construction of the polypeptide backbone that is common to all proteins.

these volumes overlapping are avoided. In addition to steric considerations the conformation angle of the C—N inter amino acid bond has structural characteristics that restrict its freedom to two main conformation angles of 0 and 180 degrees.

The twenty distinct amino acid sidechains also have some freedom in their spatial configurations, again mainly from rotation around bond axes. The differing spatial configurations of sidechain atoms specifically resulting from rotation along bond axes are called *rotamers*, short for rotational isomers.

Figure 1.6 shows the structure of the amino acid Phenylalanine and illustrates and labels the bond axial rotational degrees of freedom of both the backbone and sidechain atoms.

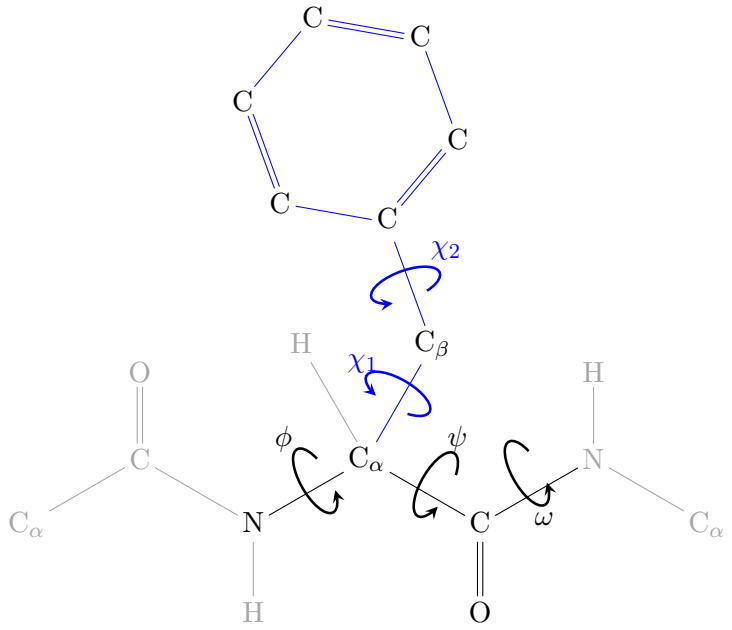


Figure 1.6: A representation of the structure of a Phenylalanine amino acid within a protein chain. The backbone bonds are shown in black and the sidechain bonds in blue. ϕ, ψ and ω are the backbone conformation angles. The rotamers of the sidechains are defined by the bond *torsion angles*, χ_n; hence, χ_1 and χ_2 are the sidechain conformation angles of Phenylalanine. Differing spatial configurations of the sidechain atoms resulting from variation in sidechain conformation atoms are called rotamers.

The physical interactions that lead to the final protein fold are not considered in this text as they are far too involved to look at in any detail; however, it is worth noting that in practice certain patterns of local structure tend to show up regularly. Specifically, sidechain rotamers tend to prefer certain conformations, this can be seen in the example in figure 1.7 which shows a random sample of twenty experimentally determined structures of the Phenylalanine amino acid taken from the Protein Data Bank. Additionally, certain spatial arrangements of

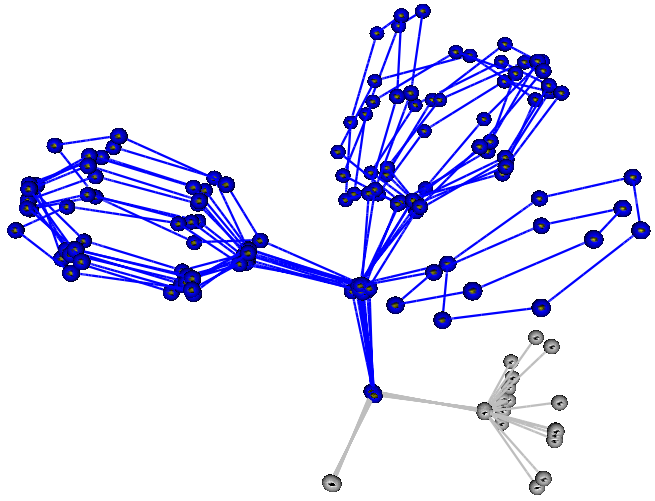


Figure 1.7: A plot of the experimentally determined atomic locations of a random sample of twenty Phenylalanine amino acids taken from the Protein Data Bank. The groupings indicate preferred low energy configurations of the sidechain atoms.

subsections of the polypeptide backbone occur regularly. These common arrangements are referred to as *secondary structure* and are one of the standard levels used when describing protein structure.

1.3.2 Describing protein structure

The important terminology for protein structure is now described. The term *sequence* when applied to proteins refers to an ordered one dimensional list of the constituent amino acids of the protein. A piece of structure formed from a single set of consecutive residues is called a *local* structure.

Primary Structure The amino acid sequence.

Secondary Structure Structural arrangements of the polypeptide backbone. Secondary structure types include α-helix, β-sheet and loops.

α-**helix** Consecutive residues forming a helix.

β-**sheet** Lateral interactions of two or more independent sets of residues; each set is formed of a strand of consecutive residues, but the strands may be widely separated within amino acid sequence. In general sheets are not flat. Central strands in the sheet have two neighbours and edge strands in the sheet have a single neighbour. There are various sub-types of β-sheet, these include **parallel** β-**sheet**, **antiparallel** β-**sheet**, β-**hairpin**, β-**barrel** and β-**bulge**.

loops The regions of the sequence that join the sheets and helices are called *loops*.

Supersecondary Structure Many proteins show similar patterns of structural interactions between the helices and sheets in the structure. When these patterns occur in *local* structures they are called *supersecondary structure*. Supersecondary structures lie somewhere between secondary and tertiary structure in the hierarchy. Common supersecondary structures include the α-**helix hairpin**, the β-**hairpin** and β-α-β **unit**.

Tertiary Structure Spatial arrangement of the secondary structure elements. There are common ways that elements of secondary structure pack together. *Folding pattern* is often used as a synonym for *tertiary structure*.

Quarternary Structure The spatial arrangement of *subunits*; a subunit being one of a curated set of common sequences. A protein made of a single subunit is called *monomeric*. A subunit in a larger protein that is a monomeric protein in its own right is called a *domain*. A protein made up of multiple domains is called a *modular protein*. Most domains only combine with a limited set of other domains, and certain domain combinations are very common.

There is no universally agreed mechanism for identifying domains within protein structures and so domain mappings differ between protein categorisation databases. Manually identified domains, such as those in SCOP, are considered to be more reliable than automatically identified domains (Holland et al. 2006) and in general the domains defined in SCOP tend to be bigger than those in other classification databases (Csaba et al. 2009).

In protein science *Similarity* refers to an objective measurement of resemblance, whereas *homology* includes inference that sequences have a common evolutionary ancestor i.e. homology is an inference from *similarity* – note that the use of the term homology in this

context is closer to the typical usage in biology rather than chemistry. Evolutionary mutation occurs both at the sequence and the domain level i.e. evolution seems to try out domain combinations as well as sequence combinations. Similar sequences *do* often produce similar structures, but the converse is very often not true; i.e. similar spatial structure is often found with very different sequence. Additionally similar function is often found in proteins which are both non-homologous and have no similar structure. The biologically active part of a protein structure is often a very small part of the whole, typically ten percent; however, labelling one part of the structure as active should not necessarily be taken to mean that the rest of the structure is unimportant scaffolding; for example energy reduction from bonds in the "scaffolding" may effectively subsidise increased energy configurations in the active parts.

1.4 Software and programming languages

The primary objective of the software implementations written as part of this work is to produce trustworthy results; for the results to be trustworthy the code must be tested, documented, parsimonious and available. To this end the proof of concept, reference implementations are written using scripting languages. Scripting languages offer readability and brevity, and rapid development times when compared to low level languages such as C; the readability along with the brevity also serve to reduce the number of coding errors. Software is written using the scripting languages `R` (R Core Team 2016) and Perl (Christiansen et al. 2012). All methodological work is implemented in `R`; this was chosen because of its familiarity to the statistics community and the reliability of the statistical functions. The primary reference used in understanding the internals of `R` was Chambers (2010). Processing and fetching protein data files is implemented using `Perl`; `Perl` is better suited to text processing and system tasks. All open source packages and scripts incorporated in, or used alongside, code written by the author are detailed in the code section of the bibliography.

1.5 Thesis structure

The core of the remainder of this thesis consists of three chapters.

Chapter 2 quantifies the complexity of the unlabelled partial matching problem in Section 2.2 and continues on in Sections 2.3 to review existing Bayesian techniques targeted at addressing unlabelled partial matching in the context of protein alignment. Section 2.4 describes a reference implementation of the method described in Green and Mardia (2006) having argued that Green and Mardia (2006) is representative of the spectrum of Bayesian techniques described in the literature. Finally in Section 2.7 we consider the sensitivity of the Green and Mardia (2006) method to the required prior information and also show that without favourable starting conditions that the method will not scale to proteins of typical size in the Protein Data Bank.

Chapter 3 details the development of a novel new algorithm which can be used to find both global and good local solutions to the unlabelled partial matching problem in the general protein case by applying constraints specific to the problem problem. In Section 3.2 we begin with a description of an approximate distribution of size-and-shape distance calculated between configurations with known labelling where the difference in configurations is modelled by spherical normal errors. Using this model, in Section 3.3 we develop a method to select a set of starting matching matrices which are taken forward into Section 3.4 where we extend the mechanism to iteratively find additional matches. This idea is further developed to produce a greedy algorithm that is able to reproduce the results from Green and Mardia (2006) and that we show is generally applicable to Protein Data Bank data. Finally in Section 3.5 we propose a difference measure that can be used to quantify the similarity between proteins.

In Chapter 4 we discuss in detail the structure of the SCOP2 database as it relates to querying category relationships, and detail the acquisition of data from the Protein Data Bank along with the criteria for data exclusion. We then continue to apply the new algorithm and difference measure proposed in Chapter 3 to protein atom co-ordinate data from the Protein Data Bank and quantify the success of the algorithm at reproducing the classifications from the SCOP2 categorisation database.

2

Recovering labelling using Bayesian Hierarchical Models

2.1 Overview

This chapter reviews existing methods based on Bayesian hierarchical models for determining the optimal labelling of unlabelled and optionally partially matching sets of landmarks. The motivation for this chapter is to understand the mechanism by which these methods achieve a solution, consider the applicability of the methods in the general case and then take this understanding forward to Chapter 3 where we offer a novel, related method for the mass comparison of protein structures.

Detailed discussion in this chapter is based on the method described in Green and Mardia (2006), but the author believes that similar arguments to those presented would also apply to the alternate models described in Dryden, Hirst, et al. (2007) and Schmidler (2007). It is also noted that for the elements of the discussion that focus on the number of steps taken by the Markov process to reach the optimal solution that it may have been more appropriate to consider an optimisation technique such as the Expectation Maximisation (EM) algorithm suggested in Green and Mardia (2006, Section 3.7) and Kent et al. (2010)

and Chui and Rangarajan (2000). This assertion is not refuted but our choice to consider MCMC methods is based on a preference for the availability of the posterior distribution as a diagnostic resource, our interest in minor modes (discussed in detail in Chapter 3), and a belief that the critical component of both approaches is the likelihood and that a particular form of the likelihood will influence both methodologies similarly.

The chapter is organised as follows: Section 2.2 quantifies the complexity of the unlabelled partial matching problem. Section 2.3 considers the formulation of the Green and Mardia (2006) method in detail. Section 2.4 looks at our reference implementation of the method from Green and Mardia (2006) and covers the reproduction of the results given in that paper. Section 2.5 considers sensitivity to required prior information and the information available to the likelihood when the process is far from the mode. These considerations are important when looking at the applicability of the method to unsupervised comparison of large protein data sets containing large protein structures. Finally in Section 2.7 we summarise the aspects of Green and Mardia (2006) that we will take forward to Chapter 3 where we develop a related method suitable for large scale unsupervised matching.

2.2 Quantification of the problem complexity

Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be two configurations of landmarks where $\boldsymbol{A}$ is a $m \times d$ matrix containing the co-ordinates of m landmarks in $\mathbb{R}^d$ and $\boldsymbol{B}$ is a $n \times d$ matrix containing the co-ordinates of n landmarks in $\mathbb{R}^d$. Let $\boldsymbol{a}_j$ be the jth landmark co-ordinates in $\boldsymbol{A}$ and let $\boldsymbol{b}_k$ be the kth landmark co-ordinates in $\boldsymbol{B}$. The order of the landmarks in $\boldsymbol{A}$ and $\boldsymbol{B}$ hold no significance other than that their indices are used to identify the landmarks.

Labelling is a property ascribed between distinct sets of landmarks that describes correspondence between individual landmarks; that is, landmarks in the distinct configurations that have the same label are "labelled" as being matched. The labelling between configurations $\boldsymbol{A}$ and $\boldsymbol{B}$ can be encapsulated in a $m \times n$ *matching matrix*, $\boldsymbol{M}$, such that

$$M_{j,k} = \begin{cases} 1 & \text{if } \boldsymbol{a}_j \text{ matches } \boldsymbol{b}_k \\ 0 & \text{otherwise} \end{cases} \qquad \text{where} \quad j \in \{1, \ldots, m\} \text{ and } k \in \{1, \ldots, n\} \, .$$

In what follows the additional restriction is applied that a landmark in $\boldsymbol{A}$ can at most be

matched to one landmark in $\boldsymbol{B}$ and vice versa. This implies that

$$\begin{aligned} \sum_{j=1}^{m} M_{j,k} &\in \{0,1\} \quad \text{when } k \text{ is fixed} \\ \sum_{k=1}^{n} M_{j,k} &\in \{0,1\} \quad \text{when } j \text{ is fixed} \end{aligned} \tag{2.1}$$

and so the total number of matched landmarks is given by

$$L = \sum_{j=1}^{m} \sum_{k=1}^{n} M_{j,k} \ .$$

Without loss of generality $\boldsymbol{A}$ and $\boldsymbol{B}$ are assigned such that $m \leq n$, also implying that $L \leq m$. Considering the total number of possible matching matrices:

$L = 0$ There is one matching matrix where there are no matches.

$L = 1$ There are n available column positions in the row that contains the match and m possible rows, giving nm possible matching matrices.

$L = 2$ There are n available column positions for the first row that contains a match. The second row that contains a match then has $n - 1$ available column positions as a result of the conditions in (2.1). There are $\binom{m}{2}$ ways to choose the first and second rows giving $\binom{m}{2} n(n-1)$ possible matching matrices.

$L = 3$ There are n available column positions for the first row that contains a match, $n-1$ available column positions for the second row that contains a match, and $n-2$ available column positions for the third row that contains a match. There are $\binom{m}{3}$ ways to choose the first, second and third rows giving $\binom{m}{3} n(n-1)(n-2)$ possible matching matrices.

In general for L matches the possible number of matching matrices is given by

$$\begin{aligned} S_L &= \binom{m}{L} \, n \, (n-1) \, (n-2) \, \ldots \, (n-(L-1)) \\ &= \binom{m}{L} \, \frac{n!}{(n-L)!} \\ &= L! \binom{m}{L} \binom{n}{L} \qquad \text{using} \qquad \binom{n}{L} = \frac{n!}{L!(n-L)!} \ . \end{aligned} \tag{2.2}$$

Hence the number of all possible matching matrices is

$$S = \sum_{L=0}^{m} L! \binom{m}{L} \binom{n}{L} . \tag{2.3}$$

As is demonstrated by the tables in Figure 2.1 the size of the space of all possible matching matrices gets very large even for relatively modest values of m and n.

L	0	1	2	3	4	5	total
S_L	1	25	200	600	600	120	1546

(a) $m = 5$, $n = 5$

L	0	1	2	3	4	5	6	7	8	9	10	total
S_L	1	100	4,050	86,400	1,058,400	7,620,480	31,752,000	72,576,000	81,648,000	36,288,000	3,628,800	234,662,231

(b) $m = 10$, $n = 10$

L	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	total
S_L	1	225	22050	1.2e06	4.5e07	1.1e09	1.8e10	2.1e11	1.7e12	9.1e12	3.3e13	7.4e13	9.9e13	6.9e13	2.0e13	1.3e12	3.1e14

(c) $m = 15$, $n = 15$

Figure 2.1: Illustrations of the size of the space of matching matrices for various small values of m and n, using (2.2) and (2.3).

2.3 Green and Mardia (2006) in detail

Interest is in drawing inference about the matching matrix and the translation and rotation of the configurations to achieve registration are considered nuisance parameters. The two main approaches to dealing with these nuisance parameters are represented by Green and Mardia (2006) and Dryden, Hirst, et al. (2007). Green and Mardia (2006) use prior distributions on the rotation and translation parameters to obtain a posterior distribution over the transformations and the matching matrix and then the marginal posterior for the matching matrix is obtained by integrating out the nuisance transformation. Dryden, Hirst, et al. (2007) minimise the shape distance using Procrustes registration; the Dryden, Hirst, et al. (2007) method will not be discussed in any detail but will be considered later in Section 2.5 when the domain of applicability of the Green and Mardia (2006) method is discussed.

We consider Green and Mardia (2006) for the comparison of configurations of landmarks in $\mathbb{R}^3$. Green and Mardia (2006) use a hierarchical Bayesian model

$$\pi\left(\boldsymbol{M}, \boldsymbol{\Gamma}, \boldsymbol{\tau}, \sigma \mid \boldsymbol{A}, \boldsymbol{B}\right) \propto \mathrm{L}\left(\boldsymbol{A}, \boldsymbol{B} \mid \boldsymbol{M}, \boldsymbol{\Gamma}, \boldsymbol{\tau}, \sigma\right) \pi(\boldsymbol{M}) \, \pi(\boldsymbol{\Gamma}) \, \pi(\boldsymbol{\tau}) \, \pi(\sigma) \tag{2.4}$$

where $\boldsymbol{M}$ is a matching matrix, $\boldsymbol{\Gamma} \in \mathrm{SO}(3)$, $\boldsymbol{\tau} = (\tau_x, \tau_y, \tau_z)^{\mathsf{T}}$ is a translation, and σ is a standard deviation related to the error in the location of both sets of landmarks. The prior distributions of $\boldsymbol{M}$, $\boldsymbol{\Gamma}$, $\boldsymbol{\tau}$ and σ are assumed to be independent. In Green and Mardia (2006) $\boldsymbol{\Gamma}$ is initially generalised to an affine transformation. In the model that follows we assume the final simplifying assumptions from Green and Mardia (2006) from the outset; namely that, $\boldsymbol{\Gamma}$ is a member of the special orthogonal group, and that the data are three dimensional landmarks.

Sampling of the posterior distribution is achieved using MCMC with a mixture of Gibbs and Metropolis-Hastings steps. It is assumed that for all but the matching matrix $\boldsymbol{M}$ the prior information is weak and weakly informative prior distributions are chosen for $\boldsymbol{\Gamma}$, $\boldsymbol{\tau}$ and σ to simplify posterior sampling. The final model is given as

$$\pi\left(\boldsymbol{M}, \boldsymbol{\Gamma}, \boldsymbol{\tau}, \sigma \mid \boldsymbol{A}, \boldsymbol{B}\right) \propto \pi(\boldsymbol{\Gamma})\, \pi(\boldsymbol{\tau})\, \pi(\sigma) \prod_{\substack{j,k: \\ M_{j,k}=1}} \left[\frac{\rho}{\lambda \sigma^3}\, \phi_3\left(\frac{\boldsymbol{a}_j - \boldsymbol{\Gamma}\boldsymbol{b}_k - \boldsymbol{\tau}}{\sigma\sqrt{2}}\right)\right]$$

where $\phi_3\left(\cdot\right)$ is a standard normal density in $\mathbb{R}^3$ with mean $\boldsymbol{\mu} = \boldsymbol{0}$ and standard deviation $\boldsymbol{\Sigma} = \boldsymbol{I}_3$. The prior distributions of $\boldsymbol{\Gamma}$, $\boldsymbol{\tau}$ and σ are chosen to be the *matrix Fisher*, *multivariate normal* and *inverse gamma* respectively. The resulting full conditionals of $\boldsymbol{\tau}$ and σ are *multivariate normal* and *inverse gamma* respectively, with $\boldsymbol{\Gamma}$ represented by a zyx *Euler Angle-axis Sequence*. The full conditionals of θ_x and θ_z are von Mises distributions. The full conditionals for θ_x, θ_z, $\boldsymbol{\tau}$ and σ are completely known and so Gibbs sampling is used. $\boldsymbol{M}$ and θ_y are sampled using Metropolis-Hastings.

2.3.1 The model

The following model is the same as that described by Green and Mardia (2006). The sets of landmarks $\{\boldsymbol{a}_j\}$ and $\{\boldsymbol{b}_k\}$ are considered to be noisy observations on a set of true locations $\{\boldsymbol{\mu}_i\}$. The observation errors are assumed to be spherical normal. Each true location, $\boldsymbol{\mu}_i$, can be represented at most one time in each of $\{\boldsymbol{a}_j\}$ and $\{\boldsymbol{b}_k\}$. The mappings of the indices j and k to i are unknown and represented by $\{\xi_j\}$ and $\{\eta_k\}$ respectively.

$\{\boldsymbol{a}_j\}$ and $\{\boldsymbol{\mu}_i\}$ occupy the same space. $\{\boldsymbol{b}_k\}$ occupies a space that is related to the $\{\boldsymbol{\mu}_i\}$ by a rigid body transformation of a rotation, $\boldsymbol{\Gamma} \in \mathrm{SO}(3)$ and a translation $\boldsymbol{\tau} = (\tau_x, \tau_y, \tau_z)$.

The relationships between $\{\boldsymbol{a}_j\}$, $\{\boldsymbol{b}_k\}$ and $\{\boldsymbol{\mu}_i\}$ are explicitly given by

$$\begin{aligned} \boldsymbol{a}_j &= \boldsymbol{\mu}_{\xi_j} + \boldsymbol{\epsilon}_j \\ \boldsymbol{\Gamma}\boldsymbol{b}_k + \boldsymbol{\tau} &= \boldsymbol{\mu}_{\eta_k} + \boldsymbol{\epsilon}_k \end{aligned} \tag{2.5}$$

where $\boldsymbol{\epsilon}_j, \boldsymbol{\epsilon}_k \sim \mathrm{N}_3(\boldsymbol{0}, \sigma^2 \boldsymbol{I})$.

The matching relationship of landmarks in $\{\boldsymbol{a}_j\}$ and $\{\boldsymbol{b}_k\}$ is represented in terms of ξ_j and η_k such that

$$M_{j,k} = \begin{cases} 1 & \text{if } \xi_j = \eta_k \\ 0 & \text{otherwise} \end{cases}$$

which is exactly equivalent to that described in Section 2.2 above.

It is assumed that N true locations of the landmarks $\{\boldsymbol{\mu}_i\}$ are realised by a homogeneous spatial Poisson process of intensity λ over some region with volume V in $\mathbb{R}^3$. There are four possibilities for each realisation of a true location: neither an $\boldsymbol{a}_j$ nor a $\boldsymbol{b}_k$ matches the $\boldsymbol{\mu}_i$, only an $\boldsymbol{a}_j$ matches the $\boldsymbol{\mu}_i$, only a $\boldsymbol{b}_k$ matches the $\boldsymbol{\mu}_i$, or both an $\boldsymbol{a}_j$ and a $\boldsymbol{b}_k$ match the $\boldsymbol{\mu}_i$. It is assumed that these four possibilities occur independently and that the probabilities of each possibility are parameterised respectively as $1 - p_a - p_b - \rho p_a p_b$, p_a, p_b and $\rho p_a p_b$ where ρ is some measure of the tendency for landmarks to match. The number of realisations of each of the four possible outcomes are assumed to be independent Poisson random variables with counts of $N - m - n + L$, $m - L$, $n - L$ and L, and means of $\lambda V(1 - p_a - p_b - \rho p_a p_b)$, $\lambda V p_a$, $\lambda V p_b$ and $\lambda V \rho p_a p_b$ respectively, with L being the total number of events where both an $\boldsymbol{a}_j$ and a $\boldsymbol{b}_k$ match a particular $\boldsymbol{\mu}_i$. The count for neither an $\boldsymbol{a}_j$ nor a $\boldsymbol{b}_k$ matches the $\boldsymbol{\mu}_i$ comes from $N - m - n + L = N - (m - l) - (n - L) - L$.

The conditional distribution of L and N is therefore proportional to the product of four

Poisson densities parameterised as follows

$$\begin{aligned}\pi\left(L, N \mid m, n, \lambda, V, \rho, p_a, p_b\right) \propto & \\ & \frac{e^{-\lambda V(1-p_a-p_b-\rho p_a p_b)}(\lambda V(1-p_a-p_b-\rho p_a p_b))^{N-m-n+L}}{(N-m-n+L)!} \\ \times & \frac{e^{-\lambda V p_a}(\lambda V p_a)^{m-L}}{(m-L)!} \\ \times & \frac{e^{-\lambda V p_b}(\lambda V p_b)^{n-L}}{(n-L)!} \\ \times & \frac{e^{-\lambda V \rho p_a p_b}(\lambda V \rho p_a p_b)^{L}}{L!} .\end{aligned}$$

Only the first term in the product contains N, which can be integrated out as follows

$$\begin{aligned}\sum_{N=m+n-L}^{\infty} \frac{e^{-k} k^{(N-m-n+L)}}{(N-m-n+L)!} &= e^{-k}\left(\frac{k^0}{0!}+\frac{k^1}{1!}+\frac{k^2}{2!}+\ldots\right) \qquad \text{where } k=\lambda V(1-p_a-p_b-\rho p_a p_b) \\ &= e^{-k}\sum_{n=0}^{\infty}\frac{k^n}{n!} = e^{-k}e^{k} = 1 \qquad \text{(Taylor expansion)}\end{aligned}$$

leaving the conditional distribution of L as

$$\begin{aligned}\pi\left(L \mid m, n, \lambda, V, \rho, p_a, p_b\right) &\propto \frac{e^{-\lambda V p_a}(\lambda V p_a)^{m-L}}{(m-L)!} \times \frac{e^{-\lambda V p_b}(\lambda V p_b)^{n-L}}{(n-L)!} \times \frac{e^{-\lambda V \rho p_a p_b}(\lambda V \rho p_a p_b)^{L}}{L!} \\ &\propto \frac{(\lambda V p_a)^{m-L}}{(m-L)!} \times \frac{(\lambda V p_b)^{n-L}}{(n-L)!} \times \frac{(\lambda V \rho p_a p_b)^{L}}{L!} \\ &\propto \frac{(\lambda V p_a)^{-L}(\lambda V p_b)^{-L}(\lambda V \rho p_a p_b)^{L}}{(m-L)!(n-L)!L!} \\ &\propto \frac{\left(\frac{\rho}{\lambda V}\right)^{L}}{(m-L)!(n-L)!L!} . \qquad (2.6)\end{aligned}$$

Now considering (2.2)

$$\begin{aligned}L!\binom{m}{L}\binom{n}{L} &= L! \times \frac{m!}{L!(m-L)!} \times \frac{n!}{L!(n-L)!} \\ &= \frac{m!n!}{L!(m-L)!(n-L)!} ,\end{aligned}$$

then

$$\pi\left(L \mid m, n, \lambda, V, \rho, p_a, p_b\right) \propto L!\binom{m}{L}\binom{n}{L}\left(\frac{\rho}{\lambda V}\right)^{L}$$

and so

$$\pi\left(L \mid m, n, \lambda, V, \rho, p_a, p_b\right) = \frac{L!\binom{m}{L}\binom{n}{L}\left(\frac{\rho}{\lambda V}\right)^{L}}{\sum_{l=0}^{m}\{l!\binom{m}{l}\binom{n}{l}\left(\frac{\rho}{\lambda V}\right)^{l}\}} .$$

Using the property that each row and column of $\overset{m\times n}{\boldsymbol{M}}$ must sum to 1 or 0 (2.1), for a given L there will be $L!\binom{m}{L}\binom{n}{L}$ possible $\boldsymbol{M}$s consistent with L matches. Assuming each of these possible $\boldsymbol{M}$s is equally likely

$$\pi(\boldsymbol{M}|L) = \left\{ L!\binom{m}{L}\binom{n}{L} \right\}^{-1}$$

and so assuming conditional on m, n, λ, V, ρ, p_a and p_b

$$\begin{aligned}
\pi(\boldsymbol{M}) = \pi(\boldsymbol{M}, L) &= \pi(\boldsymbol{M}|L)\ \pi(L) \\
&= \frac{1}{L!\binom{m}{L}\binom{n}{L}} \times \frac{L!\binom{m}{L}\binom{n}{L}\ \left(\frac{\rho}{\lambda V}\right)^L}{\sum_{l=0}^{m}\{l!\binom{m}{L}\binom{n}{L}\ \left(\frac{\rho}{\lambda V}\right)^l\}} \\
&= \frac{\left(\frac{\rho}{\lambda V}\right)^L}{\sum_{l=0}^{m}\{l!\binom{m}{L}\binom{n}{L}\ \left(\frac{\rho}{\lambda V}\right)^l\}}\ . \qquad (2.7)
\end{aligned}$$

When MCMC is applied to a model where the dimensionality of the unknowns is itself an unknown, or as Peter Green concisely put it "the number of things you don't know is one of the things you don't know"[1], then an extension to Metropolis-Hastings is required. The usual MCMC approach to this type of trans-dimensional problems is called *reversible jump MCMC* and is presented in Green (1995). The role of the matching matrix in the Green and Mardia (2006) model mediating the addition and removal of hidden point locations suggests the need for a reversible jump MCMC consideration, however this is not required for the reason detailed in Green and Mardia (2006, Section 3.1)

> Our model has another similarity with a mixture formulation, in that as M varies, the number of hidden points needed to generate all the observed data also varies, and thus there seems to be a 'variable-dimension' aspect to the model. However, here our approach of integrating out the hidden point locations eliminates the variable-dimension parameter, so that reversible jump MCMC is not needed.

[1] Green 2003.

2.3.2 Likelihood

The likelihood contribution of the landmarks in $\boldsymbol{A}$ and $\boldsymbol{B}$ given a particular $\boldsymbol{M}$ are derived using the relationship given in (2.5) which is repeated here for convenience

$$\begin{aligned} \boldsymbol{a}_j &= \boldsymbol{\mu}_{\xi_j} + \boldsymbol{\epsilon}_j \qquad \text{where} \quad \boldsymbol{\epsilon}_j \sim \mathrm{N}_3(\boldsymbol{0}, \sigma^2 \boldsymbol{I}) \\ \boldsymbol{\Gamma}\boldsymbol{b}_k + \boldsymbol{\tau} &= \boldsymbol{\mu}_{\eta_k} + \boldsymbol{\epsilon}_k \qquad \text{where} \quad \boldsymbol{\epsilon}_k \sim \mathrm{N}_3(\boldsymbol{0}, \sigma^2 \boldsymbol{I}) \ . \end{aligned}$$

The densities for non-matching landmarks – that is when $M_{j,k} = 0$ which is equivalent to when a $\boldsymbol{\mu}_i$ is either matched by a $\boldsymbol{a}_j$ or a $\boldsymbol{b}_k$ but not both – are: for a $\boldsymbol{\mu}_i$ that is only matched by an $\boldsymbol{a}_j$

$$\mathrm{f}\left(\boldsymbol{a}_j - \boldsymbol{\mu}_{\xi_j}\right)$$

and for a $\boldsymbol{\mu}_i$ that is only matched by an $\boldsymbol{b}_k$

$$\mathrm{f}\left(\boldsymbol{\Gamma}\boldsymbol{b}_k + \boldsymbol{\tau} - \boldsymbol{\mu}_{\eta_k}\right) \ .$$

The densities for matched landmarks – that is when $M_{j,k} = 1$ which is equivalent to when a $\boldsymbol{\mu}_i$ is matched by both an $\boldsymbol{a}_j$ and a $\boldsymbol{b}_k$ – are

$$\mathrm{f}\left(\boldsymbol{a}_j - \boldsymbol{\mu}_{\xi_j}\right) \mathrm{f}\left(\boldsymbol{\Gamma}\boldsymbol{b}_k + \boldsymbol{\tau} - \boldsymbol{\mu}_{\eta_k}\right) \ .$$

Where in each case $\mathrm{f}(\cdot)$ is a trivariate normal, parameterised by σ, such that

$$\mathrm{f}(\boldsymbol{x}) \ = \ \frac{1}{\sigma^3}\, \boldsymbol{\phi}_3\left(\frac{\boldsymbol{x}}{\sigma}\right) \ = \ \frac{1}{\sigma}\, \phi\left(\frac{x_1}{\sigma}\right) \times \frac{1}{\sigma}\, \phi\left(\frac{x_2}{\sigma}\right) \times \frac{1}{\sigma}\, \phi\left(\frac{x_3}{\sigma}\right)$$

where $\phi(\cdot)$ is a univariate standard normal.

The combined likelihood for all matched and non matched landmarks given a particular $\boldsymbol{M}$

is then given by

$$\begin{aligned}\mathrm{L}(\boldsymbol{a},\boldsymbol{b}\mid \boldsymbol{M},\boldsymbol{\Gamma},\boldsymbol{\tau},\sigma) = & \prod_{\substack{j:\\ M_{j,k}=0}} \frac{1}{V}\int_V \mathrm{f}\left(\boldsymbol{a}_j-\boldsymbol{\mu}_{\xi_j}\right)\mathrm{d}\boldsymbol{\mu} \\ & \times \prod_{\substack{k:\\ M_{j,k}=0}} \frac{1}{V}\int_V \mathrm{f}\left(\boldsymbol{\Gamma}\boldsymbol{b}_k+\boldsymbol{\tau}-\boldsymbol{\mu}_{\eta_k}\right)\mathrm{d}\boldsymbol{\mu} \\ & \times \prod_{\substack{j,k:\\ M_{j,k}=1}} \frac{1}{V}\int_V \mathrm{f}\left(\boldsymbol{a}_j-\boldsymbol{\mu}_{\xi_j}\right)\mathrm{f}\left(\boldsymbol{\Gamma}\boldsymbol{b}_k+\boldsymbol{\tau}-\boldsymbol{\mu}_{\eta_k}\right)\mathrm{d}\boldsymbol{\mu}\ .\end{aligned}$$

Assuming that V is large the integrals of the densities in the first and second terms can be approximated to one such that

$$\mathrm{L}(\boldsymbol{a},\boldsymbol{b}\mid \boldsymbol{M},\boldsymbol{\Gamma},\boldsymbol{\tau},\sigma) \approx \left(\frac{1}{V}\right)^{m-L}\left(\frac{1}{V}\right)^{n-L}\left(\frac{1}{V}\right)^{L}\prod_{\substack{j,k:\\ M_{j,k}=1}}\int_V \mathrm{f}\left(\boldsymbol{a}_j-\boldsymbol{\mu}_{\xi_j}\right)\mathrm{f}\left(\boldsymbol{\Gamma}\boldsymbol{b}_k+\boldsymbol{\tau}-\boldsymbol{\mu}_{\eta_k}\right)\mathrm{d}\boldsymbol{\mu}\ . \tag{2.8}$$

Considering the third term of (2.8):

Let

$$\begin{aligned}\boldsymbol{z} &= (\boldsymbol{a}_j-\boldsymbol{\mu})-(\boldsymbol{\Gamma}\boldsymbol{b}_k+\boldsymbol{\tau}-\boldsymbol{\mu}) = \boldsymbol{a}_j-\boldsymbol{\Gamma}\boldsymbol{b}_k-\boldsymbol{\tau} \\ \boldsymbol{u} &= \boldsymbol{\Gamma}\boldsymbol{b}_k+\boldsymbol{\tau}-\boldsymbol{\mu}\ .\end{aligned} \tag{2.9}$$

Hence

$$\boldsymbol{z}+\boldsymbol{u} = (\boldsymbol{a}_j-\boldsymbol{\Gamma}\boldsymbol{b}_k-\boldsymbol{\tau})+(\boldsymbol{\Gamma}\boldsymbol{b}_k+\boldsymbol{\tau}-\boldsymbol{\mu}) = \boldsymbol{a}_j-\boldsymbol{\mu}$$

and

$$\begin{aligned}\boldsymbol{\mu} &= \boldsymbol{\Gamma}\boldsymbol{b}_k+\boldsymbol{\tau}-\boldsymbol{u} \\ \mathrm{d}\boldsymbol{\mu} &= -\,\mathrm{d}\boldsymbol{u}\ .\end{aligned}$$

Note that $\boldsymbol{z}$ is independent of $\boldsymbol{\mu}$.

Now assuming that V is large enough to ignore edge effects

$$\begin{aligned}\int_{\mathbb{R}^3}\mathrm{f}\left(\boldsymbol{a}_j-\boldsymbol{\mu}_{\xi_j}\right)\mathrm{f}\left(\boldsymbol{\Gamma}\boldsymbol{b}_k+\boldsymbol{\tau}-\boldsymbol{\mu}_{\eta_k}\right)\mathrm{d}\boldsymbol{\mu} &= \int_{\mathbb{R}^3}\frac{1}{\sigma^3}\boldsymbol{\phi}_3\left(\frac{\boldsymbol{a}_j-\boldsymbol{\mu}}{\sigma}\right)\frac{1}{\sigma^3}\boldsymbol{\phi}_3\left(\frac{\boldsymbol{\Gamma}\boldsymbol{b}_k+\boldsymbol{\tau}-\boldsymbol{\mu}}{\sigma}\right)\mathrm{d}\boldsymbol{\mu} \\ &= -\int_{\mathbb{R}^3}\frac{1}{\sigma^3}\boldsymbol{\phi}_3\left(\frac{\boldsymbol{z}+\boldsymbol{u}}{\sigma}\right)\frac{1}{\sigma^3}\boldsymbol{\phi}_3\left(\frac{\boldsymbol{u}}{\sigma}\right)\mathrm{d}\boldsymbol{u}\ .\end{aligned}$$

The implicit diagonal covariance matrix implies that

$$\begin{aligned}\int_{\mathbb{R}^3} \frac{1}{\sigma^3}\,\boldsymbol{\phi}_3\left(\frac{\boldsymbol{z}+\boldsymbol{u}}{\sigma}\right)\frac{1}{\sigma^3}\,\boldsymbol{\phi}_3\left(\frac{\boldsymbol{u}}{\sigma}\right)\mathrm{d}\boldsymbol{u}&\\ = \frac{1}{\sigma^6}\int_{\mathbb{R}}\int_{\mathbb{R}}\int_{\mathbb{R}}\phi\left(\frac{z_1+u_1}{\sigma}\right)\phi\left(\frac{z_2+u_2}{\sigma}\right)\phi\left(\frac{z_3+u_3}{\sigma}\right)&\\ \phi\left(\frac{u_1}{\sigma}\right)\phi\left(\frac{u_2}{\sigma}\right)\phi\left(\frac{u_3}{\sigma}\right)\,\mathrm{d}u_1\,\mathrm{d}u_2\,\mathrm{d}u_3\,.&\end{aligned}$$

Now

$$\begin{aligned}&\int_{\mathbb{R}}\frac{1}{\sigma}\,\phi\left(\frac{z_1+u_1}{\sigma}\right)\,\frac{1}{\sigma}\,\phi\left(\frac{u_1}{\sigma}\right)\,\mathrm{d}u_1\\ &\quad= \frac{1}{2\pi\sigma^2}\int_{\mathbb{R}}\exp\left\{-\frac{1}{2\sigma^2}\left(z_1+u_1\right)^2\right\}\,\exp\left\{-\frac{1}{2\sigma^2}u_1^2\right\}\mathrm{d}u_1\\ &\quad= \frac{1}{2\pi\sigma^2}\int_{\mathbb{R}}\exp\left\{-\frac{1}{2\sigma^2}\left(z_1^2+2u_1^2+2z_1u_1\right)\right\}\mathrm{d}u_1\\ &\quad= \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left\{-\frac{1}{2\sigma^2}\left(\frac{z_1}{\sqrt{2}}\right)^2\right\}\int_{\mathbb{R}}\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left\{-\frac{1}{2\sigma^2}\left(u_1\sqrt{2}+\frac{z_1}{\sqrt{2}}\right)^2\right\}\mathrm{d}u_1\\ &\quad= \frac{1}{\sqrt{2}}\,\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left\{-\frac{1}{2\sigma^2}\left(\frac{z_1}{\sqrt{2}}\right)^2\right\}\\ &\quad= \frac{1}{\sigma\sqrt{2}}\,\phi\left(\frac{z_1}{\sigma\sqrt{2}}\right)\,.\end{aligned}$$

Hence

$$\begin{aligned}\int_V \mathrm{f}\left(\boldsymbol{a}_j-\boldsymbol{\mu}_{\xi_j}\right)\mathrm{f}\left(\boldsymbol{\Gamma}\boldsymbol{b}_k+\boldsymbol{\tau}-\boldsymbol{\mu}_{\eta_k}\right)\mathrm{d}\boldsymbol{\mu} &= \left(\frac{1}{\sigma\sqrt{2}}\right)^3\boldsymbol{\phi}_3\left(\frac{\boldsymbol{z}}{\sigma\sqrt{2}}\right)\\ &= \left(\frac{1}{\sigma\sqrt{2}}\right)^3\boldsymbol{\phi}_3\left(\frac{\boldsymbol{a}_j-\boldsymbol{\Gamma}\boldsymbol{b}_k-\boldsymbol{\tau}}{\sigma\sqrt{2}}\right)\,. \qquad (2.10)\end{aligned}$$

Substituting (2.10) into (2.8)

$$\mathrm{L}(\boldsymbol{a},\boldsymbol{b}\mid\boldsymbol{M},\boldsymbol{\Gamma},\boldsymbol{\tau},\sigma)\approx\left(\frac{1}{V}\right)^{m+n-L}\prod_{\substack{j,k:\\ M_{j,k}=1}}\left[\left(\frac{1}{\sigma\sqrt{2}}\right)^3\boldsymbol{\phi}_3\left(\frac{\boldsymbol{a}_j-\boldsymbol{\Gamma}\boldsymbol{b}_k-\boldsymbol{\tau}}{\sigma\sqrt{2}}\right)\right]\,. \qquad (2.11)$$

In order to substitute this likelihood into the Bayesian model, (2.4), note that

$$\begin{aligned}\pi\left(\boldsymbol{M},\boldsymbol{\Gamma},\boldsymbol{\tau},\sigma \mid \boldsymbol{A},\boldsymbol{B}\right) &\propto \mathrm{L}\left(\boldsymbol{A},\boldsymbol{B} \mid \boldsymbol{M},\boldsymbol{\Gamma},\boldsymbol{\tau},\sigma\right)\pi(\boldsymbol{M})\,\pi(\boldsymbol{\Gamma})\,\pi(\boldsymbol{\tau})\,\pi(\sigma)\\ &\propto \frac{\mathrm{L}\left(\boldsymbol{A},\boldsymbol{B},\boldsymbol{M} \mid \boldsymbol{\Gamma},\boldsymbol{\tau},\sigma\right)}{\pi(\boldsymbol{M})}\,\pi(\boldsymbol{M})\,\pi(\boldsymbol{\Gamma})\,\pi(\boldsymbol{\tau})\,\pi(\sigma)\\ &\propto \mathrm{L}\left(\boldsymbol{A},\boldsymbol{B},\boldsymbol{M} \mid \boldsymbol{\Gamma},\boldsymbol{\tau},\sigma\right)\pi(\boldsymbol{\Gamma})\,\pi(\boldsymbol{\tau})\,\pi(\sigma)\ ,\end{aligned}$$

and that multiplying (2.7) and (2.11) we get

$$\mathrm{L}(\boldsymbol{a},\boldsymbol{b},\boldsymbol{M} \mid \boldsymbol{\Gamma},\boldsymbol{\tau},\sigma) \propto \prod_{\substack{j,k:\\ M_{j,k}=1}} \left[\left(\frac{\rho}{\lambda}\right)\left(\frac{1}{\sigma\sqrt{2}}\right)^{3}\boldsymbol{\phi}_3\left(\frac{\boldsymbol{a}_j-\boldsymbol{\Gamma}\boldsymbol{b}_k-\boldsymbol{\tau}}{\sigma\sqrt{2}}\right)\right]\ . \tag{2.12}$$

Hence the Bayesian model, (2.4), becomes

$$\begin{aligned}\pi\left(\boldsymbol{M},\boldsymbol{\Gamma},\boldsymbol{\tau},\sigma \mid \boldsymbol{A},\boldsymbol{B}\right) &\propto \mathrm{L}\left(\boldsymbol{A},\boldsymbol{B} \mid \boldsymbol{M},\boldsymbol{\Gamma},\boldsymbol{\tau},\sigma\right)\pi(\boldsymbol{M})\,\pi(\boldsymbol{\Gamma})\,\pi(\boldsymbol{\tau})\,\pi(\sigma)\\ &\propto \mathrm{L}\left(\boldsymbol{a},\boldsymbol{b},\boldsymbol{M} \mid \boldsymbol{\Gamma},\boldsymbol{\tau},\sigma\right)\pi(\boldsymbol{\Gamma})\,\pi(\boldsymbol{\tau})\,\pi(\sigma)\\ &\propto \pi(\boldsymbol{\Gamma})\,\pi(\boldsymbol{\tau})\,\pi(\sigma)\prod_{\substack{j,k:\\ M_{j,k}=1}} \left[\left(\frac{\rho}{\lambda}\right)\left(\frac{1}{\sigma\sqrt{2}}\right)^{3}\boldsymbol{\phi}_3\left(\frac{\boldsymbol{a}_j-\boldsymbol{\Gamma}\boldsymbol{b}_k-\boldsymbol{\tau}}{\sigma\sqrt{2}}\right)\right]\ .\end{aligned} \tag{2.13}$$

2.3.3 Prior distributions and sampling conditional posterior distributions

Prior knowledge of $\boldsymbol{\Gamma}$, σ and $\boldsymbol{\tau}$ is assumed to be weak and so weakly informative priors are chosen which facilitate posterior analysis.

2.3.3.1 Rotation

Considering the prior on the rotation $\boldsymbol{\Gamma}$, from (2.13) the full conditional distribution of $\boldsymbol{\Gamma}$ is

$$\begin{aligned}\pi\left(\boldsymbol{\Gamma} \mid \boldsymbol{M},\boldsymbol{\tau},\sigma,\boldsymbol{A},\boldsymbol{B}\right) &\propto\ \pi\left(\boldsymbol{\Gamma}\right)\prod_{\substack{j,k:\\ M_{j,k}=1}}\boldsymbol{\phi}_3\left(\frac{\boldsymbol{a}_j-\boldsymbol{\Gamma}\boldsymbol{b}_k-\boldsymbol{\tau}}{\sigma\sqrt{2}}\right)\\ &\propto\ \pi(\boldsymbol{\Gamma})\ \exp\left\{-\frac{1}{2}\sum_{\substack{j,k:\\ M_{j,k}=1}}\left(\frac{\|\boldsymbol{a}_j-\boldsymbol{\Gamma}\boldsymbol{b}_k-\boldsymbol{\tau}\|}{\sigma\sqrt{2}}\right)^{2}\right\}\ .\end{aligned} \tag{2.14}$$

Now

$$\begin{aligned}
\exp\left\{\|\boldsymbol{a}_j - \boldsymbol{\Gamma}\boldsymbol{b}_k - \boldsymbol{\tau}\|^2\right\} &= \exp\left\{\mathrm{Tr}\left[\left((\boldsymbol{a}_j - \boldsymbol{\tau}) - \boldsymbol{\Gamma}\boldsymbol{b}_k\right)^{\mathsf{T}}\left((\boldsymbol{a}_j - \boldsymbol{\tau}) - \boldsymbol{\Gamma}\boldsymbol{b}_k\right)\right]\right\} \qquad \|\boldsymbol{X}\| = \left[\mathrm{Tr}\left(\boldsymbol{X}^{\mathsf{T}}\boldsymbol{X}\right)\right]^{\frac{1}{2}} \\
&\propto \exp\left\{\mathrm{Tr}\left[-\left(\boldsymbol{a}_j - \boldsymbol{\tau}\right)^{\mathsf{T}}\boldsymbol{\Gamma}\boldsymbol{b}_k - \left(\boldsymbol{\Gamma}\boldsymbol{b}_k\right)^{\mathsf{T}}\left(\boldsymbol{a}_j - \boldsymbol{\tau}\right)\right]\right\} \qquad \text{removing non-}\boldsymbol{\Gamma}\text{ terms} \\
&\propto \exp\left\{\mathrm{Tr}\left[-2\boldsymbol{b}_k\left(\boldsymbol{a}_j - \boldsymbol{\tau}\right)^{\mathsf{T}}\boldsymbol{\Gamma}\right]\right\} \qquad \begin{array}{l}\mathrm{Tr}(\boldsymbol{X}) = \mathrm{Tr}(\boldsymbol{X}^{\mathsf{T}}) \\ \mathrm{Tr}(\boldsymbol{X}\boldsymbol{Y}) = \mathrm{Tr}(\boldsymbol{Y}\boldsymbol{X}) \\ \mathrm{Tr}(\boldsymbol{X}+\boldsymbol{Y}) = \mathrm{Tr}(\boldsymbol{X}) + \mathrm{Tr}(\boldsymbol{Y}) \\ \mathrm{Tr}(c\boldsymbol{X}) = c\,\mathrm{Tr}(\boldsymbol{X})\ ,\end{array} \\
&\propto \exp\left\{-2\,\mathrm{Tr}\left[\boldsymbol{b}_k\left(\boldsymbol{a}_j - \boldsymbol{\tau}\right)^{\mathsf{T}}\boldsymbol{\Gamma}\right]\right\}
\end{aligned}$$

(Householder 1953, p. 39)

so continuing from (2.14)

$$\pi\left(\boldsymbol{\Gamma} \mid \boldsymbol{M}, \boldsymbol{\tau}, \sigma, \boldsymbol{A}, \boldsymbol{B}\right) \propto\ \pi\left(\boldsymbol{\Gamma}\right)\ \exp\left\{\mathrm{Tr}\left[\frac{1}{2\sigma^2}\sum_{\substack{j,k:\\ M_{j,k}=1}}\left[\boldsymbol{b}_k\left(\boldsymbol{a}_j - \boldsymbol{\tau}\right)^{\mathsf{T}}\boldsymbol{\Gamma}\right]\right]\right\}\ . \qquad (2.15)$$

It is noted that the second term of (2.15) is proportional to the density of the Matrix Fisher distribution[2]

$$\mathrm{f}(\boldsymbol{X} \mid \boldsymbol{F}) \propto \exp\left\{\mathrm{Tr}\left[\boldsymbol{F}^{\mathsf{T}}\boldsymbol{X}\right]\right\}$$

and hence a prior of the form

$$\pi(\boldsymbol{\Gamma}) \propto \exp\left\{\mathrm{Tr}\left[\boldsymbol{F}_0^{\mathsf{T}}\boldsymbol{\Gamma}\right]\right\}$$

delivers conditional conjugacy such that (2.15) becomes

$$\pi\left(\boldsymbol{\Gamma} \mid \boldsymbol{M}, \boldsymbol{\tau}, \sigma, \boldsymbol{A}, \boldsymbol{B}\right) \propto\ \exp\left\{\mathrm{Tr}\left[\boldsymbol{F}^{\mathsf{T}}\boldsymbol{\Gamma}\right]\right\}$$

where

$$\boldsymbol{F} = \boldsymbol{F}_0 + \frac{1}{2\sigma^2}\sum_{\substack{j,k:\\ M_{j,k}=1}}\left[\boldsymbol{b}_k\left(\boldsymbol{a}_j - \boldsymbol{\tau}\right)^{\mathsf{T}}\right]\ . \qquad (2.16)$$

The rotation, $\boldsymbol{\Gamma}$, can be represented by the *Euler Angle-axis Sequence*[3]

[2] Mardia and Jupp 2000, Section 13.2.3.

[3] An *Euler Angle* is an angle of rotation about a co-ordinate axis. When used to multiply from the left the given *Euler Angle-axis Sequence* is a rotation about the x-axis by ϕ, followed by a rotation around the new y-axis by θ, followed by a rotation about the newest z-axis by ψ (Kuipers 2002, Chapter 4).

$$
\begin{aligned}
\boldsymbol{\Gamma} &= \boldsymbol{\Gamma}_z(\psi)\boldsymbol{\Gamma}_y(\theta)\boldsymbol{\Gamma}_x(\phi) \\
&= \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{pmatrix} \\
&= \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & -\sin\theta\sin\phi & -\sin\theta\cos\phi \\ 0 & \cos\phi & -\sin\phi \\ \sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{pmatrix} \\
&= \begin{pmatrix} \cos\psi\cos\theta & -\cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & -\cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ \sin\psi\cos\theta & -\sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & -\sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi \\ \sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{pmatrix}
\end{aligned}
$$

where

$$
\phi \in (-\pi, \pi] \qquad \theta \in (-\frac{\pi}{2}, \frac{\pi}{2}] \qquad \psi \in (-\pi, \pi] \; .
$$

The full conditional of the rotation written in terms of these Euler angles is

$$
\propto \; \exp\left\{ \mathrm{Tr}\left[\boldsymbol{F}^{\mathsf{T}} \boldsymbol{\Gamma} \right] \right\} \cos\theta
$$

where the $\cos\theta$ term results from the uniform distribution of rotations having volume measure

$$
\frac{1}{8\pi^2} \cos\theta \, \mathrm{d}\psi \, \mathrm{d}\theta \, \mathrm{d}\phi \; .
$$

Noting that

$$
\mathrm{Tr}\left[\boldsymbol{X}^{\mathsf{T}} \boldsymbol{Y} \right] = \sum_{i,j} X_{ij} Y_{ij}
$$

the conditional distributions of the individual Euler angles can be written as

$$
\pi\left(\psi \mid \boldsymbol{M}, \boldsymbol{\tau}, \sigma, \boldsymbol{A}, \boldsymbol{B}, \theta, \phi\right) \propto \; \exp\left\{ c_\psi \cos\psi + s_\psi \sin\psi \right\} \tag{2.17}
$$

$$
\pi\left(\theta \mid \boldsymbol{M}, \boldsymbol{\tau}, \sigma, \boldsymbol{A}, \boldsymbol{B}, \psi, \phi\right) \propto \; \exp\left\{ c_\theta \cos\theta + s_\theta \sin\theta \right\} \cos\theta \tag{2.18}
$$

$$
\pi\left(\phi \mid \boldsymbol{M}, \boldsymbol{\tau}, \sigma, \boldsymbol{A}, \boldsymbol{B}, \psi, \theta\right) \propto \; \exp\left\{ c_\phi \cos\phi + s_\phi \sin\phi \right\} \tag{2.19}
$$

where

$$
\begin{aligned}
c_\psi &= F_{1,1}\cos\theta - F_{1,2}\sin\theta\sin\phi - F_{1,3}\sin\theta\cos\phi + F_{2,2}\cos\phi - F_{2,3}\sin\phi \\
&= (F_{2,2} - F_{1,3}\sin\theta)\cos\phi + (-F_{2,3} - F_{1,2}\sin\theta)\sin\phi + F_{1,1}\cos\theta \\
s_\psi &= -F_{1,2}\cos\phi + F_{1,3}\sin\phi + F_{2,1}\cos\theta - F_{2,2}\sin\theta\sin\phi - F_{2,3}\sin\theta\cos\phi \\
&= (-F_{1,2} - F_{2,3}\sin\theta)\cos\phi + (F_{1,3} - F_{2,2}\sin\theta)\sin\phi + F_{2,1}\cos\theta
\end{aligned}
$$

$$
\begin{aligned}
c_\theta &= F_{1,1}\cos\psi + F_{2,1}\sin\psi + F_{3,2}\sin\phi + F_{3,3}\cos\phi \\
s_\theta &= -F_{1,2}\cos\psi\sin\phi - F_{1,3}\cos\psi\cos\phi - F_{2,2}\sin\psi\sin\phi - F_{2,3}\sin\psi\cos\phi + F_{3,1} \\
&= (-F_{1,2}\sin\phi - F_{1,3}\cos\phi)\cos\psi + (-F_{2,2}\sin\phi - F_{2,3}\cos\phi)\sin\psi + F_{3,1}
\end{aligned}
$$

$$
\begin{aligned}
c_\phi &= -F_{1,2}\sin\psi - F_{1,3}\cos\psi\sin\theta + F_{2,2}\cos\psi - F_{2,3}\sin\psi\sin\theta + F_{3,3}\cos\theta \\
&= (F_{2,2} - F_{1,3}\sin\theta)\cos\psi + (-F_{1,2} - F_{2,3}\sin\theta)\sin\psi + F_{3,3}\cos\theta \\
s_\phi &= -F_{1,2}\cos\psi\sin\theta + F_{1,3}\sin\psi - F_{2,2}\sin\psi\sin\theta - F_{2,3}\cos\psi + F_{3,2}\cos\theta \\
&= (-F_{2,3} - F_{1,2}\sin\theta)\cos\psi + (F_{1,3} - F_{2,2}\sin\theta)\sin\psi + F_{3,2}\cos\theta\ .
\end{aligned}
$$

The conditional distributions of ψ (2.17) and ϕ (2.19) are proportional to the Von Mises distribution[4]

$$
\begin{aligned}
\mathrm{f}(x \mid \kappa, \alpha) &= \frac{1}{2\pi I_0(\kappa)} \exp\{\kappa\cos(x-\alpha)\} \\
&= \frac{1}{2\pi I_0(\kappa)} \exp\{\kappa\cos\alpha\cos x + \kappa\sin\alpha\sin x\}
\end{aligned}
$$

which can be sampled using Gibbs steps.

The Von Mises distribution is sampled using the algorithm described in Best and Fisher (1979). The conditional for θ (2.18) is sampled using a random walk Metropolis step with a uniform perturbation on $[-0.1, 0.1]$. It is assumed that there will be no prior knowledge of the rotation and hence the parameter matrix for the Matrix Fisher distribution, $\boldsymbol{F}_0$, is a zero matrix.

[4] Mardia and Jupp 2000, Section 3.5.17.

2.3.3.2 Translation and sigma

The conditional distributions for $\boldsymbol{\tau}$ and σ use standard normal and inverse gamma assumption respectively (Gelman et al. 2013, Sections 2.5 and 2.6) giving the following full conditional distributions which can be sampled using a Gibbs sampler.

The prior for $\boldsymbol{\tau}$ is

$$\boldsymbol{\tau} \sim \mathrm{N}_3\left(\boldsymbol{\mu}_\tau\ ,\ \sigma_\tau^2 \boldsymbol{I}\right)$$

and the full conditional for $\boldsymbol{\tau}$

$$\boldsymbol{\tau} \mid \boldsymbol{M}, \boldsymbol{\Gamma}, \sigma, \boldsymbol{A}, \boldsymbol{B} \sim \mathrm{N}_3\left(\frac{\boldsymbol{\mu}_\tau/\sigma_\tau^2 + \sum_{j,k:M_{j,k}=1}\left(\boldsymbol{a}_j - \boldsymbol{\Gamma}\boldsymbol{b}_k\right)/2\sigma^2}{1/\sigma_\tau^2 + L/2\sigma^2}\ ,\ \frac{1}{1/\sigma_\tau^2 + L/2\sigma^2}\right). \tag{2.20}$$

The prior for σ is

$$\sigma^{-2} \sim \Gamma\left(\alpha\ ,\ \beta\right)$$

and the full conditional for σ

$$\sigma^{-2} \mid \boldsymbol{M}, \boldsymbol{\Gamma}, \boldsymbol{\tau}, \boldsymbol{A}, \boldsymbol{B} \sim \Gamma\left(\alpha + 3L/2\ ,\ \beta + \frac{1}{4}\sum_{j,k:M_{j,k}=1}\left\|\boldsymbol{a}_j - \boldsymbol{\Gamma}\boldsymbol{b}_k - \boldsymbol{\tau}\right\|^2\right). \tag{2.21}$$

2.3.3.3 The matching matrix

The matching matrix is updated with a Metropolis-Hastings step. The acceptance probability is given by

$$\mathrm{p}\left(\boldsymbol{M}', \boldsymbol{M}\right) = \min\left\{\frac{\pi\left(\boldsymbol{M}' \mid \boldsymbol{\Gamma}, \boldsymbol{\tau}, \sigma, \boldsymbol{A}, \boldsymbol{B}\right)}{\pi\left(\boldsymbol{M} \mid \boldsymbol{\Gamma}, \boldsymbol{\tau}, \sigma, \boldsymbol{A}, \boldsymbol{B}\right)}\frac{\mathrm{q}\left(\boldsymbol{M} \mid \boldsymbol{M}'\right)}{\mathrm{q}\left(\boldsymbol{M}' \mid \boldsymbol{M}\right)},\ 1\right\}$$

where $\boldsymbol{M}'$ is the proposed matching matrix and $\mathrm{q}(.)$ are the proposal probabilities.

As described in Green and Mardia (2006, section 3.4) the matching matrix is perturbed by considering one possible match at a time as follows. A landmark is chosen at random from the set of all landmarks across the two configurations.

if the choice is matched

[1a] With probability p^* the proposal is to delete the match.

[1b] With probability $1-p^*$ the proposal is to switch the match to a currently unmatched landmark, chosen at random.

else the choice is un-matched

[2] The proposal is to add a match to a currently unmatched landmark, chosen at random.

Here p^* is a parameter introduced to weight the choice between [1a] and [1b]; for the results given in Green and Mardia (2006) p^* is set to 0.5.

The ratios $\frac{\mathrm{q}(\boldsymbol{M}|\boldsymbol{M}')}{\mathrm{q}(\boldsymbol{M}'|\boldsymbol{M})}$, given that the landmark under consideration was chosen from configuration $\boldsymbol{A}$, are calculated as follows

$$\frac{\mathrm{q}\left(\boldsymbol{M} \mid \boldsymbol{M}'\right)}{\mathrm{q}\left(\boldsymbol{M}' \mid \boldsymbol{M}\right)} = \begin{cases} \frac{\frac{1}{m}}{\frac{1}{m}} \frac{\frac{1}{L_u^{(B)}+1}}{p^*} = \frac{1}{(L_u^{(B)}+1)p^*} & \text{for [1a]} \\ \frac{\frac{1}{m}}{\frac{1}{m}} \frac{1-p^*}{1-p^*} = 1 & \text{for [1b]} \\ \frac{\frac{1}{m}}{\frac{1}{m}} \frac{p^*}{\frac{1}{L_u^{(B)}}} = L_u^{(B)} p^* & \text{for [2]} \end{cases} \tag{2.22}$$

where $L_u^{(B)}$ is the number of unmatched landmarks in configuration $\boldsymbol{B}$ when the chain is in state $\boldsymbol{M}$. The ratios $\frac{\pi(\boldsymbol{M}'|\boldsymbol{\Gamma},\boldsymbol{\tau},\sigma,\boldsymbol{A},\boldsymbol{B})}{\pi(\boldsymbol{M}|\boldsymbol{\Gamma},\boldsymbol{\tau},\sigma,\boldsymbol{A},\boldsymbol{B})}$ for [1a], [1b] and [2] are straightforward to calculate from the posterior distribution (2.13)

For [1a]

$$\begin{aligned} \frac{\pi\left(\boldsymbol{M}' \mid \boldsymbol{\Gamma}, \boldsymbol{\tau}, \sigma, \boldsymbol{A}, \boldsymbol{B}\right)}{\pi\left(\boldsymbol{M} \mid \boldsymbol{\Gamma}, \boldsymbol{\tau}, \sigma, \boldsymbol{A}, \boldsymbol{B}\right)} &= \frac{\prod_{\substack{j,k:M_{j,k}=1, \\ j\neq j_i, k\neq k_i}} \left[\left(\frac{\rho}{\lambda}\right) \left(\frac{1}{\sigma\sqrt{2}}\right)^3 \boldsymbol{\phi}_3 \left(\frac{\boldsymbol{a}_j - \boldsymbol{\Gamma}\boldsymbol{b}_k - \boldsymbol{\tau}}{\sigma\sqrt{2}}\right) \right]}{\prod_{j,k:M_{j,k}=1} \left[\left(\frac{\rho}{\lambda}\right) \left(\frac{1}{\sigma\sqrt{2}}\right)^3 \boldsymbol{\phi}_3 \left(\frac{\boldsymbol{a}_j - \boldsymbol{\Gamma}\boldsymbol{b}_k - \boldsymbol{\tau}}{\sigma\sqrt{2}}\right) \right]} \\ &= \frac{1}{\left(\frac{\rho}{\lambda}\right) \left(\frac{1}{\sigma\sqrt{2}}\right)^3 \boldsymbol{\phi}_3 \left(\frac{\boldsymbol{a}_{j_i} - \boldsymbol{\Gamma}\boldsymbol{b}_{k_i} - \boldsymbol{\tau}}{\sigma\sqrt{2}}\right)} \\ &= \frac{\left(\sigma\sqrt{2}\right)^3}{\left(\frac{\rho}{\lambda}\right) \boldsymbol{\phi}_3 \left(\frac{\boldsymbol{a}_{j_i} - \boldsymbol{\Gamma}\boldsymbol{b}_{k_i} - \boldsymbol{\tau}}{\sigma\sqrt{2}}\right)} \,. \end{aligned} \tag{2.23}$$

For [1b]

$$\frac{\pi\left(\boldsymbol{M}' \mid \boldsymbol{\Gamma}, \boldsymbol{\tau}, \sigma, \boldsymbol{A}, \boldsymbol{B}\right)}{\pi\left(\boldsymbol{M} \mid \boldsymbol{\Gamma}, \boldsymbol{\tau}, \sigma, \boldsymbol{A}, \boldsymbol{B}\right)} = \frac{\prod_{\substack{j,k:M_{j,k}=1,\\ j\neq j_{i'}, k\neq k_{i'}}} \left[\left(\frac{\rho}{\lambda}\right)\left(\frac{1}{\sigma\sqrt{2}}\right)^3 \boldsymbol{\phi}_3\left(\frac{\boldsymbol{a}_j - \boldsymbol{\Gamma}\boldsymbol{b}_k - \boldsymbol{\tau}}{\sigma\sqrt{2}}\right)\right]}{\prod_{\substack{j,k:M_{j,k}=1\\ j\neq j_i, k\neq k_i}} \left[\left(\frac{\rho}{\lambda}\right)\left(\frac{1}{\sigma\sqrt{2}}\right)^3 \boldsymbol{\phi}_3\left(\frac{\boldsymbol{a}_j - \boldsymbol{\Gamma}\boldsymbol{b}_k - \boldsymbol{\tau}}{\sigma\sqrt{2}}\right)\right]}$$
$$= \frac{\boldsymbol{\phi}_3\left(\frac{\boldsymbol{a}_{j_{i'}} - \boldsymbol{\Gamma}\boldsymbol{b}_{k_{i'}} - \boldsymbol{\tau}}{\sigma\sqrt{2}}\right)}{\boldsymbol{\phi}_3\left(\frac{\boldsymbol{a}_{j_i} - \boldsymbol{\Gamma}\boldsymbol{b}_{k_i} - \boldsymbol{\tau}}{\sigma\sqrt{2}}\right)} . \tag{2.24}$$

For [2]

$$\frac{\pi\left(\boldsymbol{M}' \mid \boldsymbol{\Gamma}, \boldsymbol{\tau}, \sigma, \boldsymbol{A}, \boldsymbol{B}\right)}{\pi\left(\boldsymbol{M} \mid \boldsymbol{\Gamma}, \boldsymbol{\tau}, \sigma, \boldsymbol{A}, \boldsymbol{B}\right)} = \frac{\prod_{j,k:M_{j,k}=1} \left[\left(\frac{\rho}{\lambda}\right)\left(\frac{1}{\sigma\sqrt{2}}\right)^3 \boldsymbol{\phi}_3\left(\frac{\boldsymbol{a}_j - \boldsymbol{\Gamma}\boldsymbol{b}_k - \boldsymbol{\tau}}{\sigma\sqrt{2}}\right)\right]}{\prod_{\substack{j,k:M_{j,k}=1,\\ j\neq j_i, k\neq k_i}} \left[\left(\frac{\rho}{\lambda}\right)\left(\frac{1}{\sigma\sqrt{2}}\right)^3 \boldsymbol{\phi}_3\left(\frac{\boldsymbol{a}_j - \boldsymbol{\Gamma}\boldsymbol{b}_k - \boldsymbol{\tau}}{\sigma\sqrt{2}}\right)\right]}$$
$$= \frac{\left(\frac{\rho}{\lambda}\right)\boldsymbol{\phi}_3\left(\frac{\boldsymbol{a}_{j_i} - \boldsymbol{\Gamma}\boldsymbol{b}_{k_i} - \boldsymbol{\tau}}{\sigma\sqrt{2}}\right)}{\left(\sigma\sqrt{2}\right)^3} . \tag{2.25}$$

2.4 Reference implementation

A reference implementation of Green and Mardia (2006) was written by this author as an `R` package, this includes a set of formal tests. The code for the package is given in Appendix B.1.

2.4.1 Testing strategy

It is important that there is significant confidence in the implementation of the described method. For this reason we applied a formalised testing strategy resulting in a set of tests that are packaged with the code. Functionality that can be broken out into units that produce deterministic return values, such as the conversion of Euler angles to a rotation matrix, is tested with scripts that are packaged as formal test scripts run under '`R CMD check`'. There is a single unit of functionality broken out that does not produce deterministic output – the '`rvonmises`' function produces random samples from a Von-Mises distribution – we tested this function by producing large samples for various parameter sets and then visually comparing histograms of the returned samples to an overlaid plot of the density. The code for these tests is packaged as an example file in the package.

The core flow of the code that iterates over the Gibbs and Metropolis steps is constructed such that a test script can replace any step or combination of steps by predetermined fixed values, hence the functionality of a single (or combination of) steps can be tested in the presence of the "correct" values produced by the other steps. We created a set of test using data from the first eight landmarks of the '1cyd' protein from the Green and Mardia (2006) data set. The $\boldsymbol{A}$ landmarks of the test data are the eight '1cyd' landmarks which have been centred to the origin. The $\boldsymbol{B}$ landmarks are the $\boldsymbol{A}$ landmarks re-ordered, translated and then rotated, with a small Gaussian perturbation applied. The ordering, rotation, translation and error are then known for the relationship of $\boldsymbol{A}$ to $\boldsymbol{B}$. We wrote a series of unit tests to check that the implementation can recover each of the values used to create the test data; these tests are summarised in Table 2.1.

test	held	free
σ	$\boldsymbol{M}$, $\boldsymbol{\tau}$, θ_z, θ_y, θ_x	
θ_z	$\boldsymbol{M}$, $\boldsymbol{\tau}$, θ_y, θ_x	θ_z, σ
θ_y	$\boldsymbol{M}$, $\boldsymbol{\tau}$, θ_z, θ_x	θ_y, σ
θ_x	$\boldsymbol{M}$, $\boldsymbol{\tau}$, θ_z, θ_y	θ_x, σ
$\boldsymbol{\tau}$	$\boldsymbol{M}$, θ_z, θ_y, θ_x	$\boldsymbol{\tau}$, σ
$\boldsymbol{M}$	$\boldsymbol{\tau}$, θ_z, θ_y, θ_x	$\boldsymbol{M}$, σ

Table 2.1: Summary of the tests used to check the validity of the core MCMC workflow.

The size of the space of all possible matching matrices for eight landmarks versus eight landmarks is $25,738$ (2.3). A series of test runs demonstrated that $100,000$ matrix updates is sufficient to sample the posterior when checking the matching matrix update proportion of the code. In each case the test is performed by checking plots of the posterior distributions against the known true values.

2.4.2 Reproducing the results from Green and Mardia (2006)

Going forward in this chapter when considering our reference implementation the terminology will reflect that Green and Mardia (2006) "made 10 updates to M in each sweep". Hence in what follows an iteration refers to a sweep over all parameters in the model – so one iteration incorporates a sequence of 10 updates to the matching matrix – and hence 1,000 iterations involves 10,000 updates of the matching matrix and 1,000 updates to all other parameters.

Using the data from Green and Mardia (2006), 2000 runs of 1,000,000 iterations each discarding a burn in of 200,000 iterations were completed using the reference implementation.[5] For all runs the algorithm was initiated with a zero matching matrix. A point estimate of the most probable matches was made by minimising the posterior expected loss with the cost ratio set to not discriminate between the undesirability of false positives and false negatives, this was implemented as described in Green and Mardia (2006, Section 3.5). Of the 2000 runs only 1 failed to produce the "36 most probable matches". The 36 most probable matches are given in Table 2.2.

Matrix index		Residue seq No.	
row	column	1cyd	1a27
1	21	14	9
2	22	15	10
3	23	16	11
4	24	17	12
5	25	18	13
6	26	19	14
7	27	20	15
9	28	38	36
10	29	39	37
11	30	40	38
12	31	43	41
13	32	59	64
14	33	60	65
15	34	61	66
16	35	62	67
17	36	83	90
18	37	84	91
19	38	85	92
20	39	86	93
22	40	106	113
23	41	134	140
24	42	135	141
25	1	136	142
26	2	137	143
38	3	138	144
39	5	143	149
40	6	146	152
27	7	149	155
28	43	153	159
29	8	179	185
30	9	180	186
31	10	181	187
32	44	182	188
33	45	184	190
34	46	185	191
35	11	186	192

Table 2.2: Details of the best 36 matches found between the 1cyd and 1a27 proteins in Green and Mardia (2006).

Green and Mardia (2006) states

[5] The 'plain text file' from `http://www.maths.bris.ac.uk/~peter/Align/` retrieved on 2016-01-20. Details of the data, and the data values, are given in Appendix A.1 in the form of our package documentation.

> After short runs of 50,000 sweeps, we tested whether or not the threshold log-posterior value had been exceeded, and if not the run was abandoned. Eighty-three out of the 100 runs passed this test, and these were allowed to run on for a further 450,000 sweeps. Every one of these 83 long runs provided exactly the same set of 36 most probable matches.

We chose our log-posterior threshold from examining trace plots from several runs started with the correct 36 matches. With this 78% of our 2,000 runs were below the log-posterior threshold after 50,000 iterations (500,000 updates to the matching matrix). In our test we ran the full set of iterations in all 2,000 cases and in all of the 22% not passing the log posterior threshold after 50,000 iterations, the run continued on to produce the correct point estimate of the best 36 matches.

Figure 2.2 shows the trace plots from a typical run of our reference implementation.

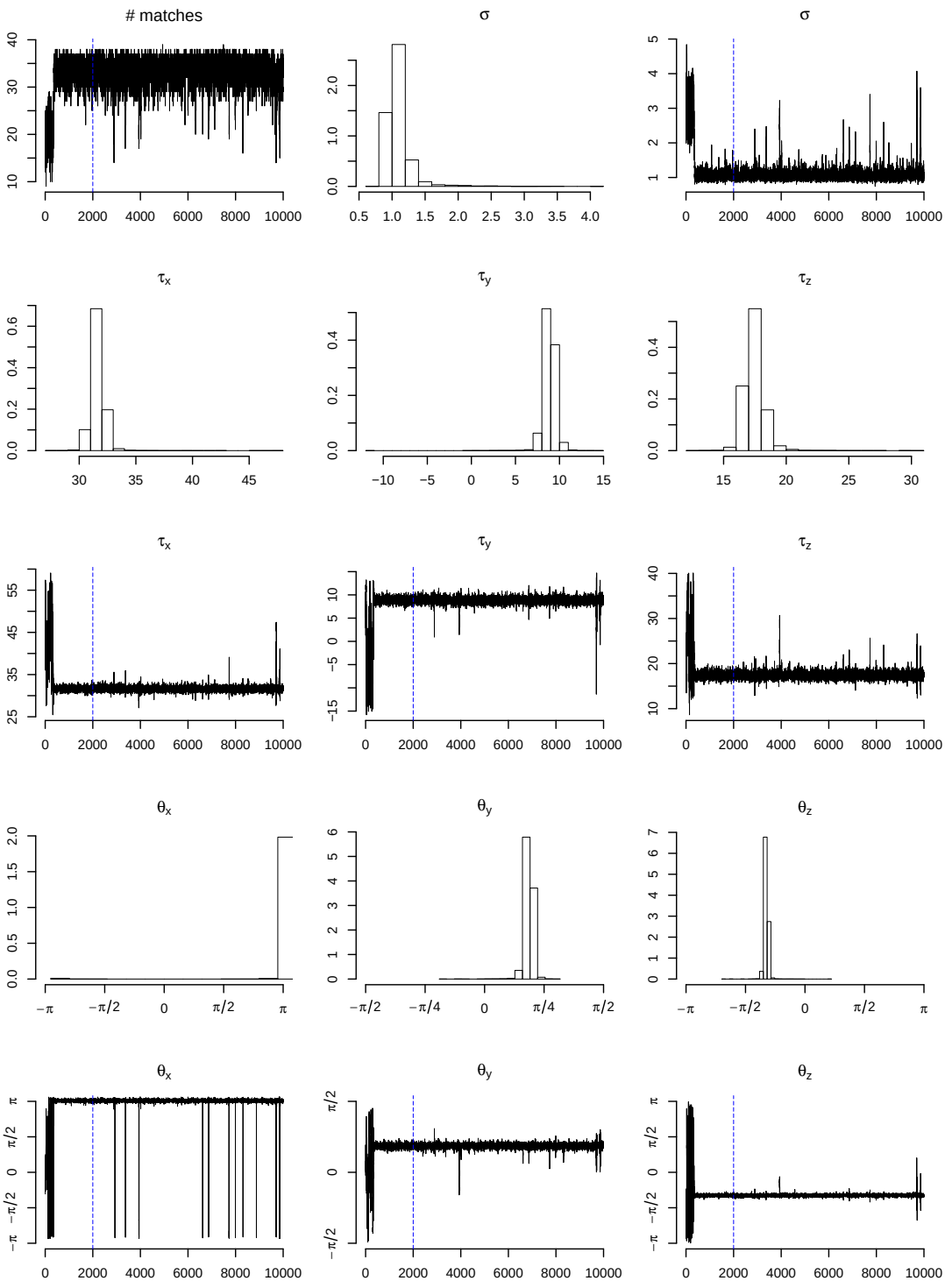


Figure 2.2: Typical set of trace plots from a run of the reference implementation of Green and Mardia (2006). The trace plots represent a run of 1,000,000 iterations (10,000,000 matching matrix updates). The plots are of a thinned sample where 1 in every hundred iterations is kept.

Figure 2.3 zooms in on the first 100,000 iterations of the number of matches trace plot from Figure 2.2. Notice the very pronounced shift in mode which was typical of all runs;

this will be discussed in detail in Section 2.5.3.

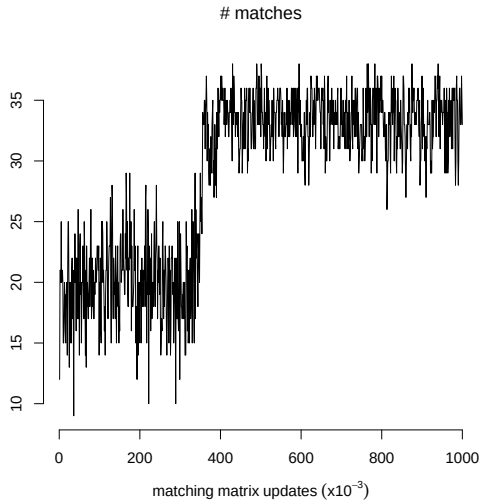


Figure 2.3: The first 100,000 iterations of the number of matches trace plot from Figure 2.2. Notice the very pronounced shift in mode which was typical of all runs.

Of the 2,000 runs only a single run failed to produce all 36 matches. This run produced 35 of the 36 matches, the discrepancy resulting from a temporary jump out of the mode just after the burn in period. This run passed was below the log-posterior threshold. The trace plots from the failure are given in Figure 2.4.

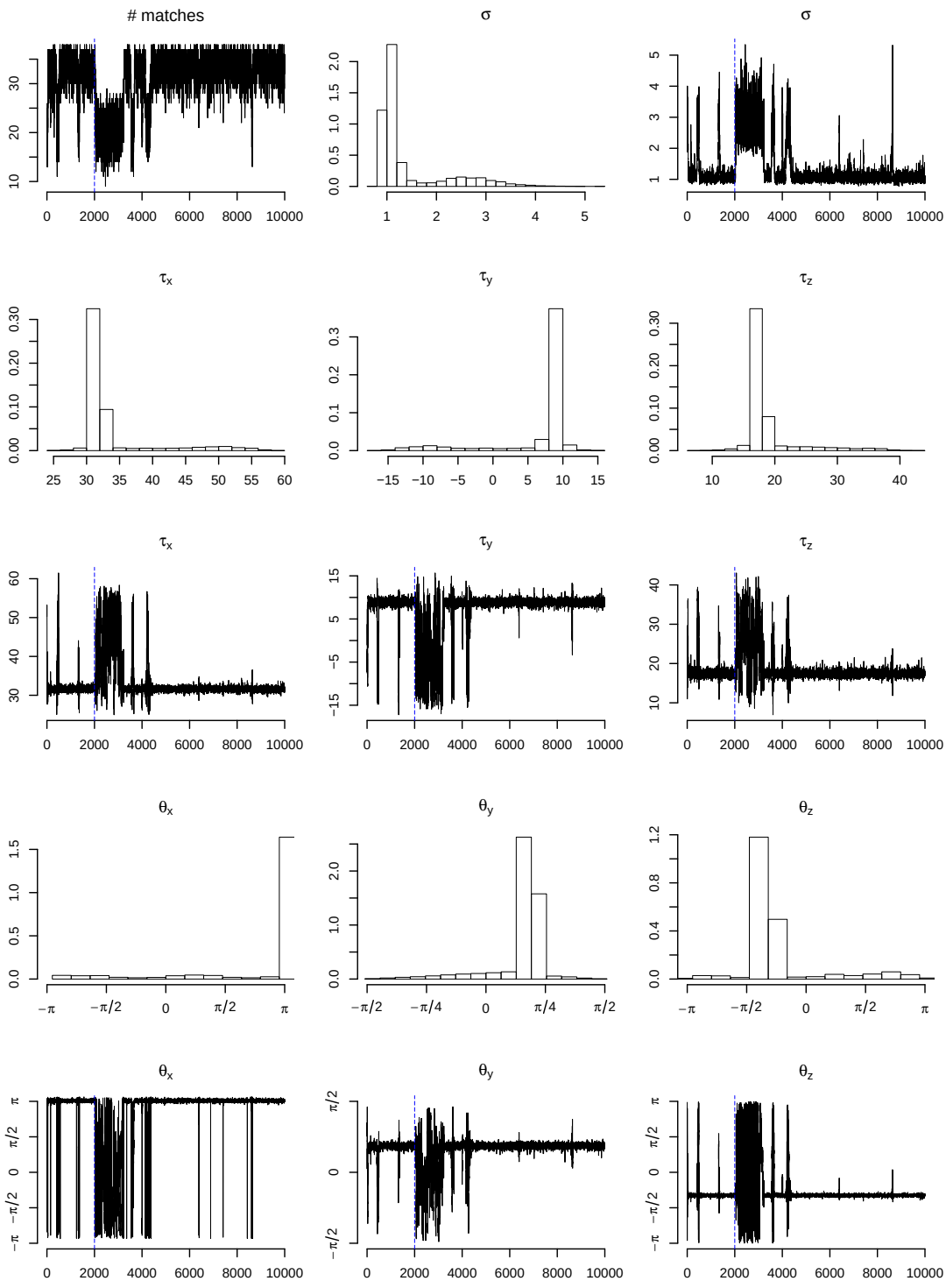


Figure 2.4: Trace plots from the single failure of 2000 runs of the reference implementation of Green and Mardia (2006). Details are the same as Figure 2.2.

2.5 Considerations for unsupervised matching

This section considers the applicability of the Green and Mardia (2006) method for unsupervised matching from three perspectives: the required prior information relating to the expected number of matches as manifested in the hyper parameter λ/ρ, scalability in the context of the information available through the likelihood to differentiate correct matches from incorrect matches when finding early correct matches, and dealing with multimodality in the posterior.

2.5.1 Hyper-parameter λ/ρ

The hyper-parameter λ is the rate of the Poisson process that produces the true locations and ρ "is a certain measure of the tendency a priori for points to be matched". The guidance from Green and Mardia (2006, Section 4.3) for setting the value of λ/ρ

> It is clear from inspection that setting λ/ρ equal to $(m-\bar{L})(n-\bar{L})/\bar{L}V$ yields a mode of L that is within 1 of $\bar{L}$, ... As long as V is known, or at least a representative value is provided, and the analyst is able to make a prior guess L at the number of matches, this suggests a reasonable way to specify λ/ρ. The posterior distribution for L tracks the prior rather closely, confirming that the raw data carry little information about the number of matches.

To see this consider the substitution of $(m-\bar{L})(n-\bar{L})/\bar{L}V$ into the conditional distribution for L given in (2.6)

$$\begin{aligned}\pi(L \mid \dots) &\propto \frac{\left(\frac{\rho}{\lambda V}\right)^L}{L!(m-L)!(n-L)!} = \frac{\left(\frac{\bar{L}}{(m-\bar{L})(n-\bar{L})}\right)^L}{L!(m-L)!(n-L)!} \\ &\propto \begin{cases} \frac{C}{(m-\bar{L})^2(n-\bar{L})^2} & \text{if } L=\bar{L} \\ \frac{C\bar{L}}{(\bar{L}+1)(m-\bar{L})^2(n-\bar{L})^2} & \text{if } L=\bar{L}+1 \\ \frac{C}{(m-\bar{L}+1)(n-\bar{L}+1)\,(m-\bar{L})(n-\bar{L})} & \text{if } L=\bar{L}-1 \end{cases} \end{aligned} \tag{2.26}$$

where

$$C = \frac{\left(\frac{\bar{L}}{(m-\bar{L})(n-\bar{L})}\right)^{(\bar{L}-1)}}{(\bar{L}-1)!(m-\bar{L}-1)!(n-\bar{L}-1)!}\,.$$

(2.26) illustrates that with the given substitution $L=\bar{L}$ is at least a local maxima.

In practice the volume term is an estimate and hence we re-examine (2.26) and add a factor,

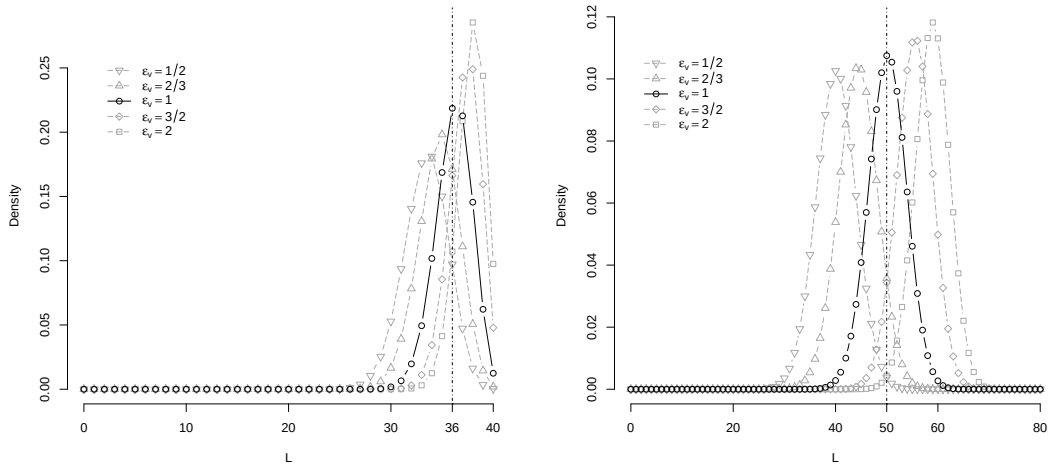
ϵ_v, that represents the error in the estimate of V. Now (2.26) becomes

$$\begin{aligned}\pi(L \mid \dots) &\propto \frac{\left(\frac{\bar{L}}{(m-\bar{L})(n-\bar{L})}\,\epsilon_v\right)^L}{L!(m-L)!(n-L)!} \\ &\propto \begin{cases} \frac{C\epsilon_v}{(m-\bar{L})^2(n-\bar{L})^2} & \text{if } L = \bar{L} \\ \frac{C\bar{L}\,\epsilon_v^2}{(\bar{L}+1)(m-\bar{L})^2(n-\bar{L})^2} & \text{if } L = \bar{L}+1 \\ \frac{C}{(m-\bar{L}+1)(n-\bar{L}+1)\,(m-\bar{L})(n-\bar{L})} & \text{if } L = \bar{L}-1 \end{cases}\end{aligned} \tag{2.27}$$

where

$$C = \frac{\left(\frac{\bar{L}}{(m-\bar{L})(n-\bar{L})}\,\epsilon_v\right)^{(\bar{L}-1)}}{(\bar{L}-1)!(m-\bar{L}-1)!(n-\bar{L}-1)!}\,.$$

(2.27) illustrates that without knowledge of ϵ_v there is no information available about the mode of $\pi(L \mid \dots)$. The plots in Figure 2.5 show the distribution for L given in (2.6) having substituted $\lambda/\rho = (m-\bar{L})(n-\bar{L})/\bar{L}V\epsilon_v$. The plots demonstrate the interaction of the estimation quality of V with the mode of $\pi(L \mid \dots)$; the pronounced sensitivity to ϵ_v indicates the importance of both prior information on the number of matches and the volume in which these matches are enclosed.

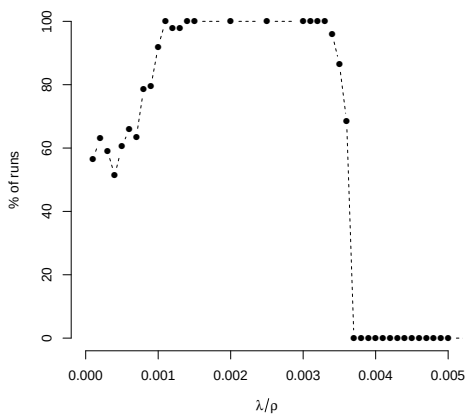


(a) $m = 40$, $n = 63$, $\bar{L} = 36$ **(b)** $m = 80$, $n = 100$, $\bar{L} = 50$

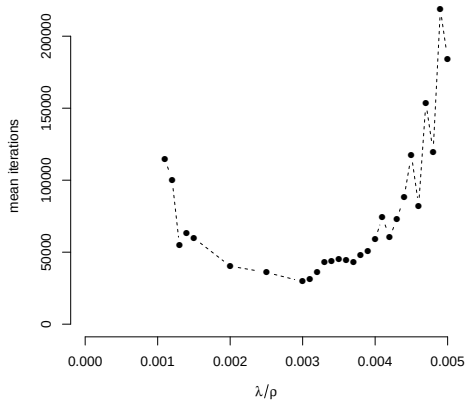
Figure 2.5: Plots of the distribution for L given in (2.6) having substituted $\lambda/\rho = (m-\bar{L})(n-\bar{L})/\bar{L}V\epsilon_v$. The plots illustrate the sensitivity of the prior distribution of L to the estimate of V. (a) illustrates the example given in Green and Mardia (2006). (b) is given both for additional clarity and to demonstrate that the influence of the quality of V increases as the number of landmarks in the configurations increase.

Figure 2.6 shows the sensitivity of the method around the optimum value of $\lambda/\rho = 0.003$

for the Green and Mardia (2006) data. Notable is that there is a significant drop off the effectiveness of the method outside of the values $0.0011 \leq \lambda/\rho \leq 0.0033$; the skew in this range could be taken to suggest that estimates of λ/ρ are better to be underestimated rather than overestimated, although there is indication from the plot in Figure 2.5a that this is a facet of the closeness of the correct number of matches to the maximum possible correct number of matches (i.e. the number of landmarks in configuration $\boldsymbol{A}$).



(a) The percentage of runs that produce a point estimate with at least 34 correct matches and no incorrect matches.



(b) The mean number of iterations before the Markov chain first produces a matching matrix with at least 34 correct matches and no incorrect matches.

Figure 2.6: Sensitivity of the Green and Mardia (2006) method to the λ/ρ hyperparameter. Each data point is derived from 50 runs of the reference implementation using the parameters from Green and Mardia (2006) but with a total run of 1,000,000 iterations, a burn in of 200,000 iterations (and 10 updates to the matching matrix per iteration). The minimum value tested was $\lambda/\rho = 0.0001$.

Figure 2.7 shows a typical set of traces for a value just outside of $0.0011 \leq \lambda/\rho \leq 0.0033$; the traces indicate a tendency to jump away from the mode.

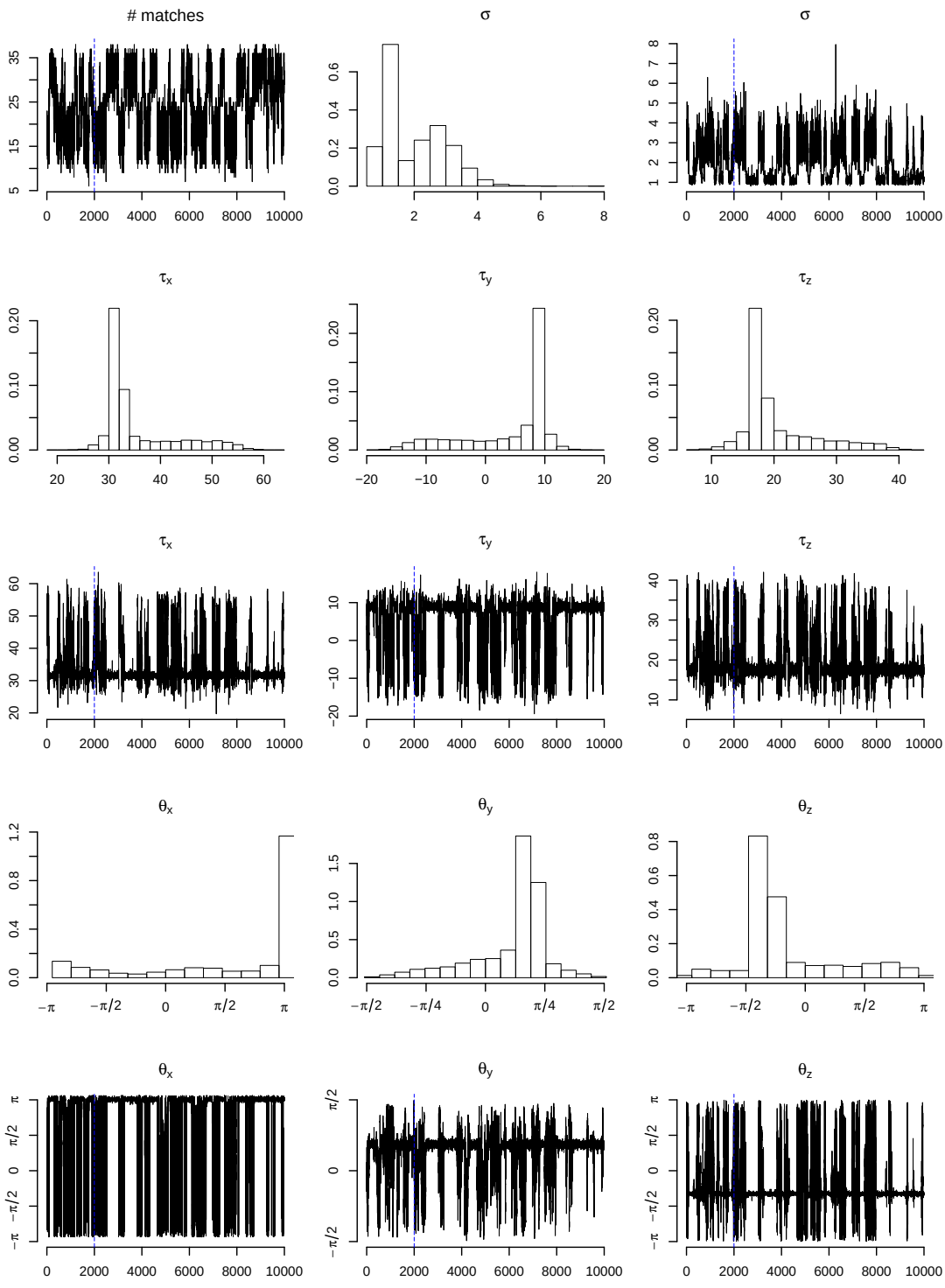


Figure 2.7: A typical set of traces for a value just outside of $0.0011 \leq \lambda/\rho \leq 0.0033$; the traces indicate a tendency to jump away from the mode. Run details are the same as those described in Figure 2.2.

2.5.2 Calculating a minimum enclosing volume

Section 2.5.1 indicates the need for prior information as to the number and spatial arrangement of the true matches so that an estimation can be made of their enclosing volume. This section is somewhat of an aside as it illustrates a simple numerical method for finding minimal enclosing volumes. The ideal simplified shape to use as a minimal enclosing volume for a general point set is the smallest enclosing ellipsoid as the volume approximation error is independent of the shape of the covering body (Gärtner and Schönherr 1997). The next best bounding shape is the isothetic bounding box or a hyper-rectangle; we have chosen to use this because the complexity of the calculation is greatly reduced.

In much of the literature the minimum volume hyper-rectangle containing a set of points is referred to as *minimal enclosing box*. Calculating a minimum enclosing box in $\mathbb{R}^3$ is solved using a deterministic algorithm in O'Rourke (1985); however, the implementation of the method is non-trivial and at the time of writing an implementation is not available in `R`, hence the much simpler brute force search described below was used. It is worth noting that other popular, simple, heuristic methods suffer from common pathological problems such as those described for principal component based methods in Dimitrov et al. (2009). An example of the type of problem seen using principal component based methods is shown in Figure 2.8 which shows the bounding box for a 2D projection of protein secondary structure landmarks. The locally dense areas of landmarks have a significant unwanted effect on the principal components resulting in a larger bounding box than is achieved by a simple grid search over the rotation angle.

Our approach to approximating the minimal enclosing box takes a set of uniformly distributed rotations in $\mathrm{SO}(3)$ calculated from spherical Fibonacci points sets of the type used in numerical integration over spheres (Hannay and Nye 2004). Each rotation in the set is applied in turn to the landmarks and then the axis aligned minimum enclosing box is calculated; the minimum calculated volume is then chosen.

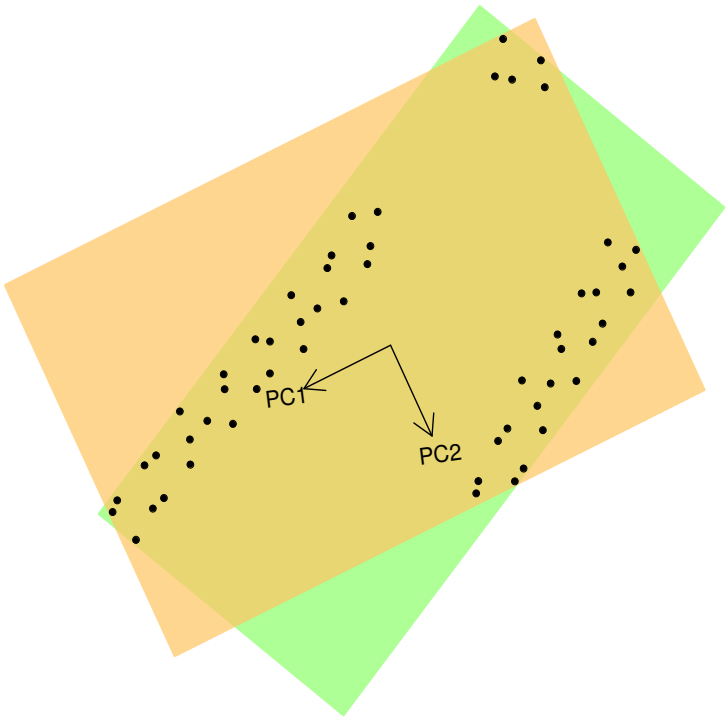


Figure 2.8: An example of the type of problem seen using principal component based methods to find a minimum bounding box. The landmarks are a 2D projection of protein secondary structure co-ordinates. The locally dense areas of landmarks have a significant unwanted effect on the principal components resulting in larger bounding box, in yellow, than is achieved by a simple grid search over the rotation angle, in green.

The procedure in detail is as follows: A set of Fibonacci points are generated on the surface of a unit sphere, then for each point the rotation matrix required to rotate the point $(0,\, 0,\, 1)$ onto the generated point is calculated using the representation described in Möller and Hughes (1999), given in (2.28).

$$\boldsymbol{R} = \begin{pmatrix} c + hv_x^2 & hv_xv_y - v_z & hv_xv_z + v_y \\ hv_xv_y + v_z & c + hv_y^2 & hv_yv_z - v_x \\ hv_xv_z - v_y & hv_yv_z + v_x & c + hv_z^2 \end{pmatrix} \tag{2.28}$$

where

$$
\begin{aligned}
f &= \text{a point from the generated Fibonacci points set} \\
x &= (1, 0, 0) \\
\boldsymbol{v} &= \boldsymbol{f} \times \boldsymbol{t} \\
c &= \boldsymbol{f} \cdot \boldsymbol{t} \\
h &= \frac{1-c}{1-c^2} = \frac{1-c}{\boldsymbol{v} \cdot \boldsymbol{v}} \, .
\end{aligned}
$$

The algorithm in Figure 2.9 is used to generate the Fibonacci points on the sphere and is based on the description in Hannay and Nye (2004).

$N = 1000$ // number of Fibonacci points
$z = -1$
$\Delta z = \frac{2}{N}$
$\phi = 0$
$\Delta\phi = \pi(1 + \sqrt{5})$
for $j = 1$ **to** N
    $l = \sqrt{1 - z^2}$
    $(x_j, y_j, z_j) = (l\cos\phi, l\sin\phi, z)$
    $z = z + \Delta z$
    $\phi = \phi + \Delta\phi$

Figure 2.9: The algorithm used to generate the Fibonacci points on a sphere. Based on Hannay and Nye (2004).

Figure 2.10a illustrates the angles described in the algorithm and figure 2.10b shows a set of small spheres centred at the generated point set on the surface of the unit sphere to illustrate their distribution over the surface.

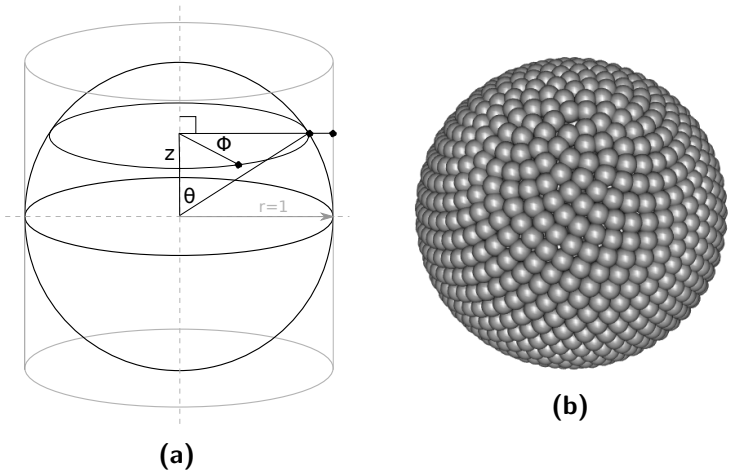


Figure 2.10: Spherical Fibonacci points sets generated over the surface of a sphere. (a) shows the arrangement of variables used in the description of the algorithm. The points are generated on a cylinder and then are projected perpendicularly from the cylinder axis onto a sphere. The sphere is aligned with its polar axis along the axis of the cylinder. The cylinder is of length $2r$, where $r = 1$ is the radius of the sphere. (b) illustrates the distribution of a set of 1000 generated points over the surface of the sphere.

2.5.3 Information available from the likelihood

The focus of this section is the information available to the likelihood. To simplify the discussion the following simplified problem is considered:

- p^* is set to 1 so that only adding and deleting matches need be considered.
- The number of landmarks in each configuration is chosen to be the same so that $L_u = L_u^{(A)} = L_u^{(B)}$.
- At the point of analysis the Markov chain will be assumed to be in a state where the rotation, $\boldsymbol{\Gamma}$, and translation, $\boldsymbol{\tau}$, are ideal for the current state of the matching matrix, $\boldsymbol{M}$. This is equivalent to translating the configurations such that the centres of mass of their currently matched landmarks are at the origin and then optimising their rotation as described in Section 1.2.5.1; as such this regime is somewhat equivalent to that described in Dryden, Hirst, et al. (2007).
- The λ/ρ term is chosen to have the ideal value.

Under these conditions and using (2.22) and (2.25) the acceptance probability for adding a match for landmark $\boldsymbol{a}_j$ to landmark $\boldsymbol{b}_k$ is given by

$$\begin{aligned} \mathrm{p}\left(\boldsymbol{M}', \boldsymbol{M}\right) &= \min\left\{ \frac{\left(\frac{\rho}{\lambda}\right)\phi_3\left(\frac{\boldsymbol{a}_j - \ddot{\boldsymbol{\Gamma}}\boldsymbol{b}_k - \ddot{\boldsymbol{\tau}}}{\sigma\sqrt{2}}\right)}{\left(\sigma\sqrt{2}\right)^3} L_u \ , \ 1 \right\} \\ &= \min\left\{\mathrm{f}\left(\boldsymbol{M}', \boldsymbol{M}\right), 1\right\} \end{aligned} \tag{2.29}$$

where $\ddot{\boldsymbol{\Gamma}}$ and $\ddot{\boldsymbol{\tau}}$ are the optimised rotation and translation. It is noted that as the Markov process is reversible a similar argument to what follows can be made for the deletion of a match.

It is clear from (2.29) that under the ideal conditions described above the variance is the critical factor and that assuming, as is the case, that σ has an uninformative prior then the information available for σ must come from the data. Now consider starting from a zero matching matrix and consecutively adding correct matches. The variance will have no data available when adding the first or second matches as it will be constructed from zero and one data points respectively. Information available to the variance will at best be marginal when adding the third point as the variance will be calculated from two data points that can only capture information from a single degree of freedom – it will not be until the fifth point is being added that all the degrees of freedom come into play. Figure 2.11 illustrates values for $\mathrm{f}\left(\boldsymbol{M}', \boldsymbol{M}\right)$ under these ideal conditions for the the Green and Mardia (2006) data set. The specific details of the simulation used in Figure 2.11 are:

- $\boldsymbol{A}$ and $\boldsymbol{B}$ are the `1cyd` and `1a27` landmark data respectively from the Green and Mardia (2006) data set.
- The values for $\mathrm{f}\left(\boldsymbol{M}', \boldsymbol{M}\right)$ are calculated by numerical integration (quadrature) using 1000 equally spaced points between the 0.0005 and 0.9995 percentiles of the variance; the variance at each point being calculated using (2.21).
- For each addition match the values for $\mathrm{f}\left(\boldsymbol{M}', \boldsymbol{M}\right)$ are given for the best case where all the previous matches are correct e.g. the values given for the addition of the fifth matches are calculated starting from a matching matrix with four correct matches.

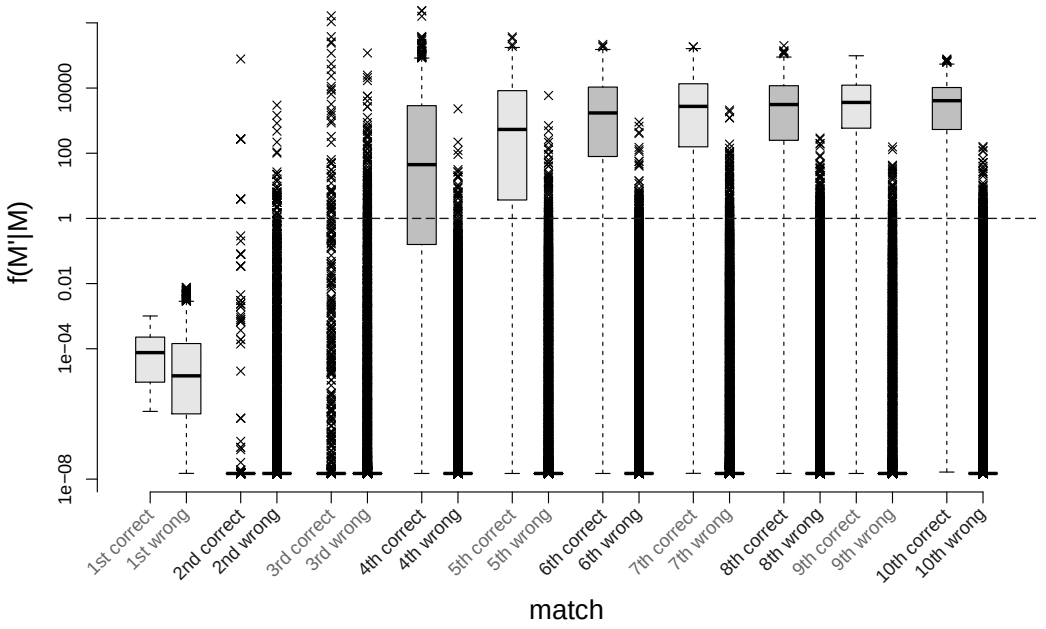


Figure 2.11: Plots of the value of $\mathrm{f}\left(\boldsymbol{M}', \boldsymbol{M}\right)$ for the idealised scenario described in Section 2.5.3. This illustrates that the acceptance probability does not have sufficient information to delineate correct from incorrect matches until adding the fourth match. The `1.490116e-08` was added to all values to allow a log scale; hence `1e-8` should be treated as zero.[6]The code used to create this plot is given in Appendix B.2.

Figure 2.11 indicates that until at least three correct matches have been found the likelihood does not contain enough information to drive the Markov process toward a mode, and hence to this point an optimisation would be considered as a random, uninformed search. This has implications for the scalability of the method with respect to its application to comparing proteins referenced by the SCOP2 database for which we need to compare structures with several hundred landmarks (Figure 2.12). For example, from (2.2) the number of possible matching matrices for L matches is given by $L!\binom{m}{L}\binom{n}{L}$, and of these $\binom{L_\mathsf{T}}{L}$ will be correct – L_T is the true number of correct matches – and hence the ratio of the number of possible matching matrices containing L correct matches to matching matrices containing at least one incorrect match, for a matching matrix containing exactly L matches, is one in

$$\frac{L!\binom{m}{L}\binom{n}{L}}{\binom{L_\mathsf{T}}{L}} \,. \tag{2.30}$$

[6] `1.490116e-08` is the value of `sqrt(.Machine$double.eps)` for the R instance on the machine where the code was executed. `.Machine$double.eps` is described by the R documentation as "the smallest positive floating-point number x such that $1 + x \neq 1$".

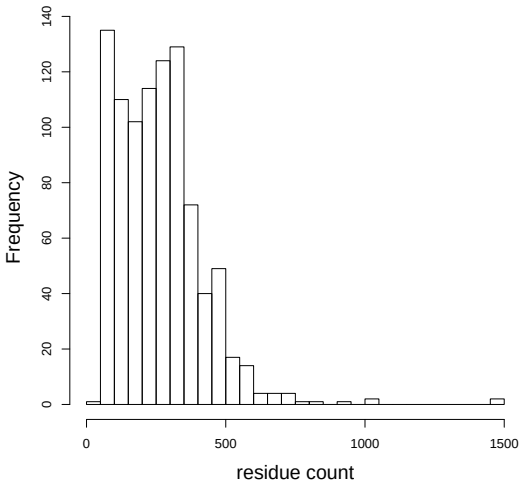


Figure 2.12: Frequency counts of the number of residues in the protein asymmetric units referenced by SCOP2. The counts have been binned, with bin widths of 50.

In the case of the example given in Green and Mardia (2006) this equates to

$$\frac{3!\binom{40}{3}\binom{63}{3}}{\binom{36}{3}} = 329,701$$

When matching the smaller proteins in SCOP2, say with 100 residues at 30% matching (the family level of categorisation is defined as having 30% and greater residue identity (Hubbard et al. 1999)) then this number rises to

$$\frac{3!\binom{100}{3}\binom{100}{3}}{\binom{30}{3}} = 38,640,724$$

As a secondary check of these calculations Figures 2.13 show histograms of the number of matching matrix updates made using the Green and Mardia (2006) data set before 1, 2, 3, and 4 correct matches are achieved. The histograms are overplotted with a hypergeometric distribution with the probability being the inverse of (2.30), which would be expected if the Markov chain was a completely unguided random walk. The plots indicate that the process is substantively a random walk until a correct 3rd match is found. Once the third correct match is found the likelihood is able to discern correct matches (as demonstrated in Figure 2.11) and the Markov process moves strongly toward the mode. This "catching" is seen in all trace plots that were manually examined, a typical example being that shown earlier in Figure 2.2.

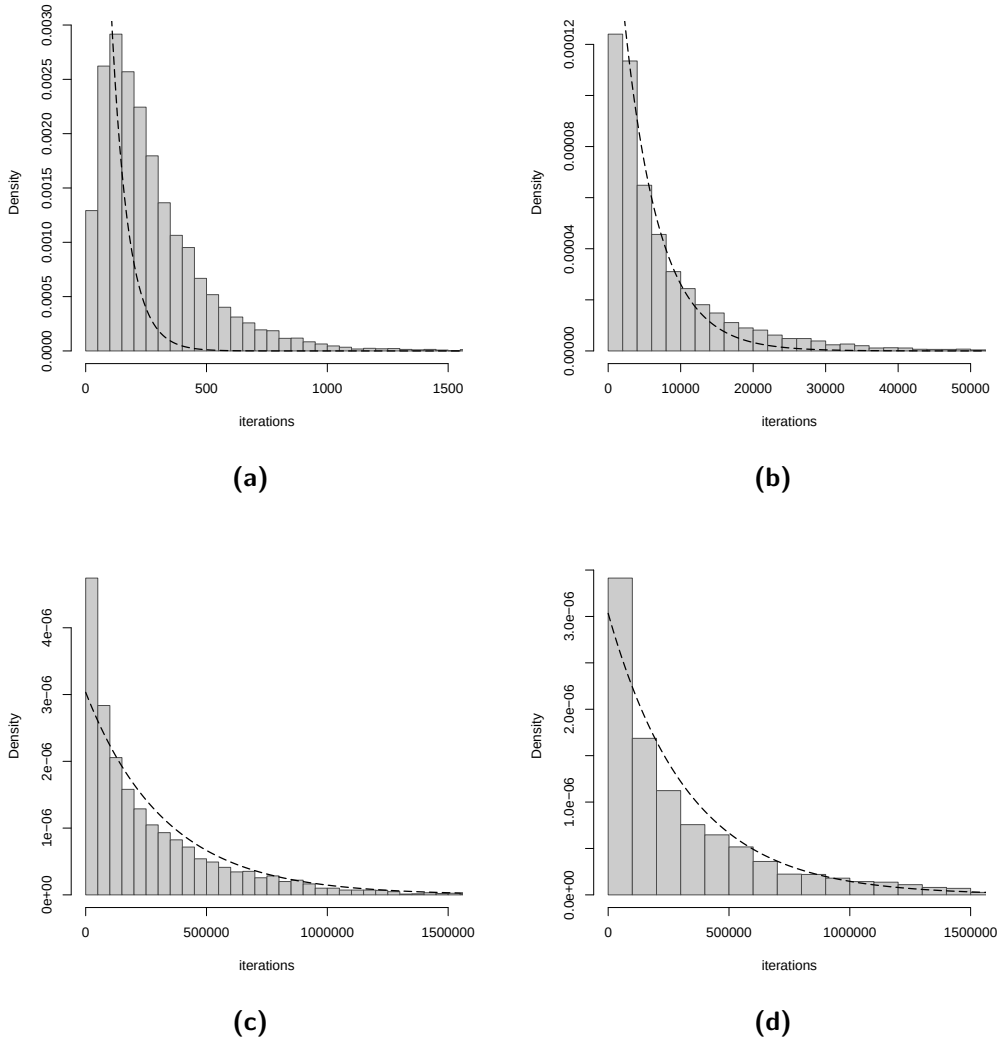


Figure 2.13: Histograms of the number of matching matrix updates taken before the first 1 (a), 2 (b), 3 (c), and 4 (d) correct matches are found. In each case a hypergeometric density is overplotted with the probability parameter calculated using (2.30). The overplotting for plots (a), (b) and (c) are calculated with the probability for 1, 2 and 3 matches respectively, and the overplot for plot (d) is again calculated with the probability for 3 matches, the same as plot (c). Each histogram represents 5,000 runs using the setup from Green and Mardia (2006).

2.6 Multimodality

In this chapter we will not present a detailed treatment of multimodality. When considering the Green and Mardia (2006) data this is not an issue as it is clear from the aligned set of protein structures (Figure 3.12c) that the data will only present one major mode of interest. It is worth noting however that a discussion relating to multimodality and techniques such

as parallel tempering[7] need to be pre-empted by addressing the size of the solution space away from the modes (Section 2.5.3). The problem of finding modes in this solution space is treated in detail in Chapter 3 to follow.

2.7 Summary

Green and Mardia (2006) offers important insights into approaching the unlabelled partial matching problem for proteins. The first of these is that their likelihood can be used to locate a single solution for their dataset of 40 and 63 landmarks which results a possible matching matrix solution space of $\sum_{L=0}^{40} L!\binom{40}{L}\binom{63}{L} \approx 3.7 \times 10^{65}$ matrices. This clearly demonstrates that this form of likelihood and the model of spherical errors is appropriate for this problem domain. We see from our analysis that once at least three matches are found the likelihood strongly drives the Markov process to the mode; this will prove to be useful for the method discussed in Chapter 3. The method is sensitive to the λ/ρ hyper-parameter for which in large scale un-supervised matching scenarios prior information is unlikely to be available, but a range of values could be suggested by considering the minimum enclosing volumes of the target configurations. The discussion in Section 2.5.3 relating to the limited information available to the likelihood is further addressed in Chapter 3.

[7] Gelman et al. 2013, Section 12.3.

3

A greedy algorithm

3.1 Overview

This chapter details the development of a new algorithm called GProtA (Greedy Protein Alignment) which can be used to find global solutions to the unlabelled partial matching problem for protein data. We begin in Section 3.2 with a description of an approximate distribution of size-and-shape distance calculated between configurations with known labelling where the difference in configurations is modelled by spherical normal errors; this distribution will be important for several of the sections in the remainder of the chapter. Section 3.3 details several protein specific constraints that allow us to specify a sufficiently small set of candidate starting matching matrices that it becomes practical to filter them based on a comparison to a distribution of shape distances from a random sample of matching matrices. In Section 3.4 we develop the ideas from Section 3.3 to allow us to iteratively add new high quality matches between the configurations. We then consolidate the ideas of sections 3.3 and 3.4 into a greedy algorithm that we propose as a general solution to the unlabelled matching problem when applied to protein data. Finally in Section 3.5 we propose a difference measure that can be used to quantify the similarity between proteins.

3.2 A spherical error model of size-and-shape distance

Consider the model where $\boldsymbol{B}$ is a copy of $\boldsymbol{A}$ with the addition of some form of random errors, $\boldsymbol{\epsilon}$

$$\boldsymbol{B} = \boldsymbol{A} + \boldsymbol{\epsilon} \, .$$

$\boldsymbol{B}$ is observed after the application of some unknown rigid transformation; the transformation consisting of a rotation, $\boldsymbol{R} \in \mathbf{SO}(m)$, and a translation by a repeated set of displacement vectors, $\boldsymbol{\tau} = (1, \ldots, 1)^{\mathsf{T}}(\tau_1, \ldots, \tau_m)$

$$\boldsymbol{B}_{\mathsf{obs}} = \boldsymbol{B}\boldsymbol{R} + \boldsymbol{\tau} \, . \tag{3.1}$$

When labelling is known, any method for optimally aligning $\boldsymbol{A}$ and $\boldsymbol{B}$ will result in an estimate of $\boldsymbol{B}$, $\widehat{\boldsymbol{B}}$, containing three sources of variance; these being from the error in estimating the translation, the error in optimising the rotation, and residual errors. Procrustes measures of distance between shapes are calculated from the squared distances between labelled landmarks in each configuration after the sum of squared distances has been minimised, for example

$$\mathrm{d}_{\mathsf{ss}}\left(\boldsymbol{A},\, \boldsymbol{B}\right) = \inf_{\widehat{\boldsymbol{R}} \in \mathbf{SO}(m) \;\&\; \widehat{\boldsymbol{\tau}} \in \boldsymbol{T}(k,m)} \left\| \boldsymbol{A} - \left(\boldsymbol{B}_{\mathsf{obs}} - \widehat{\boldsymbol{\tau}}\right) \widehat{\boldsymbol{R}}^{\mathsf{T}} \right\|_{\mathsf{F}} \tag{3.2}$$

where $\widehat{\boldsymbol{R}}$ and $\widehat{\boldsymbol{\tau}}$ are estimates of the rotation and translation applied to $\boldsymbol{B}$, and $\boldsymbol{T}(k,m)$ is the set of all $k \times m$ matrices. If the infimum over the rotation and translation are treated separately, and the infimum over translation calculated first then this is always achieved by translating the centroid of each configuration to the origin. When this translation is performed by Helmertizing the configurations, (3.2) becomes

$$\begin{aligned} \mathrm{d}_{\mathsf{ss}}\left(\boldsymbol{A},\, \boldsymbol{B}\right) &= \inf_{\boldsymbol{\Gamma} \in \mathbf{SO}(m)} \left\| \boldsymbol{H}_{\mathsf{s}}\boldsymbol{A} - \boldsymbol{H}_{\mathsf{s}}\boldsymbol{B}_{\mathsf{obs}}\boldsymbol{\Gamma} \right\|_{\mathsf{F}} \\ &= \inf_{\boldsymbol{\Gamma} \in \mathbf{SO}(m)} \left\| \boldsymbol{A}' - \boldsymbol{B}'_{\mathsf{obs}}\boldsymbol{\Gamma} \right\|_{\mathsf{F}} \end{aligned} \tag{3.3}$$

where $\boldsymbol{A}' = \boldsymbol{H}_{\mathsf{s}}\boldsymbol{A}$ and $\boldsymbol{B}'_{\mathsf{obs}} = \boldsymbol{H}_{\mathsf{s}}\boldsymbol{B}_{\mathsf{obs}}$. Note that since one result of the action of the Helmert matrix is to modify the geometric relation between the landmarks (see Section 1.2.3.1) then the minimising rotation, $\boldsymbol{\Gamma}$, is generally no longer an estimate of the rotation in (3.1).

Now consider the special case where the errors on each landmark are iid and have some form of spherical distribution

$$\boldsymbol{\epsilon} \ \sim \ \mathrm{f}\left(\boldsymbol{\mu} = \mathbf{0},\ \boldsymbol{\Sigma} = \sigma^2(I_k \otimes I_m)\right) \ . \tag{3.4}$$

In this case the actions of $\boldsymbol{H}_{\mathrm{s}}$ will not transform the covariance matrix since $\boldsymbol{H} \in \mathbf{SO}(k)$. The action of $\boldsymbol{H}_{\mathsf{s}}$ on $\boldsymbol{B}_{\mathsf{obs}}$ reduces the degrees of freedom by m as the resulting configuration is represented by a $(k-1) \times m$ matrix in the space of *derived landmarks*. The action of the optimising rotation $\boldsymbol{\Gamma} \in \mathbf{SO}(m)$ in the derived landmark space constrains the result by a further $\frac{1}{2}m(m-1)$ degrees of freedom, but the optimisation results in dependency between the residuals which no longer are iid as the original errors.

If we further specialise our treatment of the errors to consider spherical normal errors such that (3.4) becomes

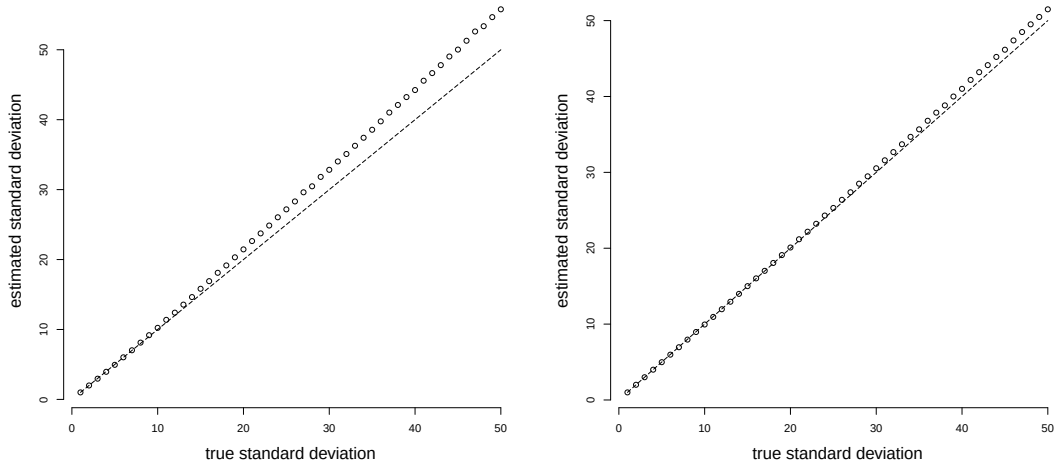
$$\boldsymbol{\epsilon} \ \sim \ \mathrm{N}\left(\boldsymbol{\mu} = \mathbf{0},\ \boldsymbol{\Sigma} = \sigma^2(I_k \otimes I_m)\right) \tag{3.5}$$

then a sample of distances of $\mathrm{d}^2_{\mathsf{ss}}\left(\boldsymbol{A},\ \boldsymbol{B}\right)$ derived from a set of independent observation of $\boldsymbol{B}$ would be approximately distributed as

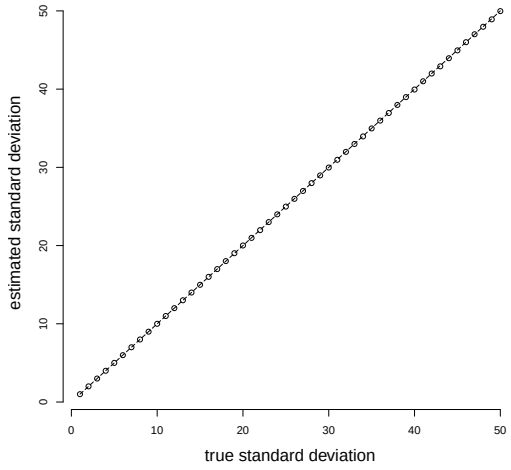
$$\sigma^2 \chi^2_{km-\frac{1}{2}m(m+1)} \ = \ \mathrm{Gamma}\left(\frac{km - \frac{1}{2}m(m+1)}{2},\, 2\sigma^2\right) \ . \tag{3.6}$$

The error in estimating $\boldsymbol{\Gamma}$ is a function of the number of landmarks being optimised over and the variance of the initial errors. We quantify the robustness of the approximation in Figure 3.1 which shows that deviation from (3.6) is minimal for protein data even with as few as 5 landmarks and a standard deviation of 10Å. In Figure Ďetailed treatment of this form of approximation can be found in Sibson (1979), Langron and Collins (1985), and Goodall (1991).

The relationship given in (3.6) will be useful later when we shall use it to help check our ability to delineate two sets of shape distances; one with the source of variance being modelled as a measurement error and the other where the variance results from landmark separations in the order of protein C_α separation distances.

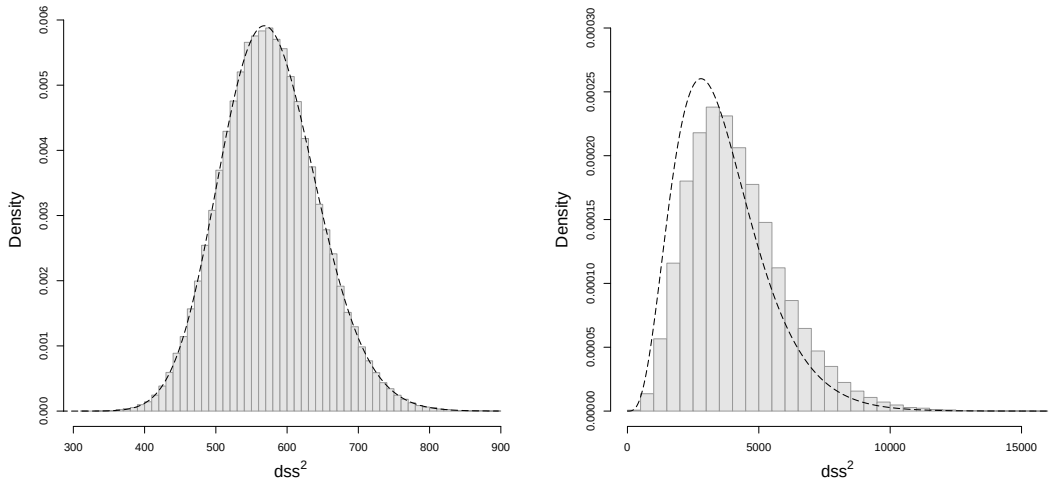


(a) $L = 5$ **(b)** $L = 10$



(c) $L = 50$

Figure 3.1: Plots of true versus estimated standard deviation, where the estimated standard deviation was obtained by maximising the likelihood of the Gamma distribution given in (3.6) over possible values of the variance. Each data point was derived from 10,000 squared chordal size-and-shape distance between configurations where configuration $\boldsymbol{A}$ is always the first L landmarks from the 1a27 protein in the Green and Mardia (2006) data and configuration $\boldsymbol{B}$ is a random sample of 10,000 configurations where each configuration is $\boldsymbol{A}$ with the addition of a spherical normal error, as described by (3.5), followed by the application of a random rotation and translation. The plots indicate that the remaining residues of the original errors on the landmarks are approximated well by the original error distribution and so that (3.6) is a good description of the distribution of chordal distances when there are at least 5 landmarks and the standard deviation is below 10Å.



(a) $L = 50,\ sd = 2$ **(b)** $L = 5,\ sd = 20$

Figure 3.2: Histograms of the squared chordal size-and-shape distance between two configurations where configuration $\boldsymbol{A}$ consists of the first ten landmarks from the 1a27 protein in the Green and Mardia (2006) data and configuration $\boldsymbol{B}$ is configuration $\boldsymbol{A}$ with the addition of a spherical normal error described by (3.5), followed by the application of a random rotation and translation. The sample used is $100,000$ generated iid $\boldsymbol{B}$ configurations. The histograms are over-plotted with the theoretical distributions described by (3.6). (a) shows a good fit for small standard errors, (b) gives an example where the approximation begins to break down for large standard errors.

3.3 Determining a set of starting matching matrices

In this section we show how to determine a set of starting matching matrices that will be used as the foundation for the GProtA algorithm. In Section 2.5.3 of the previous chapter we established a need to constrain the early stages of the matching problem; these matching matrices could also be used to solve that problem. In what is to follow we choose to derive a set of starting matching matrices with four matches, rather than the three suggested by the arguments in Section 2.5.3; this is mainly for the reason that four non-planar points defines orientation in 3 dimensional Euclidean space and this property will be important as our argument develops in Section 3.4.

Before we begin we shall define some useful terminology: Let a matching matrix containing γ matches be written as $\boldsymbol{M}^{(\gamma)}$ where

$$\forall \boldsymbol{M}^{(\gamma)} : \quad \sum_{j=1}^{m}\sum_{k=1}^{n} M_{j,k} = \gamma$$

and let a matching matrix containing γ matches where δ of the γ matches are correct be written as $\boldsymbol{M}^{(\gamma,\delta)}$. A matching matrix where $\delta = \gamma$ will be referred to as a *fully correct* matching matrix, a matching matrix where $0 < \delta < \gamma$ will be referred to as a *partially correct* matching matrix, and matching matrix where $\delta = 0$ will be referred to as a *fully incorrect* matching matrix.

3.3.1 Candidate matching matrices

From (2.3) the number of possible matching matrices containing γ matches is given by

$$S_\gamma = \gamma! \binom{m}{\gamma} \binom{n}{\gamma} \tag{3.7}$$

where m is the number of landmarks in configuration $\boldsymbol{A}$ and n is the number of landmarks in configuration $\boldsymbol{B}$. We have stated that we were going to consider matching matrices with four matches; although it seems intuitive that the number of possible matching matrices will always increase with the number of matches we first check the behaviour of (3.7).

Consider the ratio

$$\begin{aligned}
\frac{S_{\gamma+1}}{S_\gamma} &= \frac{(\gamma+1)!\binom{m}{\gamma+1}\binom{n}{\gamma+1}}{\gamma!\binom{m}{\gamma}\binom{n}{\gamma}} \\
&= \frac{(\gamma+1)!\,\frac{m!}{(\gamma+1)!(m-\gamma-1)!}\,\frac{n!}{(\gamma+1)!(n-\gamma-1)!}}{\gamma!\,\frac{m!}{\gamma!(m-\gamma)!}\,\frac{n!}{\gamma!(n-\gamma)!}} \\
&= \frac{\gamma(\gamma!)^2(m-\gamma)!(n-\gamma)!}{((\gamma+1)!)^2(m-\gamma-1)!(n-\gamma-1)!} \\
&= \frac{\gamma(m-\gamma)(n-\gamma)}{(\gamma+1)^2}\,. \qquad (3.8)
\end{aligned}$$

When this ratio is greater than 1 then S_γ is increasing with γ. The denominator and numerator are plotted separately in Figure 3.3. The region of interest is for $0 \leq \gamma \leq m$.

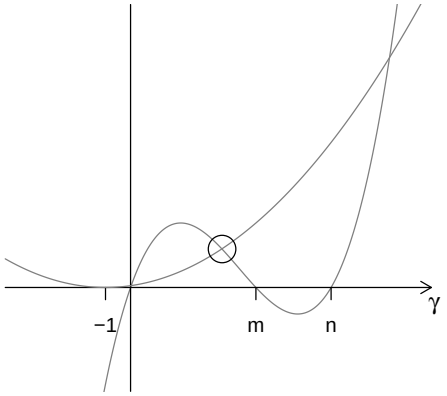


Figure 3.3: Plot of the numerator and denominator of (3.8).

The first crossing point is of little interest as in all practical problems m will be much greater than γ so this will always be very close to $\gamma = 0$. As m gets larger compared to γ the second crossing, circled in Figure 3.3, moves closer to m, indicating that the value of S_γ will be smaller for smaller values of γ that are below m. Hence when γ is small compared to m, S_γ is an increasing function of γ; meaning that there is motivation to choose a small value of γ.

Currently our set of potential candidates is the complete set of $\boldsymbol{M}^{(\gamma)}$ matching matrices and so next we introduce a problem specific constraint against this set. Recall that *local structure* explicitly refers to a piece of structure formed from a set of consecutive residues; this reflects the importance of structural similarity between continuous sections of the amino acid sequence. Using this preference we restrict consideration to sequences of consecutive C_α atoms in each of the protein chains. The set of candidate matching matrices is then based on the matches between combinations of these consecutive sequences. This restric-

tion implies that good starting matching matrices of γ matches will only be found if the proteins being compared have similarity in the conformations of γ consecutive residues. This restriction is not without precedent as it is analogous to the common use of gap penalties in protein sequence alignment (Rodriguez and Schmidler 2014). Figure 3.4 shows a set of sub-configurations with $\gamma = 4$ consecutive labelled landmarks from configurations of eight and ten landmarks.

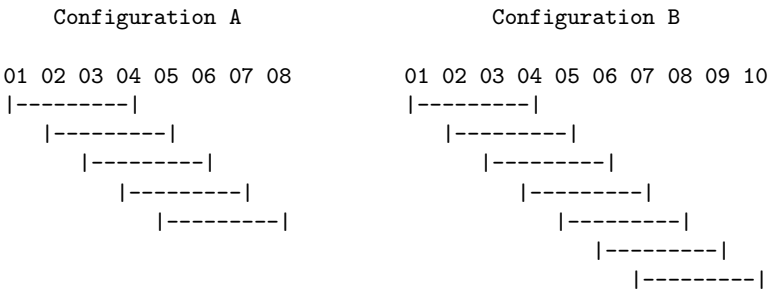


Figure 3.4: A restricted set of sub-configurations.

The total number of combinations of the complete sets of γ consecutive landmarks from each configuration is $2(m - \gamma + 1)(n - \gamma + 1)$; the factor of 2 comes from including the reverse of each of one of the set of configurations i.e. the selection from configuration A would include both the ordered sequences $(01, 02, 03, 04)$ and $(04, 03, 02, 01)$. This regime scales well for practical protein comparison.

There are two points of note to make about our use of sequences of consecutive C_α. The inclusion of the reverse of each of the configurations allows the method to capture alignments that cross along the protein sequence. Additionally we have explicitly decided not to choose the candidates based on possible matches from sequence alignments. Although sequence alignment is a mature research field we do not want to impose either the assumptions or heuristics of these techniques on our method and prefer to rely on statistical reasoning.

3.3.2 A statistical hypothesis test to constrain the set of candidates

We now cut down the set of candidates further using a statistical hypothesis test. The purpose of the test is to give us the information to decide which of a set of candidate starting matching matrices to keep. The statistic to be used is the squared chordal size-and-shape distance between the sub-configurations as defined by some candidate matching

matrix, $\boldsymbol{M}_i^{(\gamma)}$. The hypothesis test looks for evidence that a particular distance is smaller than that expected if we had just guessed – chosen at random – a matching matrix. For a given data set we have the information needed to completely define the null distribution; namely the distribution of all the distances calculated against the two configurations for every possible matching matrix, $\boldsymbol{M}^{(\gamma)}$. In practice the number of matching matrices will be too large and so we will approximate the null distribution from a random sample of these matching matrices. No attempt is made to model the null distribution directly as it is clear from the descriptions of the steric constraints in Section 1.3.1 that determining an analytic distribution would be non-trivial.

In the null distribution the density of fully correct matching matrices is very low. Of the total number of possible matching matrices (3.7) there are $\binom{L}{\gamma}$ that are fully correct, where L is the true total number of correct matches. The ratio of the number of all $\boldsymbol{M}^{(\gamma)}$ matching matrices to fully correct $\boldsymbol{M}^{(\gamma,\delta=\gamma)}$ matching matrices is

$$\frac{\#\mathcal{M}^{(\gamma)}}{\#\mathcal{M}^{(\gamma,\delta=\gamma)}} = \frac{\gamma!\binom{m}{\gamma}\binom{n}{\gamma}}{\binom{L}{\gamma}} \geq \frac{\gamma!\binom{n}{\gamma}}{1} \qquad \text{since} \quad L \leq m \leq n \tag{3.9}$$

where $\mathcal{M}^{(\gamma)}$ and $\mathcal{M}^{(\gamma,\delta=\gamma)}$ are the complete sets of the $\boldsymbol{M}^{(\gamma)}$ and $\boldsymbol{M}^{(\gamma,\delta=\gamma)}$ matching matrices respectively, and $\#\mathcal{M}^{(.)}$ is the count of elements in the set. For any realistically sized protein comparison this ratio is large even for small values of γ, for example the Green and Mardia (2006) data gives a ratio of $\frac{4!\binom{40}{4}\binom{63}{4}}{\binom{36}{4}} = 22,179,913$ for $\gamma = 4$.

As the null distribution is of the test statistic, the chordal size-and-shape distance, it is still left to show that the fully correct matching matrices result in distinctly smaller distances to partially correct and fully incorrect matching matrices. Intuitively this is the case as the chordal distance for matrices where $\delta = \gamma$ is derived from a set of matching errors whereas when $\delta \neq \gamma$ at least one of the landmark separations is of the order of the spatial separation of C_α distances (3.8Å). For the rest of this section we shall make the assumption that the chordal size-and-shape distance is a monotonic function of δ, but we shall return to this to offer more evidence at the end of Section 3.3.3.

Incorporating the selection of candidate matching matrices and the statistical hypothesis test produces the following algorithm for selecting a set of starting matching matrices

SelectStarting

1 Create a random sample of N matching matrices each containing γ matches,
$\mathcal{M}_{\mathsf{s}} = \{\overset{m\times n}{\boldsymbol{M}_i^{(\gamma)}} : i = 1 \dots N\}$.

2 For each member of $\mathcal{M}_{\mathsf{s}}$ filter the $\boldsymbol{A}$ and $\boldsymbol{B}$ configurations using the matching matrix and then calculate the squared chordal size-and-shape distance between the filtered configurations.
The resulting squared distances constitute a sample of the null distribution of squared distances
$\mathcal{D} = \left\{\delta_i : \boldsymbol{M}_i^{(\gamma)} \in \mathcal{M}_{\mathsf{s}}\,,\, \delta_i = \mathrm{d}^2_{\mathsf{ss}}\left(\boldsymbol{M}_i^{(\gamma)}, \boldsymbol{A}, \boldsymbol{B}\right)\right\}$

3 Create a set of candidate starting matching matrices
$\mathcal{M}_{\mathsf{c}} = \{\overset{m\times n}{\boldsymbol{M}_j^{(\gamma)}} : j = 1 \dots 2(m-\gamma+1)(n-\gamma+1)\}$
where the $\boldsymbol{M}_j^{(\gamma)}$ are chosen as detailed at the end of Section 3.3.1.

4 **for** each $\boldsymbol{M}_j \in \mathcal{M}_{\mathsf{c}}$

5 Calculate the p-value against the null distribution of the squared chordal size-and-shape distances.
$\text{p-value}_j = \frac{1}{\#\mathcal{D}} \sum \mathbb{1}_{(\delta_i \le \delta_j,\ \delta_j = \mathrm{d}^2_{\mathsf{ss}}(\boldsymbol{M}_j, \boldsymbol{A}, \boldsymbol{B}),\ \delta_i \in D,\ \boldsymbol{M}_j \in \mathcal{M}_{\mathsf{c}})}$

6 Adjust each p-value with a Bonferroni correction for $2(m-\gamma+1)(n-\gamma+1)$ tests.
$\text{p-adjusted}_j = \text{p-value}_j \times 2(m-\gamma+1)(n-\gamma+1)$

7 Use the p-adjusted values to select a set of starting matching matrices from $\mathcal{M}_{\mathsf{c}}$.
$\mathcal{M}_{\mathsf{start}} = \{\boldsymbol{M}_k : \boldsymbol{M}_k \in \mathcal{M}_{\mathsf{c}},\ \text{p-adjusted}_k \le \alpha\}$

Where

$$\mathrm{d}^2_{\mathsf{ss}}\left(\boldsymbol{M}^{(\gamma)}, \boldsymbol{A},\ \boldsymbol{B}\right) = \inf_{\boldsymbol{\Gamma}\in\mathbf{SO}(m)} \sum_{i=1}^{\gamma-1} \left\| \boldsymbol{H}_{\mathsf{s}}^{(\gamma)} \boldsymbol{a}_{1\dots\gamma} - \boldsymbol{H}_{\mathsf{s}}^{(\gamma)} \boldsymbol{b}_{1\dots\gamma} \boldsymbol{\Gamma} \right\|_2^2$$

and $\boldsymbol{a}$ and $\boldsymbol{b}$ are the co-ordinates of the individual landmarks in configurations $\boldsymbol{A}$ and $\boldsymbol{b}$.

It is important to note that SelectStarting is using statistical reasoning to determine the starting matching matrices. In this way we ensure that we will include all good matches from all high quality optima. The alternative of just selecting a number of the best matches would have the advantage that the numerical expense of the calculation could be controlled. The problem with this approach is that it is easy to find an example where these first best matches would all be used on some local optima thus potentially excluding the global optima or other interesting local optima. This is particularly likely to be the case where there are large regions of secondary structure in both structures.

3.3.3 A worked example of finding starting matching matrices

For this example $\boldsymbol{A}$ is arbitrarily chosen to be the first 10 landmarks of the landmark data used in Green and Mardia (2006) for the `1cyd` protein and $\boldsymbol{B}$ is generated using the model in Section 3.2 with $\sigma^2 = 0.5$. Note that in the Green and Mardia (2006) data for `1cyd`

the first ten landmarks represent a sequence of seven consecutive amino acids followed by a second sequence of three consecutive amino acids. In Figure 3.5 the two configurations used in the example are illustrated under optimal alignment, by minimising the sum of squared distance between labelled landmarks.

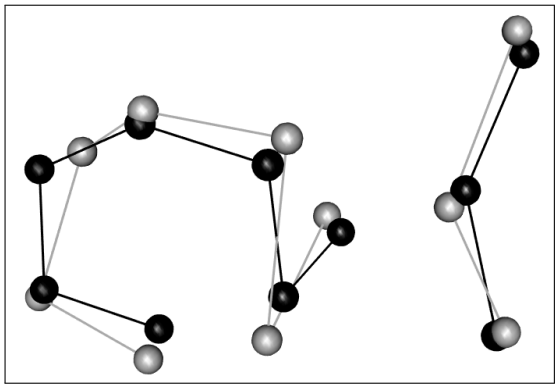


Figure 3.5: A 2D projection of the 3D configurations, under optimal alignment, used as the basis for the example.

For this example the ratio $\frac{\#\mathcal{M}^{(4)}}{\#\mathcal{M}^{(4,\delta=4)}} = 5,040 : 1$. A random sample of $2,500$ matching matrices from $\mathcal{M}^{(4)}$ was used for the null distribution; for reference the counts of δ for the sample are given in Table 3.1.

δ	0	1	2	3	4
count	1672	691	124	13	0

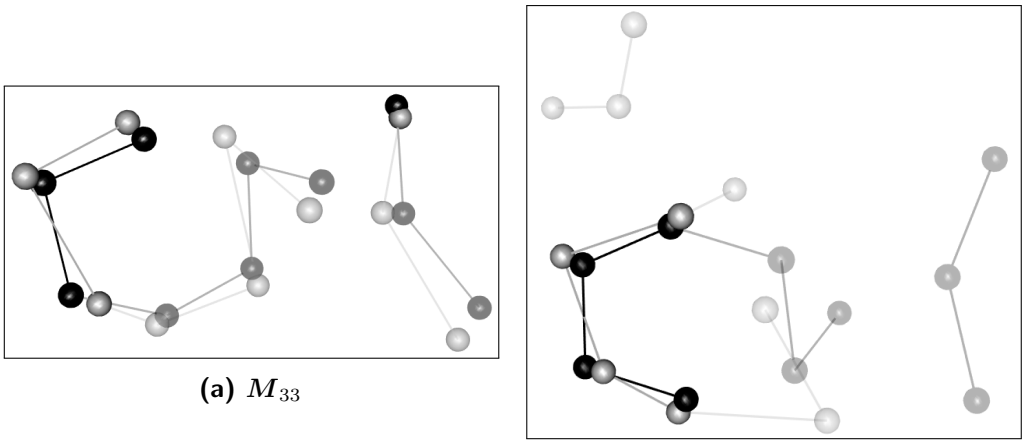
Table 3.1: Counts of the number of matrices for each value of δ in the sample of $\boldsymbol{M}^{(4)}$ used to approximate the null distribution.

There are 98 candidate matching matrices; the distance and p-values calculated for the first 49 (un-reversed) candidates are given in Table 3.2. It can be seen that at a significance level of $\alpha = 0.01$ the adjusted p-values offer evidence to reject the null hypothesis for six of the seven candidates which are in truth fully correct matching matrices. Referring to the jth matching matrix represented in the table as $\boldsymbol{M}_j$, then for $\alpha = 0.01$ there is a single false negative, $\boldsymbol{M}_{33}$; the optimal alignment using this matching matrix is shown in Figure 3.6a. There are also two matching matrices, $\boldsymbol{M}_{11}$ and $\boldsymbol{M}_{16}$, that would be significant at $\alpha = 0.05$; alignment using $\boldsymbol{M}_{11}$ is shown in Figure 3.6b. Both $\boldsymbol{M}_{11}$ and $\boldsymbol{M}_{16}$ represent good local matches in the protein fold even though they are not "correct" matches in the sense of having the correct modelled labelling. This illustrates the important point that these lower order matches may be of interest and should not be treated as unwanted local minima without first careful consideration of the motives of the correspondence search

j	δ	dss^2	p-value	adjusted p-value
1	4	1.9614	0.0000	0.0000
2	0	7.4791	0.0072	0.3528
3	0	12.0991	0.0220	1.0000
4	0	9.5913	0.0128	0.6272
5	0	42.1011	0.2912	1.0000
6	0	43.1525	0.3024	1.0000
7	0	26.2128	0.1296	1.0000
8	0	6.8120	0.0048	0.2352
9	4	2.3042	0.0000	0.0000
10	0	4.8028	0.0028	0.1372
11	0	2.9553	0.0008	0.0392
12	0	36.3985	0.2296	1.0000
13	0	36.8765	0.2336	1.0000
14	0	14.9172	0.0336	1.0000
15	0	7.7134	0.0076	0.3724
16	0	3.7859	0.0008	0.0392
17	4	2.0833	0.0000	0.0000
18	0	11.0012	0.0180	0.8820
19	0	16.3339	0.0420	1.0000
20	0	29.9842	0.1660	1.0000
21	0	16.2768	0.0416	1.0000
22	0	6.6053	0.0044	0.2156
23	0	5.8328	0.0036	0.1764
24	0	4.4049	0.0020	0.0980
25	4	2.0014	0.0000	0.0000
26	0	41.8145	0.2872	1.0000
27	0	24.8520	0.1120	1.0000
28	0	32.9267	0.1956	1.0000
29	0	44.1472	0.3132	1.0000
30	0	27.4785	0.1396	1.0000
31	0	31.2096	0.1756	1.0000
32	0	43.5972	0.3060	1.0000
33	4	2.6496	0.0004	0.0196
34	0	56.9728	0.4636	1.0000
35	0	38.8228	0.2564	1.0000
36	0	41.7516	0.2860	1.0000
37	0	37.1048	0.2356	1.0000
38	0	26.5258	0.1312	1.0000
39	0	39.0737	0.2584	1.0000
40	0	57.6983	0.4708	1.0000
41	4	0.9349	0.0000	0.0000
42	0	42.3982	0.2936	1.0000
43	0	28.6348	0.1524	1.0000
44	0	23.4574	0.1012	1.0000
45	0	21.1118	0.0804	1.0000
46	0	33.3887	0.1996	1.0000
47	0	33.6942	0.2032	1.0000
48	0	48.1966	0.3624	1.0000
49	4	0.7451	0.0000	0.0000

Table 3.2: Results of p-value calculations for the first 49 candidate starting matching matrices (the un-reversed set).

being instigated by the protein scientist. These examples also illustrate the importance of the significance level and choice of method for multiple test correction, as both alignments illustrated in Figure 3.6 would visually be considered good matches; these points will be considered in more detail in Section 3.3.4.



(a) $\boldsymbol{M}_{33}$

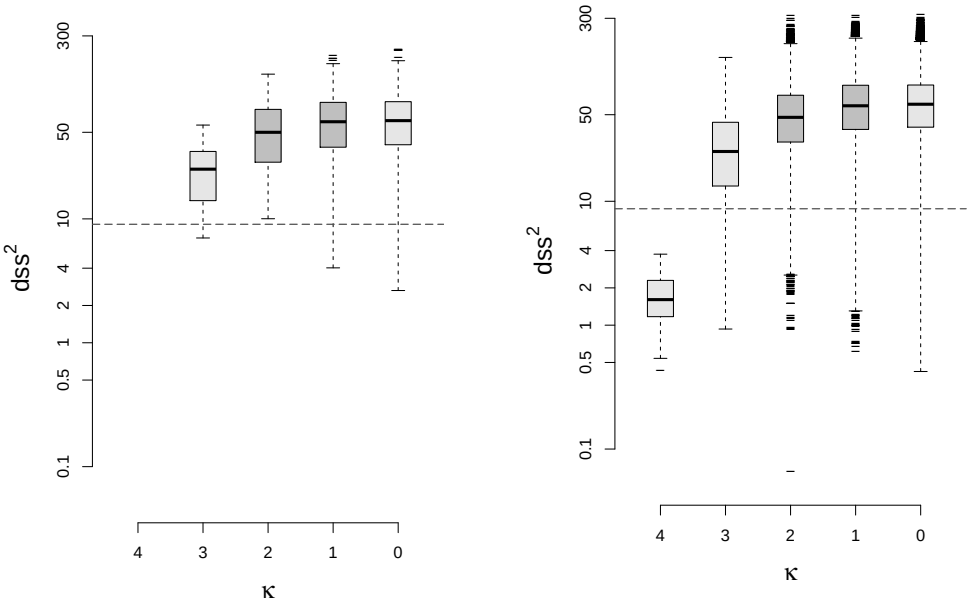
(b) $\boldsymbol{M}_{11}$

Figure 3.6: 2D projections of selected 3D configurations, under optimal alignment, after applying candidate matching matrices. For both matching matrices there is evidence to reject the null hypothesis at the $\alpha = 0.01$ significance level. (a) is the only "correct" matching matrix that is rejected. (b) is significant at $\alpha = 0.05$ and demonstrates that there is potentially interesting good local matching for a set of not "correct" matches.

Now we return to our assumption at the end of Section 3.3.2 that the chordal size-and-shape distance is a monotonic function of δ. To do this we consider the null distribution for the above example. Figure 3.7 shows the distributions of the statistic for each value δ; as the example problem is small this can be shown both for the sample and the complete set of the $1,058,400$ $\boldsymbol{M}^{(4)}$ matching matrices. The expected clear separation of the statistic calculated against fully correct and not fully correct matching matrices is seen, and in the complete set all the distances associated to $\delta = \gamma = 4$ are in the lower 0.01 percentile of the data. Of the matches where $\delta < 4$, a subset offer the statistic in the range occupied by $\delta = 4$; these are similar to the example in Figure 3.6b in that although they are not a subset of the optimal global alignment they may offer interest as local optima.

3.3.4 Further checking the selected starting matching matrices

Section 3.3.2 offers a method for finding a good set of starting matching matrices. In this section we consider the distribution of a possible alternative hypothesis, this being the distribution of the statistic under the spherical error approximation detailed in Section 3.2. As an alternative distribution in the case of the above example this makes sense but in Section 3.4 we shall argue that this model is useful for real data comparison, and that if this is the case then such a distribution offers a way to check the validity of any set of suggested starting matching matrices, offered by any method, if prior knowledge of the



(a) The null sample.

(b) The complete set of the $1,058,400$ $\boldsymbol{M}^{(4)}$ matching matrices.

Figure 3.7: Chordal size-and-shape distances partitioned by δ. The distances are calculated for both the null sample and for the complete set of the $1,058,400$ $\boldsymbol{M}^{(4)}$ matching matrices. The dotted line indicates the lower 0.01 percentile of the data. The plots have been adjusted to take account of the skewed distributions using Hubert and Vandervieren (2007) as implemented in Maechler, Todorov, et al. (2016).

parameters for the alternative can be elicited.

Figure 3.8 plots the theoretical density of the statistic for spherical errors using the value of σ^2 used to generate the test data. This density is plotted over the density formed by a histogram generated from the complete set of $1,058,400$ $\boldsymbol{M}^{(4)}$. The plot indicates the respective upper and lower 0.01 percentiles, which do not overlap.

Estimating σ^2 for real matching could be done in general using the following reasoning. For C_α atoms in protein data, steric constraints lead to an expected C_α separation of 3.8Å (Kleywegt 1997) and the variance on crystallographic determinations of the atom positions is expected to be less than 0.02 (Cruickshank 2006). We confirmed this expectation by examining a large sample ($\approx 10^6$) of distances between consecutive C_α atoms in PDB data for proteins categorised in SCOP2; this analysis found a mean of 3.80 and variance of 0.0024 (a graphical representation of this data can be seen in Figure 4.6 in Chapter 4). Starting with an assumption that a match with an error greater than or equal to 3.8Å is unreliable – as this is the distance between consecutive C_α atoms – in Figure 3.9 we

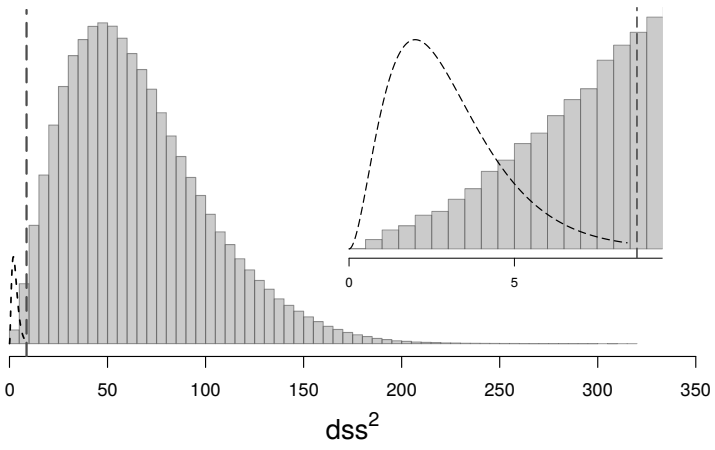


Figure 3.8: Density plot of the chordal squared distance for the complete set of the $1,058,400$ $\boldsymbol{M}^{(4)}$ matching matrices calculated against the $\boldsymbol{A}$ and $\boldsymbol{B}$ configurations used in this illustrative example. The vertical dotted line indicates the lower 0.01 percentile. The theoretical distribution (3.6) is over plotted (not to scale); this is plotted up to the upper 0.01 percentile. The inset shows an enlarged version of the region of interest in the main plot around $\text{dss}^2 = 0$.

plot the 0.0005 to 0.9995 percentile ranges of normal densities with variances of 1.14Å^2 and 0.92Å^2; these represent respectively the variance where the percentile range does not include the expected distance to the adjacent C_α, and the variance where $3.8/2$ is two standard deviations from the mean. Some heuristic following this sort of reasoning would seem reasonable for choosing the σ^2 parameter if assuming spherical errors for an alternative distribution. As can be seen from the typical set of trace plots for the Green and Mardia (2006) implementation in Figure 2.2 once the mode is found the value of σ^2 concentrates at 1.1 ± 0.2 also suggesting a similar value for the σ^2 parameter of an alternative distribution.

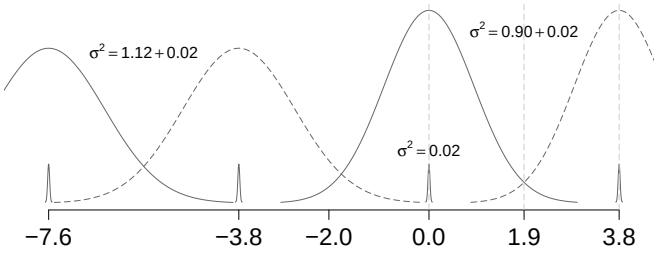


Figure 3.9: Plots of normal densities of varying variance centred with separations of their means of 3.8Å. The densities suggest a mechanism for setting a σ^2 value for the distribution of the alternative hypothesis.

3.4 The GProtA algorithm

In Section 3.3 we detail a hypothesis test that we use to determine good candidates for starting matching matrices where $\delta = 4$. In this section we develop the use of the hypothesis test to iteratively add matches to the matching matrix.

3.4.1 Extending the use of the hypothesis test to iteratively add matches

A matching matrix $\boldsymbol{M}'$ is called *adjacent (increasing)* to $\boldsymbol{M}$ if $\boldsymbol{M}'$ contains exactly one more match than $\boldsymbol{M}$, i.e.

$$\sum_{j=1}^{m}\sum_{k=1}^{n} M'_{j,k} \;=\; 1 + \sum_{j=1}^{m}\sum_{k=1}^{n} M_{j,k}$$

and similarly a matching matrix $\boldsymbol{M}'$ is defined such that it is *adjacent (decreasing)* to $\boldsymbol{M}$ if $\boldsymbol{M}'$ contains exactly one less match than $\boldsymbol{M}$, i.e.

$$1 + \sum_{j=1}^{m}\sum_{k=1}^{n} M'_{j,k} \;=\; \sum_{j=1}^{m}\sum_{k=1}^{n} M_{j,k}\ .$$

The hypothesis test described in Section 3.3.2 can then be used to filter adjacent matching matrices to the original starting matching matrices, and then as we progressively add matches by determining the adjacent matrices to those where we reject the null hypothesis at the previous iteration. This is described in the following NaiveHypothesis algorithm.

NaiveHypothesis

1 Initialise $\mathcal{M}_{\mathsf{s}}^{(\gamma=4)} = \mathcal{M}_{\mathsf{start}}^{(\gamma=4)}$, these being the starting matching matrices described in Section 3.3.

2 **for** $\gamma = 4$ **to** $m - 1$

3 Create the set, $\mathcal{M}_{\mathsf{c}}^{(\gamma+1)}$, of all the adjacent (increasing) matching matrices to the matrices in $\mathcal{M}_{\mathsf{s}}^{(\gamma)}$.

4 For each $\boldsymbol{M}_j^{(\gamma+1)} \in \mathcal{M}_{\mathsf{c}}^{(\gamma+1)}$ calculate the p-value against the null distribution of the squared chordal size-and-shape distances, $\text{p-value}_j = \frac{1}{\#\mathcal{S}} \sum \mathbb{1}_{\left(\delta_i \leq \delta_j,\ \delta_j = \mathrm{d}_{\mathsf{ss}}^2\left(\boldsymbol{M}_j^{(\gamma+1)}, \boldsymbol{A}, \boldsymbol{B}\right),\ \delta_i \in \mathcal{S},\ \boldsymbol{M}_j^{(\gamma+1)} \in \mathcal{M}_{\mathsf{c}}^{(\gamma+1)}\right)}$
where $\mathcal{S}$ is a sample of the null distribution of squared distances for $(\gamma + 1)$ matches
$\mathcal{S} = \left\{\delta_i \,:\, \delta_i = \mathrm{d}_{\mathsf{ss}}^2\left(\boldsymbol{M}_i^{(\gamma+1)}, \boldsymbol{A}, \boldsymbol{B}\right),\ \boldsymbol{M}_i^{(\gamma+1)} \,:\, i = 1 \ldots N\right\}$

5 Adjust each p-value with a Bonferroni correction for $\#\mathcal{M}_{\mathsf{c}}^{(\gamma+1)}$ tests.

6 Use the p-adjusted values to select a set of starting matching matrices from $\mathcal{M}_{\mathsf{s}}^{(\gamma+1)}$.
$\mathcal{M}_{\mathsf{s}}^{(\gamma+1)} = \left\{\boldsymbol{M}_k^{(\gamma+1)} \,:\, \boldsymbol{M}_k^{(\gamma+1)} \in \mathcal{M}_{\mathsf{c}}^{(\gamma+1)},\ \text{p-adjusted}_k \leq \alpha\right\}$

When this algorithm was applied to the data from the illustrative example of Section 3.3.3 the minimum chordal size-and-shape distance for the set of $\mathcal{M}_{\mathsf{s}}^{(\gamma=10)}$ was for the "correct" matching matrix and the routine performed a total of 2382 size-and-shape distance calculations. The number of calculations performed for each value of γ are illustrated in Figure 3.10.

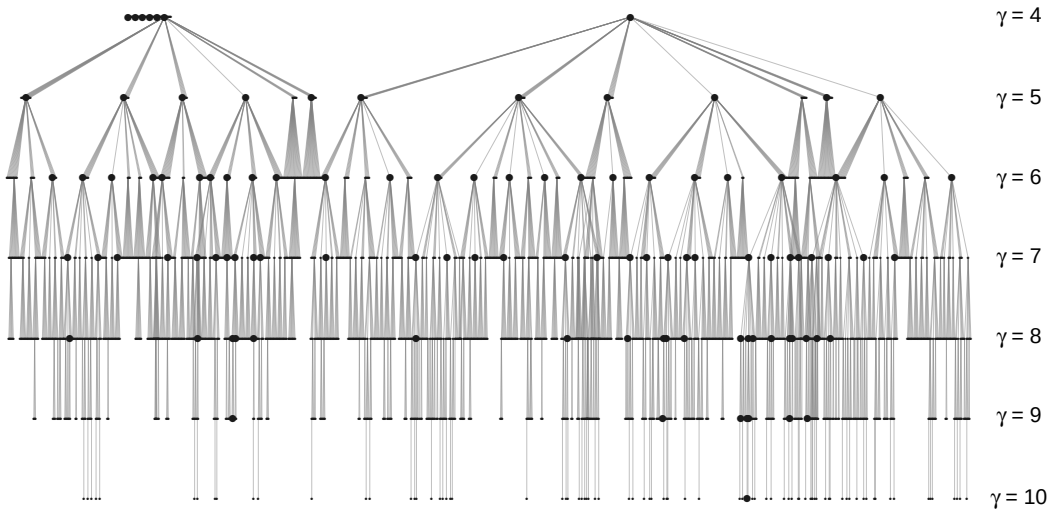


Figure 3.10: Illustration of the number of calculations carried out using the NaiveHypothesis algorithm to find the optimum matching matrix between two configurations of 10 landmarks. The landmarks used were those from the illustrative example of Section 3.3.3. Each edge in the plot represents a size-and-shape distance calculation; in total there are 2382 calculations. The larger black dots represent matching matrices where all the matches are correct.

3.4.2 Greedily following multiple paths to find multiple solutions

In the NaiveHypothesis implementation many of the fully correct matching matrices in the starting matching matrices for $\gamma = 4$ are not being selected to go forward. This is

a result of the combination of the significance level, set at 0.01, and what is most likely a too conservative adjustment for multiple tests in the Bonferroni correction. If we were able to correctly identify all of the fully correct matching matrices at each level then this method would not scale to comparing real protein configurations as there are $\binom{L}{\gamma}$ fully correct matching matrices at each level, where L is the true number of correct matches. For example a pair of configurations that in truth had 100 correct matches would have $\binom{100}{4} = 3,921,225$ matching matrices with $\gamma = 4$ correct matches, which may be manageable, but by the time the algorithm reaches $\gamma = 10$ correct matches there are $\binom{100}{10} \approx 10^{13}$ correct matching matrices.

The overly conservative filtering of results in the NaiveHypothesis algorithm reflects the intentional mechanism of a greedy algorithm (Cormen et al. 2009, Chapter 16) in that at each γ level a greedy algorithm would choose the locally optimal solution or set of solutions. This mechanism presents a clear problem with our NaiveHypothesis algorithm in that it is sensitive to being misdirected by local optima; it is easy to envisage a scenario where the set of solutions chosen to continue to the next level is from a strongly similar subset or subsets of matches that are not part of a larger set that is the global optimum. Our chosen method to avoid this issue is to treat each starting matching matrix separately; in the discussion to follow we will refer to each separate treatment as a branch. The treatment of each starting matching matrix as the base for a separate optimisation leads to the following two further considerations. Firstly, multiple branches may converge on the same matching matrix hence a mechanism is needed to merge such branches to stop duplication of effort. Secondly, branches that lead to local optima will at some point become sub-optimal and hence a stopping criteria must be introduced.

Assuming that the chordal-size-and-shape distance is within the support of the null distribution then the p-value is a monotonic function of the size-and-shape distance. Hence, for a greedy algorithm – where only the single best solution is chosen at each stage – the calculated size-and-shape distances will suffice for selection; this removes the requirement to calculate a null distribution for each γ, and this also avoids the issue of there being a loss of resolution in the tail of the distribution.

We now modify the NaiveHypothesis algorithm considering the above discussion giving a new algorithm that we will refer to as GProtA.

Align($\boldsymbol{M}, \boldsymbol{A}, \boldsymbol{B}$)

1 Create sub-configurations of $\boldsymbol{A}$ and $\boldsymbol{B}$ using $\boldsymbol{M} \Rightarrow \boldsymbol{A}_M$, $\boldsymbol{B}_M$

2 Centre $\boldsymbol{A}_M$ and $\boldsymbol{B}_M$ at the origin $\Rightarrow \boldsymbol{A}_{Mc}$, $\boldsymbol{B}_{Mc}$, $\boldsymbol{\tau} = \boldsymbol{\tau}_B - \boldsymbol{\tau}_A$

3 Optimally rotate $\boldsymbol{B}_{Mc}$ onto $\boldsymbol{A}_{Mc}$ by minimising the sum of squared distances $\Rightarrow \boldsymbol{B}_{Mc}^{(\text{opt})}$, $\boldsymbol{R}$

4 **return** $(\boldsymbol{R}, \boldsymbol{\tau})$

GProtA($\boldsymbol{A}, \boldsymbol{B}$)

1 Initialise $\mathcal{M}_{\text{b}}^{(\gamma=4)} = \mathcal{M}_{\text{start}}^{(\gamma=4)}$

(these being the starting matching matrices described in Section 3.3)

2 $b_{\text{stopped}} = \{\}$

3 **for** $\gamma = 4$ **to** $m - 1$

4 **for** each $b \in \{b \,:\, b = 1 \ldots \#\mathcal{M}_{\text{b}}^{(\gamma=4)}$ AND $b \notin b_{\text{stopped}}\}$ // b represents the index of a branch

5 $(\boldsymbol{R}, \boldsymbol{\tau}) =$ Align$(\boldsymbol{M}_b^{(\gamma)}, \boldsymbol{A}, \boldsymbol{B})$

6 Create the set, $\mathcal{M}_{\text{c}}^{(\gamma+1)}$, of all the adjacent (increasing) matching matrices to $\boldsymbol{M}_{\text{b}}^{(\gamma)}$.

7 **for** each $\boldsymbol{M}_j^{(\gamma+1)} \in \mathcal{M}_{\text{c}}^{(\gamma+1)}$

8 **if** the error on the latest match $>$ tol

9 reject $\boldsymbol{M}_j^{(\gamma+1)}$

10 **elseif** $\boldsymbol{M}_j^{(\gamma+1)} ==$ an already processed $\boldsymbol{M}^{(\gamma+1)}$

11 merge branches

12 Calculate the squared chordal size-and-shape distances, $\delta_j = \text{d}_{\text{ss}}^2\left(\boldsymbol{M}_j^{(\gamma+1)}, \boldsymbol{A}, \boldsymbol{B}\right)$

13 **if** all $\mathcal{M}_{\text{c}}^{(\gamma+1)}$ were rejected

14 add b to b_{stopped}

15 **else**

16 Set $\boldsymbol{M}_b^{(\gamma+1)} = \boldsymbol{M}_j^{(\gamma+1)}$ where j is the index of $\min_j \{\delta_j\}$

An assumption of the greedy algorithm is that at each γ-level the mechanism for choosing the next match is able to distinguish between good and bad matches. In the case of matching the C_α co-ordinate data this is confirmed by simulation, where for values of $\gamma \geq 4$ plots were produced as in Section 3.3.4 that show increasing separation between the size-and-shape distance resulting from modelled Gaussian errors and that found by random samples of matching matrices. The mechanism is then dependant on having high quality initial guesses, which we achieved through the mechanism of the statistical hypothesis test described in Section 3.3.2. If the errors between matches in the real C_α co-ordinate data are well approximated by our modelled Gaussian errors then we should see that the iterative adding of matches will follow a path progressively improving alignment with the mean of the residuals reducing at a rate better than $\sqrt{\gamma}$. This is seen to be the case as we demonstrate using the Green and Mardia (2006) data.

A value needs to be set for the cutoff tolerance (line 8 of GProtA). We set this value

heuristically to 2.7 which is the limit of the 99% confidence interval for a normal distribution with $\sigma^2 = 1.1$ – this is the variance seen in the trace plots for the Green and Mardia (2006) implementation (Figure 2.2) once the mode is found. As previously shown in Figure 3.9 this value is also a sensible intuitive cutoff when considering a spherical normal error model and protein atom separations around the expected separation of 3.8Å. We hence use this value going forward for all protein comparisons.

Figure 3.11 shows the operation of the GProtA algorithm against the Green and Mardia (2006) data and indicates how far each branch continues before it is caught by the stopping criteria or merges with another branch. The longest lived branch, branch 1, continues to 35 matches and the 35 matches agree with the non-marginal matches obtained using the Green and Mardia (2006) method. Figure 3.12 shows a selection of 2D projections of the optimally aligned configurations for the matching matrices from the terminations of a selection of branches.

From Section 3.2, under the spherical error model for a set of "correct" matches the squared chordal size-and-shape distance is approximately distributed as a gamma distribution parameterised with the shape parameter $\alpha = \frac{\gamma m - \frac{1}{2}m(m+1)}{2}$ and rate parameter $\beta = 2\sigma^2$. The expected value of this distribution is

$$\frac{\alpha}{\beta} = \frac{\gamma m - \frac{1}{2}m(m+1)}{4\sigma^2} \tag{3.10}$$

where $m = 3$ as we are working in $\mathbb{R}^3$. As new matches are added and hence γ increases all other terms in (3.10) remain constant so we would expect that if the model assumption holds, and if the new correct matches are added in random sequence, then the squared chordal size-and-shape distance will increase proportionally to γ. This is seen using the Green and Mardia (2006) data and is shown in Figure 3.13 (crosses). When instead of adding randomly chosen correct matches the next best match is added (circles in Figure 3.13) the squared chordal size-and-shape distance will increase at a rate equal to or faster than that of adding randomly chosen matches; this is also seen. The increase when adding next best matches using the Green and Mardia (2006) data is very closely proportional to γ^3, this relation is repeated when we examine alignments for C_α data in Chapter 4; the author is unable explain the specific additional factor of γ^2.

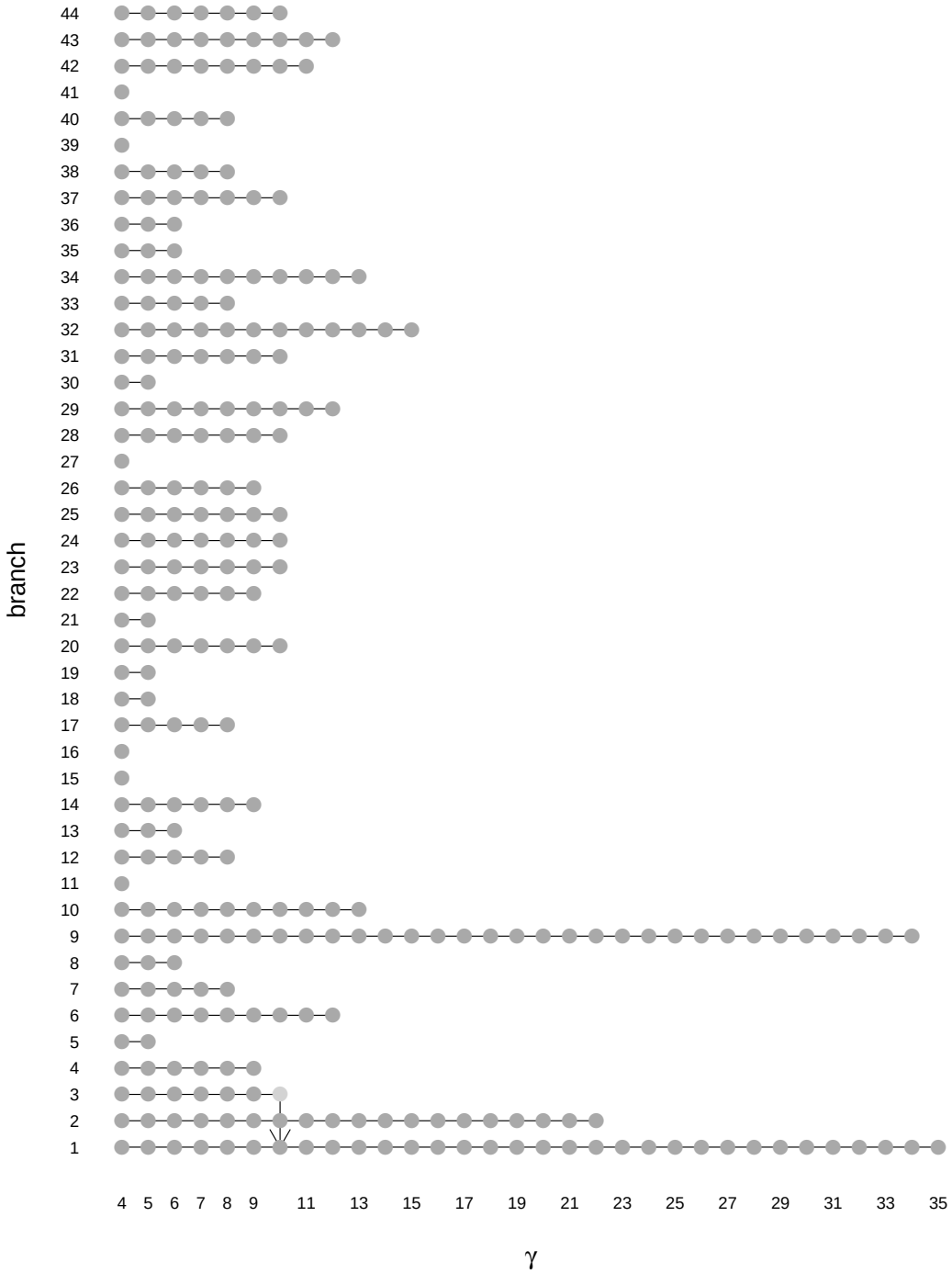
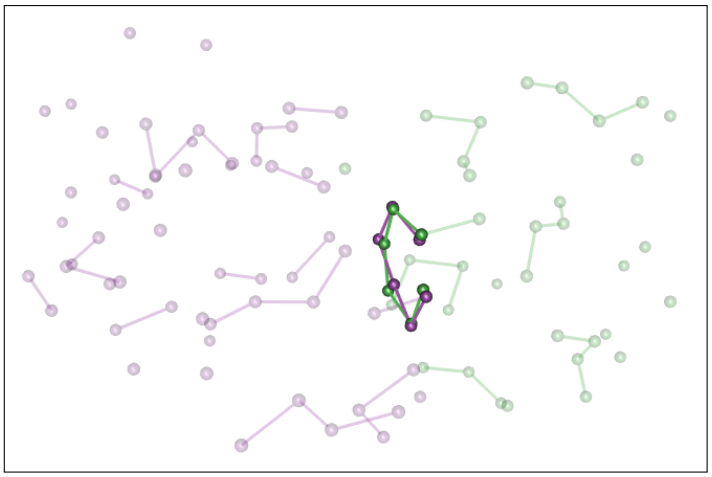


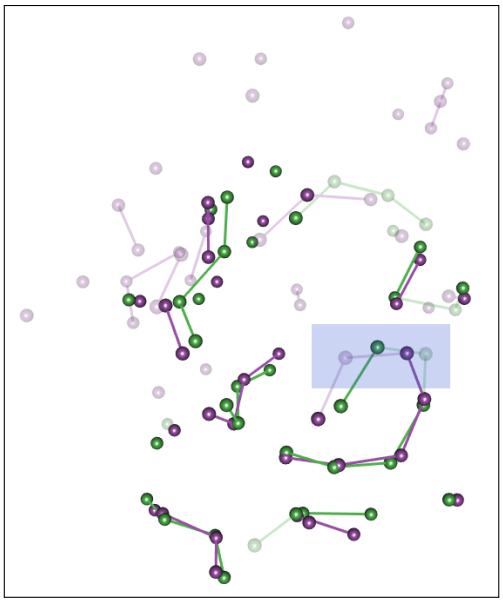
Figure 3.11: The operation of the GProtA algorithm against the Green and Mardia (2006) data. The branches run vertically from top to bottom and are either caught by the stopping criteria in which case they just end, or merge with another branch in which case a line is shown bridging the two branches. The code used is given in Appendix B.3.

The GProtA algorithm potentially will require a maximum of

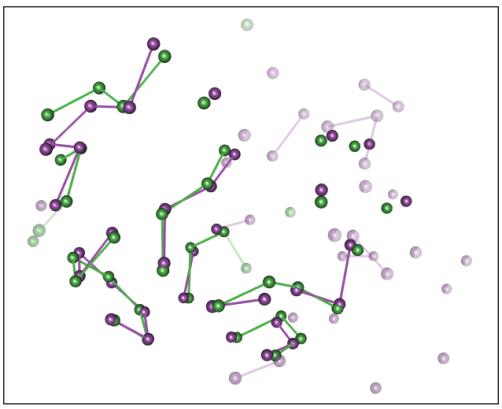
$$\#\mathcal{M}_{\text{start}}^{(\gamma=4)} \times \sum_{\gamma=4}^{m-1} (m-\gamma)(n-\gamma)$$



(a) Branch 36: Matches between 6 C_αs on the outer edges of each protein with no other structure overlap.



(b) Branch 9: Matches between 34 C_αs. This branch contains many of the "correct" matches but early on mismatched two C_αs in the highlighted area.



(c) Branch 1: Matches between 35 C_αs. These matches agree with those obtained using the Green and Mardia (2006) method.

Figure 3.12: A selection of 2D projections of the optimally aligned (minimise squared distance between labelled landmarks) configurations for the matching matrices from the terminations of selected branches.

size-and-shape calculations. Now

$$\sum_{\gamma=4}^{m-1} \gamma = \frac{1}{2}m(m+1) - \frac{1}{2}3(3+1) - m$$
$$= \frac{1}{2}(m^2 - m - 12)$$

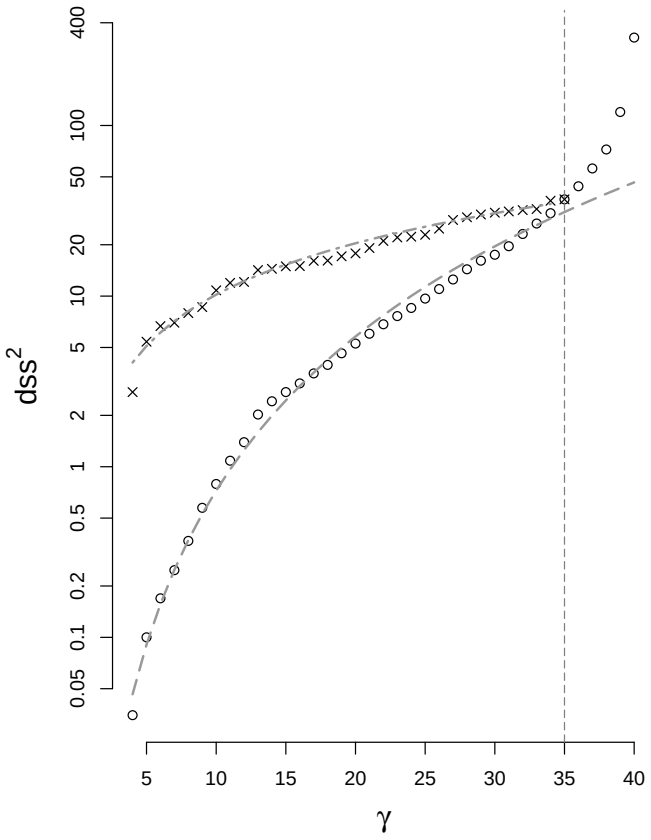


Figure 3.13: Plot of results from the GProtA algorithm against the Green and Mardia (2006) data. The crosses show an example of the relationship of the squared chordal size-and-shape distance to γ as new correct matches are added in random sequence; this closely follows the over plotted line which is proportional to γ. The circles show the squared distance as the consecutive next best match is added; again this very closely follows the overlaid curve, which in this case is proportional to γ^3.

$$\begin{aligned}\sum_{\gamma=4}^{m-1}\gamma^2 &= \frac{1}{6}m(m+1)(2m+1) - \frac{1}{6}3(3+1)(6+1) - m^2 \\ &= \frac{1}{6}(2m^3 - 3m^2 + m - 84)\end{aligned}$$

$$
\begin{aligned}
\sum_{\gamma=4}^{m-1}(m-\gamma)(n-\gamma) &= \sum_{\gamma=4}^{m-1}(mn - m\gamma - n\gamma + \gamma^2) \\
&= \frac{1}{6}(6m^2n - 30mn \\
&\quad - 3m^3 + 3m^2 + 36m \\
&\quad - 3m^2n + 3mn + 36n \\
&\quad + 2m^3 - 3m^2 + m - 84) \\
&= -\frac{1}{6}(m^2 - 7m + 12)(m - 3n + 7)
\end{aligned}
$$

Since in practice the configurations are assigned such that $m \leq n$ then each branch is at worst an $\mathcal{O}(m^2n)$ calculation and there is potential for parallelisation of calculation at each γ level. When used against the Green and Mardia (2006) data, for which the results were given above, 42 branches were followed giving a theoretical maximum number of size-and-shape distance calculations of $44 \times \frac{1}{6}(40^2 - 7 \times 40 + 12)(40 - 3 \times 63 + 7) = 1,387,056$. For the results shown $4,440$ size-and-shape calculations were required with a further $481,668$ being replaced by a single vector length calculation (steps 8 and 9 in the GProtA algorithm).

3.5 A difference measure for proteins

Implicit in our optimisation of the rotation – using SVD to solve the orthogonal Procrustes problem as described in Section 1.2.5.1 – is a modelling assumption that the errors are spherical and iid; we are in effect forcing this to be the case as we are minimising the squared distances with the sum of the distances set to zero. It was shown that this assumption is plausible in Section 3.2 which then makes use of the RMSD, a standard measure of quality for closely related proteins, a good choice as the impact of increasing the number of matches is primarily to reduce the variance. The RMSD is related to the chordal size-and-shape distance detailed in Section 1.2.5.3 by

$$
\mathrm{d}_{\mathsf{rmsd}} = \frac{\mathrm{d}_{\mathsf{ss}}}{\sqrt{L}} \tag{3.11}
$$

where L is the number of matched landmarks.

The nature of the RMSD is to reflect the average quality of individual matches between landmarks, and it does not take account of the number of matches. For example if protein

A matches protein B with the same RMSD as protein A matches protein C, but there are more matched residues between A and B than A and C we would consider protein B to be a better match to protein A than is protein C.

Binary similarity measures offer a measure of the similarity of two sequences as a function of a set of *presence/absence coefficients*. These measures are constructed from the four counts: $\#a$ – present in both $\boldsymbol{A}$ and $\boldsymbol{B}$, $\#b$ – present in $\boldsymbol{A}$, absent in $\boldsymbol{B}$, $\#c$ – absent in $\boldsymbol{A}$, present in $\boldsymbol{B}$ and $\#d$ – absent in both $\boldsymbol{A}$ and $\boldsymbol{B}$. This is illustrated in the Venn diagram in Figure 3.14. In our case $\#d$ has no utility, $\#a$ equates to the number of matches, and

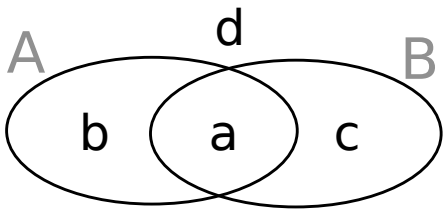


Figure 3.14: Venn diagram showing the presence /absence counts used in binary similarity measures.

$\#b$ and $\#c$ equate to the numbers of unmatched landmarks in configurations $\boldsymbol{A}$ and $\boldsymbol{B}$ respectively.

The simplest linear binary similarity measure was chosen from the extensive list referenced in Hayek (1994, Chapter 9); this is the *Kulczynski coefficient*

$$\frac{1}{2}\left(\frac{a}{a+b}+\frac{a}{a+c}\right)$$

which using our notation for matches becomes

$$s_b = \frac{1}{2}\left(\frac{L}{L+L_{\mathsf{u}}^{(A)}}+\frac{L}{L+L_{\mathsf{u}}^{(B)}}\right) \tag{3.12}$$

where L is the number of matches, $L_{\mathsf{u}}^{(A)}$ is the number of vertices not matched from $\boldsymbol{A}$, and $L_{\mathsf{u}}^{(B)}$ is the number of vertices not matched from $\boldsymbol{B}$. On examination we also see that

the Kulczynski coefficient is equivalent to the average of the conditional probabilities

$$\begin{aligned}
&\frac{\mathrm{P}(\text{match in } \boldsymbol{B} \mid \text{match in } \boldsymbol{A}) \;+\; \mathrm{P}(\text{match in } \boldsymbol{A} \mid \text{match in } \boldsymbol{B})}{2} \\
&\quad \equiv \frac{1}{2}\,(\,\mathrm{P}(\boldsymbol{B} \mid \boldsymbol{A}) + \mathrm{P}(\boldsymbol{A} \mid \boldsymbol{B})\,) \qquad \text{simplifying notation} \\
&\quad = \frac{1}{2}\,(\,\mathrm{P}(\boldsymbol{B} \cap \boldsymbol{A})/\,\mathrm{P}(\boldsymbol{A}) + \mathrm{P}(\boldsymbol{A} \cap \boldsymbol{B})/\,\mathrm{P}(\boldsymbol{B})\,) \\
&\quad = \frac{1}{2}\,(a/(a+b) + a/(a+c))
\end{aligned}$$

which captures the essence of the comparison we are interested in.

We need to combine (3.12) and (3.11). Since we will always get at least one match then s_{b} is in the range $(0, 1]$ with a value of 1 indicating that all landmarks in both configurations are matched. We want more matching to reduce the distance measure and therefore s_{b} should be a divisor. Beyond this there is no obviously apparent reasoning to suggest anything other the simple combination given in (3.13):

$$d = \frac{d_{\mathsf{rmsd}}}{s_b} \qquad \text{where} \quad s_b = \tfrac{1}{2}\left(\frac{L}{L+L_{\mathsf{u}}^{(A)}} + \frac{L}{L+L_{\mathsf{u}}^{(B)}}\right) . \tag{3.13}$$

To test the efficacy of (3.13) we compared a set of proteins chosen from their descriptions in the original SCOP database. We used SCOP rather than SCOP2 because SCOP is a hierarchical classification system which simplifies this test comparison. We chose a test set of proteins were each protein differs from the next by one SCOP classification level; the set chosen and their relationship to each other is given in table 3.3.

Using the set of proteins shown in table 3.3 we calculated the distance measure (3.13) between each protein in the set and the protein with the most specific classification; this is the eight pairs

{ (1oau,1oau), (1oau,1ocw), (1oau,1ohq), (1oau,1nko), (1oau,1lds), (1oau,2w0p), (1oau,2j71), (1oau,3erj) }

We then used a comparison of the order of this set ordered by the distance measure to the set ordered by SCOP classification as a measure of the efficacy of the difference measure. The distance measure (3.13) gave the correct order. Neither the shape measure alone nor the binary similarity measure recovered the ordering.

It is noted that Davies et al. (2007) propose a binary similarity measure derived from

PDB ID	class	fold	super family	family	domain	species	protein
1oau	48724	48725	48726	48727	88543	88560	92720
1ocw	48724	48725	48726	48727	88543	88560	92779
1ohq	48724	48725	48726	48727	88543	88562	93028
1nko	48724	48725	48726	48727	89176	89177	85829
1lds	48724	48725	48726	48942	88600	88602	73858
2w0p	48724	48725	81296	81290	141023	141024	153739
2j71	48724	49451	49452	158932	158933	158937	147896
3erj	51349	102461	102462	102463	117608	117609	158194

(a) Proteins and their SCOP classifications labels.

similarity to 1oau	PDB ID							
	1oau	1ocw	1ohq	1nko	1lds	2w0p	2j71	3erj
same class	yes	yes	yes	yes	yes	yes	yes	no
same fold	yes	yes	yes	yes	yes	yes	no	no
same super family	yes	yes	yes	yes	yes	no	no	no
same family	yes	yes	yes	yes	no	no	no	no
same domain	yes	yes	yes	no	no	no	no	no
same species	yes	yes	no	no	no	no	no	no
same protein	yes	no	no	no	no	no	no	no

(b) Table clarifying the classification relationship of each protein to protein 1oau.

Table 3.3: A set of proteins chosen from the SCOP database such that one differs from the next by one classification level.

the Poisson model in Green and Mardia (2006), but they do not propose the measure in combination with a measure of shape difference.

4

Protein structure classification

4.1 Overview

In this chapter we apply the GProtA algorithm and difference measure proposed in Chapter 3 to protein atom co-ordinate data from the Protein Data Bank. Our aim is to reproduce classifications from the SCOP2 categorisation database. Before we begin the categorisation experiments we discuss in detail the structure of the SCOP2 database as it relates to querying category relationships, and detail the acquisition of data from the Protein Data Bank along with our criteria for excluding malformed data from this study.

4.2 The SCOP2 database

In this chapter we will use the typographical convention of writing database table names and database attributes (column names) in fixed width fonts, so the 'domain' database table is written `domain`; this helps, for example, to distinguish references to the `domain` database table from use of the noun 'domain' when referring to a functional unit of a protein.

Version two of the *Structural Classification of Proteins* database (SCOP2) (Andreeva et al. 2014) contains manually curated classifications for a set of *protein domains* whose

experimentally-determined atomic co-ordinates are available in the *Protein Data Bank*[1]. SCOP2 data is released in the form of a MySQL[2] database. Appendix C.1 gives details of obtaining and installing the database.

SCOP2 only provides documentation for the two web based interface tools, the *Browser* and *Graph* tools[3], and there is no documentation relating to the MySQL database. Examination of the database reveals a data structure that is not normalised, in the *database design* sense (Date 2012, Chapter 3), and so there is no referential integrity to help reverse engineer the data representation. The following describes the inference used to determine the entity representations and relationships.

The usual definition of a *domain* is as a contiguous region of the protein amino acid chain that folds into a semi-independent, compact and stable structure (Richardson 1981, Section II.I). Within the SCOP2 data the `domains` table is used somewhat differently; from the SCOP2 documentation:

> In SCOP2, there is no a priori division of protein structures into domains, in which one domain size fits all possible relationships. The protein domain is defined as a unit of relationship and its boundaries are dependent on a given relationship. In general, the protein domain, representing a child node of the SCOP2 graph, is larger than that representing its parental node domain of the same protein, especially if there is more than one parental node for a given child node. For example, Fold is an attribute of a single structural domain, but the domains representing Family and Superfamily can span over more than one structural domain. Similarly, the Family domain may contain more than one non-overlapping Superfamily domains.

Hence in the database a particular `domain`, as identified by a `domains.dom_name`, represents the part of a specific protein that relates to a specific relationship; it is allowed that different `domains` can be defined as having exactly the same constituent residues. Despite the reuse of existing terminology this is a useful facet of the data structure as the `domains` then capture the parts of a protein's structure that the curators have considered in the determination of a specific relationship. The `domains` are represented as one-to-many subsections of *asymmetric units*; here asymmetric unit is used only in the sense that it is protein chain data that is represented in the asymmetric unit structural elements (`_struct_asym`) in the PDB data. The illustrations of alignments in Figure 4.19 later in this chapter give a feel for the structural relationship between a domain and an asymmetric unit.

The `domains.serial`, `domains.pdb_code`, `domains.pdb_chain`, `domains.pdb_begin` and `domains.pdb_end` at-

[1] Detailed in Section 4.3.
[2] `http://dev.mysql.com/doc/`
[3] The SCOP2 documentation pages were last accessed 2017-11-12 at `http://scop2.mrc-lmb.cam.ac.uk/about.html`.

tributes relate to the PDB entry for the domain. The domains.seq_begin, domains.seq_end, domains.exd_db_name, domains.ext_db_id and domains.seq_length attributes relate to entries in some other database; in 3,540 of the 3,566 domains this relates to annotation in the *SWISS-PROT protein sequence database* (Bairoch and Apweiler 2000), this is when domains.exd_db_name equals SPROT. For the remaining 25 domains domains.exd_db_name equals PDB, and in these cases the meaning of the external database related attribute entries is unclear. There are 3,566 distinct domains described in the SCOP2 domains table, which are derived from 995 proteins. Table 4.1 contains an illustrative set of entries from the domains table.

node	dom_name	serial	pdb_code	pdb_chain	pdb_begin	pdb_end	seq_begin	seq_end	exd_db_name	ext_db_id	seq_length	
PR:5000536	PR-8003547-1F3UA	1	1F3U	A	2	119	2	119	SPROT	P13984	249	(1)
SF:3000142	SF-8003550-1F3UA	1	1F3U	A	2	119	2	119	SPROT	P13984	249	
SF:3000142	SF-8003556-1F3UB	1	1F3U	B	5	153	5	153	SPROT	P35269	517	(2)
PR:5000865	PR-8003553-1F3UB	1	1F3U	B	5	153	5	153	SPROT	P35269	517	
SP:6001114	SP-8003629-1J6UA	1	1J6U	A	1	446	1	446	SPROT	Q9WY73	457	(3)
HF:1100005	HF-8003639-1J6UA	1	1J6U	A	102	282	102	282	SPROT	Q9WY73	457	
CF:2000082	CF-8003589-3DBHA	1	3DBH	A	6	168	6	168	SPROT	Q13564	534	(4)
CF:2000082	CF-8003589-3DBHA	2	3DBH	A	488	534	488	534	SPROT	Q13564	534	
CF:2000082	CF-8003592-3DBHB	1	3DBH	B	39	204	60	225	SPROT	Q8TBC4	463	(5)
CF:2000082	CF-8003592-3DBHB	2	3DBH	B	288	348	309	369	SPROT	Q8TBC4	463	
CF:2000021	CF-8017502-2Z3YA	1	2Z3Y	A	274	312	274	312	SPROT	O60341	852	(6)
CF:2000021	CF-8017502-2Z3YA	2	2Z3Y	A	570	654	570	654	SPROT	O60341	852	
CF:2000021	CF-8017502-2Z3YA	3	2Z3Y	A	764	831	764	831	SPROT	O60341	852	
CF:2000124	CF-8003551-1F3U_	1	1F3U	A	2	119	2	119	SPROT	P13984	249	(7)
CF:2000124	CF-8003551-1F3U_	2	1F3U	B	5	153	5	153	SPROT	P35269	517	

Figure 4.1: Example of data from the SCOP2 `domains` table: (1) and (2) Domains entries from the same protein that contain exactly the same set of residues; (3) Example of domain entries with overlapping sets of residues; (4), (5), (6) and (7) Examples of domain entries made from multiple separated residue sets; (7) includes residues from distinct asymmetric units.

4.3 The Protein Data Bank

Protein Data Bank refers to the *PDB archive* which is a repository for freely available molecular structure data. The *worldwide Protein Data Bank* (wwPDB) (Berman et al. 2003) refers to a set of organisations that offer windows onto the data of the PDB archive through internet accessible resources, i.e. the wwPDB makes the source files of the PDB archive available. The PDB archive file format is `PDBx/mmCIF`, where `mmCIF` stands for *macromolecular Crystallographic Information File*; the mmCIF specification is an extension of Version 1.1 of the *CIF* format[4] created by the *Union of Crystallography* (IUCr) for the description of crystallographic experiments and their results. The use of mmCIF files by the PDB is described by the *PDB Data Dictionary*.[5] The modal count of residues per protein in the Protein Data Bank is 259.[6]

The atomic co-ordinate data in the PDBx/mmCIF files is encoded by the `ATOM` records. The SCOP2 MySQL database `domains` table attributes map to the PDBx/mmCIF `ATOM` records as follows:

`pdb_code`**:** This is simply the *Protein Data Bank ID*[7] used to identify the protein containing the domain.

`pdb_chain`**:** A particular asymmetric unit. In the PDB data the asymmetric units are captured by `_struct_asym`[8] entries. The `_struct_asym.id` is referenced by the `_atom_site.label_asym_id`[9] in the `ATOM` data. The `ATOM` entries contain an alternate free text field, the `_atom_site.auth_asym_id`[10], that is documented as

> [an] alternative identifier for _atom_site.label_asym_id that may be provided by an author in order to match the identification used in the publication that describes the structure

The correspondence between the `pdb_chain` SCOP2 attribute and the PDB is undocumented. Comparison of the two data sets indicates that the `pdb_chain` maps to the alternative free text field, `_atom_site.auth_asym_id`.

[4] `http://www.iucr.org/resources/cif/spec/version1.1/cifsyntax`
[5] PDBx/mmCIF Data Dictionary Version 4.064 retrieved from `http://mmcif.wwpdb.org/` during May of 2016.
[6] The mode residue count was taken from `https://www.rcsb.org/pdb/static.do?p=general_information/pdb_statistics` (retrieved 2017-10-03).
[7] `http://www.rcsb.org/pdb/staticHelp.do?p=help/advancedsearch/pdbIDs.html`
[8] `http://mmcif.wwpdb.org/dictionaries/mmcif_pdbx_v40.dic/Categories/struct_asym.html`
[9] `http://mmcif.wwpdb.org/dictionaries/mmcif_pdbx_v40.dic/Items/_atom_site.label_asym_id.html`
[10] `http://mmcif.wwpdb.org/dictionaries/mmcif_pdbx_v40.dic/Items/_atom_site.auth_asym_id.html`

pdb_begin **and** pdb_end**:** The order of the amino acid residues is captured in the PDB data by the _entity_poly_seq.num[11] entries. These entries are required to be sequential integers and are referenced by the _atom_site.label_seq_id[12] entries in the ATOM data. It is not required that there are ATOM entries for every _entity_poly_seq entry. The ATOM entries contain an alternate sequence id that is an uncontrolled field that is neither guaranteed to be sequential or an integer, the _atom_site.auth_seq_id[13], which is documented as

> may be provided by an author in order to match the identification used in the publication that describes the structure

The correspondence between the pdb_begin and pdb_end SCOP2 attributes and the PDB is undocumented. Comparison of the two data sets indicates that pdb_begin and pdb_end SCOP2 entries map to the alternative _atom_site.auth_seq_id.

The ATOM record can optionally include an _atom_site.pdbx_PDB_model_num[14] entry. The pdbx_PDB_model_num differentiates multiple data sets for the same molecule; when there are multiple pdbx_PDB_model_num entries in the file this indicates that there are multiple different experimental determinations of the structure. The SCOP2 data contains no reference to the model and there is no machine readable information within the PDB file indicating if one model is preferred to another. When multiple models are present we arbitrarily choose the first model in the file.

Each ATOM record contains an _atom_site.occupancy[15] attribute. When a residue is observed with a single conformation then there is a single ATOM record for each residue atom with an _atom_site.occupancy of 1.00. A residue may be observed in multiple conformations; when this is the case some or all of the residue atoms will have multiple ATOM records, each with a _atom_site.occupancy value of less than 1.00. Each of the multiple ATOM records has a _atom_site.label_alt_id[16] attribute, and this attribute is used to separate the ATOM records for the particular residue into distinct conformations. There is no PDB documentation indicating if a label_alt_id for one residue relates to the same label_alt_id in another residue, i.e. if the 'A' label_alt_id is chosen for one residue the 'A' label_alt_id should be chosen for

[11] http://mmcif.wwpdb.org/dictionaries/mmcif_pdbx_v40.dic/Items/_entity_poly_seq.num.html
[12] http://mmcif.wwpdb.org/dictionaries/mmcif_pdbx_v40.dic/Items/_atom_site.label_asym_id.html
[13] http://mmcif.wwpdb.org/dictionaries/mmcif_pdbx_v40.dic/Items/_atom_site.auth_asym_id.html
[14] http://mmcif.wwpdb.org/dictionaries/mmcif_pdbx_v40.dic/Items/_atom_site.pdbx_PDB_model_num.html
[15] http://mmcif.wwpdb.org/dictionaries/mmcif_pdbx_v40.dic/Items/_atom_site.occupancy.html
[16] http://mmcif.wwpdb.org/dictionaries/mmcif_pdbx_v40.dic/Items/_atom_site.label_alt_id.html – This is documented as referencing atom_sites_alt records (http://mmcif.wwpdb.org/dictionaries/mmcif_pdbx_v40.dic/Categories/atom_sites_alt.html), however there are no atom_sites_alt records in any PDB file relating to the SCOP2 domains.

all other residues. The only relevant comment in the documentation gives the none specific advice[17]

> Tip: When dealing with PDB entries with multiple co-ordinates, you often need to pay close attention. It is not always possible to select just the 'A' conformations and throw away the 'B' conformations. You need to look carefully in each case and make sure that there are not any bad contacts between mobile sidechains.

It is not required that all atoms within a residue demonstrating multiple conformations have multiple occupancy i.e. some atoms within the residue may only have a single `ATOM` record. The SCOP2 database entries make no reference to the multiple conformations. For this analysis the `ATOM` records relating to the conformation with the highest occupancy are chosen and a random choice is made when the occupancy levels are equal. Each residue is treated in isolation i.e. the choice of a particular `_atom_site.label_alt_id` for one residue does not affect our choice for other residues. Figure 4.2 shows parts of the `_atom_site` section of a `PDBx/mmCIF` representing two residues with multiple occupancy.

[17] `http://pdb101.rcsb.org/learn/guide-to-understanding-pdb-data/dealing-with-coordinates` fetched November 2016.

```
loop_
_atom_site.group_PDB
_atom_site.id
_atom_site.type_symbol
_atom_site.label_atom_id
_atom_site.label_alt_id   <---
_atom_site.label_comp_id
_atom_site.label_asym_id
_atom_site.label_entity_id
_atom_site.label_seq_id
_atom_site.pdbx_PDB_ins_code
_atom_site.Cartn_x
_atom_site.Cartn_y
_atom_site.Cartn_z
_atom_site.occupancy      <---
_atom_site.B_iso_or_equiv
_atom_site.Cartn_x_esd
_atom_site.Cartn_y_esd
_atom_site.Cartn_z_esd
_atom_site.occupancy_esd
_atom_site.B_iso_or_equiv_esd
_atom_site.pdbx_formal_charge
_atom_site.auth_seq_id
_atom_site.auth_comp_id
_atom_site.auth_asym_id
_atom_site.auth_atom_id
_atom_site.pdbx_PDB_model_num

                 |                                  |
                 V                                  V
ATOM   11   N N   . MET A 1 3   ? 15.024 41.549 24.920 1.00 11.73 ? ? ? ? ? ? 1    MET A N   1
ATOM   12   C CA  A MET A 1 3   ? 16.444 41.315 25.096 0.52 12.33 ? ? ? ? ? ? 1    MET A CA  1
ATOM   13   C CA  B MET A 1 3   ? 16.479 41.289 24.980 0.48 13.55 ? ? ? ? ? ? 1    MET A CA  1
ATOM   14   C C   . MET A 1 3   ? 16.816 39.898 25.524 1.00 16.61 ? ? ? ? ? ? 1    MET A C   1
ATOM   15   O O   . MET A 1 3   ? 17.725 39.192 25.083 1.00 24.95 ? ? ? ? ? ? 1    MET A O   1
ATOM   16   C CB  A MET A 1 3   ? 17.061 42.228 26.163 0.52 15.64 ? ? ? ? ? ? 1    MET A CB  1
ATOM   17   C CB  B MET A 1 3   ? 17.236 42.331 25.781 0.48 12.27 ? ? ? ? ? ? 1    MET A CB  1
ATOM   18   C CG  A MET A 1 3   ? 18.527 42.466 25.830 0.52 14.39 ? ? ? ? ? ? 1    MET A CG  1
ATOM   19   C CG  B MET A 1 3   ? 18.497 41.930 26.525 0.48 14.49 ? ? ? ? ? ? 1    MET A CG  1
ATOM   20   S SD  A MET A 1 3   ? 19.683 41.263 26.464 0.52 27.74 ? ? ? ? ? ? 1    MET A SD  1
ATOM   21   S SD  B MET A 1 3   ? 19.412 43.348 27.173 0.48 16.91 ? ? ? ? ? ? 1    MET A SD  1
ATOM   22   C CE  A MET A 1 3   ? 21.050 42.371 27.032 0.52 3.76  ? ? ? ? ? ? 1    MET A CE  1
ATOM   23   C CE  B MET A 1 3   ? 20.260 42.746 28.621 0.48 11.15 ? ? ? ? ? ? 1    MET A CE  1

ATOM   1264 N N   A PRO A 1 155 ? 36.210 35.207 56.471 0.60 5.17  ? ? ? ? ? ? 153  PRO A N   1
ATOM   1265 N N   B PRO A 1 155 ? 36.238 35.184 56.468 0.41 4.75  ? ? ? ? ? ? 153  PRO A N   1
ATOM   1266 C CA  A PRO A 1 155 ? 36.375 35.083 57.929 0.60 3.94  ? ? ? ? ? ? 153  PRO A CA  1
ATOM   1267 C CA  B PRO A 1 155 ? 36.429 35.046 57.924 0.41 3.94  ? ? ? ? ? ? 153  PRO A CA  1
ATOM   1268 C C   A PRO A 1 155 ? 35.765 36.201 58.790 0.60 4.09  ? ? ? ? ? ? 153  PRO A C   1
ATOM   1269 C C   B PRO A 1 155 ? 35.908 36.202 58.790 0.41 4.37  ? ? ? ? ? ? 153  PRO A C   1
ATOM   1270 O O   A PRO A 1 155 ? 35.416 35.968 59.957 0.60 5.95  ? ? ? ? ? ? 153  PRO A O   1
ATOM   1271 O O   B PRO A 1 155 ? 35.807 36.072 60.021 0.41 3.24  ? ? ? ? ? ? 153  PRO A O   1
ATOM   1272 C CB  A PRO A 1 155 ? 37.898 35.046 58.107 0.60 5.74  ? ? ? ? ? ? 153  PRO A CB  1
ATOM   1273 C CB  B PRO A 1 155 ? 37.947 34.930 58.115 0.41 5.30  ? ? ? ? ? ? 153  PRO A CB  1
ATOM   1274 C CG  A PRO A 1 155 ? 38.420 34.493 56.821 0.60 5.86  ? ? ? ? ? ? 153  PRO A CG  1
ATOM   1275 C CG  B PRO A 1 155 ? 38.473 34.449 56.804 0.41 5.64  ? ? ? ? ? ? 153  PRO A CG  1
ATOM   1276 C CD  A PRO A 1 155 ? 37.512 35.013 55.788 0.60 5.78  ? ? ? ? ? ? 153  PRO A CD  1
ATOM   1277 C CD  B PRO A 1 155 ? 37.553 35.033 55.797 0.41 5.47  ? ? ? ? ? ? 153  PRO A CD  1
```

Figure 4.2: A sample of the `_atom_site` section of a PDBx/mmCIF from the PDB entry with PDB ID 2CAR. The chosen residues demonstrate multiple conformations.

When we process each PDBx/mmCIF file we capture a set of metadata to be investigated as possible covariates when assessing the efficacy of the derived distance measure as a predictor of SCOP2 classification. Specifically these are:

- The date of submission, or the date of update, to the PDB; this is derived from the `database_PDB_rev` records.[18]
- The experimental method.[19]
- The resolution.[20] When the experimental method is 'NMR' no resolution is given as "non-diffraction methods such as NMR do not have a resolution specified"[21]. A method for inferring a resolution for NMR entries is posited by Berjanskii et al. (2012), however this is not considered here.

We derive two data sets from the PDBx/mmCIF data files relating to the SCOP2 domains. These are sets of atomic co-ordinates for each unique PDB ID and asymmetric unit pair, and sets of atomic co-ordinates for each domain identified by `domains.dom_name`. These data are cached in the form of co-ordinate CSV files; Figure 4.3 illustrates a snippet of the content of one of these files.

[18] http://mmcif.wwpdb.org/dictionaries/mmcif_std.dic/Categories/database_PDB_rev.html
[19] http://mmcif.wwpdb.org/dictionaries/mmcif_std.dic/Items/_exptl.method.html
[20] http://mmcif.wwpdb.org/dictionaries/mmcif_std.dic/Items/_refine.ls_d_res_high.html
[21] http://www.rcsb.org/pdb/staticHelp.do?p=help/advancedsearch/xRayResolution.html

serial	label_atom_id	label_alt_id	label_comp_id	cartn_x	cartn_y	cartn_z	occupancy	auth_asym_id	auth_seq_id
1	N	NA	MET	-12.817	34.708	32.066	1	A	1
1	CA	NA	MET	-11.663	35.505	32.484	1	A	1
1	C	NA	MET	-11.155	36.336	31.331	1	A	1
1	O	NA	MET	-11.533	36.107	30.171	1	A	1
1	CB	NA	MET	-10.552	34.586	32.994	1	A	1
1	CG	NA	MET	-9.186	34.949	32.513	1	A	1
1	SD	NA	MET	-7.936	34.291	33.604	1	A	1
1	CE	NA	MET	-8.908	33.216	34.674	1	A	1
1	N	NA	GLU	-10.311	37.308	31.630	1	A	2
1	CA	NA	GLU	-9.728	38.123	30.555	1	A	2
1	C	NA	GLU	-8.412	37.497	30.198	1	A	2
1	O	NA	GLU	-7.578	37.245	31.069	1	A	2

Figure 4.3: A sample of the atom co-ordinate data captured from the PDBx/mmCIF for each SCOP2 domain. The data is displayed as a table for clarity. The `label_alt_id`, `occupancy` and `auth_asym_id` are supplied as a record of the choices made in the selection of the records from the PDBx/mmCIF file.

In addition we add a set of tables to our local copy of the SCOP2 MySQL database that capture metadata and issues relating to the PDB entries that were found when processing the PDB files. The `pdb` and `domain` tables contain metadata about the full protein entries and domains respectively. The `pdb_issue`, `domain_issue` and `asym_unit_issue` tables contain details of issues relating to the data associated with the PDB entries, domains and asymmetric units respectively. Figure 4.4 shows these tables. The `*_issue` issues are discussed in Section 4.4.

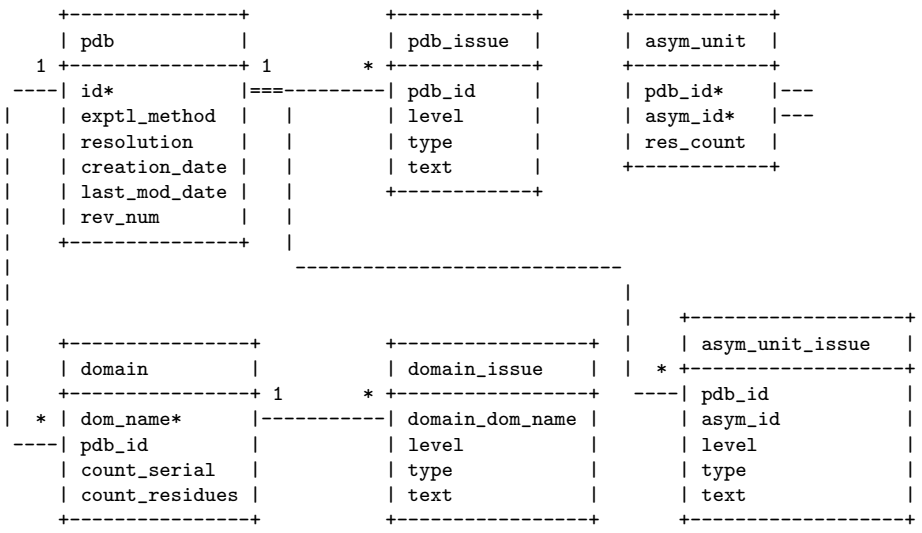


Figure 4.4: The tables added to the SCOP2 MySQL database containing metadata relating to the PDB entries. The `*_issue` tables capture the issues found when processing the `PDBx/mmCIF` files.

The code used to efficiently fetch and cache the PDB mmCIF files is given in Appendix C.3 and the code used to generate the issue table (Figure 4.4) is given in Appendix C.2. The code in Appendix C.2 also creates a CSV file for each domain and asymmetric unit included in the SCOP2 database; these files contain the ATOM records relevant to each domain and asymmetric unit.

4.4 Excluded data

The `PDBx/mmCIF` files were fetched on 2016-12-27 and the SCOP2 database was the release dated as 2014-02-05.

Five issues relating to the PDB data are considered when deciding which data to include in each part of the analysis.

1. **The entry has been marked as 'obsolete'**

   Obsoletion of PDB entries is covered in the 'Changes to Entries' section of the 'wwPDB Processing Procedures and Policies Document'.[22]

2. **The data is malformed**

   Some PDBx/mmCIF files contain malformed data. Usually this is as a result of the submitter either misunderstanding or intentionally subverting the data definition. A common example is where residues with multiple conformations are submitted without using the `label_alt_id` and `occupancy` mechanism that is illustrated in Figure 4.2; instead the data presents multiple `ATOM` rows for each residue atom with some arbitrary modification to the `label_atom_id` to indicate additional conformations. There is no automated way to process these types of entry.

3. **PDB validation errors**

   Validation of the PDB data against established physical principles is captured in the PDB validation reports;[23] the results of the validation reports are included in the published PDBx/mmCIF data files.[24] Validation reports were made available after the release of SCOP2.[25]

4. **Residue data is missing**

   In the SCOP2 data, for each row in the `domains` table there are `pdb_begin` and `pdb_end` attributes referencing the `_atom_site.auth_seq_id` entries in the PDBx/mmCIF data. Domains are noted where residues are missing from the data in the range between `pdb_begin` and `pdb_end`.

5. **The PDB entry has been modified post the release of SCOP2**

   SCOP2 does not state the version or date of the PDB entry that was used to make relationship determinations. PDB entries are noted where the `PDBx/mmCIF` file has been modified since the date of the SCOP2 release.

In SCOP2 the `domains` table references 3,566 distinct domains derived from 995 PDB entries. Issues 1 and 2 are treated as 'errors' and all data where issues 1 or 2 are observed are excluded from all analysis. Data 'errors' account for 82 PDB entries which in turn affect

[22] `http://www.wwpdb.org/documentation/policy#toc\_changes` (Accessed June 2016)
[23] `http://wwpdb.org/validation/validation-reports` (viewed June 2016)
[24] `http://mmcif.wwpdb.org/dictionaries/mmcif_pdbx_v40.dic/Groups/validate_group.html`
[25] wwPDB News 2016-03-05 *New NMR and 3DEM Validation Reports for Archived PDB Structure* `http://wwpdb.org/news/news?year=2016#5764490799cccf749a90cdf4`.

264 domains. Figure 4.5 gives example rows from our pdb_iisue, domain_issue, asym_unit_iisue tables. A summary of counts for ERRORs and WARNINGs is given in Table 4.1.

Issue Type	Level	PDBs	PDB IDs	Domains
1 (obsolete)	ERROR	6	1UUL, 1WE3, 2FB1, 2GY9, 2H7M, 2PLI	14
2 (malformed)	ERROR	76	1A5Z, 1AU7, 1AWC, 1B72, 1B8I, 1BC8, 1BLO, 1CF7, 1D5Y, 1DP7, 1DUX, 1E3O, 1F6O, 1FJL, 1GAD, 1GDT, 1H88, 1H8A, 1HDG, 1HLV, 1IC8, 1IF1, 1IG7, 1IGN, 1IJW, 1J75, 1JGG, 1K61, 1K78, 1K82, 1LE8, 1LFU, 1LLC, 1LMB, 1PDN, 1PJJ, 1PP7, 1PUE, 1PUF, 1PZG, 1QBJ, 1R2Z, 1REP, 1RM4, 1SFU, 1U78, 1VDC, 1WOT, 1WOU, 1YO5, 1ZH5, 1ZQ3, 2AO7, 2A94, 2C6Y, 2EC9, 2FO1, 2IRF, 2J5K, 2LKX, 2NRA, 2P6R, 2PIO, 2PJP, 2R1J, 2R5Y, 2UUB, 2XOJ, 3BDN, 3CMC, 3CRO, 3HTS, 3L2C, 6PAX, 9ANT, 9LDT	250
3 (validation)	WARNING	884		3039
4 (missing residue data)	WARNING	229		640
5 (changed post SCOP2)	WARNING	38	1DCQ, 1DXL, 1F76, 1GRO, 1GUZ, 1HD2, 1HKU, 1IOZ, 1I1O, 1I8T, 1IUK, 1JAY, 1KS9, 1LDM, 1LLD, 1LNQ, 1LSS, 1MV8, 1OBB, 1OBF, 1PJQ, 1QMV, 1TOF, 1UXG, 1VAO, 1VJI, 1VYR, 2BI7, 2BKA, 2F8A, 2FXA, 2G82, 2OAN, 2RDZ, 2V64, 3HNA, 3OL4, 3TQU	130

Table 4.1: Counts relating to errors and warnings relating to domains entries in SCOP2. Note that ERRORs are exclusive such that a PDB entry identified as having an ERROR is excluded and these PDBs and their associated domains are not considered for, or included in WARNING. WARNINGs are not exclusive so a particular domain and / or PDB entry could be associated with all three WARNING types. The WARNING counts relating to PDBs are counts of distinct PDBs i.e. if a PDB entry is the parent of multiple domains with a particular WARNING the PDB is only counted once.

Issue types 3, 4 and 5 are treated as 'warnings'; data is not excluded that have WARNINGs against them but these may be referenced when discussing results. Apart from issue type 5 it can be assumed that the included data is as it was seen by those making the categorisations. In the case of issue type 5 when the data has been modified post the release of SCOP2 these warnings will only be considered if they apply to data where our categorisation fails to agree with SCOP2.

pdb_id	level	type	text
2FB1	ERROR	OBSOLETE	missing mmCIF file
1D5Y	ERROR	UNEXPECTED_ON_ATOM_INSERT	DBD::mysql::st execute failed: Bad label_atom_id: >>O5'<<
2EC9	ERROR	UNEXPECTED_ON_ATOM_KEEP_INSERT	DOMAIN CF-8003777-2EC9H - DBD::mysql::db do failed: Duplicate entry 'N-60-H' for key 'label_atom_id'

(a) Example ERROR rows from the pdb_issue table.

pdb_id	level	type	text
3OL4	WARN	MODIFIED_POST_SCOP2	mmCIF file last modified 2015-04-22
3DBH	WARN	VALIDATION	36 A (torsion), 38 A (torsion), 39 A (torsion), 63 A (torsion), 65 A (torsion), 66 A (torsion), 79 A <snip>

domain_dom_name	level	type	text
CF-8002341-1MIJA	WARN	AUTH_SEQ_ID_MISSING	1314, 1315, 1316, 1317, 1318, 1319, 1320, 1321, 1322, 1323, 1324, 1325, 1326
SP-8002542-1JT6B	WARN	VALIDATION	4 (torsion), 18 (torsion), 21 (close_contact), 33 (torsion), 41 (torsion), 44 (torsion), 89 (torsion <snip>

pdb_id	asym_id	level	type	text
1OFC	X	WARN	VALIDATION	734 (rmsd_angle), 751 (peptide_omega), 752 (peptide_omega), 756 (torsion), 794 (chiral), 795 (torsio <snip>

(b) Example WARNING rows from the pdb_issue, domain_issue, asym_unit_issue tables.

Figure 4.5: Example ERROR and WARNING rows relating to PDB data.

Figure 4.6 shows boxplots of the RMSD distances between consecutive C_α atoms for random samples of domains taken from the sets of domains with and without warnings. The expected distance is 3.8Å (Kleywegt 1997). This demonstrates that in data both with and without warnings there are relatively few significant departures from the expected distance, and also shows that the PDB validation warnings do not always show departures. As any significant departures may impact on the efficacy of our matching method departures will be checked when our matching fails to agree with SCOP2; this check will be calculated directly against the data and we will not rely on the PDB validation reports.

Figure 4.6: A plot comparing the distance between consecutive alpha carbons for domains that are flagged with validation warnings in the PDB and those that are not. The two groups are made up of random samples of domains that are in SCOP2. Domains where processing found data errors are excluded.

4.5 SCOP2 classification

The following is an edited down version of the SCOP2 *classification category definitions* taken from `http://scop2.mrc-lmb.cam.ac.uk/about.html` on 2016-08-15.

Evolutionary relationships:

Hyperfamily: A common region shared by different superfamilies. Unlike a Superfamily region, it can be smaller than a structural domain, as it does not include the superfamily-specific regions. The Hyperfamily level helps separate the most populated and structurally-diverse superfamilies.

Superfamily: A common structural region shared by different protein families. The Superfamily domain spans from the beginning of the first common secondary structural element to the end of the last common conserved structural element. Importantly, the domains representing Family and Superfamily levels can span over more than one structural domain. Moreover, the Family region may contain more than one non-overlapping Superfamily region.

Family: A conserved sequence region shared by closely-related proteins. The boundaries of a family domain region are frequently similar to those derived by sequence family databases.

Species: Generally the individual gene product represented by its full-length sequence. A notable exception is viral polyproteins, for which protein species correspond to mature posttranslational products.

Protein: This groups together orthologous proteins and is defined as a sub-sequence that can be found on its own. In general, at this level the boundaries of the relationship are the same as in Species. The exceptions are fusion proteins found in some organisms, the Protein parts of which correspond to stand-alone proteins in other biological species.

Structural relationships:

Fold: An attribute of structural domain defined strictly on the basis of global structural features as originally described. These features are: the composition of secondary structures in the domain core, their architecture and topology. If the structural domains of evolutionarily related proteins differ by any of these parameters, they can be classified into different folds.

IUPR: (Intrinsically Unstructured Protein Region.) This annotates the protein regions that do not adopt globular folded structure. They are present in some of the experimentally-determined structures, where they adopt an ordered conformation upon binding to other macromolecules. IUPRs may occur in both individual proteins and protein sequence families.

Table 4.2 gives counts of the number of distinct protein relationships of each type encoded in the SCOP2 database.

Protein relationships	category	rank	count
Evolutionary relationships	Hyper Family	HF	296
	Super Family	SF	492
	Family	FA	606
	Protein	PR	635
	Species	SP	780
Structural relationships	Fold	CF	681
	IUPR		

Table 4.2: Count of the number of distinct protein relationships encoded in the SCOP2 database.

An important point to note is that the classification system is not hierarchical (as it was in SCOP). Although traditionally the evolutionary relationships from superfamily to protein are represented as a tree hierarchy, in SCOP2 each specific instance of a categorisation is represented as a node in an acyclic graph, which is implemented in the database using a slightly modified version of a *Levelled Adjacency List Model* implemented in the `closure` table. An implication of this is that a protein may not have a categorisation at a particular level even when it has a categorisation at the level above and below, for example a protein with both superfamily and species categorisations may not be categorised under family. In addition a protein may be represented with multiple domains that belong to different categorisations at the same level of evolutionary relationship.

In its basic form the GProtA algorithm detailed in Chapter 3 is particularly suited to finding the best global alignment of structures, which in turn is then particularly suited to comparison at the family level which is typically the sequence level comparison made when deducing evolutionary history. If a family categorisation is correctly attributed then most higher level categorisations can be inferred either uniquely or with a small number of alternatives, and the number of possible lower level categories is significantly reduced. Family, species and protein level categorisations are all generally expected to be global matches at the asymmetric unit level, of these the family categorisation would be expected to be the smallest subset of structure being matched.

4.6 Comparing SCOP2 families

For this section we consider the complete set of protein domains taken from SCOP2 family categories that include at least 10 members and where the members are associated to asymmetric units of no greater than 250 residues. This set consists of 5 families containing a total of 72 domains. The complete set of these data is given in Table 4.3.[26] There are 92 of the 295 family categories defined in SCOP2 that contain more than one protein.

For the first illustration we pick one of the domains at random, `FA-8002409-1FTZA`, and calculate the distance, using GProtA for alignment and the difference measure in (3.13), to all other members of the set. The results are plotted in Figure 4.7. Of the 22 members of

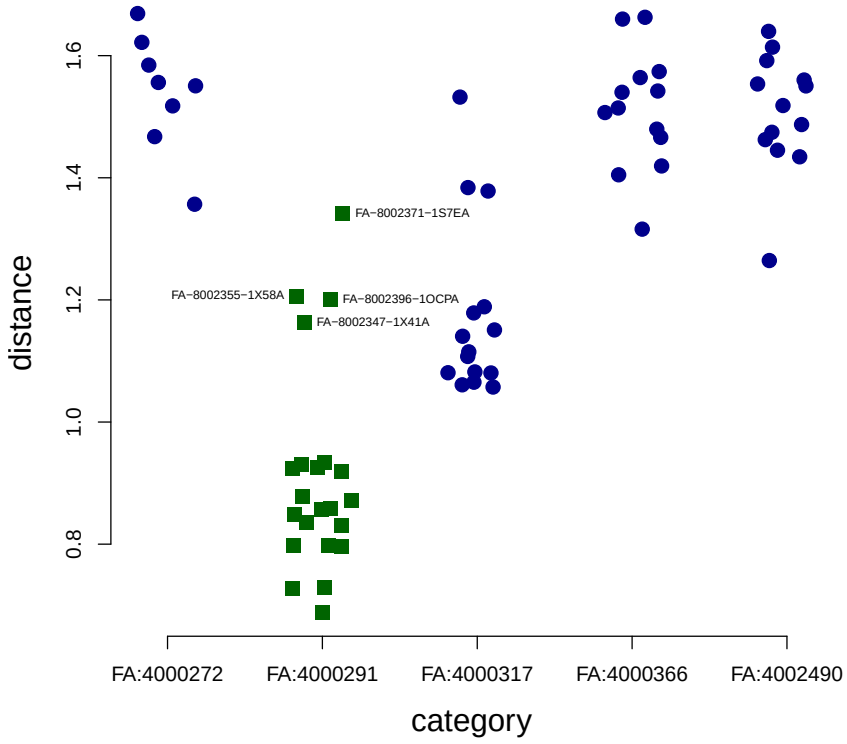


Figure 4.7: Distance calculated between the domain `FA-8002409-1FTZA` and all of the other 71 domains in the set. The horizontal alignment of the groups is jittered to help distinguish between individual members.

the `FA:4000291` family (that contains `FA-8002409-1FTZA` domain) 18 distinguish correctly from the other 4 families. For the 4 incorrectly distinguished domains there is no apparent error in the matching from visual examination of the alignment. There is also no apparent

[26] The SQL used to find this set is given in Appendix C.4.1.

domain	PDB ID	asym unit	node	domain res count	asym unit res count	exptl method	last mod date	dom warn	asym unit warn
FA-8002765-1FT9A	1FT9	A	FA:4000272	132	210	X-RAY DIFFRACTION	2009-02-24	1	1
FA-8002757-1ZYBA	1ZYB	A	FA:4000272	147	230	X-RAY DIFFRACTION	2011-07-13	1	1
FA-8002725-1I5ZA	1I5Z	A	FA:4000272	132	201	X-RAY DIFFRACTION	2009-02-24	1	1
FA-8002746-3E5UC	3E5U	C	FA:4000272	139	219	X-RAY DIFFRACTION	2011-07-13	1	1
FA-8002761-2ZCWA	2ZCW	A	FA:4000272	113	194	X-RAY DIFFRACTION	2009-02-24	0	1
FA-8002731-2OZ6A	2OZ6	A	FA:4000272	129	201	X-RAY DIFFRACTION	2011-08-10	1	1
FA-8002753-2GAUA	2GAU	A	FA:4000272	137	218	X-RAY DIFFRACTION	2011-07-13	1	1
FA-8002769-2BGCA	2BGC	A	FA:4000272	135	235	X-RAY DIFFRACTION	2011-07-13	1	1
FA-8002357-2CRAA	2CRA	A	FA:4000291	57	70	SOLUTION NMR	2009-02-24	1	1
FA-8002396-1OCPA	1OCP	A	FA:4000291	66	67	SOLUTION NMR	2009-02-24	1	1
FA-8002353-1X2NA	1X2N	A	FA:4000291	59	73	SOLUTION NMR	2009-02-24	1	1
FA-8002355-1X58A	1X58	A	FA:4000291	49	62	SOLUTION NMR	2009-02-24	1	1
FA-8002387-1HDPA	1HDP	A	FA:4000291	63	63	SOLUTION NMR	2009-02-24	1	1
FA-8002351-2E1OA	2E1O	A	FA:4000291	57	70	SOLUTION NMR	2009-02-24	1	1
FA-8002403-1BW5A	1BW5	A	FA:4000291	65	66	SOLUTION NMR	2009-02-24	1	1
FA-8002344-2CQXA	2CQX	A	FA:4000291	59	72	SOLUTION NMR	2009-02-24	1	1
FA-8002386-1FTTA	1FTT	A	FA:4000291	67	68	SOLUTION NMR	2009-02-24	1	1
FA-8002371-1S7EA	1S7E	A	FA:4000291	46	147	SOLUTION NMR	2009-02-24	1	1
FA-8002369-2ECCA	2ECC	A	FA:4000291	63	76	SOLUTION NMR	2009-02-24	1	1
FA-8002413-2ECBA	2ECB	A	FA:4000291	76	89	SOLUTION NMR	2009-02-24	1	1
FA-8002378-1ENHA	1ENH	A	FA:4000291	54	54	X-RAY DIFFRACTION	2009-02-24	0	0
FA-8002365-1WH7A	1WH7	A	FA:4000291	67	80	SOLUTION NMR	2009-02-24	1	1
FA-8002347-1X41A	1X41	A	FA:4000291	45	60	SOLUTION NMR	2009-02-24	1	1
FA-8002411-1VNDA	1VND	A	FA:4000291	76	77	SOLUTION NMR	2009-02-24	1	1
FA-8002363-1WH5A	1WH5	A	FA:4000291	67	80	SOLUTION NMR	2009-02-24	1	1
FA-8002356-2CUFA	2CUF	A	FA:4000291	76	95	SOLUTION NMR	2009-02-24	1	1
FA-8002332-1UHSA	1UHS	A	FA:4000291	64	72	SOLUTION NMR	2009-02-24	1	1
FA-8002367-1WI3A	1WI3	A	FA:4000291	58	71	SOLUTION NMR	2009-02-24	1	1
FA-8002345-2CUEA	2CUE	A	FA:4000291	67	80	SOLUTION NMR	2009-02-24	1	1
FA-8002409-1FTZA	1FTZ	A	FA:4000291	69	70	SOLUTION NMR	2009-02-24	1	1
FA-8002350-1X2MA	1X2M	A	FA:4000291	51	64	SOLUTION NMR	2009-02-24	1	1
FA-8002301-1XC5A	1XC5	A	FA:4000317	68	68	SOLUTION NMR	2009-02-24	1	1
FA-8002293-2CKXA	2CKX	A	FA:4000317	83	83	X-RAY DIFFRACTION	2009-02-24	0	0
FA-8002295-2CRGA	2CRG	A	FA:4000317	57	70	SOLUTION NMR	2009-02-24	1	1
FA-8002302-2CU7A	2CU7	A	FA:4000317	65	72	SOLUTION NMR	2009-02-24	1	1
FA-8002307-1FEXA	1FEX	A	FA:4000317	59	59	SOLUTION NMR	2009-02-24	1	1
FA-8002290-2CJJA	2CJJ	A	FA:4000317	63	63	X-RAY DIFFRACTION	2011-07-13	1	1
FA-8002311-1A5JA	1A5J	A	FA:4000317	55	110	SOLUTION NMR	2009-02-24	1	1
FA-8002297-2IW5B	2IW5	B	FA:4000317	65	133	X-RAY DIFFRACTION	2011-07-13	1	1
FA-8002312-1UG2A	1UG2	A	FA:4000317	81	95	SOLUTION NMR	2009-02-24	1	1
FA-8002299-2CQQA	2CQQ	A	FA:4000317	59	72	SOLUTION NMR	2009-02-24	1	1
FA-8002303-1WGXA	1WGX	A	FA:4000317	60	73	SOLUTION NMR	2009-02-24	1	1
FA-8002294-2AJEA	2AJE	A	FA:4000317	97	97	SOLUTION NMR	2009-02-24	1	1
FA-8002310-1A5JA	1A5J	A	FA:4000317	53	110	SOLUTION NMR	2009-02-24	1	1
FA-8002300-2CQRA	2CQR	A	FA:4000317	60	73	SOLUTION NMR	2009-02-24	1	1
FA-8002251-1IRZA	1IRZ	A	FA:4000317	64	64	SOLUTION NMR	2009-02-24	1	1
FA-8004157-1MYOA	1MYO	A	FA:4000366	118	118	SOLUTION NMR	2009-02-24	1	1
FA-8004154-1IKND	1IKN	D	FA:4000366	215	215	X-RAY DIFFRACTION	2009-02-24	1	1
FA-8004166-1OY3D	1OY3	D	FA:4000366	207	220	X-RAY DIFFRACTION	2009-02-24	1	1
FA-8004080-1YCSB	1YCS	B	FA:4000366	130	193	X-RAY DIFFRACTION	2009-02-24	1	1
FA-8004144-3UI2A	3UI2	A	FA:4000366	138	234	X-RAY DIFFRACTION	2012-06-06	1	1
FA-8004163-1OT8A	1OT8	A	FA:4000366	186	209	X-RAY DIFFRACTION	2011-07-13	1	1
FA-8004153-3B7BA	3B7B	A	FA:4000366	234	236	X-RAY DIFFRACTION	2009-02-24	1	1
FA-8004104-1IHBA	1IHB	A	FA:4000366	156	156	X-RAY DIFFRACTION	2009-02-24	1	1
FA-8004107-1BI7B	1BI7	B	FA:4000366	125	125	X-RAY DIFFRACTION	2009-02-24	1	1
FA-8004159-1K1AA	1K1A	A	FA:4000366	228	228	X-RAY DIFFRACTION	2009-02-24	1	1
FA-8004097-1BD8A	1BD8	A	FA:4000366	156	156	X-RAY DIFFRACTION	2009-02-24	1	1
FA-8004128-3C5RA	3C5R	A	FA:4000366	122	125	X-RAY DIFFRACTION	2011-07-13	1	1
FA-8004111-1UOHA	1UOH	A	FA:4000366	223	223	X-RAY DIFFRACTION	2011-07-13	1	1
FA-8017485-2P4PA	2P4P	A	FA:4002490	82	84	X-RAY DIFFRACTION	2011-07-13	1	1
FA-8017483-2NQWA	2NQW	A	FA:4002490	87	87	X-RAY DIFFRACTION	2011-07-13	1	1
FA-8017481-2O3GA	2O3G	A	FA:4002490	76	76	X-RAY DIFFRACTION	2011-07-13	1	1
FA-8017489-3LLBA	3LLB	A	FA:4002490	81	81	X-RAY DIFFRACTION	2010-03-16	1	1
FA-8017479-2P13A	2P13	A	FA:4002490	85	85	X-RAY DIFFRACTION	2011-07-13	1	1
FA-8017474-4HG0A	4HG0	A	FA:4002490	81	232	X-RAY DIFFRACTION	2013-02-06	1	1
FA-8017482-2R2ZA	2R2Z	A	FA:4002490	84	84	X-RAY DIFFRACTION	2011-07-13	1	1
FA-8017488-2P3HA	2P3H	A	FA:4002490	98	98	X-RAY DIFFRACTION	2011-07-13	1	1
FA-8017478-2PLSA	2PLS	A	FA:4002490	84	86	X-RAY DIFFRACTION	2011-07-13	1	1
FA-8017486-2OAIA	2OAI	A	FA:4002490	80	80	X-RAY DIFFRACTION	2011-07-13	1	1
FA-8017477-2RK5A	2RK5	A	FA:4002490	84	86	X-RAY DIFFRACTION	2009-02-24	1	1
FA-8017487-3DEDA	3DED	A	FA:4002490	87	87	X-RAY DIFFRACTION	2011-07-13	1	1
FA-8017480-3LAEA	3LAE	A	FA:4002490	78	81	X-RAY DIFFRACTION	2010-01-19	1	1

Table 4.3: The set of protein domains used to test agreement with SCOP2 categorisation at the family level. These are the complete sets of SCOP2 family categories that include at least 10 members and where the members are associated to asymmetric units of no greater than 250 residues. Residue counts are given for both the domain and associated asymmetric unit.

correlation in their captured metadata. In all 4 cases the domain length is at the short end of the distribution; `FA-8002347-1X41A`, `FA-8002371-1S7EA` and `FA-8002355-1X58A` being the three shortest domains in our data set. However, there does not seem an obvious explanation for the incorrect distinction.

The comparison of domains offers evidence that our alignment and distance measure work well when the important region of the protein has been isolated, i.e. the domain region has been identified by the expert as the region of the asymmetric unit that contains the residues important in identifying the family category.

Next we make a comparison for the same set using the asymmetric unit associated with the domain. If we are correctly identifying the global best match this should offer similar results to the domain comparison, as the family categorisation is generally attributed to global similarity at the asymmetric unit level. Note that the asymmetric unit definitions are taken from the Protein Data Bank.

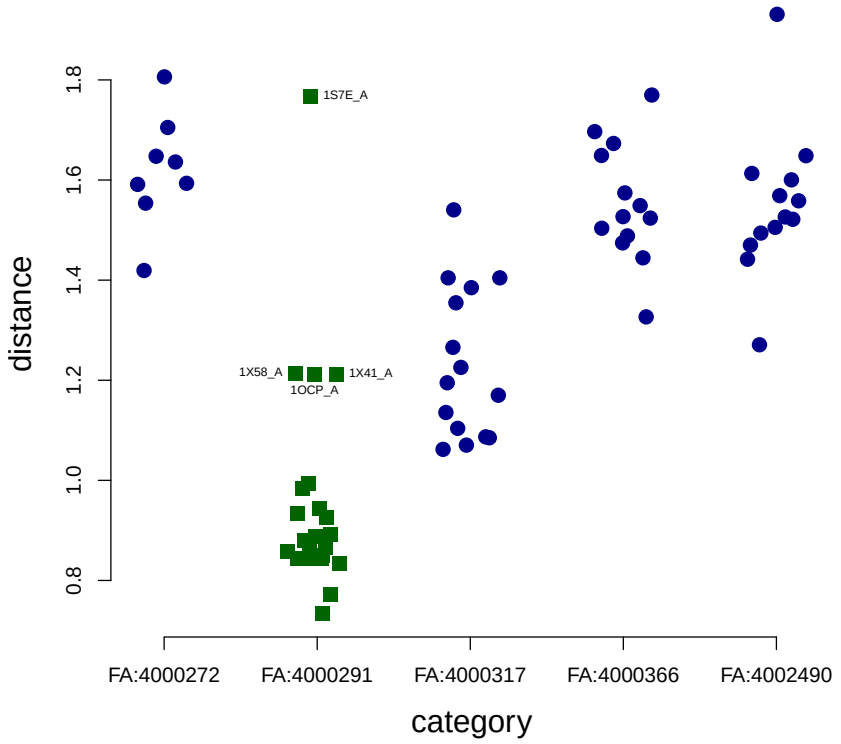


Figure 4.8: Distance calculated between the asymmetric unit `1FTZ_A` from which the domain `FA-8002409-1FTZA` is derived and all of the other 70 asymmetric units from which the domains in the group were derived. The horizontal alignment of the groups is jittered to help distinguish between individual members.

The results for the comparison using the asymmetric units, Figure 4.8, are consistent with the comparisons made using the domain residues, Figure 4.7. As with the domain comparison 4 asymmetric units failed to correctly distinguish the categorisation, and these four are

the asymmetric units from which the failing domains are derived.

In this comparison there are 70 rather than 71 comparisons made as the asymmetric unit 1A5J_A contains two separate domains both of which are in the FA:4000317 family category. Both sequences are of the same length and examination of our alignment results show that subsets of the two sequences were in every case the first and second best matches in comparisons within family, although their order was not consistent across all comparisons.

4.6.1 Clustering

In this section we look at a brute force comparison of every domain in our test group with every other domain, using GProtA for alignment and the difference measure in (3.13). For the 72 domains in the group this requires $\binom{72}{2} = 2556$ individual comparisons. The first examination of the results is done using *multidimensional scaling* using the cmdscale command from R. The results, shown in Figure 4.9, are poor for delineating the FA:4000291 and FA:4000317 categories, and an examination of the eigenvalues suggests that dimension reduction techniques will be poor at clustering these data using two or three dimensions since the first three dimensions only explain about 40% of the variation in the data.[27]

[27] Only marginally improved results were achieved using the isoMDS and sammon functions from the MASS package.

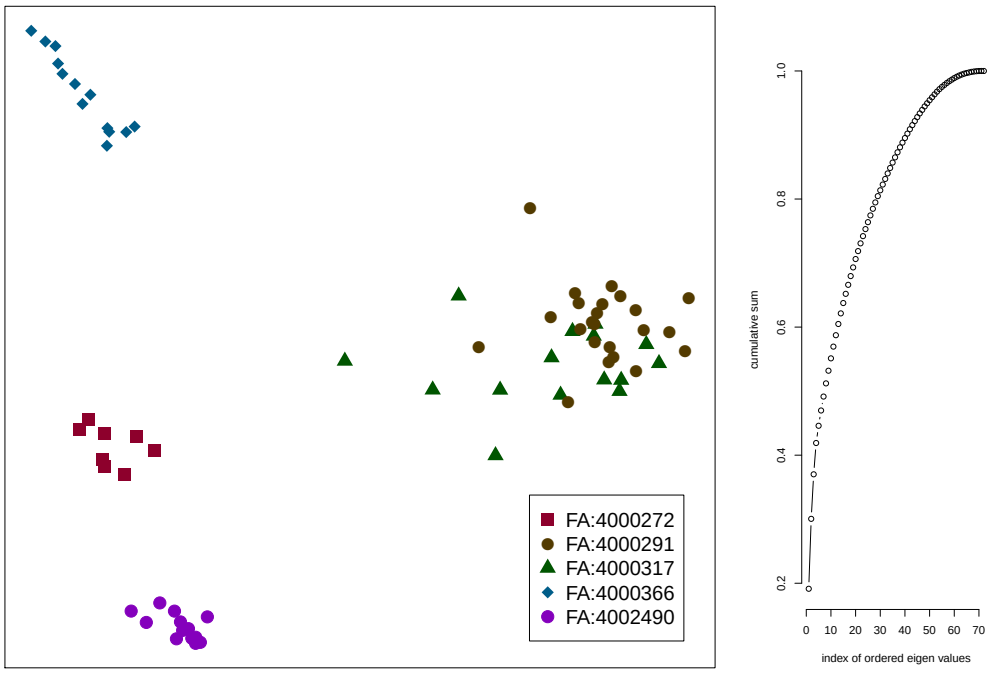


Figure 4.9: MDS plot against the distance matrix of the complete set of pairwise comparisons in our test group of domains. The accompanying plot of the cumulative sum of the ordered eigenvalues shows that less than 30% of the variance in the data is captured in the first 2 dimensions.

Going forward we opt to use density based clustering; the following uses the DBSCAN (Density Based Spatial Clustering of Applications with Noise) method detailed in Ester et al. (1996).

DBSCAN is parameterised using Eps and a $MinPts$ values and uses the following definitions:

Eps-neighbourhood A point q is in the Eps-neighbourhood of a point p if: $\mathrm{dist}(p,q) \leq Eps$.

core point A point p is a core point if the number of points that are in its Eps-neighbourhood: $N_{Eps}(p) \geq MinPts$.

border point A point p is a border point if the number of points that are in its Eps-neighbourhood: $0 < N_{Eps}(p) < MinPts$.

directly density-reachable A point q is directly density-reachable from a point p if q is in the Eps-neighbourhood of p and p is a core point.

density-reachable A point q is density reachable from a point p_1 if a chain can be made $(p_1, p_2, \ldots, p_n, q)$ where each point p_i is a core point.

density-connected Points p and q are density-connected if there is a point r from which both p and q are density reachable.

Using these definitions a cluster is defined as "a set of density-connected points which is maximal wrt. density-reachability", and noise is defined as those points that do not belong to a cluster. Figure 4.12 gives a diagrammatic overview of the above definitions.

It is difficult to demonstrate that DBSCAN is an effective method of clustering using images of large protein data sets and so we make an aside to show a visual representation of the results of clustering a set of randomly selected Aspartate residues. There follows a brief explanation of these data. The data are pairwise size-and-shape distances of a sample of experimentally determined atomic co-ordinates for the amino acid Aspartate taken from PDB files. The atoms from an amino acid that are not part of the protein backbone form the *sidechain* and the different *conformations* of a sidechain are called its *rotamers*, short for *rotational isomers*. Rotations along the axis of bonds within a sidechain, called *torsion angles*, result in the different conformations. Torsion angles are preferred that avoid minimise proximety between atoms (steric hindrance). The torsion angles for Aspartate are illustrated in Figure 4.10.

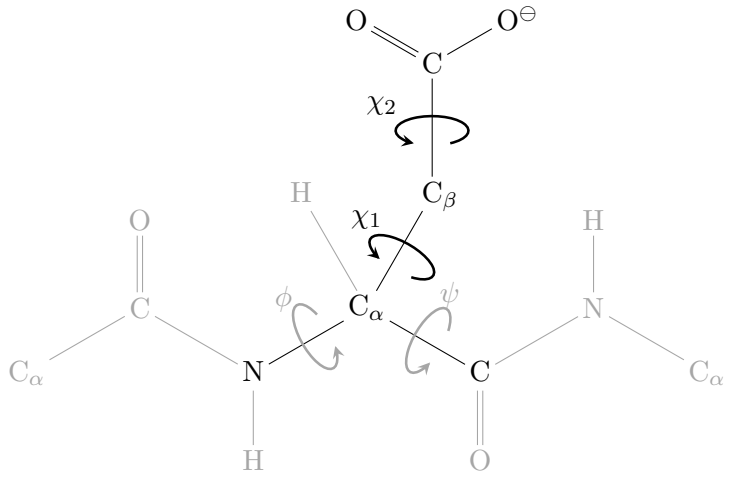
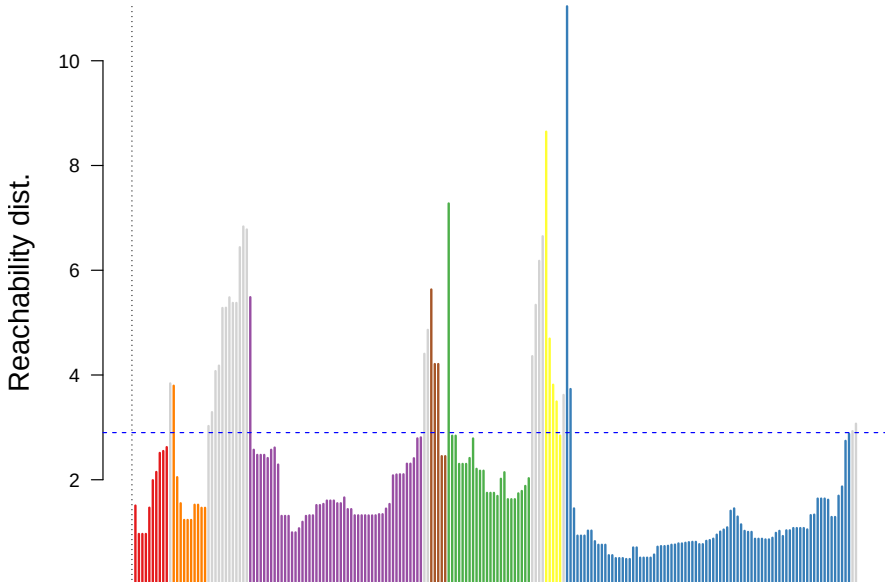
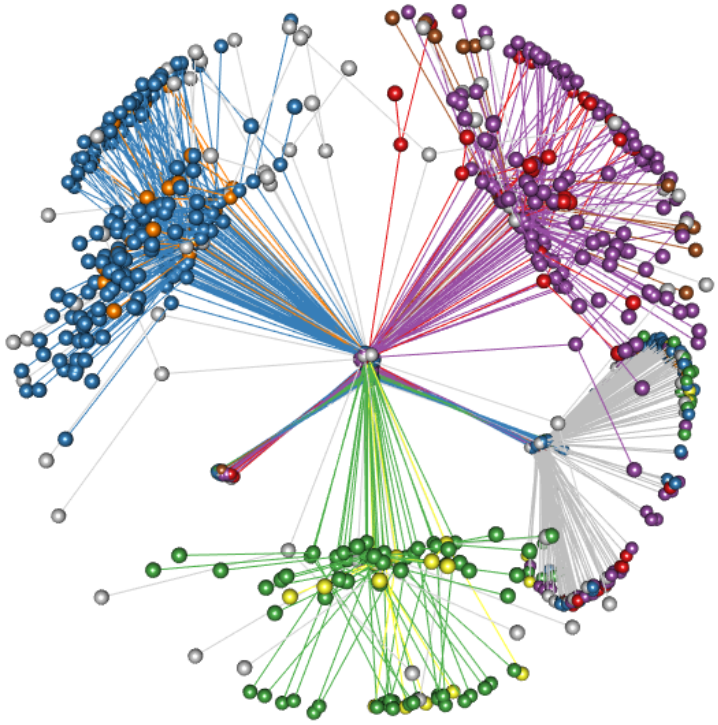


Figure 4.10: To a first approximation the torsion angles χ_n define the rotamers of the sidechain. This figure illustrates the torsion angles χ_1 and χ_2 that define the rotamers of Aspartate.

The results of the clustering the Aspartate data are shown in Figures 4.11, the function of the reachability plot will be described later in this section. The code for this example is given in Appendix B.4 and we use the implementation of DBSCAN from the `R DBSCAN` package.



(a) Reachability plot.



(b) Registered rotamers, coloured by cluster. The different clusters within the visual groupings, for example the green and yellow, are as a result of the torsion angle χ_2 differing by approximately π radians.

Figure 4.11: Results of clustering of a random selection of Aspartate residues using DBSCAN.

Returning to the clustering of the protein domain data: The eps and `minPts` parameters are set at 1.9 and 6 respectively for the domain data, and at 2.1 and 6 for the asymmetric unit data. The parameters were determined using the procedure outlined in Ester et al. (1996, Section 4.2). The associated k-nearest neighbour distance plots are given in Figure 4.13.

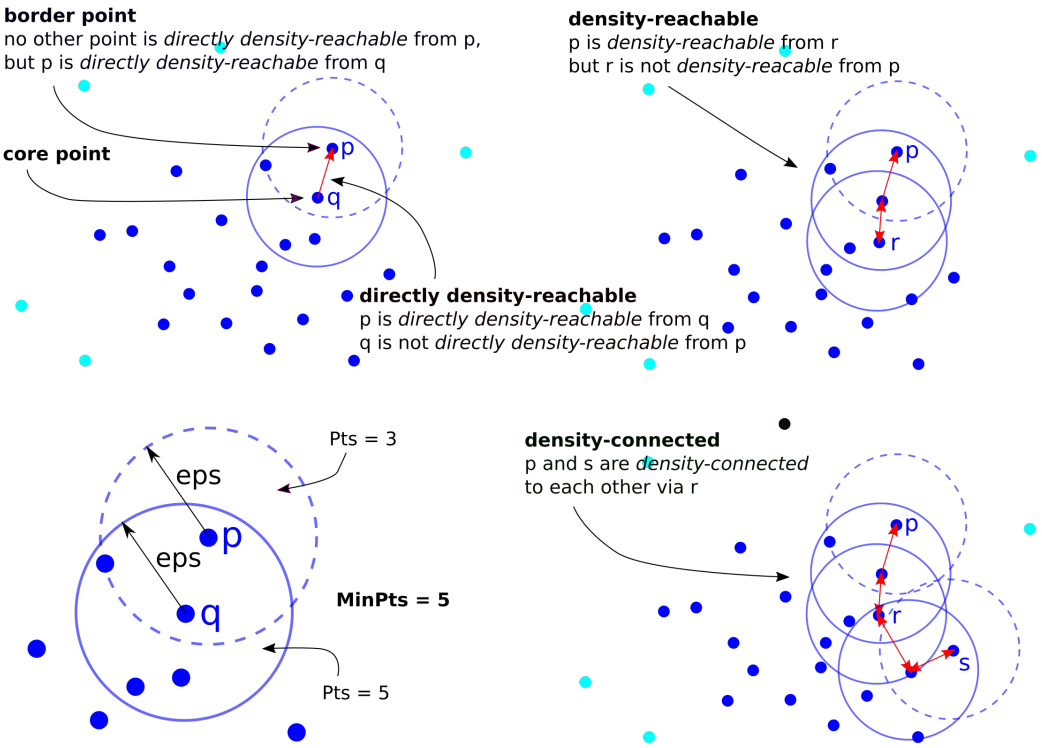
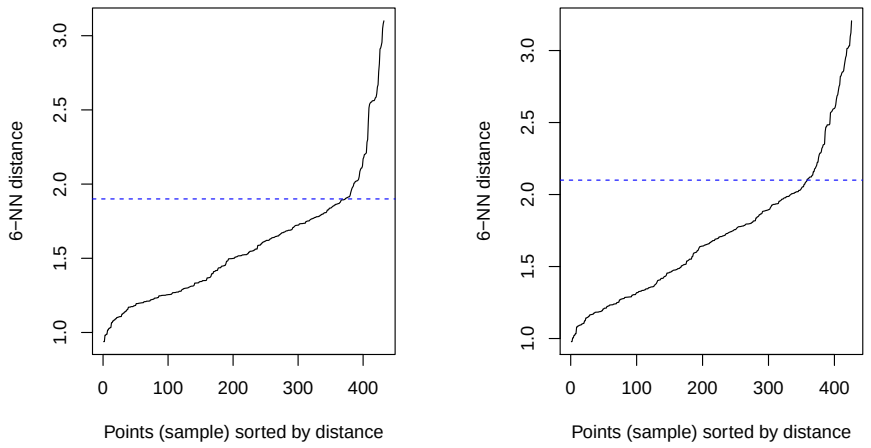


Figure 4.12: Overview of DBSCAN terminology.



(a) domain data **(b)** asymmetric unit data

Figure 4.13: kth nearest neighbour distance plots used to calculate the eps parameter for the DBSCAN clustering.

A useful way to visualise the clustering structure discovered by density based clustering is to traverse the data by plotting the distances to the nearest neighbour whilst only visiting each data point once. This method is formalised in the OPTICS algorithm (Ordering Points To Identify the Clustering Structure) detailed in Ankerst et al. (1999). Plots of the

clustering structure for both the domains and asymmetric units are given in Figure 4.14 and Figure 4.15 respectively. The detailed results of the clustering are given in Tables 4.4.

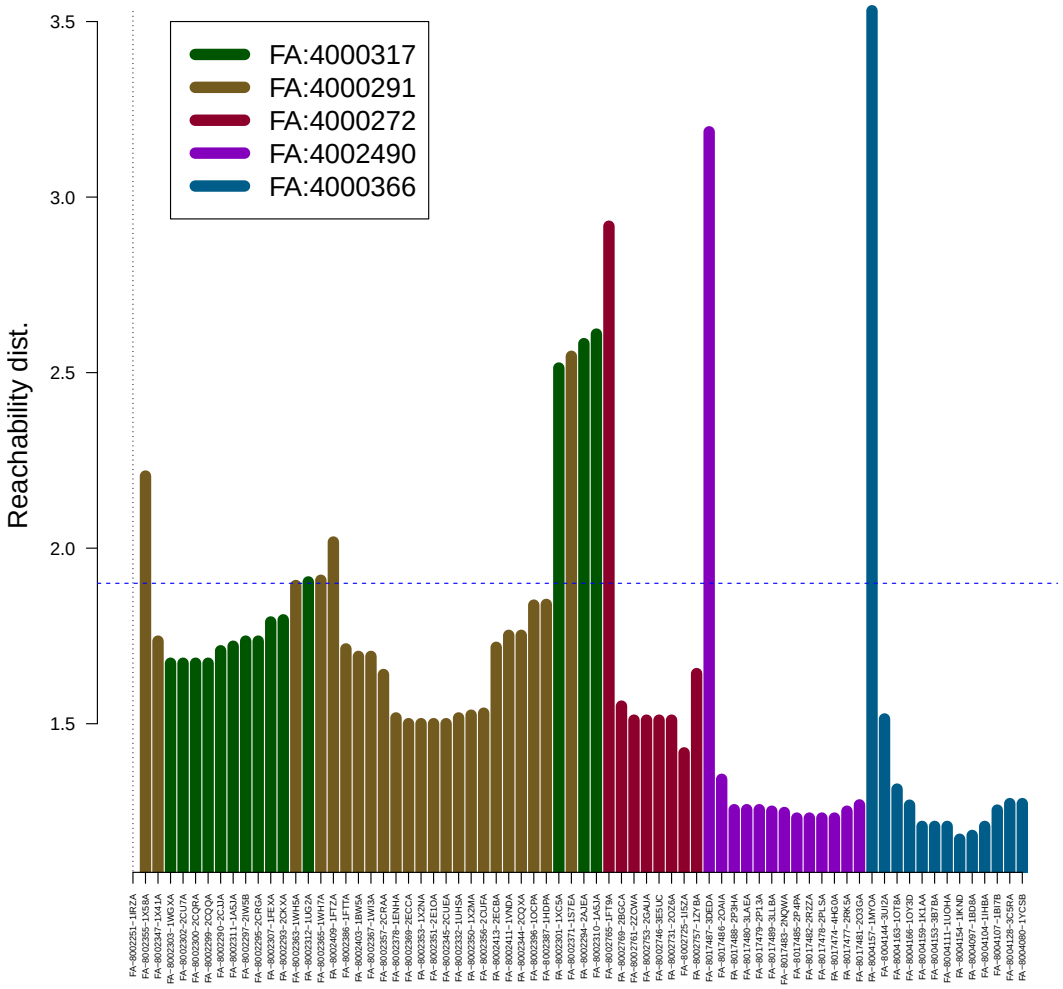


Figure 4.14: OPTICS plot of the clustering structure for the domain data.

The two nearest neighbour distance plots, Figures 4.14 and 4.15, show very similar structure for the domains and asymmetric units, which reflects the fact that the residue matches are very similar. The choice of eps parameter used for the DBSCAN clustering is marked on the plots (horizontal dotted line); the choices are seen to be well chosen for delineating the five families. The structure within the plots reveals that the FA:4000272, FA:4002490 and FA:4000366 family proteins demonstrate tighter clustering than the FA:4000317 and FA:4000291 families suggesting less variation and hence a lower energy structure being characteristic of these families.

Of note in these results is that we have correctly clustered into 5 distinct sets 62 of the

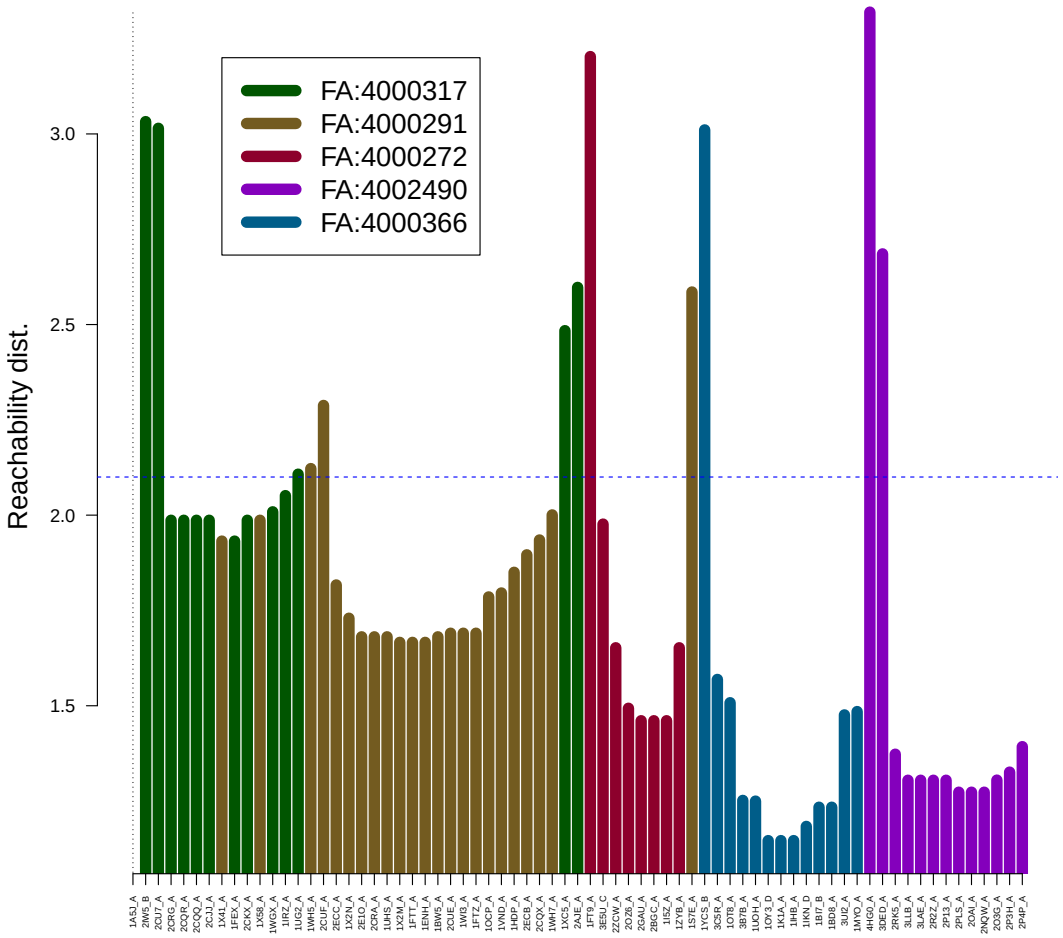


Figure 4.15: OPTICS plot of the clustering structure for the asymmetric unit data.

71 asymmetric units, and of the remaining 9 only two are incorrectly clustered with the remaining 7 being indeterminate. The clustering has been achieved using no information from SCOP2 except that used to make the original choice of the 71 asymmetric units. Here we have discovered a set of SCOP2 family categorisations directly from the data available in the Protein Data Bank using GProtA and our proposed difference measure. This strongly suggests that for global matching the difference measure is capturing the same information from the data as the expert curator.

domain	family	cluster
FA-8002371-1S7EA	FA:4000291	-
FA-8002251-1IRZA	FA:4000317	-
FA-8002294-2AJEA	FA:4000317	-
FA-8002301-1XC5A	FA:4000317	-
FA-8002310-1A5JA	FA:4000317	-
FA-8002312-1UG2A	FA:4000317	-
FA-8002347-1X41A	FA:4000291	1 ←
FA-8002355-1X58A	FA:4000291	1 ←
FA-8002363-1WH5A	FA:4000291	1 ←
FA-8002290-2CJJA	FA:4000317	1
FA-8002293-2CKXA	FA:4000317	1
FA-8002295-2CRGA	FA:4000317	1
FA-8002297-2IW5B	FA:4000317	1
FA-8002299-2CQQA	FA:4000317	1
FA-8002300-2CQRA	FA:4000317	1
FA-8002302-2CU7A	FA:4000317	1
FA-8002303-1WGXA	FA:4000317	1
FA-8002307-1FEXA	FA:4000317	1
FA-8002311-1A5JA	FA:4000317	1
FA-8002332-1UHSA	FA:4000291	2
FA-8002344-2CQXA	FA:4000291	2
FA-8002345-2CUEA	FA:4000291	2
FA-8002350-1X2MA	FA:4000291	2
FA-8002351-2E1OA	FA:4000291	2
FA-8002353-1X2NA	FA:4000291	2
FA-8002356-2CUFA	FA:4000291	2
FA-8002357-2CRAA	FA:4000291	2
FA-8002365-1WH7A	FA:4000291	2
FA-8002367-1WI3A	FA:4000291	2
FA-8002369-2ECCA	FA:4000291	2
FA-8002378-1ENHA	FA:4000291	2
FA-8002386-1FTTA	FA:4000291	2
FA-8002387-1HDPA	FA:4000291	2
FA-8002396-1OCPA	FA:4000291	2
FA-8002403-1BW5A	FA:4000291	2
FA-8002409-1FTZA	FA:4000291	2
FA-8002411-1VNDA	FA:4000291	2
FA-8002413-2ECBA	FA:4000291	2
FA-8002725-1I5ZA	FA:4000272	3
FA-8002731-2OZ6A	FA:4000272	3
FA-8002746-3E5UC	FA:4000272	3
FA-8002753-2GAUA	FA:4000272	3
FA-8002757-1ZYBA	FA:4000272	3
FA-8002761-2ZCWA	FA:4000272	3
FA-8002765-1FT9A	FA:4000272	3
FA-8002769-2BGCA	FA:4000272	3
FA-8004080-1YCSB	FA:4000366	4
FA-8004097-1BD8A	FA:4000366	4
FA-8004104-1IHBA	FA:4000366	4
FA-8004107-1BI7B	FA:4000366	4
FA-8004111-1UOHA	FA:4000366	4
FA-8004128-3C5RA	FA:4000366	4
FA-8004144-3UI2A	FA:4000366	4
FA-8004153-3B7BA	FA:4000366	4
FA-8004154-1IKND	FA:4000366	4
FA-8004157-1MYOA	FA:4000366	4
FA-8004159-1K1AA	FA:4000366	4
FA-8004163-1OT8A	FA:4000366	4
FA-8004166-1OY3D	FA:4000366	4
FA-8017474-4HGOA	FA:4002490	5
FA-8017477-2RK5A	FA:4002490	5
FA-8017478-2PLSA	FA:4002490	5
FA-8017479-2P13A	FA:4002490	5
FA-8017480-3LAEA	FA:4002490	5
FA-8017481-2O3GA	FA:4002490	5
FA-8017482-2R2ZA	FA:4002490	5
FA-8017483-2NQWA	FA:4002490	5
FA-8017485-2P4PA	FA:4002490	5
FA-8017486-2OAIA	FA:4002490	5
FA-8017487-3DEDA	FA:4002490	5
FA-8017488-2P3HA	FA:4002490	5
FA-8017489-3LLBA	FA:4002490	5

(a) domain data

asym unit	family	cluster
1S7E_A	FA:4000291	-
1A5J_A	FA:4000317	-
1UG2_A	FA:4000317	-
1XC5_A	FA:4000317	-
2AJE_A	FA:4000317	-
2IW5_B	FA:4000317	-
4HGO_A	FA:4002490	-
1X41_A	FA:4000291	1 ←
1X58_A	FA:4000291	1 ←
1FEX_A	FA:4000317	1
1IRZ_A	FA:4000317	1
1WGX_A	FA:4000317	1
2CJJ_A	FA:4000317	1
2CKX_A	FA:4000317	1
2CQQ_A	FA:4000317	1
2CQR_A	FA:4000317	1
2CRG_A	FA:4000317	1
2CU7_A	FA:4000317	1
1BW5_A	FA:4000291	2
1ENH_A	FA:4000291	2
1FTT_A	FA:4000291	2
1FTZ_A	FA:4000291	2
1HDP_A	FA:4000291	2
1OCP_A	FA:4000291	2
1UHS_A	FA:4000291	2
1VND_A	FA:4000291	2
1WH5_A	FA:4000291	2
1WH7_A	FA:4000291	2
1WI3_A	FA:4000291	2
1X2M_A	FA:4000291	2
1X2N_A	FA:4000291	2
2CQX_A	FA:4000291	2
2CRA_A	FA:4000291	2
2CUE_A	FA:4000291	2
2CUF_A	FA:4000291	2
2E1O_A	FA:4000291	2
2ECB_A	FA:4000291	2
2ECC_A	FA:4000291	2
1FT9_A	FA:4000272	3
1I5Z_A	FA:4000272	3
1ZYB_A	FA:4000272	3
2BGC_A	FA:4000272	3
2GAU_A	FA:4000272	3
2OZ6_A	FA:4000272	3
2ZCW_A	FA:4000272	3
3E5U_C	FA:4000272	3
1BD8_A	FA:4000366	4
1BI7_B	FA:4000366	4
1IHB_A	FA:4000366	4
1IKN_D	FA:4000366	4
1K1A_A	FA:4000366	4
1MYO_A	FA:4000366	4
1OT8_A	FA:4000366	4
1OY3_D	FA:4000366	4
1UOH_A	FA:4000366	4
1YCS_B	FA:4000366	4
3B7B_A	FA:4000366	4
3C5R_A	FA:4000366	4
3UI2_A	FA:4000366	4
2NQW_A	FA:4002490	5
2O3G_A	FA:4002490	5
2OAI_A	FA:4002490	5
2P13_A	FA:4002490	5
2P3H_A	FA:4002490	5
2P4P_A	FA:4002490	5
2PLS_A	FA:4002490	5
2R2Z_A	FA:4002490	5
2RK5_A	FA:4002490	5
3DED_A	FA:4002490	5
3LAE_A	FA:4002490	5
3LLB_A	FA:4002490	5

(b) asymmetric unit data

Table 4.4: Results of density based clustering on the pairwise distance data for the test group of 72 domains and 71 asymmetric units. A cluster value of '-' indicates that the data point was identified as noise. Incorrect category assignment is highlighted using an arrow.

4.6.2 Characteristic signatures

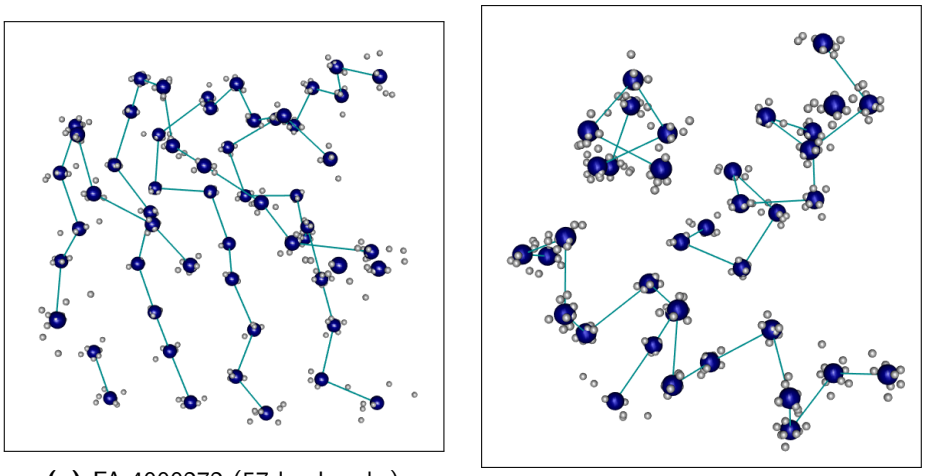
In this section we create a characteristic signature for each category family. The characteristic signature is a set of landmarks that represent the identifying structure of the family. If such a characteristic signature can be identified then rather than following our previous procedure of brute force pairwise comparison followed by clustering we could simply compare a new configuration to each characteristic signature to see which it most closely resembles. Although derived differently the characteristic signature of a category is analogous to the *templates* constructed in Mardia, Nyirongo, et al. (2011) to be representative of pharmacophores.

In the first instance we derive a characteristic signature from the clustered asymmetric units. Although the clustered domains would offer a marginally better starting point, the use of the clustered asymmetric units is interesting as the asymmetric unit definitions are available in the PDB, and so this suggests the possibility of a primarily data driven approach to family level categorisation. The basic procedure has the following steps applied in turn for each cluster

CharacteristicSignature

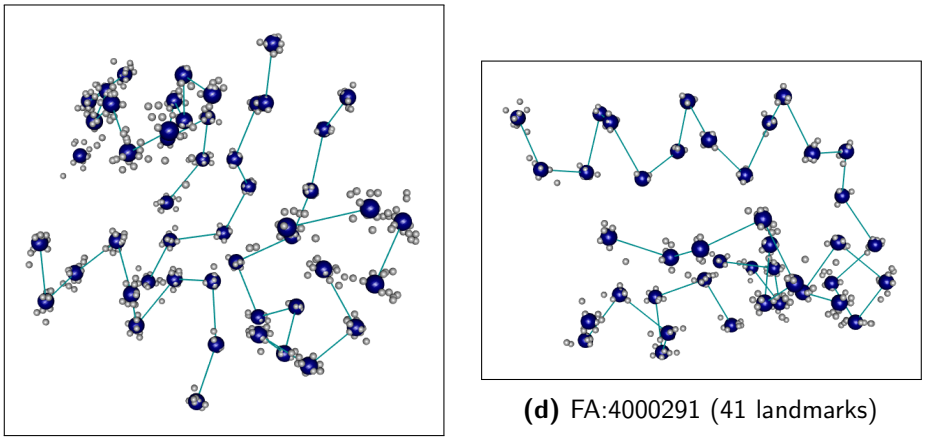
1 **for** each asymmetric units cluster
2 Find the configuration that has the smallest mean RMSD for the pairwise alignments to each other configuration in the cluster and call this the central configuration. (Use the labelling calculated during clustering.)
3 Identify the subset of landmarks in the central configuration that are matched to a landmark in all other of the cluster configurations and call these the characteristic landmarks. (You will now have identified the same number of characteristic landmarks in each configuration of the cluster.)
4 Calculate a mean shape using the characteristic landmarks for each configuration in the cluster using *partial Procrustes analysis* (Dryden and Mardia 2016, Section 7.5). This mean shape is the characteristic signature of the cluster.

The mean shape derived above is used as the characteristic signature for the family associated with the cluster. When we follow this procedure using the clustered asymmetric unit data we establish the characteristic signatures illustrated in Figure 4.16, the co-ordinates of which are given in Appendix A.2.



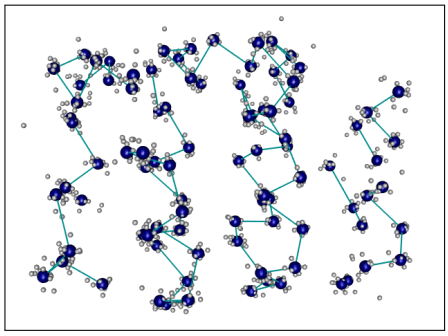
(a) FA:4000272 (57 landmarks)

(b) FA:4000317 (37 landmarks)



(c) FA:4002490 (50 landmarks)

(d) FA:4000291 (41 landmarks)



(e) FA:4000366 (93 landmarks)

Figure 4.16: 2D projections of characteristic signatures. The blue spheres represent the locations of the landmarks used for the characteristic signatures. The smaller grey spheres are the aligned sets of landmarks used in the partial Procrustes analysis. The co-ordinates are given in Appendix A.2.

First we check that the signatures work using the data that we used to create them. We would expect this to produce results very similar to the clustering that we illustrated in the nearest neighbour distance plot for the pairwise comparisons of all asymmetric units, Figure 4.15, and this is the case. The distances obtained by comparing each of the 71 asymmetric units to the family characteristic signatures are given in the form of a boxplot

in Figure 4.17. The boxplots show good separation between distances to the characteristic signature for members of the characteristic signature's family compared to asymmetric units not belonging to the characteristic signature's family.

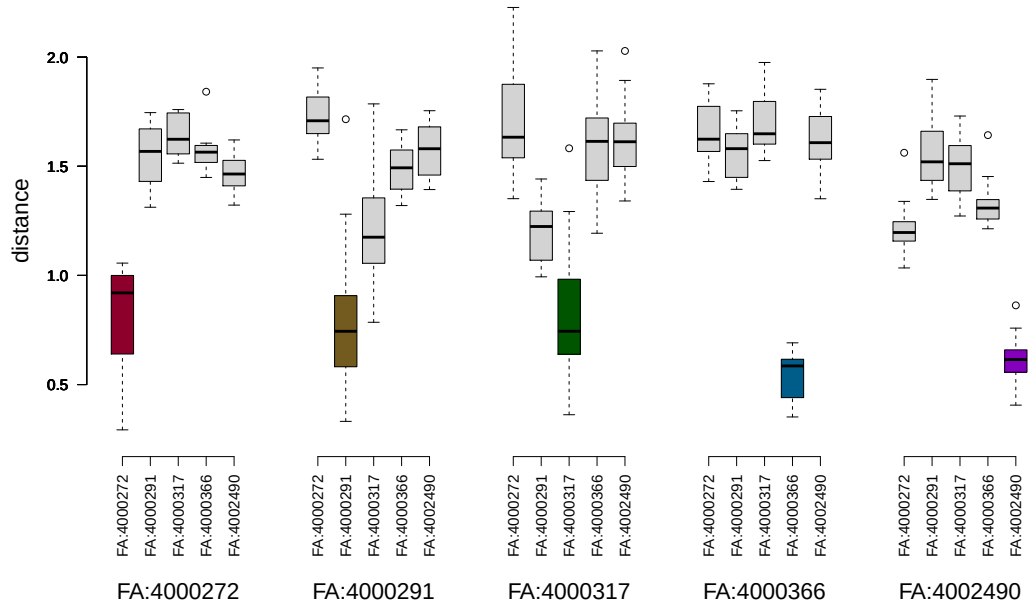


Figure 4.17: Boxplot of distances obtained by comparing each of the 71 asymmetric units to each of the five family characteristic signatures. For each comparison to a characteristic signature the results have been grouped by their family according to SCOP2, the boxes of the groups that have the same family as the characteristic signature are coloured.

Next we look at comparisons using data that was not part of the original data set. Figure 4.18 and Table 4.5 show the distances between the 5 larger asymmetric units belonging to the `FA:`4000366 family category with each of the characteristic signatures derived from the clustered asymmetric units. The details of these new 5 larger asymmetric units are given in Table 4.6. The results show a clear distinction in distances when compared against the correct family category.

asym unit	characteristic signature	dss^2	matches	m	n	sb	d
1DCQ_A	FA:4000272	7.107	36	57	276	0.251	1.768
1N11_A	FA:4000272	6.856	36	57	404	0.234	1.861
1S70_B	FA:4000272	6.902	33	57	291	0.234	1.952
1SW6_A	FA:4000272	7.234	34	57	254	0.246	1.876
1WDY_A	FA:4000272	6.750	35	57	285	0.245	1.793
1DCQ_A	FA:4000291	6.566	31	41	276	0.266	1.732
1N11_A	FA:4000291	5.812	32	41	404	0.256	1.666
1S70_B	FA:4000291	6.970	30	41	291	0.258	1.868
1SW6_A	FA:4000291	6.649	31	41	254	0.270	1.717
1WDY_A	FA:4000291	6.557	35	41	285	0.285	1.519
1DCQ_A	FA:4000317	6.501	28	37	276	0.261	1.843
1N11_A	FA:4000317	6.952	30	37	404	0.258	1.863
1S70_B	FA:4000317	6.731	30	37	291	0.271	1.750
1SW6_A	FA:4000317	6.290	27	37	254	0.259	1.864
1WDY_A	FA:4000317	6.437	30	37	285	0.271	1.706
1DCQ_A	FA:4000366	5.049	89	93	276	0.366	0.650
1N11_A	FA:4000366	2.829	93	93	404	0.344	0.508
1S70_B	FA:4000366	6.653	85	93	291	0.352	0.795
1SW6_A	FA:4000366	5.249	80	93	254	0.351	0.730
1WDY_A	FA:4000366	6.486	91	93	285	0.368	0.725
1DCQ_A	FA:4002490	7.133	36	50	276	0.267	1.667
1N11_A	FA:4002490	6.887	34	50	404	0.241	1.866
1S70_B	FA:4002490	7.038	34	50	291	0.255	1.786
1SW6_A	FA:4002490	7.134	32	50	254	0.251	1.881
1WDY_A	FA:4002490	5.619	34	50	285	0.256	1.590

Table 4.5: Results of the comparison between the 5 larger asymmetric units belonging to the FA:4000366 category described in Table 4.6 and each of the characteristic signatures for the five families in our original test group.

asym unit	residue count	exptl method	last mod date	warnings
1DCQ_A	276	X-RAY DIFFRACTION	2016-12-14	1
1N11_A	404	X-RAY DIFFRACTION	2009-02-24	1
1S70_B	291	X-RAY DIFFRACTION	2009-02-24	1
1SW6_A	254	X-RAY DIFFRACTION	2009-02-24	1
1WDY_A	285	X-RAY DIFFRACTION	2009-02-24	1

Table 4.6: 5 larger asymmetric units belonging to the FA:4000366 category that were not part of the original test group.

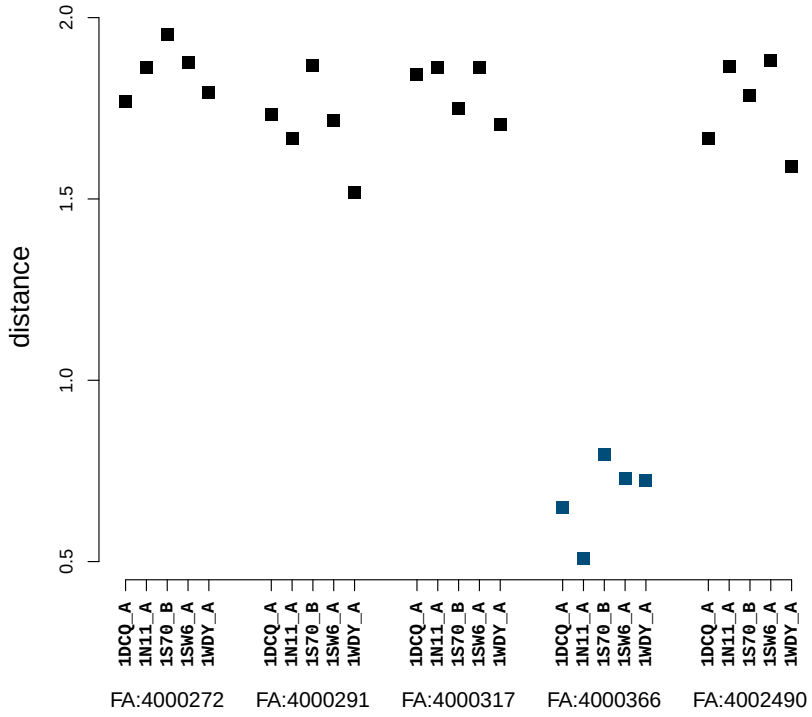
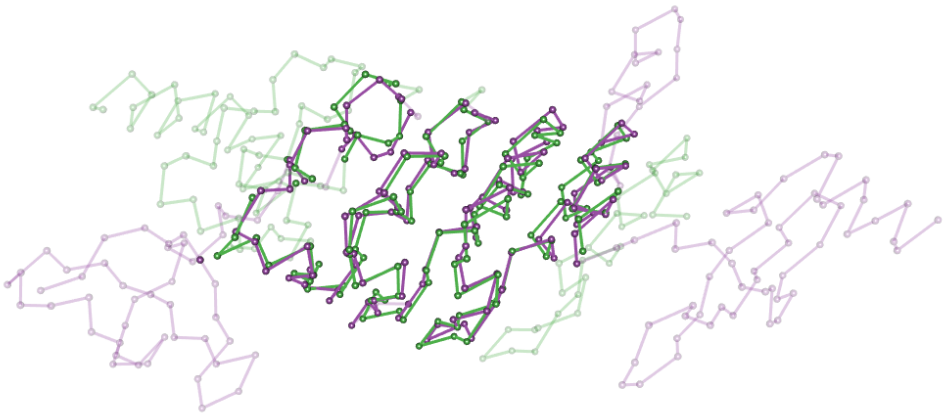
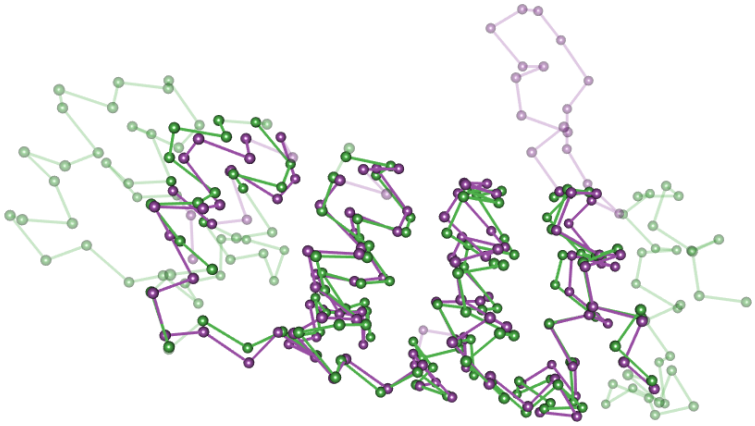


Figure 4.18: Results of the comparison between the 5 larger asymmetric units belonging to the `FA:4000366` category described in Table 4.6 and each of the characteristic signatures for the five families in our original test group.

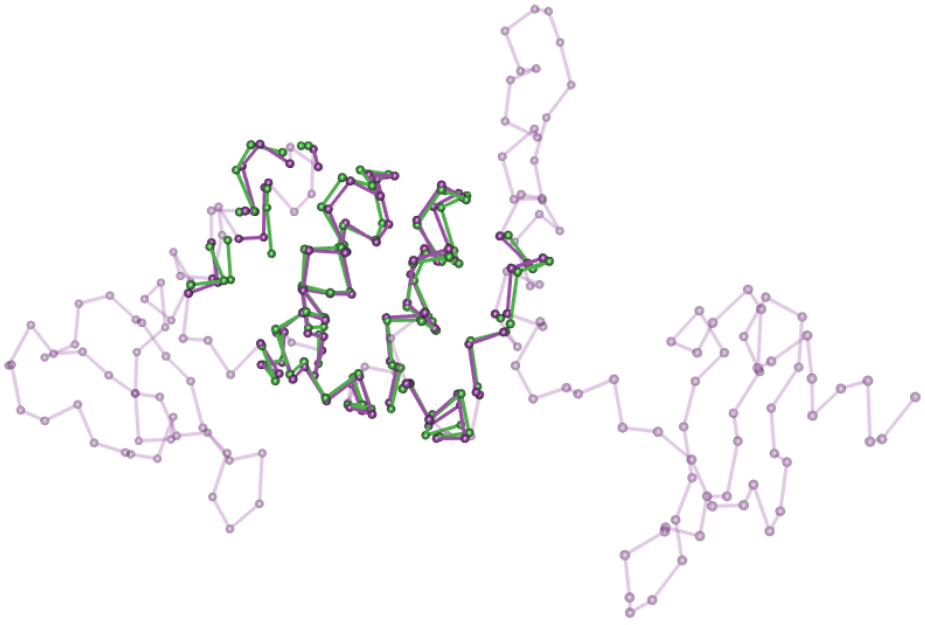
To summarise the different data sets that we have applied the GProtA algorithm to so far, Figure 4.19 shows examples of alignment between asymmetric units, alignment between domains as defined in SCOP2, and alignment between an asymmetric unit and a family characteristic signature.



(a) Alignment of the asymmetric units 1OT8_A and 3UI2_A.



(b) Alignment of the domains for the FA:4000366 family as defined by SCOP2 for the asymmetric units in (a), FA-8004163-1OT8A and FA-8004144-3UI2A.



(c) Alignment of the 3UI2_A asymmetric unit from (a) with the characteristic signature for the SCOP2 family FA:4000366.

Figure 4.19: Examples of alignment using the GProtA algorithm.

Finally in this section we derive a characteristic signature directly from a set of SCOP2 domains. To do this we consider the complete set of asymmetric units that are assigned the family category FA:4000157; these are detailed in Table 4.7.

asym unit	residue count	exptl method	last mod date	warnings
1I5Z_A*	201	X-RAY DIFFRACTION	2009-02-24	1
2OZ6_A*	201	X-RAY DIFFRACTION	2011-08-10	1
1ZYB_A	230	X-RAY DIFFRACTION	2011-07-13	1
1FT9_A*	210	X-RAY DIFFRACTION	2009-02-24	1
3E5U_C*	219	X-RAY DIFFRACTION	2011-07-13	1
2GAU_A	218	X-RAY DIFFRACTION	2011-07-13	1
2ZCW_A	194	X-RAY DIFFRACTION	2009-02-24	1
2BGC_A	235	X-RAY DIFFRACTION	2011-07-13	1

Table 4.7: The complete set of asymmetric units that are assigned the SCOP2 family category FA:4000157.

Of these we randomly select 4 (indicated by a * in the table) to use to create a characteristic signature. The characteristic signature is created using the same procedure as was used with a cluster of the asymmetric units except that here we use the 4 randomly chosen units rather than those from the core of the cluster. The resulting characteristic signature is shown in Figure 4.20.

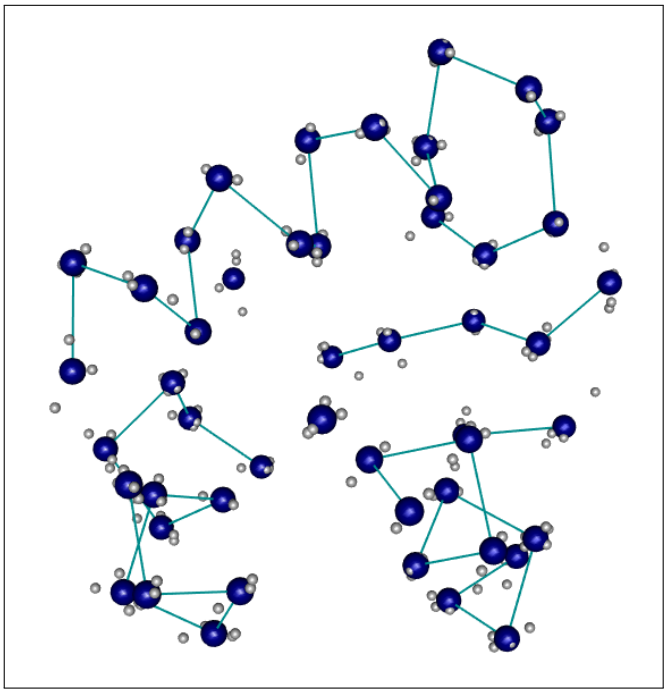


Figure 4.20: Characteristic signature for family FA:4000157 (49 landmarks).

Distances are then calculated between the remaining 4 asymmetric units categorised in

family FA:4000157 and all of the 5 plus 1 characteristic signatures. The results of this comparison are given in Figure 4.21 and Table 4.8.

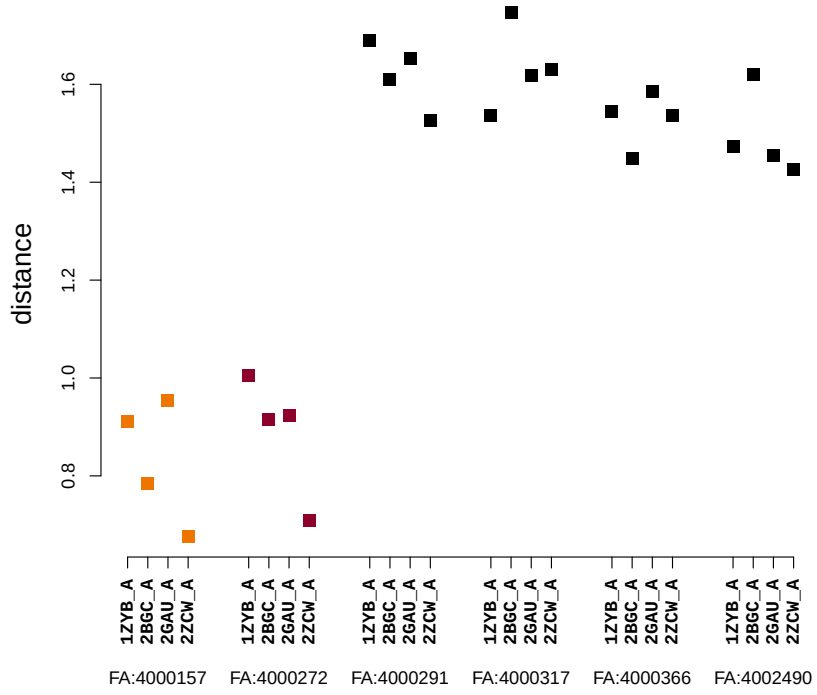


Figure 4.21: Results of the comparison between the 4 unstarred asymmetric units belonging to the FA:4000157 category described in Table 4.7 and each of the characteristic signatures for the five families in our original test group along with the newly characteristic signatures of FA:4000157 derived from the domains associated with the starred asymmetric units in Table 4.7.

asym unit	characteristic signature	dss^2	matches	m	n	sb	d
1ZYB_A	FA:4000157	4.048	46	49	230	0.325	0.912
1ZYB_A	FA:4000272	7.039	57	57	230	0.349	1.006
1ZYB_A	FA:4000291	7.173	32	41	230	0.280	1.689
1ZYB_A	FA:4000317	4.799	28	37	230	0.270	1.535
1ZYB_A	FA:4000366	7.168	47	93	230	0.253	1.545
1ZYB_A	FA:4002490	5.999	36	50	230	0.277	1.474
2BGC_A	FA:4000157	3.257	48	49	235	0.332	0.784
2BGC_A	FA:4000272	5.775	57	57	235	0.348	0.916
2BGC_A	FA:4000291	6.929	33	41	235	0.285	1.610
2BGC_A	FA:4000317	6.678	29	37	235	0.275	1.747
2BGC_A	FA:4000366	6.553	48	93	235	0.255	1.449
2BGC_A	FA:4002490	5.875	33	50	235	0.260	1.621
2GAU_A	FA:4000157	5.217	49	49	218	0.342	0.955
2GAU_A	FA:4000272	6.086	57	57	218	0.354	0.924
2GAU_A	FA:4000291	7.012	32	41	218	0.283	1.653
2GAU_A	FA:4000317	6.352	30	37	218	0.284	1.618
2GAU_A	FA:4000366	6.609	44	93	218	0.245	1.585
2GAU_A	FA:4002490	7.186	39	50	218	0.295	1.455
2ZCW_A	FA:4000157	2.764	49	49	194	0.351	0.677
2ZCf_A	FA:4000272	3.785	57	57	194	0.364	0.709
2ZCW_A	FA:4000291	7.183	34	41	194	0.301	1.526
2ZCW_A	FA:4000317	6.737	30	37	194	0.291	1.629
2ZCW_A	FA:4000366	7.033	45	93	194	0.257	1.537
2ZCW_A	FA:4002490	6.446	37	50	194	0.293	1.426

Table 4.8: Full results set for Figure 4.21.

The family FA:4000157 was specifically chosen for this test because, unusually for SCOP2, all asymmetric units containing a domain categorised as FA:4000157 also contain a domain in a separate part of the asymmetric unit with a different family category, namely FA:4000272. The results demonstrate that by isolating the characteristic signature of each family category we are able to correctly identify when an asymmetric unit is given multiple categories.

4.7 Comparing SCOP2 superfamilies and hyperfamilies

A hyperfamily categorisation offers a general description of the protein structure. For example, these are the first three of ten defined hyperfamilies:

- Core: 3-helices, bundle, closed or partly opened, right-handed twist, up-and down (HF:1100001 – hyperfamily DNA/RNA-binding 3-helical bundle).
- 3 layer core: alpha/beta/alpha, parallel beta-sheet of 6 strands, order 321456; common region begins with strand 1 and ends with strand 6; the core is packed against the helix that, depending on a particular superfamily member, either precedes strands 1 (rare), or follows strand 6 (common) (HF:1100002 – hyperfamily Rossmann-like nucleotide-binding lobe).

- Variable number of helices and little beta structure, not a true fold (HF:1100003 – hyperfamily Multiheme cytochromes).

These descriptions allow for significant variability in structure and so no attempt was made to reproduce categorisation at this level. It is noted that the `relation` and `closure` tables in the SCOP2 database imply a hierarchy in the levels of categorisation, an example of which is captured in Figure 4.22. However, although membership of a hyperfamily is suggested by membership of a superfamily or family, in practice this is not always the case as the asymmetric unit may not have associated domains in the database for those categorisations. For example, in Figure 4.22 the asymmetric unit `1WH7_A` has a SCOP2 domain associated to the family `FA:4000291` but not ones for the superfamily or hyperfamily in the implied hierarchy.

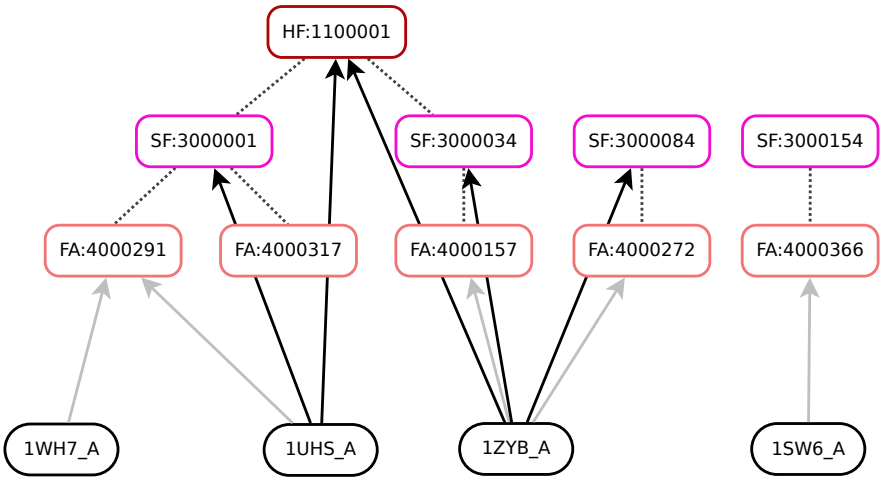


Figure 4.22: Example of domains associated with a selection of hyperfamilies, superfamilies and families.

A superfamily categorisation offers a mixture of general descriptions and specific detail of protein structure. For example, these are the three superfamily definitions that cover asymmetric units that we have already examined:

- Contains a small beta-sheet (wing) (SF:3000034 – superfamily Winged helix DNA-binding domain).
- One turn of helix is made by two pairs of antiparallel strands linked with short turns has appearance of a sandwich of distinct architecture and jelly-roll topology (SF:3000084 – superfamily cAMP-binding domain-like).
- Consists only of helices (SF:3000001 – superfamily Homeodomain-like).

SF:3000001 is very general which suggests that we will be unable to identify a characteristic signature – actually identifying structures that only contain helices is possible from the metadata available in the PDB. The descriptions of SF:3000034 and SF:3000084 are more specific, but in the case of SF:3000084 it is unclear if the description "linked with short turns" implies that the two parts of the structure need not be globally aligned.

Using the procedure to create characteristic signatures described in the previous section, Figure 4.23 shows the results of a small study against the SF:3000001, SF:3000034 and SF:3000084 families using the asymmetric units shown in Figure 4.22. As expected the

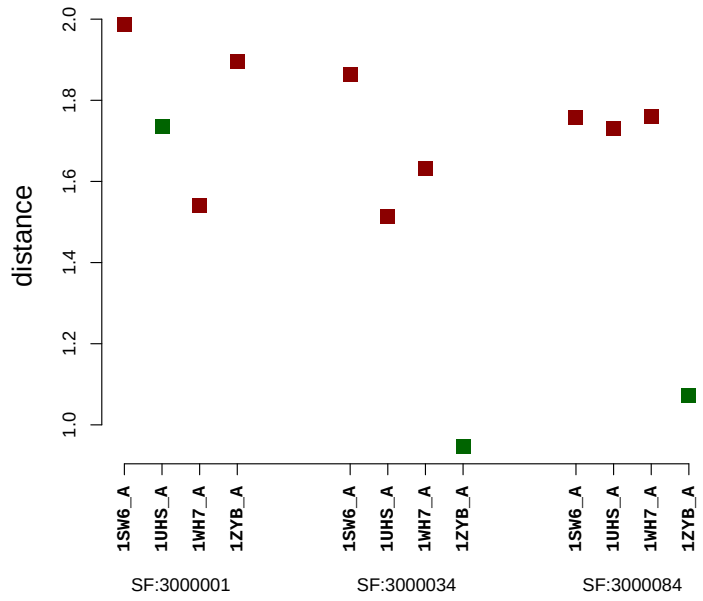


Figure 4.23: Comparison of a selection of asymmetric units against characteristic signatures of superfamily categories. The green points indicate the asymmetric unit that does belong to the category for each group.

plot in Figure 4.23 shows no ability to determine the SF:3000001 superfamily but suggests potential for the SF:3000034 and SF:3000084 categories.

4.8 Processing times

In Section 3.4.2 we estimated the worst case order of the GProtA algorithm as $\mathcal{O}(m^2n)$ size-and-shape distance calculation, these calculations being the most expensive part of the algorithm, and m and n being the number of landmarks in configurations $\boldsymbol{A}$ and $\boldsymbol{B}$

respectively. In Figure 4.24 we show a scatter plot of the time taken against $m + n$ for multiple runs of the algorithm when comparing asymmetric units and domains. This data set was chosen as all the runs are from a single ring-fenced machine and so the times are comparable. The overplotted blue points are the data fitted against $\alpha\, m^2 n + c$, the general agreement in the structure offers validation of the order calculation. The fit produces $\alpha = 0.00057$, this value reflects the number of calculations that are avoided by setting the tolerance in the algorithm and is dependant on the ratio of the variance of the error between true matches and the variance of the spatial separation of the landmarks. The plot suggests a mechanism for predicting run times for larger configurations.

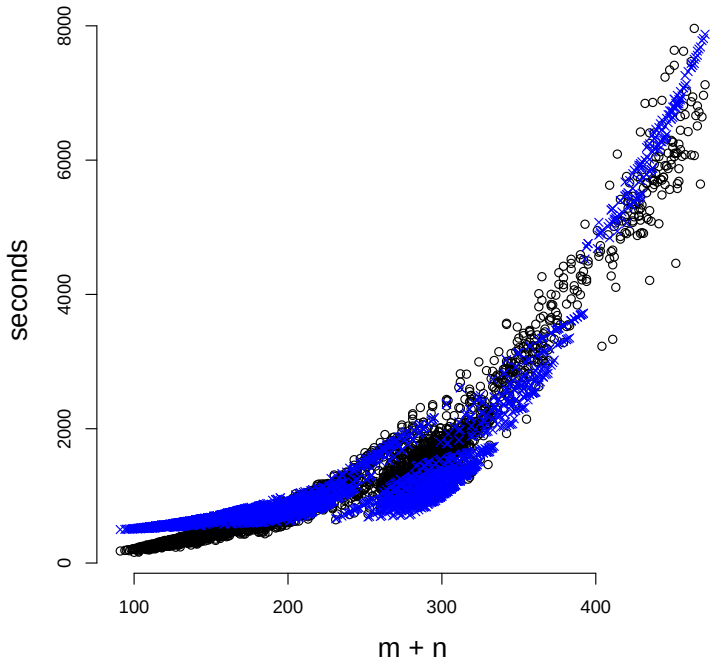


Figure 4.24: Time taken for multiple runs of the GProtA when comparing protein asymmetric units. The blue points are fitted against $\alpha m^2 n + c$ which gave $\alpha = 0.00057$.

5

Discussion

In Chapter 2 we looked in detail at the method in Green and Mardia (2006); this method provides both the motivation and the basis for many of the ideas developed for the GProtA algorithm described in Chapter 3. The outcome of the analysis was the identification of two limitations related to applying Green and Mardia (2006) to large scale unsupervised matching. The first limitation relates to the information available to the likelihood which results in the search for the first few matches being essentially random; such a random search will not scale to large configurations because of the exponential nature of the size of the space of possible matching matrices. A solution to this problem was offered in Chapter 3 where a set of starting matching matrices – used as the base for each of the branches in the GProtA algorithm – can be used as a set of over-dispersed starting points (Gelman et al. 2013, Section 11.4) for multiple simulated Markov chains in the Green and Mardia (2006) method. The second limitation relates to the need for prior knowledge of the number of matches which is captured in the λ/ρ parameter; the sensitivity to this was shown to be significant and we were unable to determine a mechanism to derive or approximate a value from the data.

In Chapter 3 we proposed a new algorithm called GProtA for solving the unlabelled, partial matching problem within the protein alignment problem domain. SCOP2 considers only the structure of the C_α atoms and does not distinguish the type of amino acids. Steric

considerations offer the constraint that C_α atoms should be separated by at least a distance of 3.8Å and that consecutive C_α atoms in a protein chain should have a separation very close to 3.8Å; this allows us to make modelling assumptions about the maximum errors allowed for a match to be valid. We also use the constraint that a matching solution must contain matches between at least four consecutive C_α atoms in each protein chain. This constraint is valid for matching protein fold descriptions as it follows directly from the domain-specific definition of local structure. The four consecutive C_α constraint gives a manageable set of candidate starting matching matrices that we test for evidence against a null hypothesis that the matching matrices were chosen at random. The resulting set of starting matching matrices are then individually pursued, iteratively adding matches using a greedy algorithm that in essence chooses the match with the smallest p-value against a null hypothesis that the new match is chosen at random. The results of this GProtA algorithm is a set of optima from which we choose the one with the greatest number of matches, and the remaining local optima may be of use to an investigator (but were not used in this work). Using a pre-cached null distribution the GProtA algorithm implemented in `R` finds the solution for the Green and Mardia (2006) data in approximately 45 seconds using a single core of a `Intel Core i7-3520M 2.90GHz` processor. Also in Chapter 3 we proposed a difference measure for the matched proteins that combines a binary similarity measure with the RMSD calculated against the aligned configurations.

In Section 4.6.1 of Chapter 4 we were able to show that we could discover a set of SCOP2 family categorisations directly from the data available in the Protein Data Bank using GProtA and our proposed difference measure. This strongly suggests that for global matching the difference measure is capturing the same information from the data as the expert curator. As the algorithm was applied directly against asymmetric units, as defined in the PDB, this also suggests the potential for the discovery of new family level similarity in as yet uncategorised proteins.

Using the curated domain definitions from SCOP2 we created characteristic signatures to give a single reference point for comparisons against a family category. An important consequence of defining a configuration that acts as a characteristic signature is that we then are assured that the global best match is the required solution. Characteristic signatures therefore also offer potential in unsupervised matching of superfamily categories, as a superfamily category may be derived from a subset of a family region and so may not be equivalent

to a global alignment. A clear limitation of the characteristic signature mechanism when applied to the SCOP2 database is the relatively small size of the database. This manifests in that only 92 of the 295 families defined have been applied to more than one protein, with only 40 being applied to 3 or more proteins; 3 is a practical minimum number of family members before there is enough information to define a useful characteristic signature.

Currently, the mechanism for deriving the characteristic signatures from the SCOP2 domain definitions is quite simple and only uses a small amount of the available information; an important piece of future work is to find a more comprehensive solution to this problem probably based on the construction of templates in Mardia, Nyirongo, et al. (2011).

Currently we have no mechanism to match the superfamilies where multiple instances of local structure are described with their relative position and orientation being free. The GProtA algorithm makes available the local optima derived from each branch and so there is potential for future research incorporating methods such as those developed by the *Critical Assessment of protein Structure Prediction* project (CASP) (Adcock et al. 2016) for combining distinct local matches.

Appendices

Appendix A

Data tables

A.1 Green and Mardia (2006) landmarks

The landmark data was retrieved from `http://www.maths.bris.ac.uk/~peter/Align/nicola.txt` (plain text file) on 2016-01-20. From the description of the source:

> Origin of data: the two examples are taken from the pdb data bank (a data bank of protein structures about 33,000 proteins available on the website `http://www.rcsb.org/pdb/`) and their pdb codes are 1cyd (the x configuration, 40 points) and 1a27 (the y configuration, 63 points). These two proteins belongs to the same family in what is called a SCOP classification (the family is Tyrosine-dependent oxireductase).

The two sets of landmarks are derived from experimentally determined co-ordinates for a subset of the amino acids in their respective proteins. The current Protein Data Bank data files are `http://www.rcsb.org/pdb/files/1cyd.cif` and `http://www.rcsb.org/pdb/files/1a27.cif`. The table row names are the `SeqNo` from the source data file which references the `_atom_site.label_seq_id`[1] in the Protein Data Bank data. The table rows are not ordered by `SeqNo`, this reflects the data source and not the Protein Data Bank. The `AA` table column is the `AA` column from the source data file and is the standard single letter amino acid identifier and equates to the `_atom_site.label_comp_id`[2] in the Protein Data Bank data. The `Grp` table column is the `Grp` column from the source data file. This is not specified in the PDB data and indicates membership of one of a set of of amino acid groupings, the source of which is not specified in the data. The `x`, `y` and `z` table columns are the derived amino acid co-ordinates. In Green and Mardia (2006) the

[1] `http://mmcif.wwpdb.org/dictionaries/mmcif_pdbx_v40.dic/Items/_atom_site.label_seq_id.html`
[2] `http://mmcif.wwpdb.org/dictionaries/mmcif_pdbx_v40.dic/Items/_atom_site.label_comp_id.html`

co-ordinates are described as being of "the centres of gravity of the amino acids". It is assumed that these were derived from the _atom_site.Cartn_x, _atom_site.Cartn_y and _atom_site.Cartn_z from the Protein Data Bank data.

$`1a27`

	AA	Grp	x	y	z
142	S	polar	11.025	4.871	0.160
143	V	hydrophobic	9.952	7.763	-2.125
144	G	glycine	8.368	4.823	-4.003
147	M	hydrophobic	4.696	8.465	-5.391
149	L	hydrophobic	6.845	4.892	-10.564
152	N	polar	7.394	-0.619	-10.226
155	Y	polar	6.923	-2.595	-5.550
185	C	polar	14.936	7.175	1.208
186	G	glycine	15.532	7.818	-2.508
187	P	hydrophobic	17.855	5.625	-4.572
192	F	hydrophobic	19.001	-3.613	-4.107
193	M	hydrophobic	18.666	-2.704	-7.750
218	Y	polar	12.409	3.386	-18.180
221	H	polar	16.164	6.563	-17.341
222	S	polar	16.649	4.763	-14.018
225	V	hydrophobic	18.443	8.564	-11.840
259	F	hydrophobic	11.985	14.693	-4.509
262	L	hydrophobic	7.456	16.258	-7.132
279	M	hydrophobic	6.434	11.881	-14.028
282	E	charged	10.655	12.932	-16.619
9	G	glycine	18.051	-7.229	6.714
10	C	polar	21.652	-6.572	7.700
11	S	polar	23.632	-8.613	5.123
12	S	polar	24.082	-5.417	3.065
13	G	glycine	22.962	-1.857	2.371
14	I	hydrophobic	21.065	0.299	4.872
15	G	glycine	20.787	-2.443	7.492
36	L	hydrophobic	20.001	-11.513	8.524
37	R	charged	17.847	-13.806	6.381
38	D	charged	20.834	-16.125	5.996
41	T	polar	26.167	-14.058	4.337
64	L	hydrophobic	15.786	-13.813	11.074
65	D	charged	13.749	-14.524	7.956
66	V	hydrophobic	10.738	-12.135	7.944
67	R	charged	8.493	-14.567	5.933
90	N	polar	14.423	-2.637	7.110
91	A	hydrophobic	14.228	-5.856	5.148
92	G	glycine	12.779	-5.295	1.662
93	L	hydrophobic	12.653	-6.493	-1.961
113	V	hydrophobic	8.345	-9.707	2.620
140	T	polar	12.996	2.001	6.206
141	G	glycine	10.761	3.973	3.851
159	K	charged	5.900	0.544	-0.187
188	V	hydrophobic	20.630	4.149	-2.397
190	T	polar	23.152	-1.038	-2.104
191	A	hydrophobic	22.705	-3.783	-4.760
195	K	charged	18.746	-7.927	-8.157
226	F	hydrophobic	20.621	6.145	-9.867
94	G	glycine	12.235	-4.847	-5.329
95	L	hydrophobic	10.206	-6.019	-8.297
110	V	hydrophobic	7.165	-8.887	-2.605
114	N	polar	7.752	-5.964	2.390
145	G	glycine	5.831	4.545	-1.113
146	L	hydrophobic	4.948	8.258	-1.554
153	D	charged	3.640	-1.184	-9.854
154	V	hydrophobic	3.836	-3.971	-7.258
156	C	polar	5.193	0.809	-5.111
157	A	hydrophobic	2.049	-1.074	-4.061
158	S	polar	4.127	-2.681	-1.314
160	F	hydrophobic	2.641	2.546	-0.113
163	E	charged	2.949	3.940	4.681
183	I	hydrophobic	12.924	6.969	7.094
184	E	charged	12.086	8.074	3.530

$`1cyd`

	AA	Grp	x	y	z
14	G	glycine	47.297	-3.124	21.673
15	A	hydrophobic	48.890	-6.527	22.166
16	G	glycine	50.774	-5.660	25.364
17	K	charged	48.000	-6.147	27.923
18	G	glycine	44.442	-7.348	28.422
19	I	hydrophobic	42.242	-8.488	25.570
20	G	glycine	44.937	-7.862	23.009
37	V	hydrophobic	51.849	-4.186	18.012
38	T	polar	51.871	-2.283	21.289
39	R	charged	53.063	0.915	22.967
40	T	polar	55.232	-0.358	25.854
43	D	charged	55.141	-5.983	27.393
59	V	hydrophobic	54.890	2.040	18.270
60	D	charged	52.512	4.657	19.606
61	L	hydrophobic	49.489	4.190	17.324
62	G	glycine	48.321	7.742	18.033
83	N	polar	42.241	-3.550	19.597
84	A	hydrophobic	44.352	-0.650	20.956
85	A	hydrophobic	42.349	2.045	22.748
86	L	hydrophobic	42.692	4.346	25.724
102	R	charged	44.371	11.711	22.039
106	V	hydrophobic	44.403	6.259	19.174
134	V	hydrophobic	37.155	-4.076	19.165
135	S	polar	34.387	-1.714	20.108
136	S	polar	32.332	-1.529	23.301
137	M	hydrophobic	28.723	-1.789	24.405
149	Y	polar	35.493	7.259	24.722
153	K	charged	33.149	3.738	19.801
179	P	hydrophobic	31.877	-6.104	25.315
180	T	polar	30.579	-4.818	28.626
181	V	hydrophobic	33.088	-4.253	31.456
182	V	hydrophobic	35.973	-6.790	31.411
184	T	polar	41.577	-5.807	32.609
185	D	charged	43.448	-2.637	33.586
186	M	hydrophobic	40.885	-0.618	31.636
187	G	glycine	37.870	-2.371	33.166
190	V	hydrophobic	36.341	2.371	34.675
138	V	hydrophobic	29.134	1.967	25.013
143	F	hydrophobic	28.267	5.221	30.820
146	L	hydrophobic	33.006	8.051	29.247

A.2 Co-ordinates of alpha carbon atoms for family characteristic signatures

	x	y	z
19	7.426	-5.637	-10.815
20	6.160	-7.342	-7.755
21	7.131	-6.200	-4.315
22	6.579	-7.768	-0.959
23	5.917	-5.983	2.212
24	5.882	-7.390	5.726
25	3.303	-6.289	8.261
26	4.055	-2.844	9.660
27	6.382	-1.860	6.825
28	5.861	1.506	5.155
29	5.140	1.315	1.447
30	5.119	5.057	0.954
31	5.812	8.075	3.072
32	4.412	11.501	3.015
33	6.805	13.793	1.523
38	5.162	5.139	-9.495
39	1.401	3.779	-8.820
40	0.777	0.689	-6.854
41	-1.771	-2.043	-7.105
42	-2.503	-4.201	-4.076
43	-2.571	-7.832	-5.153
44	-2.954	-9.410	-1.741
45	-3.110	-8.144	1.805
46	-4.266	-4.993	3.521
47	-2.808	-1.524	3.795
48	-3.658	1.493	5.883
49	-3.512	5.064	4.636
50	-2.618	7.529	7.333
51	-1.950	11.119	7.870
59	-5.668	9.624	11.371
60	-6.343	6.503	9.542
62	-8.622	4.403	5.117
63	-7.865	0.758	4.691
64	-7.140	-0.625	1.247
65	-7.699	-4.235	0.338
66	-6.912	-6.530	-2.524
67	-7.467	-5.207	-5.828
68	-7.112	-1.568	-5.072
69	-4.825	1.058	-6.509
70	-2.954	3.594	-4.429
83	0.385	13.126	0.443
84	-0.212	10.298	3.090
85	1.641	7.321	4.458
86	0.763	3.820	3.333
87	1.582	1.047	5.792
88	1.109	-2.710	5.523
89	-1.436	-3.949	8.055
90	-0.797	-7.540	7.091
91	1.932	-9.253	4.722
92	1.166	-7.678	1.364
93	2.015	-8.267	-2.264
94	1.982	-5.167	-4.422
95	2.661	-4.425	-8.064
96	4.349	-1.137	-8.814
99	4.032	6.771	-13.802
103	0.708	6.850	-18.715
117	-10.874	6.521	-11.338

Table A.1: FA:4000272

	x	y	z
8	3.536	2.283	-9.050
9	5.763	2.598	-6.072
10	4.026	4.333	-3.200
11	5.170	5.473	0.219
12	3.482	3.665	3.104
13	2.555	6.996	4.625
14	0.459	8.042	1.628
15	-3.042	8.878	2.814
16	-5.773	6.570	1.598
17	-7.696	9.532	0.296
18	-4.744	10.561	-1.828
19	-4.189	7.031	-3.058
21	-7.872	9.057	-5.595
22	-4.771	8.403	-8.048
24	-9.461	5.903	-9.012
25	-10.601	3.227	-6.057
35	-5.662	6.515	7.297
36	-2.645	5.490	5.251
37	-1.728	2.881	2.687
38	0.561	1.160	5.185
39	-2.291	1.046	7.691
40	-4.568	-0.506	5.102
41	-2.003	-3.156	4.329
42	-1.380	-3.941	7.964
43	-5.089	-4.217	8.566
44	-5.524	-6.603	5.653
50	7.378	-8.737	5.789
52	7.538	-9.599	-0.288
53	4.135	-10.137	1.193
54	1.302	-9.308	-1.115
55	-2.331	-8.537	-0.683
56	-5.060	-8.126	-3.189
57	-7.779	-5.621	-2.766
58	-10.123	-4.535	-5.950
60	-5.093	-4.048	-7.052
61	-2.156	-6.068	-5.910
62	-0.083	-4.492	-3.171
63	3.398	-5.825	-2.464
64	5.572	-4.759	0.444
65	8.820	-3.593	-1.062
66	10.495	-2.089	1.968
67	9.923	-2.158	5.683
71	6.542	0.253	9.385
72	5.843	-0.434	5.798
73	7.899	2.124	3.994
74	7.364	1.366	0.286
75	4.683	-0.593	-1.524
76	4.411	-1.670	-5.141
77	0.914	-1.401	-6.593
78	-0.107	-3.236	-9.720

Table A.2: `FA:4002490`

	x	y	z
146	4.878	11.727	-10.997
147	6.660	13.340	-8.135
149	5.608	8.399	-7.083
150	9.276	9.183	-6.790
151	8.951	10.631	-3.337
154	12.316	7.182	-1.904
155	10.525	7.757	1.314
157	7.960	12.037	4.038
158	8.135	15.712	3.981
159	8.009	15.844	0.210
160	4.962	13.618	0.081
161	3.429	16.229	2.365
163	2.993	17.548	-2.748
164	-0.276	16.363	-1.309
165	-1.162	20.233	-1.109
176	-3.016	10.246	-5.164
177	-4.549	7.632	-7.390
178	-2.132	7.991	-10.069
179	-0.749	4.868	-11.583
180	2.596	4.438	-13.278
182	3.170	-0.790	-14.268
183	-0.086	0.039	-12.641
184	1.394	0.928	-9.310
185	0.075	3.785	-7.229
186	2.126	5.713	-4.699
187	0.558	3.569	-1.979
188	1.746	0.387	-3.679
189	5.268	1.792	-3.716
190	5.200	2.906	-0.091
191	3.891	-0.501	0.976
192	6.505	-2.358	-1.028
193	9.307	-0.342	0.529
194	7.816	-0.596	3.992
195	7.407	3.100	4.581
196	4.731	3.034	7.255
197	4.656	6.775	7.743
198	4.107	7.440	4.035
199	1.433	4.751	3.875
200	-0.352	6.461	6.764
201	-0.123	9.889	5.154
202	-1.426	8.601	1.822
203	-4.319	6.841	3.519
204	-5.131	9.921	5.566
205	-5.211	11.846	2.301
206	-7.698	9.485	0.729
207	-5.635	7.019	-1.208
208	-7.659	4.009	-2.282
209	-6.428	1.194	-0.064
210	-7.921	-1.398	-2.367
211	-6.366	-0.216	-5.612
212	-5.168	-3.090	-7.788
213	-2.070	-2.984	-9.916
217	-4.315	-7.520	-9.602
218	-2.379	-7.122	-6.408
219	-3.564	-4.714	-3.752
220	-1.266	-2.930	-1.319
221	-2.061	-5.664	1.218
222	-0.791	-8.332	-1.166
223	2.451	-6.406	-1.479
224	2.812	-5.898	2.268
225	2.210	-9.604	2.757
226	4.674	-10.597	0.067
227	7.276	-8.303	1.541
228	6.664	-9.562	5.027
229	5.906	-6.151	6.475
230	3.717	-6.872	9.485
231	3.454	-3.279	10.661
232	2.255	-2.211	7.229
233	-0.213	-5.077	7.065
234	-1.591	-4.090	10.447
235	-1.955	-0.494	9.307
236	-3.717	-1.447	6.090
237	-6.099	-3.730	7.933
238	-6.866	-0.727	10.417
240	-9.942	-0.812	5.985
241	-8.233	-3.020	3.427
242	-10.124	-5.991	2.019
243	-8.341	-9.050	3.302
244	-10.065	-11.288	0.818
245	-9.252	-9.580	-2.442
247	-5.330	-11.427	-7.279
248	-5.326	-12.605	-11.121
249	-4.920	-16.052	-9.205
251	-4.547	-15.734	-4.384
252	-5.374	-13.709	-1.302
253	-2.844	-11.845	0.800
254	-2.904	-14.726	3.268
256	-0.221	-16.123	0.038
265	1.755	-12.027	9.012
266	-0.313	-15.120	8.532
267	-1.450	-14.875	12.095
268	-2.507	-11.289	11.716
269	-4.879	-12.188	8.402

Table A.3: `FA:4000366`

	x	y	z
16	-0.214	10.693	-7.841
17	1.290	7.467	-9.465
18	1.813	6.047	-6.032
19	-1.711	6.755	-4.892
21	-1.667	2.254	-7.401
22	-2.347	2.026	-3.619
23	-5.999	2.741	-4.319
24	-6.269	-0.041	-6.874
25	-4.897	-2.491	-4.324
26	-7.211	-1.292	-1.586
27	-10.277	-1.823	-3.778
29	-8.380	-5.996	-0.487
31	-5.471	-5.660	4.083
32	-2.282	-6.706	2.443
33	-0.055	-9.394	3.727
34	3.636	-9.087	4.331
35	4.397	-10.646	1.055
36	2.227	-8.274	-0.891
37	3.187	-5.198	0.999
39	6.163	-6.280	-3.069
40	4.020	-3.069	-3.539
41	6.818	-1.189	-1.864
42	9.047	-2.119	-4.722
43	6.517	-1.339	-7.340
44	5.559	2.024	-6.075
45	8.901	2.732	-4.531
46	7.673	3.298	-0.996
47	8.670	1.829	2.365
48	6.759	-1.063	3.934
49	5.953	1.268	6.812
50	4.516	3.955	4.557
51	2.319	1.485	2.691
52	1.050	0.018	5.951
53	0.108	3.406	7.376
54	-1.656	4.455	4.165
55	-3.813	1.315	4.230
56	-4.569	1.807	7.924
57	-5.736	5.394	7.461
58	-7.741	4.481	4.378
59	-9.767	1.873	6.239
60	-10.563	4.342	8.922

Table A.4: `FA:4000291`

	x	y	z
20	-9.950	-6.368	2.478
21	-8.074	-3.394	3.454
22	-8.888	-1.341	6.511
24	-5.618	2.185	8.784
25	-4.021	0.062	6.107
26	-5.269	2.313	3.352
27	-3.891	5.379	5.097
28	-0.581	3.705	5.767
29	-0.288	2.770	2.108
30	-1.019	6.276	0.956
31	1.510	7.687	3.368
32	4.082	5.135	2.325
33	3.602	5.815	-1.339
34	4.418	9.410	-0.665
35	7.673	8.340	1.142
37	7.521	6.308	-4.485
44	6.736	-1.646	-2.966
45	8.830	-3.947	-0.885
46	9.378	-0.952	1.334
47	5.738	-0.025	1.536
48	4.815	-3.556	2.494
50	5.081	-0.411	6.721
51	2.196	-2.395	6.365
54	0.149	-7.129	3.506
55	2.509	-8.414	0.943
56	4.296	-6.493	-1.759
57	2.047	-7.862	-4.411
58	-1.011	-7.070	-2.389
59	0.112	-3.491	-2.030
60	0.793	-3.172	-5.724
61	-2.593	-4.530	-6.618
62	-4.233	-2.133	-4.217
63	-2.471	0.777	-5.788
64	-3.964	-0.011	-9.099
65	-7.530	0.077	-7.424
66	-6.700	3.236	-5.785
67	-5.386	4.866	-8.762

Table A.5: `FA:4000317`

Appendix B

Code

B.1 Reference implementation of Green and Mardia (2006)

```
pkg_vars <- new.env()
pkg_vars$unittest_GM_fix_M       <- FALSE
pkg_vars$unittest_GM_fix_theta_x <- FALSE
pkg_vars$unittest_GM_fix_theta_y <- FALSE
pkg_vars$unittest_GM_fix_theta_z <- FALSE
pkg_vars$unittest_GM_fix_tau     <- FALSE
pkg_vars$unittest_GM_fix_sigma   <- FALSE


R_matrix <- function (theta) {
    cx <- cos(theta['x'])
    sx <- sin(theta['x'])
    cy <- cos(theta['y'])
    sy <- sin(theta['y'])
    cz <- cos(theta['z'])
    sz <- sin(theta['z'])
    R <- matrix(c(
        cz*cy, -cz*sy*sx-sz*cx, -cz*sy*cx+sz*sx,
        sz*cy, -sz*sy*sx+cz*cx, -sz*sy*cx-cz*sx,
           sy,           cy*sx,           cy*cx
    ), byrow = TRUE, nrow = 3)
    return( R )
}


M_filter <- function(M, A, B) {
    return( list(a = A[, M@i[M@x == 1]+1, drop = FALSE], b = B[, M@j[M@x == 1]+1, drop = FALSE]) )  # sometimes there are stray zero
        entries
}


# m = nrow(a), n = nrow(b)
# x is in the range 1 to n+m
# M is a matching matrix
M_propose <- function(x, M, m, n) {
    if ( x > m ) {
        # we chose from B
        k <- x - m
        j <- M@i[M@j+1 == k]+1  # will be numeric(0) if there is no match
        matched <- length(j) > 0  # should be 0 or 1
        if(! matched) { j <- NA }
        available <- setdiff(1:m, M@i + 1)
        if(length(available) == 0) {
```

```
            j_prop <- NA
        }
        else {
            if(length(available) == 1) {
                j_prop <- available
            } else {
                j_prop <- sample(available, 1) # behaviour changes if this is not a vector
            }
        }
        k_prop <- k
    }
    else {
        # we chose from A
        j <- x
        k <- M@j[M@i+1 == j]+1
        matched <- length(k) > 0
        if(! matched) { k <- NA }
        j_prop <- j
        available <- setdiff(1:n, M@j + 1)
        if(length(available) == 0) {
            k_prop <- NA
        }
        else {
            if(length(available) == 1) {
                k_prop <- available
            } else {
                k_prop <- sample(available, 1)
            }
        }
    }
    return( list(matched = matched, j = j, k = k, j_prop = j_prop, k_prop = k_prop) )
}


# x is a three vector
tristdnorm <- function(x) {
      return( prod( dnorm(x) ) )
}


# wrap [-pi/2, pi/2)
wrap <- function(a) {
    a = (a + pi/2) %% pi
    a <- ifelse(a < 0, a + pi, a)
    return(a - pi/2)
}

mcmc_GM <- function(A, B, mcmc_params) {
```

```
m <- ncol(A)
n <- ncol(B)

defaults = list(
    iterations = list(
        total = 1000000,
        burn = 200000,
        thin = 100,
        Mm = 10  # M multiplier - M updates per iteration
    ),
    starting_values = list(
        sigma = 1.0,
        theta = c(x = 0.0, y = 0.0, z = 0.0), # radians
        tau   = c(x = 0.0, y = 0.0, z = 0.0),
        M = uniqTsparse(as(Matrix(0, nrow = m, ncol = n), "dgTMatrix"))
    ),
    config = list(
        delta_theta_y = 0.1,
        tau_sigma = 50,
        tau_mu = c(0.0, 0.0, 0.0),
        sigma_alpha = 1,
        sigma_beta = 36,
        p_star = 0.5,
        lambda_over_rho = 0.003
    )
)
if(is.null(mcmc_params)){
    params <- defaults
} else {
    params <- list.merge(defaults, mcmc_params)
}

if(class(params$starting_values$M) != "dgTMatrix") {
    stop("params$starting_values$M not a Matrix of class 'dgTMatrix'")
}
if(! all(params$starting_values$M@x == as.integer(params$starting_values$M@x))) {
    stop("some content of params$starting_values$M is not integer")
}
params$config$tau_mu <- matrix(params$config$tau_mu, ncol = 1, dimnames = list(c('x', 'y', 'z')))

# stats for metropolis acceptance rate
stats <- list(
    theta_y_accept = 0,
    M_delete_propose = 0,
    M_delete_accept = 0,
    M_switch_propose = 0,
    M_switch_accept = 0,
    M_add_propose = 0,
```

```
    M_add_accept = 0
)

# how many to keep (burn in wanted for visuals)
N <- with(params$iterations, floor(total / thin))
# somewhere to put the sample
sample <- list(
    theta = data.frame(x = numeric(N), y = numeric(N), z = numeric(N)),
    tau   = data.frame(x = numeric(N), y = numeric(N), z = numeric(N)),
    sigma = numeric(N),
    M     = list()[1:N],
    L     = integer(N)
)

# starting values
sigma = params$starting_values$sigma
theta = params$starting_values$theta
tau   = matrix(params$starting_values$tau, ncol = 1, dimnames = list(c('x', 'y', 'z')))
M     = params$starting_values$M

# it is slightly more efficient to cache randomness
cache <- list(
    # matrix
    M_choice = sample(m+n, params$iterations$total * params$iterations$Mm + params$iterations$Mm, replace = TRUE),
    runif_01 = runif(params$iterations$total * params$iterations$Mm + params$iterations$Mm),
    runif_02 = runif(params$iterations$total * params$iterations$Mm + params$iterations$Mm),
    # transformations
    runif_03 = runif(floor(params$iterations$total)),
    runif_rot_01 = runif(floor(params$iterations$total), min = -params$config$delta_theta_y, max = params$config$delta_theta_y)
)

# fix values if unit testing
if(pkg_vars$unittest_GM_fix_theta_x) {
    theta['x'] <- pkg_vars$unittest_GM_values_theta_x
}
if(pkg_vars$unittest_GM_fix_theta_y) {
    theta['y'] <- pkg_vars$unittest_GM_values_theta_y
}
if(pkg_vars$unittest_GM_fix_theta_z) {
    theta['z'] <- pkg_vars$unittest_GM_values_theta_z
}
if(pkg_vars$unittest_GM_fix_tau) {
    tau <- pkg_vars$unittest_GM_values_tau
}
if(pkg_vars$unittest_GM_fix_sigma) {
    sigma <- pkg_vars$unittest_GM_values_sigma
}
if(pkg_vars$unittest_GM_fix_M) {
```

```
    M <- pkg_vars$unittest_GM_values_M
}

R <- R_matrix( theta )
L <- length(M@x)
nu <- n - L  # num non matched landmarks in B

k <- 1  # keep counter
for(i in 1:params$iterations$total) {

    if( ! pkg_vars$unittest_GM_fix_M ) {
        # multiple matrix updates per iteration
        for(ii in 0:(params$iterations$Mm - 1)) {
            iii <- i * params$iterations$Mm + ii

            # M - matches - metropolis-hastings
            choice <- M_propose(cache$M_choice[iii], M, m, n)
            if ( choice$matched ) {
                # currently matched
                if ( cache$runif_01[iii] <= params$config$p_star ) {
                    # propose deleting the match
                    stats$M_delete_propose <- stats$M_delete_propose + 1
                    if( L > 1 ) {  # we would always reject if no matches
                        accept <- params$config$lambda_over_rho * (sigma * sqrt(2))^3 /
                                  ( tristdnorm((A[ , choice$j, drop = FALSE] - R %*% B[, choice$k, drop = FALSE] - tau) / (sigma *
                                      sqrt(2))) * params$config$p_star * ( nu + 1 ))
                        if ( cache$runif_02[iii] < min(1, accept) ) {
                            M[choice$j, choice$k] <- 0
                            stats$M_delete_accept <- stats$M_delete_accept + 1
                        }
                    }
                }
                else {
                    # propose a switch
                    stats$M_switch_propose <- stats$M_switch_propose + 1
                    if( ! any(is.na(c(choice$j_prop, choice$k_prop))) ) {  # none available, must reject
                        accept <- tristdnorm( (A[ , choice$j_prop, drop = FALSE] - R %*% B[ , choice$k_prop, drop = FALSE] - tau) / (
                            sigma * sqrt(2))) /
                                  tristdnorm( (A[ , choice$j,      drop = FALSE] - R %*% B[ , choice$k,      drop = FALSE] - tau) / (
                                      sigma * sqrt(2)))
                        if ( cache$runif_02[iii] < min(1, accept) ) {
                            M[choice$j, choice$k] <- 0
                            M[choice$j_prop, choice$k_prop] <- 1
                            stats$M_switch_accept <- stats$M_switch_accept + 1
                        }
                    }
                }
            }
```

```
        else {
            # not currently matched
            # propose a match
            stats$M_add_propose <- stats$M_add_propose + 1
            if( ! any(is.na(c(choice$j_prop, choice$k_prop))) ) {  # none available or all already matched, must reject
                accept <- (tristdnorm((A[ , choice$j_prop, drop = FALSE] - R %*% B[, choice$k_prop, drop = FALSE] - tau) / (sigma
                     * sqrt(2))) * params$config$p_star * nu) /
                          (params$config$lambda_over_rho * (sigma * sqrt(2))^3)
                if ( cache$runif_02[iii] < min(1, accept) ) {
                    M[choice$j_prop, choice$k_prop] <- 1
                    stats$M_add_accept <- stats$M_add_accept + 1
                }
            }
        }

        L <- length( M@x )
        nu <- n - L
    }
}

f <- M_filter(M, A, B)

# R - rotation
# with F_0 being the zero matrix F is given by (3rd equation, section 3.1, GM 2006)
ff <- t((f$b %*% t(f$a - tau[ , rep(1, L)])) / (2 * sigma^2))   # TODO explain the transpose

# theta_x - gibbs
cy <- cos(theta['y'])
sy <- sin(theta['y'])
cz <- cos(theta['z'])
sz <- sin(theta['z'])
if( ! pkg_vars$unittest_GM_fix_theta_x ) {
    ax <- (ff[2, 2] - ff[1, 3] * sy) * cz + (- ff[1, 2] - ff[2, 3] * sy) * sz + ff[3, 3] * cy
    bx <- (- ff[2, 3] - ff[1, 2] * sy) * cz + (ff[1, 3] - ff[2, 2] * sy) * sz + ff[3, 2] * cy
    aa <- atan2(bx, ax)
    kk <- ifelse(sin(aa) == 0, 0, bx / sin(aa))
    theta['x'] <- rvonmises(1, k = kk, a = aa)
}

# theta_y - metropolis-hastings
cx <- cos(theta['x'])
sx <- sin(theta['x'])
if( ! pkg_vars$unittest_GM_fix_theta_y ) {
    theta_y_prop <- wrap(theta['y'] + cache$runif_rot_01[i])
    ay <- ff[1, 1] * cz + ff[2, 1] * sz + ff[3, 2] * sx + ff[3, 3] * cx
    by <- (- ff[1, 2] * sx - ff[1, 3] * cx) * cz + (- ff[2, 2] * sx - ff[2, 3] * cx) * sz + ff[3, 1]
    accept <- exp((ay * cos(theta_y_prop) + by * sin(theta_y_prop)) - (ay * cy + by * sy)) * cos(theta_y_prop) / cy
    if ( cache$runif_03[i] < min(1, accept) ) {
```

```
            theta['y'] <- theta_y_prop
            stats$theta_y_accept <- stats$theta_y_accept + 1
        }
    }

    # theta_z - gibbs
    cy <- cos(theta['y'])
    sy <- sin(theta['y'])
    if( ! pkg_vars$unittest_GM_fix_theta_z ) {
        az <- (ff[2, 2] - ff[1, 3] * sy) * cx + (- ff[2, 3] - ff[1, 2] * sy) * sx + ff[1, 1] * cy
        bz <- (- ff[1, 2] - ff[2, 3] * sy) * cx + (ff[1, 3] - ff[2, 2] * sy) * sx + ff[2, 1] * cy
        aa <- atan2(bz, az)
        kk <- ifelse(sin(aa) == 0, 0, bz / sin(aa))
        theta['z'] <- rvonmises(1, k = kk, a = aa)
    }

    R <- R_matrix( theta )
    Rb <- R %*% f$b

    # tau - translation - gibbs
    if( ! pkg_vars$unittest_GM_fix_tau ) {
        tau_denominator <- params$config$tau_sigma^(-2) + L * 0.5 * sigma^(-2)
        tau_mean <- (params$config$tau_mu / params$config$tau_sigma^(2) + (rowSums(f$a - Rb) / (2 * sigma^2))) / tau_denominator
        tau_cv <- diag(1 / tau_denominator, 3)
        tau[ , 1] <- mvrnorm(1, tau_mean, tau_cv)
    }

    # sigma^-2 - variance of noisy measurements - gibbs
    if( ! pkg_vars$unittest_GM_fix_sigma ) {
        sigma_shape <- params$config$sigma_alpha + 1.5 * L
        sigma_rate <- params$config$sigma_beta + 0.25 * norm(f$a - Rb - tau[ , rep(1, L)], type = 'F')^2
        sigma <- 1 / sqrt(rgamma(1, sigma_shape, rate = sigma_rate))
    }

    # store
    if ( i %% params$iterations$thin == 0 ) {
        sample$theta[k, ] <- theta
        sample$tau[k, ]   <- tau
        sample$sigma[k]   <- sigma
        sample$M[[k]]     <- uniqTsparse(M)
        sample$L[k]       <- L
        k <- k + 1
    }
}

return(bprota_mcmc(
    sample = sample,
    sample_metadata = list(
```

```
            total = N,
            burn  = with(params$iterations, floor(burn / thin))
        ),
        accept_rates = with(stats, c(
            theta_y = theta_y_accept / params$iterations$total,
            M_delete = M_delete_accept / M_delete_propose,
            M_switch = M_switch_accept / M_switch_propose,
            M_add = M_add_accept / M_add_propose))
    ))
}

estimate_loss <- function(x, A, B) {

    Ms <- x$sample$M[x$sample_metadata$burn:x$sample_metadata$total]
    M_sum <- as.matrix(Reduce('+', Ms))
    Pjk <- M_sum / length(Ms)
    K <- 0.5
    loss_jk <- K - Pjk  # const = 0, l_ab = l_ba = 0.5

    # pad
    m <- nrow(M_sum)
    n <- ncol(M_sum)
    if ( m < n ) {
        loss_jk <- rbind(loss_jk, matrix(0.5, nrow = n - m, ncol = n))
    }

    # linear assignment
    M <- (lp.assign(loss_jk, direction = 'min'))$solution
    # bug in lp.assign are not exactly 1 so exact equality tests would have unexpected results
    M <- round(M)

    # filter
    # if an assignment has +ve loss remove it
    M[loss_jk >= 0] = 0
    # remove padding
    M <- M[1:m, ]

    return(bprota_estimate(
        M = uniqTsparse(as(M, "dgTMatrix")),
        A = A,
        B = B
    ))
}

align <- function(
    A,
    B,
    mcmc_params = NULL
```

```
) {
    if( ! is.matrix(A) || ! is.matrix(B) || ncol(A) < 2 || ncol(B) < 2 || nrow(A) != 3 || nrow(B) != 3 )
        stop("'A' and 'B' must be 3 row matrices, each with at least two columns.")
    if( ncol(A) > ncol(B) ) stop("'B' must not have a greater number of columns than 'B'.")

    mcmc_rv <- mcmc_GM(
        A = A,
        B = B,
        mcmc_params = mcmc_params
    )

    estimate_rv <- estimate_loss(
        x = mcmc_rv,
        A = A,
        B = B
    )

    rv <- bprota(
        mcmc = mcmc_rv,
        estimate = estimate_rv,
        A = A,
        B = B,
        CALL = match.call()
    )

    return( rv )
}

rvonmises <- function(n, k, a) {
    if(k == 0){
        vm <- runif(n = n, min = 0, max = 2*pi)
    } else {
        a <- a + pi
        vm <- c(1:n)
        aa <- 1 + (1 + 4 * (k^2))^0.5
        bb <- (aa - (2 * aa)^0.5)/(2 * k)
        rr <- (1 + bb^2)/(2 * bb)
        obs <- 1
        while(obs <= n) {
            U1 <- runif(1, 0, 1)
            zz <- cos(pi * U1)
            ff <- (1 + rr * zz)/(rr + zz)
            cc <- k * (rr - ff)
            U2 <- runif(1, 0, 1)
            if(cc * (2 - cc) - U2 > 0) {
                U3 <- runif(1, 0, 1)
                vm[obs] <- sign(U3 - 0.5) * acos(ff) + a
                vm[obs] <- vm[obs] %% (2 * pi)
```

```
                obs <- obs + 1
            }
            else {
                if(log(cc/U2) + 1 - cc >= 0) {
                    U3 <- runif(1, 0, 1)
                    vm[obs] <- sign(U3 - 0.5) * acos(ff) + a
                    vm[obs] <- vm[obs] %% (2 * pi)
                    obs <- obs + 1
                }
            }
        }
    }
    vm <- vm - pi
    return(vm)
}

dvonmises <- function (theta, k, a) {
    rv <- 1/(2 * pi * besselI(x = k, nu = 0, expon.scaled = TRUE)) * (exp(cos(theta - a) - 1))^k
    return(rv)
}

# constructor
bprota_estimate <- function(M, A, B) {
    obj <- structure(
        list(
            M = M,
            A = A,
            B = B
        ),
        class = 'bprota_estimate'
    )

    invisible(obj)
}



# s3 methods
# ----------
# ref: http://cran.r-project.org/doc/manuals/r-devel/R-exts.html#Generic-functions-and-methods
# for explanation of argument names

#summary.bprota_estimate <- function(object,...) {
#    print("TODO\n")
#    invisible(object)
#}
```

```
print.bprota_estimate <- function(x,...) {
    print(x$M)
    invisible(x)
}

plot.bprota_estimate <- function(x, ...) {
    A_matches <- rowSums(x$M)
    B_matches <- colSums(x$M)
    ff <- M_filter(x$M, x$A, x$B)

    # witch A and B
    A <- t(x$A)
    B <- t(x$B)
    a <- t(ff$a)
    b <- t(ff$b)
    ac <- scale(a, center = TRUE, scale = FALSE)
    bc <- scale(b, center = TRUE, scale = FALSE)
    R <- optimal_rotation(ac, bc)
    Ap <- A - matrix(attr(ac, 'scaled:center'), byrow = TRUE, ncol = 3, nrow = nrow(A))
    Bp <- B - matrix(attr(bc, 'scaled:center'), byrow = TRUE, ncol = 3, nrow = nrow(B))
    BpR <- Bp %*% R
    plot3d(Ap, type = 's', radius = 0.4, box = FALSE, col = '#4daf4a', axes = FALSE, xlab = '', ylab = '', zlab = '', alpha = ifelse(A_
        matches, 1.0, 0.3))
    plot3d(BpR, type = 's', radius = 0.4, add = TRUE, col = '#984ea3', alpha = ifelse(B_matches, 1.0, 0.3))

    A_index <- as.integer(rownames(A))
    for(i in 1:(nrow(A)-1)) {
        if(A_index[i] == (A_index[i+1] - 1)) {
            plot3d(Ap[i:(i+1), ], col = '#4daf4a', type = 'l', lwd = 3, add = TRUE, alpha = ifelse(A_matches[i] == 1 && A_matches[i+1] ==
                 1, 1.0, 0.3))
        }
    }

    B_index <- as.integer(rownames(B))
    for(i in 1:(nrow(B)-1)) {
        if(B_index[i] == (B_index[i+1] - 1)) {
            plot3d(BpR[i:(i+1), ], col = '#984ea3', type = 'l', lwd = 3, add = TRUE, alpha = ifelse(B_matches[i] == 1 && B_matches[i+1]
                == 1, 1.0, 0.3))
        }
    }

    invisible(x)
}

# constructor
bprota_mcmc <- function(sample, sample_metadata, accept_rates) {
    obj <- structure(
        list(
```

```
            sample = sample,
            sample_metadata = sample_metadata,
            accept_rates = accept_rates
        ),
        class = 'bprota_mcmc'
    )
    invisible(obj)
}


# s3 methods
# ----------
# ref: http://cran.r-project.org/doc/manuals/r-devel/R-exts.html#Generic-functions-and-methods
# for explanation of argument names

plot.bprota_mcmc <- function(x, ...) {
    cache_par <- par(no.readonly = TRUE) # save default

    par(mfrow = c(5, 3), mar = c(4, 3, 3, 1), lwd = 0.5, bty = 'n')
    # matches
    plot(x$sample$L, type = 'l', main = expression(paste('# matches')), xlab = '', ylab = '')
    abline(v = x$sample_metadata$burn, col = 'blue', lty = 5)
    # sigma
    hist(x$sample$sigma[x$sample_metadata$burn:x$sample_metadata$total], freq = FALSE, breaks = 20, main=expression(sigma), xlab = '',
        ylab = '')
    plot(x$sample$sigma, type = 'l', main=expression(sigma), xlab = '', ylab = '')
    abline(v = x$sample_metadata$burn, col = 'blue', lty = 5)
    # tau
    hist(x$sample$tau[ , 'x'][x$sample_metadata$burn:x$sample_metadata$total], freq = FALSE, breaks = 20, main = expression(tau [x]),
        xlab = '', ylab = '')
    hist(x$sample$tau[ , 'y'][x$sample_metadata$burn:x$sample_metadata$total], freq = FALSE, breaks = 20, main = expression(tau [y]),
        xlab = '', ylab = '')
    hist(x$sample$tau[ , 'z'][x$sample_metadata$burn:x$sample_metadata$total], freq = FALSE, breaks = 20, main = expression(tau [z]),
        xlab = '', ylab = '')
    plot(x$sample$tau[ , 'x'], type = 'l', main = expression(tau [x]), xlab = '', ylab = '')
    abline(v = x$sample_metadata$burn, col = 'blue', lty = 5)
    plot(x$sample$tau[ , 'y'], type = 'l', main = expression(tau [y]), xlab = '', ylab = '')
    abline(v = x$sample_metadata$burn, col = 'blue', lty = 5)
    plot(x$sample$tau[ , 'z'], type = 'l', main = expression(tau [z]), xlab = '', ylab = '')
    abline(v = x$sample_metadata$burn, col = 'blue', lty = 5)
    # theta
    hist(x$sample$theta[ , 'x'][x$sample_metadata$burn:x$sample_metadata$total], freq = FALSE, breaks = 20, xlim=c(-pi, pi), main =
        expression(theta [x]), xlab = '', ylab = '', xaxt = 'n')
    axis(1, at = c(-pi, -pi/2, 0, pi/2, pi), labels = c(expression(-pi), expression(-pi/2), expression(0), expression(pi/2), expression(
        pi)), las = 0)
    hist(x$sample$theta[ , 'y'][x$sample_metadata$burn:x$sample_metadata$total], freq = FALSE, breaks = 20, xlim=c(-pi/2, pi/2), main =
        expression(theta [y]), xlab = '', ylab = '', xaxt = 'n')
    axis(1, at = c(-pi/2, -pi/4, 0, pi/4, pi/2), labels = c(expression(-pi/2), expression(-pi/4), expression(0), expression(pi/4),
```

```
        expression(pi/2)), las = 0)
    hist(x$sample$theta[ , 'z'][x$sample_metadata$burn:x$sample_metadata$total], freq = FALSE, breaks = 20, xlim=c(-pi, pi), main =
        expression(theta [z]), xlab = '', ylab = '', xaxt = 'n')
    axis(1, at = c(-pi, -pi/2, 0, pi/2, pi), labels = c(expression(-pi), expression(-pi/2), expression(0), expression(pi/2), expression(
        pi)), las = 0)
    plot(x$sample$theta[ , 'x'], type = 'l', main = expression(theta [x]), ylim = c(-pi, pi), xlab = '', ylab = '', yaxt = 'n')
    abline(v = x$sample_metadata$burn, col = 'blue', lty = 5)
    axis(2, at = c(-pi, -pi/2, 0, pi/2, pi), labels = c(expression(-pi), expression(-pi/2), expression(0), expression(pi/2), expression(
        pi)), las = 0)
    plot(x$sample$theta[ , 'y'], type = 'l', main = expression(theta [y]), ylim = c(-pi/2, pi/2), xlab = '', ylab = '', yaxt = 'n')
    abline(v = x$sample_metadata$burn, col = 'blue', lty = 5)
    axis(2, at = c(-pi/2, 0, pi/2), labels = c(expression(-pi/2), expression(0), expression(pi/2)), las = 0)
    plot(x$sample$theta[ , 'z'], type = 'l', main = expression(theta [z]), ylim = c(-pi, pi), xlab = '', ylab = '', yaxt = 'n')
    abline(v = x$sample_metadata$burn, col = 'blue', lty = 5)
    axis(2, at = c(-pi, -pi/2, 0, pi/2, pi), labels = c(expression(-pi), expression(-pi/2), expression(0), expression(pi/2), expression(
        pi)), las = 0)

    par(cache_par)
    invisible(x)
}
```

B.2 Distribution of acceptance probabilities for perturbations of the matching matrix (Figure 2.11)

```
library(phdutils)
library(Matrix)
library(rgl)
library(robustbase)

# x is a three vector
tristdnorm <- function(x) {
      return( prod( dnorm(x) ) )
}

A <- as.matrix(green_mardia_2006_active_sites$`1cyd`[, c('x', 'y', 'z')])
B <- as.matrix(green_mardia_2006_active_sites$`1a27`[, c('x', 'y', 'z')])

m <- nrow(A)
n <- nrow(B)

rho_o_lambda <- 1 / 0.003

M_best_36 <- readRDS('best_36.rds')
correct_indexes <- summary(M_best_36)
correct <- paste(correct_indexes$i, correct_indexes$j, sep = "_")

probs <- function(M, sigma_beta) {
    L <- length(M@x)

    if(L == 0) {
        a <- A
        b <- B
        R <- diag(3)

        this_beta <- sigma_beta
    } else {
        f <- M_filter(M, A, B)

        ac <- scale(f$a, center = TRUE, scale = FALSE)
        bc <- scale(f$b, center = TRUE, scale = FALSE)
        R <- optimal_rotation(ac, bc)

        a <- A - matrix(attributes(ac)$`scaled:center`, nrow = nrow(A), ncol =3, byrow = TRUE)
        b <- (B - matrix(attributes(bc)$`scaled:center`, nrow = nrow(B), ncol =3, byrow = TRUE)) %*% R

        this_beta <- sigma_beta + 0.25 * norm(ac - bc %*% R, type = 'F')^2
    }
```

```
    this_alpha <- sigma_alpha + 1.5 * L

    vals_this <- seq(
        from = qgamma(0.0005, shape = this_alpha, rate = this_beta),
        to = qgamma(0.9995, shape = this_alpha, rate = this_beta),
        length.out = 1000
    )
    density_this <- dgamma(vals_this, shape = this_alpha, rate = this_beta)
    density_this <- 0.999 * density_this / sum(density_this)
    sigma_this <- 1 / sqrt(vals_this)

    id <- setdiff(0:(nrow(M)-1), M@i) + as.integer(1)
    jd <- setdiff(0:(ncol(M)-1), M@j) + as.integer(1)

    p <- expand.grid(i = id, j = jd, KEEP.OUT.ATTRS = FALSE)

    dss2 <- rowSums((a[p$i, , drop = FALSE] - b[p$j, , drop = FALSE])^2)

    LuA <- m - L + 1

    z <- sapply(sqrt(dss2), function(x) {
        # calculations for quadrature (integration) over sigma
        # acceptance probability
        h <- sapply(sigma_this, function(y){ tristdnorm( x / (y * sqrt(2))) }, USE.NAMES = FALSE)
        sum( density_this * (LuA * rho_o_lambda * h / (sigma_this * sqrt(2))^3) )
    }, USE.NAMES = FALSE)

    # assume M is correct
    M_adj_c <- paste(p$i, p$j, sep = "_") %in% correct

    return( list(good = z[M_adj_c], bad = z[! M_adj_c]) )
}

sigma_alpha <- 1
sigma_beta <- 1/36

results <- list()

M <- as(Matrix(0, nrow = m, ncol = n), 'dgTMatrix')
results[[1]] <- probs(M , sigma_beta )

for(i in 1:9) {
    for(ii in 1:100) {
        # make a starting matrix
        M <- as(Matrix(0, nrow = m, ncol = n), 'dgTMatrix')
        if(i == 1) {
            if(ii == 37) { break }
            M[ as.matrix(correct_indexes[i, c('i', 'j')]) ] <- 1
```

```
        } else {
            M[ as.matrix(correct_indexes[sample(36, i), c('i', 'j')]) ] <- 1
        }
        M <- uniqTsparse(M)
        #
        rv <- probs(M, sigma_beta )
        if(ii == 1) {
            results[[i+1]] <- rv
        } else {
            results[[i+1]]$good <- c(results[[i+1]]$good, rv$good)
            results[[i+1]]$bad <- c(results[[i+1]]$bad, rv$bad)
        }
    }
}
#saveRDS(results, 'results.rds')

#pdf(file = 'two_regimes_01.pdf', height = 7, width = 10)
par(mar = c(8, 5, 4, 2) + 0.1)
labels <- c('1st correct', '1st wrong', '2nd correct', '2nd wrong', '3rd correct', '3rd wrong', '4th correct', '4th wrong', '5th correct'
    , '5th wrong', '6th correct', '6th wrong', '7th correct', '7th wrong', '8th correct', '8th wrong', '9th correct', '9th wrong', '10th
    correct', '10th wrong')
at <- c(1,2, 3.5,4.5, 6,7, 8.5,9.5, 11,12, 13.5,14.5, 16,17, 18.5,19.5, 20.5,21.5, 23,24)
err <- sqrt(.Machine$double.eps)
vals <- adjbox(list(
    results[[1]]$good +err,
    results[[1]]$bad +err,
    results[[2]]$good +err,
    results[[2]]$bad +err,
    results[[3]]$good +err,
    results[[3]]$bad +err,
    results[[4]]$good +err,
    results[[4]]$bad +err,
    results[[5]]$good +err,
    results[[5]]$bad +err,
    results[[6]]$good +err,
    results[[6]]$bad +err,
    results[[7]]$good +err,
    results[[7]]$bad +err,
    results[[8]]$good +err,
    results[[8]]$bad +err,
    results[[9]]$good +err,
    results[[9]]$bad +err,
    results[[10]]$good +err,
    results[[10]]$bad +err
), log = 'y',  xlab = "", ylab = "f(M'|M)", cex.lab = 1.5, frame = FALSE, col = c(gray(0.9), gray(0.9), gray(0.75), gray(0.75)), pch = 4,
     pars = list(xaxt = 'n', yaxt = 'n'), at = at)
axis(1, at = at, labels = FALSE)
text(at, 1e-9, srt = 45, adj = 1, labels = labels, xpd = TRUE, col = c(gray(0.4), gray(0.4), gray(0.1), gray(0.1)))
```

```
mtext(side = 1, text = "match", line = 5, cex = 1.5)
ylab = c(1e-16, 1e-12, 1e-8, 1e-4, 0.01, 1, 100, 1e4, 1e6)
axis(2, at = ylab, labels = ylab)
abline(h = 1, lty = 5, col = gray(0.1), lwd = 1.2)
#dev.off()
```

B.3 GProtA

```
# hack to stop R CMD check warnings
M = parent_M = gamma = dss2 = branch = status = NULL

dss2 <- function(M, A, B, Hs) {
    f <- M_filter(M, A, B)
    ah <- Hs %*% f$a
    bh <- Hs %*% f$b
    min_f_norm(ah, bh)^2
}

M_adjacent_greedy <- function(M, A, B) {
    f <- M_filter(M, A, B)
    ac <- scale(f$a, center = TRUE, scale = FALSE)
    bc <- scale(f$b, center = TRUE, scale = FALSE)
    R <- optimal_rotation(ac, bc)

    a <- A - matrix(attributes(ac)$`scaled:center`, nrow = nrow(A), ncol =3, byrow = TRUE)
    b <- (B - matrix(attributes(bc)$`scaled:center`, nrow = nrow(B), ncol =3, byrow = TRUE)) %*% R

    summary_M <- summary(M)
    id <- setdiff(1:(nrow(M)), summary_M$i)
    jd <- setdiff(1:(ncol(M)), summary_M$j)

    p <- expand.grid(i = id, j = jd)

    dss2 <- rowSums((a[p$i, , drop = FALSE] - b[p$j, , drop = FALSE])^2)
    wanted <- which.min(dss2)

    L <- length(M@x)
    M@i[L+1] <- as.integer(p$i[wanted]-1)
    M@j[L+1] <- as.integer(p$j[wanted]-1)
    M@x[L+1] <- 1

    return( list(M = uniqTsparse(M), dss2 = dss2[wanted]) )
}

consec_subs <- function(X, gamma){
    lapply(1:(nrow(X) - gamma + 1), function(x){
        return( x:(x + gamma - 1) )
    })
}

get_candidates <- function(A, B, gamma = 4) {
```

```
    A_sets <- consec_subs(A, gamma)
    A_sets <- c(A_sets, lapply(A_sets, rev))  # plus the reverse
    B_sets <- consec_subs(B, gamma)
    set_combinations <- expand.grid(1:length(A_sets), 1:length(B_sets))

    Mc <- apply(set_combinations, 1, function(x) {
        T <- sparseMatrix(i = A_sets[[x[1]]], j = B_sets[[x[2]]], x = rep(1, gamma), dims = c(nrow(A), nrow(B)), giveCsparse = FALSE)
        uniqTsparse(T)
    })
    return(Mc)
}

get_starting <- function(A, B, Mc, snull, fdr = 0.01, max_starts = NULL) {
    gamma <- sum(Mc[[1]])
    Hs <- helmert(gamma, submatrix = TRUE)
    d <- sapply(Mc, dss2, A = A, B = B, Hs = Hs)
    p_value <- sapply(d, function(x) { sum(snull <= x) / length(snull) })
    p_adjusted <- p.adjust(p_value, method = 'bonferroni')
    significant <- which(p_adjusted <= fdr)

    # filter out overlaps
    oo <- order(d)
    keep <- c()
    used <- c()
    for(i in oo) {
        if(! i %in% significant) { break }  # must be significant
        f <- summary(Mc[[i]])
        g <- paste(f$i, f$j, sep = '_')
        if(any(g %in% used)) { next }
        used <- c(used, g)
        keep <- c(keep, i)
    }

    if(! is.null(max_starts) && length(keep) > max_starts ) {
        keep <- keep[1:max_starts]
    }

    return( list( matrices = Mc[keep], dss2 = d[keep]) )
}


align <- function(A, B, snull, tolerance = 3.51, fdr = 0.01, starting_gamma = 4, max_branches = NULL) {

    Mc <- get_candidates(A, B, gamma = starting_gamma)
    Ms <- get_starting(A, B, Mc, snull, fdr, max_starts = max_branches)

    # status
    # *C*ontinue from here on branch
```

```
    # *J*oined with other branch (this one is a repeat with a different parent [denormalized])
    # *S*topped

    this_Ms <- Ms[['matrices']]
    lMs <- length(this_Ms)
    results <- data.table(
        M = this_Ms,
        parent_M = rep('none', lMs),
        gamma = rep(starting_gamma, lMs),
        dss2 = Ms[['dss2']],
        branch = 1:lMs,
        status = 'C',
        key = c('gamma','dss2', 'status', 'branch')
    )

    for(g in (starting_gamma + 1):(nrow(A))) {
        branches <- results[gamma == g - 1 & status == 'C']
        if(nrow(branches) == 0) { break }
        Hs <- helmert(g, submatrix = TRUE)
        tmp <- rbindlist(apply(branches, 1, function(x) {
            Mc <- M_adjacent_greedy(M = x$M, A = A, B = B)
            return( data.table(M = list(Mc$M), branch = x$branch, parent_M = list(x$M), dss2 = Mc$dss2, gamma = g, status = ifelse(sqrt(
                Mc$dss2) <= tolerance, 'C', 'S')) )
        }), use.names = TRUE, fill = TRUE)

        tmp[duplicated(M), status := 'J']
        results <- rbindlist(list(results, tmp), use.names = TRUE)
    }
    setkey(results, gamma, branch, dss2, status)
    M <- results[status == 'C'][gamma == max(gamma)][dss2 == min(dss2), M][[1]]
    gprota(
        M = M,
        results = results,
        A = A,
        B = B,
        CALL = match.call()
    )
}

null_sample <- function(A, B, gamma, N) {
    m <- nrow(A)
    n <- nrow(B)
    Hs <- helmert(gamma, submatrix = TRUE)
    Ms <- replicate(N, sparseMatrix(i = sample(m, gamma), j = sample(n, gamma), x = rep(1, gamma), dims = c(m, n), giveCsparse = FALSE))
    return( sapply(Ms, dss2, A = A, B = B, Hs = Hs) )
}
```

```
# constructor
gprota <- function(M, results, A, B, CALL) {
    obj <- structure(
        list(
            M = M,
            results = results,
            A = A,
            B = B,
            CALL = CALL,
            PACKAGE_VERSION = packageVersion('GProtA')
        ),
        class = 'gprota'
    )
    invisible(obj)
}

# s3 methods
# ----------
# ref: http://cran.r-project.org/doc/manuals/r-devel/R-exts.html#Generic-functions-and-methods
# for explanation of argument names

summary.gprota <- function(object, ...) {
    tmp <- object$results[status %in% c('C', 'J')]
    gammas <- sort(unique(tmp$gamma))
    branches <- sort(unique(tmp$branch))
    plot(tmp$gamma, tmp$branch, bty = 'n', xaxt = 'n', yaxt = 'n', xlab = expression(gamma), ylab = 'branch', cex.lab = 1.5)
    axis(1, tick = FALSE, lty = 0, las = 1, at = gammas, labels = gammas, line = -1)
    axis(2, tick = FALSE, lty = 0, las = 1, at = branches, labels = branches, line = -0.5)
    for(b in branches) {
        lines(tmp$gamma[tmp$branch == b], rep(b, sum(tmp$branch == b)))
    }
    for(f in which(tmp$status == 'J')) {
        from <- tmp[f, ]
        i <- which(M2hash(tmp[status == 'C' & gamma == from$gamma]$M) == M2hash(from$M))
        to <- tmp[status == 'C' & gamma == from$gamma][i]
        arrows(x0 = from$gamma, y0 = from$branch, x1 = to$gamma, y1 = to$branch, length = 0.2)
    }
    points(tmp$gamma, tmp$branch, pch = 16, col = ifelse(tmp$status == 'C', 'darkgray', 'lightgray'), cex = 2)
    invisible(object)
}

print.gprota <- function(x, ...) {
    print( list(
        M = x$M,
        results = x$results[, list(gamma, branch, dss2, status)],
        PACKAGE_VERSION = x$PACKAGE_VERSION,
        CALL = x$CALL
    ) )
```

```
    invisible(x)
}

plot.gprota <- function(x, branch = NULL, ...) {
    if(is.null(branch)) {
        M <- x$M
    } else {
        p_branch <- branch  # can't use a variable same as column name
        M <- x$results[status == 'C' & branch == p_branch][gamma == max(gamma), M][[1]]
    }
    A_matches <- rowSums(M)
    B_matches <- colSums(M)
    ff <- M_filter(M, x$A, x$B)
    ac <- scale(ff$a, center = TRUE, scale = FALSE)
    bc <- scale(ff$b, center = TRUE, scale = FALSE)
    R <- optimal_rotation(ac, bc)
    Ap <- x$A - matrix(attr(ac, 'scaled:center'), byrow = TRUE, ncol = 3, nrow = nrow(x$A))
    Bp <- x$B - matrix(attr(bc, 'scaled:center'), byrow = TRUE, ncol = 3, nrow = nrow(x$B))
    BpR <- Bp %*% R
    plot3d(Ap, type = 's', radius = 0.4, box = FALSE, col = '#4daf4a', axes = FALSE, xlab = '', ylab = '', zlab = '', alpha = ifelse(A_
        matches, 1.0, 0.3))
    plot3d(BpR, type = 's', radius = 0.4, add = TRUE, col = '#984ea3', alpha = ifelse(B_matches, 1.0, 0.3))

    A_index <- as.integer(rownames(x$A))
    for(i in 1:(nrow(x$A)-1)) {
        if(A_index[i] == (A_index[i+1] - 1)) {
            plot3d(Ap[i:(i+1), ], col = '#4daf4a', type = 'l', lwd = 3, add = TRUE, alpha = ifelse(A_matches[i] == 1 && A_matches[i+1] ==
                1, 1.0, 0.3))
        }
    }

    B_index <- as.integer(rownames(x$B))
    for(i in 1:(nrow(x$B)-1)) {
        if(B_index[i] == (B_index[i+1] - 1)) {
            plot3d(BpR[i:(i+1), ], col = '#984ea3', type = 'l', lwd = 3, add = TRUE, alpha = ifelse(B_matches[i] == 1 && B_matches[i+1]
                == 1, 1.0, 0.3))
        }
    }

    invisible(x)
}
```

B.4 DBSCAN example

```
library(rgl)
library(shapes)
library(phdutils)
library(dbscan)
library(RColorBrewer)

paths <- sample(list.files('~/work/phd/data/PDB/asymetric_units/', full.names = TRUE, pattern = '*.csv.gz'), 15)

sequence <- c('N', 'CA', 'C', 'O', 'CB', 'CG', 'OD1', 'OD2')
residues <- lapply(paths, function(path) {
    A_all <- read.delim(path, header=TRUE, sep=',', stringsAsFactors = FALSE)
    A <- A_all[A_all$label_comp_id == 'ASP' & A_all$label_atom_id != 'H', c('label_atom_id', 'auth_seq_id', 'cartn_x', 'cartn_y', 'cartn_
        z')]
    A <- split(A, A$auth_seq_id)
    lapply(A, function(x) {
        if(all(sequence %in% x$label_atom_id) && nrow(x) == length(sequence)) {
            rownames(x) <- x$label_atom_id
            x <- x[sequence, ]
            rv <- as.matrix(x[, c('cartn_x', 'cartn_y', 'cartn_z')])
        } else {
            stop('DAMMIT!')
        }
        return( rv )
    })
})
residues <- do.call(c, residues)

saveRDS(residues, file = 'residues.rds')

# create a distance matrix
Hs <- helmert(length(sequence)-1, submatrix = TRUE)

N <- length(residues)
d <- matrix(0, nrow = N, ncol = N)
cbs <- combn(1:N, 2)
for (i in 1:ncol(cbs)) {
    Ah <- Hs %*% residues[[cbs[1,i]]][c('N', 'CA', 'C', 'CB', 'CG', 'OD1', 'OD2'), ]  # no oxegen
    Bh <- Hs %*% residues[[cbs[2,i]]][c('N', 'CA', 'C', 'CB', 'CG', 'OD1', 'OD2'), ]
    d[cbs[1,i],cbs[2,i]] <- min_f_norm(Ah, Bh)
}
d <- d + t(d)  # make symetric


kNNdistplot(d, k = 5)
abline(h = 2.9, lty = 2, col = "blue")

rv <- dbscan(d, eps = 2.9, minPts = 5)
```

```
# to a matrix  k x m x N (shapes package likes this)
residues.m <- sapply(residues, function(x) x, simplify='array', USE.NAMES=FALSE)

cols <- c('lightgray', brewer.pal(8, 'Set1'))

bb.rot <- procGPA(residues.m[c(1,2,3,5),,])$rotated
bb.reg <- transformations(bb.rot, residues.m[c(1,2,3,5),,])

sphere_size <- 0.05
plot3d(matrix(c(0,0,0),byrow=TRUE,ncol=3), col='white', aspect=TRUE, xlab='', ylab='', zlab='', axes=FALSE)
for (i in 1:dim(residues.m)[3]) {
    this.col <- cols[rv$cluster[i] + 1]
    r <- residues.m[,,i]
    rc <- t(apply(r, 1, function(y) y + bb.reg$translation[,i]))
    rc <- rc %*%  bb.reg$rotation[,,i]
    plot3d(rc[c(4,4),], type='s', col=this.col, radius=sphere_size, add=TRUE)  # seems to least at least two coords
    plot3d(rc[c(3,4),], type='l', col='grey', add=TRUE)
    plot3d(rc[c(1,3),], type='s', col=this.col, radius=sphere_size, add=TRUE)
    plot3d(rc[c(1,2,3),], type='l', col=this.col, add=TRUE)
    plot3d(rc[c(-1,-3,-4),], type='s', col=this.col, radius=sphere_size, add=TRUE)
    plot3d(rc[c(2,5,6,7),], type='l', col=this.col, add=TRUE)
    plot3d(rc[c(6,8),], type='l', col=this.col, add=TRUE)
}

res <- optics(d, eps = 15, minPts = 5)

plot(res, col = cols[rv$cluster[res$order] + 1], lwd = 2, bty = 'n', main = "", las = 2, xaxt = "n", cex.lab = 1.5, xlab = "")
abline(h = 2.9, lty = 2, col = "blue")
```

Appendix C

SCOP2

C.1 Creating a SCOP2 MySQL database

```
# The latest version of the SCOP2 data can be found here: http://scop2.mrc-lmb.cam.ac.uk/downloads/
# As of 2016-12-06 the latest release of SCOP2 was from 2014-02-05 data (scop2-rel20140205.sql)

# Assuming admin rights on a local MySQL instance the database is created as the root user like this
# replace 'secret'

create database scop2;
CREATE USER 'scop2rw'@'localhost' IDENTIFIED BY 'secret';
GRANT ALL ON scop2.* TO 'scop2rw'@'localhost';

# allow the user to write files if they are on the local machine
# this has to be applied for all databases, files will be written as the mysql user

GRANT FILE ON *.* TO 'scop2rw'@'localhost';
flush privileges;


# To create the tables and data (linux)

# wget http://scop2.mrc-lmb.cam.ac.uk/downloads/scop2-rel20140205.sql
# mysql -u scop2rw -p scop2 < ./scop2-rel20140205.sql

# If everything has worked the list of tables should look like this

# mysql> show tables;
#  +-----------------+
#  | Tables_in_scop2 |
#  +-----------------+
#  | closure         |
#  | domains         |
#  | keywords        |
#  | kwd2nodes       |
#  | relation        |
#  | relation_type   |
#  | structural_tags |
#  | tags2nodes      |
#  | term            |
#  +-----------------+
```

C.2 Extract mmCIF to CSV and create validation tables

```
#!/usr/bin/perl

use strict;
use warnings;
use Carp;

use Getopt::Long;
use File::Path qw/make_path/;
use File::Spec;
use POSIX;
use Fcntl 'O_RDONLY', 'O_RDWR', 'O_CREAT';
use File::Slurp;
use File::Copy;
use Data::Dumper;

use Term::ReadKey;
use DBI;

use STAR::Parser;
use File::Temp qw/:POSIX/;
use IO::Uncompress::Gunzip qw(gunzip $GunzipError);
use Text::CSV_XS qw/csv/;
use List::Util qw/first max min/;
use List::MoreUtils qw/indexes/;
use Time::Piece;
use Time::Piece::MySQL;


my $help = 0;
my $quiet = 0;
my $indir = '';
my $outdir = '';
GetOptions(
    'help' => \$help,
    'quiet' => \$quiet,
    "indir=s" => \$indir,
    "outdir=s" => \$outdir,
);

if ($help) {
    require Pod::Usage;
    Pod::Usage::pod2usage(-verbose => 2);
}
```

```
croak "You must supply a directory path to the PDBx/mmCIF file directory" unless($indir && -d $indir);
croak "You must supply a directory path for the output files" unless($outdir);
make_path($outdir) unless ( -d $outdir );
my $domaindir = File::Spec->catdir($outdir, 'domains');
make_path($domaindir) unless ( -d $domaindir );
my $asymunitdir = File::Spec->catdir($outdir, 'asymetric_units');
make_path($asymunitdir) unless ( -d $asymunitdir );

my $tmpdir = File::Spec->catdir($outdir, 'tmp');
make_path($tmpdir) unless ( -d $tmpdir );
my $tmpdomaindir = File::Spec->catdir($tmpdir, 'domains');
make_path($tmpdomaindir) unless ( -d $tmpdomaindir );
my $tmpasymunitdir = File::Spec->catdir($tmpdir, 'asymetric_units');
make_path($tmpasymunitdir) unless ( -d $tmpasymunitdir );

# SCOP2 database
print 'Enter scop2rw password: ';
ReadMode('noecho');
chomp(my $scop2_pass = ReadLine(0));
ReadMode(0);
print("\n");
my $dbh = DBI->connect('DBI:mysql:database=scop2new;host=localhost', 'scop2rw', $scop2_pass, {AutoCommit => 0, 'RaiseError' => 1});
my $sth_select_dom_names  = $dbh->prepare(qq/SELECT DISTINCT(dom_name) FROM domains WHERE pdb_code = ?/);
my $sth_select_serial  = $dbh->prepare(qq/SELECT serial, pdb_chain, pdb_begin, pdb_end FROM domains WHERE dom_name = ? ORDER BY serial/);

$dbh->do(qq/
    DROP TABLE IF EXISTS atom
/) or croak $DBI::errstr;
$dbh->do(qq/
    DROP TABLE IF EXISTS atom_keep
/) or croak $DBI::errstr;
$dbh->do(qq/
    DROP TABLE IF EXISTS domain_issue
/) or croak $DBI::errstr;
$dbh->do(qq/
    DROP TABLE IF EXISTS domain
/) or croak $DBI::errstr;
$dbh->do(qq/
    DROP TABLE IF EXISTS pdb_validation
/) or croak $DBI::errstr;
$dbh->do(qq/
    DROP TABLE IF EXISTS asym_unit_issue
/) or croak $DBI::errstr;
$dbh->do(qq/
    DROP TABLE IF EXISTS pdb_issue
/) or croak $DBI::errstr;
$dbh->do(qq/
```

```
    DROP TABLE IF EXISTS pdb
/) or croak $DBI::errstr;
$dbh->do(qq/
    DROP TABLE IF EXISTS asym_unit
/) or croak $DBI::errstr;

# create metadata tables
$dbh->do(qq/
    CREATE TABLE pdb (
        id VARCHAR(4) NOT NULL,
        exptl_method VARCHAR(20),
        resolution FLOAT,
        creation_date DATE,
        last_mod_date DATE,
        rev_num INTEGER,
        PRIMARY KEY(id),
        INDEX(exptl_method),
        INDEX(resolution),
        INDEX(creation_date),
        INDEX(last_mod_date)
    )
/) or croak $DBI::errstr;
my $sth_insert_pdb = $dbh->prepare(qq/INSERT INTO pdb(
    id,
    exptl_method,
    resolution,
    creation_date,
    last_mod_date,
    rev_num
) VALUES (?, ?, ?, ?, ?, ?)/);

my $sth_update_pdb = $dbh->prepare(qq/UPDATE pdb
    SET
        exptl_method = ?,
        resolution = ?,
        creation_date = ?,
        last_mod_date = ?,
        rev_num = ?
    WHERE
        id = ?/);

$dbh->do(qq/
    CREATE TABLE pdb_issue (
        pdb_id VARCHAR(4)   NOT NULL,
        level  VARCHAR(5)   NOT NULL,
        type   VARCHAR(30)  NOT NULL,
        text   VARCHAR(150) NOT NULL,
        INDEX(pdb_id),
```

```
        INDEX(type),
        INDEX(level),
        FOREIGN KEY (pdb_id) REFERENCES pdb(id)
    )
/) or croak $DBI::errstr;
my $sth_insert_pdb_issue = $dbh->prepare(qq/INSERT INTO pdb_issue(
    pdb_id,
    level,
    type,
    text
) VALUES (?, ?, ?, ?)/);

$dbh->do(qq/
    CREATE TABLE asym_unit (
        pdb_id VARCHAR(4)   NOT NULL,
        asym_id  VARCHAR(3)  NOT NULL,
        PRIMARY KEY(pdb_id, asym_id),
        FOREIGN KEY (pdb_id) REFERENCES pdb(id)
    )
/) or croak $DBI::errstr;
my $sth_insert_asym_unit = $dbh->prepare(qq/INSERT INTO asym_unit(
    pdb_id,
    asym_id
) VALUES (?, ?)/);

$dbh->do(qq/
    CREATE TABLE asym_unit_issue (
        pdb_id VARCHAR(4)   NOT NULL,
        asym_id  VARCHAR(3)  NOT NULL,
        level  VARCHAR(5)   NOT NULL,
        type   VARCHAR(30)  NOT NULL,
        text   VARCHAR(150) NOT NULL,
        INDEX(pdb_id),
        INDEX(asym_id),
        INDEX(type),
        INDEX(level),
        FOREIGN KEY (pdb_id) REFERENCES pdb(id)
    )
/) or croak $DBI::errstr;
my $sth_insert_asym_unit_issue = $dbh->prepare(qq/INSERT INTO asym_unit_issue(
    pdb_id,
    asym_id,
    level,
    type,
    text
) VALUES (?, ?, ?, ?, ?)/);

$dbh->do(qq/
```

```
    CREATE TABLE domain (
        dom_name VARCHAR(32) NOT NULL,
        pdb_id VARCHAR(4) NOT NULL,
        count_serial INTEGER NOT NULL,
        count_residues INTEGER,
        PRIMARY KEY(dom_name),
        FOREIGN KEY(pdb_id) REFERENCES pdb(id),
        INDEX(count_serial),
        INDEX(count_residues)
    )
/) or croak $DBI::errstr;
my $sth_insert_domain = $dbh->prepare(qq/INSERT INTO domain(
        dom_name,
        pdb_id,
        count_serial,
        count_residues
) VALUES (?, ?, ?, ?)/);
my $sth_update_domain = $dbh->prepare(qq/UPDATE domain
        SET count_residues = ?
        WHERE dom_name = ?
/);
$dbh->do(qq/
    CREATE TABLE domain_issue (
        domain_dom_name VARCHAR(32)   NOT NULL,
        level  VARCHAR(5)   NOT NULL,
        type   VARCHAR(30)  NOT NULL,
        text   VARCHAR(150) NOT NULL,
        INDEX(domain_dom_name),
        INDEX(type),
        INDEX(level),
        FOREIGN KEY (domain_dom_name) REFERENCES domain(dom_name)
    )
/) or croak $DBI::errstr;
my $sth_insert_domain_issue = $dbh->prepare(qq/INSERT INTO domain_issue(
    domain_dom_name,
    level,
    type,
    text
) VALUES (?, ?, ?, ?)/);

# temporary table for processing ATOM records
$dbh->do(qq/
    CREATE TABLE atom (
        id                  INT         NOT NULL,
        label_atom_id       VARCHAR(4)  NOT NULL,
        label_alt_id        CHAR(1),
        label_comp_id       CHAR(3)     NOT NULL,
        label_asym_id       VARCHAR(3)  NOT NULL,
```

```
        label_seq_id        INT         NOT NULL,
        cartn_x             FLOAT       NOT NULL,
        cartn_y             FLOAT       NOT NULL,
        cartn_z             FLOAT       NOT NULL,
        occupancy           FLOAT       NOT NULL,
        auth_asym_id        VARCHAR(3)  NOT NULL,
        auth_seq_id         INT         NOT NULL,
        pdbx_PDB_model_num  INT         NOT NULL,
        PRIMARY KEY(id),
        INDEX(auth_asym_id),
        INDEX(auth_seq_id),
        INDEX(occupancy),
        INDEX(label_alt_id),
        INDEX(pdbx_PDB_model_num)
    )
/) or croak $DBI::errstr;
$dbh->do(qq/
    CREATE TRIGGER test_label_atom_id
    BEFORE INSERT
    ON atom
    FOR EACH ROW
        BEGIN
            IF NEW.label_atom_id NOT REGEXP '^[[:alnum:]]+\$' THEN
                SET \@msg = CONCAT('Bad label_atom_id: >>', NEW.label_atom_id, '<<');
                SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = \@msg;
            END IF;
        END;
/) or croak $DBI::errstr;
my $sth_insert_atom = $dbh->prepare(qq/INSERT INTO atom (
    id,
    label_atom_id,
    label_alt_id,
    label_comp_id,
    label_asym_id,
    label_seq_id,
    cartn_x,
    cartn_y,
    cartn_z,
    occupancy,
    auth_asym_id,
    auth_seq_id,
    pdbx_PDB_model_num
) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)/) or croak $DBI::errstr;
# all three bind variables are the auth_asym_id
my $sth_select_asym_unit = $dbh->prepare(qq|
          /* rows without a label_alt_id */
          SELECT e.*
            FROM atom e
```

```
         WHERE pdbx_PDB_model_num = 1
           AND auth_asym_id = ?
           AND label_alt_id IS NULL
    UNION
            SELECT c.*
              FROM atom c
      INNER JOIN (
                   /* ensure a single label_alt_id for each auth_seq_id:
                      - this should be the label_alt_id that offers the maximum occupancy;
                        but this is only guaranteed if all rows of the label_alt_id for a particular auth_seq_id have the same
                            occupancy
                      - note that only a subset of rows for a auth_seq_id may have a (none '.') label_alt_id */
                   SELECT MIN(a.label_alt_id) AS label_alt_id,
                          b.auth_seq_id AS auth_seq_id,
                          b.max_occupancy
                     FROM atom a
                INNER JOIN (
                             SELECT auth_seq_id,
                                    MAX(occupancy) AS max_occupancy
                               FROM atom
                              WHERE pdbx_PDB_model_num = 1
                                AND auth_asym_id = ?
                                AND label_alt_id IS NOT NULL
                              GROUP BY auth_seq_id
                            ) b
                       ON (a.occupancy = b.max_occupancy)
                 GROUP BY b.auth_seq_id,
                          b.max_occupancy
              ) d
             ON (
                      c.pdbx_PDB_model_num = 1
                AND c.auth_asym_id = ?
                AND c.auth_seq_id = d.auth_seq_id
                AND c.label_alt_id = d.label_alt_id
              )
    ORDER BY auth_seq_id, id
|) or croak $DBI::errstr;
# the constraint acts as data validation
$dbh->do(qq/
    CREATE TABLE atom_keep (
        id                  INT         NOT NULL,
        serial              TINYINT(3)  NOT NULL,
        label_atom_id       VARCHAR(4)  NOT NULL,
        label_alt_id        CHAR(1),
        label_comp_id       CHAR(3)     NOT NULL,
        cartn_x             FLOAT       NOT NULL,
        cartn_y             FLOAT       NOT NULL,
        cartn_z             FLOAT       NOT NULL,
```

```
        occupancy           FLOAT       NOT NULL,
        auth_asym_id        VARCHAR(3)  NOT NULL,
        auth_seq_id         INT         NOT NULL,
        PRIMARY KEY(id),
        UNIQUE(label_atom_id, auth_seq_id, auth_asym_id)
    )
/) or croak $DBI::errstr;
my $sth_select_atom_keep = $dbh->prepare (qq|SELECT * FROM atom_keep ORDER BY serial, auth_seq_id, id|);
my $sth_select_atom_keep_seq_ids = $dbh->prepare (qq|SELECT auth_seq_id FROM atom_keep WHERE label_atom_id = 'CA' ORDER BY auth_seq_id|);

# PDB validation
$dbh->do(qq/
    CREATE TABLE pdb_validation (
        auth_seq_id         INT         NOT NULL,
        auth_asym_id        VARCHAR(3)  NOT NULL,
        PDB_model_num       INT         NOT NULL,
        failure             VARCHAR(20) NOT NULL,
        INDEX(auth_seq_id),
        INDEX(auth_asym_id),
        INDEX(PDB_model_num),
        INDEX(failure)
    )
/) or croak $DBI::errstr;
my $sth_insert_pdb_validation = $dbh->prepare(qq/INSERT INTO pdb_validation(
    auth_seq_id,
    auth_asym_id,
    PDB_model_num,
    failure
) VALUES (?, ?, ?, ?)/);
my $sth_select_pdb_validation = $dbh->prepare(qq/
    SELECT CONCAT(auth_seq_id, ' ', auth_asym_id, ' (', failure, ')') AS failure
    FROM pdb_validation
    WHERE PDB_model_num = 1
    GROUP BY auth_seq_id, auth_asym_id, failure
    ORDER BY auth_asym_id, auth_seq_id
/);
my $sth_select_domain_validation = $dbh->prepare(qq/
    SELECT CONCAT(a.auth_seq_id, ' (', a.failure, ')') AS failure
    FROM pdb_validation a, atom_keep b
    WHERE a.auth_seq_id = b.auth_seq_id
    AND a.auth_asym_id = b.auth_asym_id
    AND a.PDB_model_num = 1
    GROUP BY a.auth_seq_id, a.failure
    ORDER BY a.auth_seq_id
/);
my $sth_select_asym_unit_validation = $dbh->prepare(qq/
    SELECT CONCAT(auth_seq_id, ' (', failure, ')') AS failure
    FROM pdb_validation
```

```
    WHERE PDB_model_num = 1
    AND auth_asym_id = ?
    GROUP BY auth_seq_id, failure
    ORDER BY auth_seq_id
);

# object for writing outpus files
my $csv = Text::CSV_XS->new ({ binary => 1, eol => "\r\n" });

# cycle through proteins
my %asym_unit;
my $pdb_ids_ref = $dbh->selectcol_arrayref(qq/SELECT DISTINCT(pdb_code) FROM domains WHERE pdb_code IS NOT NULL/);
#my $pdb_ids_ref = $dbh->selectcol_arrayref(qq/SELECT DISTINCT(pdb_code) FROM domains WHERE pdb_code IN ('2J5K')/);
PDB: foreach my $pdb_id (@$pdb_ids_ref) {
    print "\n=======\n$pdb_id\n-------\n" unless $quiet;
    unlink_tmp_csv_files();
    my %meta_pdb = (
        id => $pdb_id,
        resolution => undef,
        exptl_method => undef,
        creation_date => undef,
        last_mod_date => undef,
        rev_num => undef,
    );
    insert_pdb_meta(\%meta_pdb);
    $dbh->commit or croak $DBI::errstr;

    my $data = parse_mmcif($pdb_id);
    if(! defined($data) ){
        insert_pdb_issue($pdb_id, 'ERROR', 'OBSOLETE', 'missing mmCIF file');
        $dbh->commit or croak $DBI::errstr;
        next PDB;
    }
    my @mod_date = $data->get_item_data(-item => '_database_PDB_rev.date');
    @mod_date = map {Time::Piece->strptime($_, '%Y-%m-%d') } @mod_date;
    my $max_mod_date = max( @mod_date );
    my @rev_num = $data->get_item_data(-item => '_database_PDB_rev.num');
    %meta_pdb = (
        id => $pdb_id,
        resolution => defined $data->get_item_data(-item => '_refine.ls_d_res_high') ? ($data->get_item_data(-item => '_refine.ls_d_res_
            high'))[0] : undef,
        exptl_method => ($data->get_item_data(-item => '_exptl.method'))[0],
        creation_date => min(@mod_date),
        last_mod_date => $max_mod_date,
        rev_num => max(@rev_num),
    );
    update_pdb_meta(\%meta_pdb);
    $dbh->commit or croak $DBI::errstr;
```

```
if ($max_mod_date > Time::Piece->strptime('2014-02-05', '%Y-%m-%d')) {
    insert_pdb_issue($pdb_id, 'WARN', 'MODIFIED_POST_SCOP2', 'mmCIF file last modified '. $max_mod_date->strftime('%Y-%m-%d') );
}
# get ATOM data into tmp table
$dbh->do(qq/DELETE FROM atom/) or croak $DBI::errstr;
# get_item_data returns a list rather than a reference so we don't want to keep calling it, hence
my %atoms = (
    group_PDB => [ $data->get_item_data(-item => '_atom_site.group_PDB') ],
    id => [ $data->get_item_data(-item => '_atom_site.id') ],
    type_symbol => [ $data->get_item_data(-item => '_atom_site.type_symbol') ],
    label_atom_id => [ $data->get_item_data(-item => '_atom_site.label_atom_id') ],
    label_alt_id => [ $data->get_item_data(-item => '_atom_site.label_alt_id') ],
    label_comp_id => [ $data->get_item_data(-item => '_atom_site.label_comp_id') ],
    label_asym_id => [ $data->get_item_data(-item => '_atom_site.label_asym_id') ],
    label_seq_id => [ $data->get_item_data(-item => '_atom_site.label_seq_id') ],
    cartn_x => [ $data->get_item_data(-item => '_atom_site.Cartn_x') ],
    cartn_y => [ $data->get_item_data(-item => '_atom_site.Cartn_y') ],
    cartn_z => [ $data->get_item_data(-item => '_atom_site.Cartn_z') ],
    occupancy => [ $data->get_item_data(-item => '_atom_site.occupancy') ],
    auth_asym_id => [ $data->get_item_data(-item => '_atom_site.auth_asym_id') ],
    auth_seq_id => [ $data->get_item_data(-item => '_atom_site.auth_seq_id') ],
    pdbx_PDB_model_num => [ $data->get_item_data(-item => '_atom_site.pdbx_PDB_model_num') ],
);
eval {
    ATOM: for (0 .. scalar( @{$atoms{'id'}} - 1 )) {
        next ATOM unless $atoms{'group_PDB'}->[$_] eq 'ATOM';
        next ATOM if $atoms{'type_symbol'}->[$_] eq 'H';
        $sth_insert_atom->execute(
            $atoms{'id'}->[$_],
            $atoms{'label_atom_id'}->[$_],
            $atoms{'label_alt_id'}->[$_] eq '.' ? undef : $atoms{'label_alt_id'}->[$_],
            $atoms{'label_comp_id'}->[$_],
            $atoms{'label_asym_id'}->[$_],
            $atoms{'label_seq_id'}->[$_],
            $atoms{'cartn_x'}->[$_],
            $atoms{'cartn_y'}->[$_],
            $atoms{'cartn_z'}->[$_],
            $atoms{'occupancy'}->[$_],
            $atoms{'auth_asym_id'}->[$_],
            $atoms{'auth_seq_id'}->[$_],
            $atoms{'pdbx_PDB_model_num'}->[$_]
        );
    }
};
if($@) {
    my ($text) = $@ =~ m/^(.+) at .*\/mmcif_to.+$/;
    $dbh->rollback or croak $DBI::errstr;  # do not want any warnings
```

```
    insert_pdb_issue($pdb_id, 'ERROR', 'UNEXPECTED_ON_ATOM_INSERT', $text );
    $dbh->commit or croak $DBI::errstr;
    next PDB;
}

# capture PDB validation
$dbh->do(qq/DELETE FROM pdb_validation/) or croak $DBI::errstr;
my %validation;
for my $type (('chiral', 'close_contact', 'main_chain_plane', 'peptide_omega', 'planes', 'planes_atom', 'polymer_linkage', 'rmsd_
    angle', 'rmsd_bond', 'symm_contact', 'torsion')){
    %validation = (
        auth_seq_id => [ $data->get_item_data(-item => "_pdbx_validate_${type}.auth_seq_id") ],
        auth_asym_id => [ $data->get_item_data(-item => "_pdbx_validate_${type}.auth_asym_id") ],
        auth_seq_id_1 => [ $data->get_item_data(-item => "_pdbx_validate_${type}.auth_seq_id_1") ],
        auth_asym_id_1 => [ $data->get_item_data(-item => "_pdbx_validate_${type}.auth_asym_id_1") ],
        auth_seq_id_2 => [ $data->get_item_data(-item => "_pdbx_validate_${type}.auth_seq_id_2") ],
        auth_asym_id_2 => [ $data->get_item_data(-item => "_pdbx_validate_${type}.auth_asym_id_2") ],
        auth_seq_id_3 => [ $data->get_item_data(-item => "_pdbx_validate_${type}.auth_seq_id_3") ],
        auth_asym_id_3 => [ $data->get_item_data(-item => "_pdbx_validate_${type}.auth_asym_id_3") ],
        PDB_model_num => [ $data->get_item_data(-item => "_pdbx_validate_${type}.PDB_model_num") ],
    );
    # can have single residue or 1, 2, or 3 atoms
    if(scalar( @{$validation{'auth_seq_id_3'}} )) {
        for (0 .. scalar( @{$validation{'auth_seq_id_3'}} - 1 )) {
            $sth_insert_pdb_validation->execute(
                $validation{'auth_seq_id_3'}->[$_],
                $validation{'auth_asym_id_3'}->[$_],
                $validation{'PDB_model_num'}->[$_],
                $type
            );
        };
    } elsif(scalar( @{$validation{'auth_seq_id_2'}} )) {
        for (0 .. scalar( @{$validation{'auth_seq_id_1'}} - 1 )) {
            $sth_insert_pdb_validation->execute(
                $validation{'auth_seq_id_1'}->[$_],
                $validation{'auth_asym_id_1'}->[$_],
                $validation{'PDB_model_num'}->[$_],
                $type
            );
            $sth_insert_pdb_validation->execute(
                $validation{'auth_seq_id_2'}->[$_],
                $validation{'auth_asym_id_2'}->[$_],
                $validation{'PDB_model_num'}->[$_],
                $type
            );
        };
    } else {
        for (0 .. scalar( @{$validation{'auth_seq_id'}} - 1 )) {
```

```
            $sth_insert_pdb_validation->execute(
                $validation{'auth_seq_id'}->[$_],
                $validation{'auth_asym_id'}->[$_],
                $validation{'PDB_model_num'}->[$_],
                $type
            );
        };
    }
}
my $pdb_validation = $dbh->selectcol_arrayref( $sth_select_pdb_validation );
if( scalar(@{$pdb_validation}) ) {
    my $text = join(', ', @{$pdb_validation});
    $text = substr($text, 0, 100) . ' <snip>' if (length($text) > 100);
    insert_pdb_issue($pdb_id, 'WARN', 'VALIDATION', $text );
}

# cycle through domains for the protein
$sth_select_dom_names->execute($pdb_id) or croak $sth_select_dom_names->errstr;
croak "domain SELECT got zero rows" if ($sth_select_dom_names->rows == 0);
DOMAIN: while (my $domain = $sth_select_dom_names->fetchrow_arrayref()) {  # DISTICT
    $dbh->do(qq|DELETE FROM atom_keep|) or croak $DBI::errstr;
    my $dom_name = $domain->[0];
    print "\n$dom_name\n" unless $quiet;
    $sth_select_serial->execute($dom_name)  or croak $sth_select_serial->errstr;
    croak "serial SELECT got zero rows" if ($sth_select_serial->rows == 0);
    my %meta_domain = (
        dom_name => $dom_name,
        pdb_id => $pdb_id,
        count_serial => $sth_select_serial->rows,
        count_residues => undef,
    );
    insert_domain_meta(\%meta_domain);
    my @expected_auth_seq_ids;
    # serial, pdb_chain, pdb_begin, pdb_end
    SERIAL: while (my $s = $sth_select_serial->fetchrow_arrayref()) {
        my ($serial, $pdb_chain, $pdb_begin, $pdb_end) = @{$s};
        push(@expected_auth_seq_ids, ($pdb_begin .. $pdb_end));
        unless ( $asym_unit{join('_', $pdb_id, $pdb_chain)}++ ) {
            write_asym_unit_csv($pdb_id, $pdb_chain);    ## TODO
            my $asym_unit_validation = $dbh->selectcol_arrayref( $sth_select_asym_unit_validation, {}, $pdb_chain );
            if( scalar(@{$asym_unit_validation}) ) {
                my $text = join(', ', @{$asym_unit_validation});
                $text = substr($text, 0, 100) . ' <snip>' if (length($text) > 100);
                insert_asym_unit_issue($pdb_id, $pdb_chain, 'WARN', 'VALIDATION', $text );
            }
        };
        eval {
            $dbh->do(qq|
```

```
INSERT INTO atom_keep
      /* rows without a label_alt_id */
      SELECT id,
             '$serial',
             label_atom_id,
             label_alt_id,
             label_comp_id,
             cartn_x,
             cartn_y,
             cartn_z,
             occupancy,
             auth_asym_id,
             auth_seq_id
        FROM atom
       WHERE pdbx_PDB_model_num = 1
         AND auth_asym_id = '$pdb_chain'
         AND label_alt_id IS NULL
         AND auth_seq_id >= $pdb_begin
         AND auth_seq_id <= $pdb_end
UNION
          SELECT c.id,
                 '$serial',
                 c.label_atom_id,
                 c.label_alt_id,
                 c.label_comp_id,
                 c.cartn_x,
                 c.cartn_y,
                 c.cartn_z,
                 c.occupancy,
                 c.auth_asym_id,
                 c.auth_seq_id
            FROM atom c
      INNER JOIN (
                      /* ensure a single label_alt_id for each auth_seq_id:
                         - this should be the label_alt_id that offers the maximum occupancy;
                           but this is only guaranteed if all rows of the label_alt_id for a particular auth_seq_id
                               have the same occupancy
                         - note that only a subset of rows for a auth_seq_id may have a (none '.') label_alt_id */
                      SELECT MIN(a.label_alt_id) AS label_alt_id,
                             b.auth_seq_id AS auth_seq_id,
                             b.max_occupancy
                        FROM atom a
                 INNER JOIN (
                               SELECT auth_seq_id,
                                      MAX(occupancy) AS max_occupancy
                                 FROM atom
                                WHERE pdbx_PDB_model_num = 1
                                  AND auth_asym_id = '$pdb_chain'
```

```
                                            AND label_alt_id IS NOT NULL
                                          GROUP BY auth_seq_id
                                        ) b
                                     ON (a.occupancy = b.max_occupancy)
                                  GROUP BY b.auth_seq_id,
                                           b.max_occupancy
                               ) d
                            ON (
                                    c.pdbx_PDB_model_num = 1
                                AND c.auth_asym_id = '$pdb_chain'
                                AND c.auth_seq_id = d.auth_seq_id
                                AND c.label_alt_id = d.label_alt_id
                               )
                         WHERE c.auth_seq_id >= $pdb_begin
                           AND c.auth_seq_id <= $pdb_end
        |) or die $DBI::errstr;
    };
    if ($@) {
        $dbh->rollback or die $DBI::errstr;  # this is considered a PDB error so get rid of all deeper references
        $sth_select_serial->finish;
        $sth_select_dom_names->finish;
        my ($text) = $@ =~ m/^(.+) at .*\/mmcif_to.+$/;
        insert_pdb_issue($pdb_id, 'ERROR', 'UNEXPECTED_ON_ATOM_KEEP_INSERT', 'DOMAIN ' . $dom_name . ' - ' . $text );
        $dbh->commit or die $DBI::errstr;
        next PDB;
    }
}
$sth_select_serial->finish;

my $got_auth_seq_ids = $dbh->selectcol_arrayref($sth_select_atom_keep_seq_ids) or die $DBI::errstr;
my $count_residues = scalar( @{$got_auth_seq_ids} );
#update domain meta with count
$sth_update_domain->execute($count_residues, $dom_name) or die $DBI::errstr;

write_domain_csv($dom_name);

my ($only_expected, $only_got) = compare_lists(\@expected_auth_seq_ids, $got_auth_seq_ids);
if(scalar(@{$only_expected}) or scalar(@{$only_got})) {
    if(scalar(@{$only_expected})) {
        my $text = join(', ', @{$only_expected});
        $text = substr($text, 0, 100) . ' <snip>' if (length($text) > 100);
        insert_domain_issue($dom_name, 'WARN', 'AUTH_SEQ_ID_MISSING', $text);
    }
    if(scalar(@{$only_got})) {
        my $text = join(', ', @{$only_got});
        $text = substr($text, 0, 100) . ' <snip>' if (length($text) > 100);
        insert_domain_issue($dom_name, 'WARN', 'AUTH_SEQ_ID_UNEXPECTED', $text);
    }
```

```
        }

        # validation reports
        my $domain_validation = $dbh->selectcol_arrayref( $sth_select_domain_validation );
        if( scalar(@{$domain_validation}) ) {
            my $text = join(', ', @{$domain_validation});
            $text = substr($text, 0, 100) . ' <snip>' if (length($text) > 100);
            insert_domain_issue($dom_name, 'WARN', 'VALIDATION', $text );
        }

    }
    $sth_select_dom_names->finish;
    $dbh->commit or die $DBI::errstr;
    mv_tmp_csv_files();
}

$dbh->do(qq/
    DROP TABLE IF EXISTS atom
/) or croak $DBI::errstr;
$dbh->do(qq/
    DROP TABLE IF EXISTS atom_keep
/) or croak $DBI::errstr;
$dbh->do(qq/
    DROP TABLE IF EXISTS pdb_validation
/) or croak $DBI::errstr;

$dbh->disconnect();


# -----------
# subroutines
# -----------


sub parse_mmcif {
    my $id = shift;
    $id = lc($id);

    my $cifpath = File::Spec->catfile($indir, substr($id, 1, 2), $id . '.cif.gz');
    return(undef) unless(-f $cifpath);

    # STAR::Parser does not accept file handles so we need to write the physical unzipped file
    my $tmp_file = tmpnam();
    gunzip $cifpath => $tmp_file or croak "gunzip failed: $GunzipError\n";
    my @data = STAR::Parser->parse(
        -file => $tmp_file,
    );
    unlink $tmp_file or croak "Could not unlink $tmp_file: $!";
```

```
    die 'STAR::Parser found multiple DataBlocks' if @data != 1;

    return $data[0];
}

sub insert_pdb_meta {
    my $meta = shift;
    $sth_insert_pdb->execute(
        $meta->{'id'},
        $meta->{'exptl_method'},
        $meta->{'resolution'},
        defined $meta->{'creation_date'} ? $meta->{'creation_date'}->mysql_date : undef,
        defined $meta->{'last_mod_date'} ? $meta->{'last_mod_date'}->mysql_date : undef,
        $meta->{'rev_num'}
    ) or croak $DBI::errstr;
}

sub update_pdb_meta {
    my $meta = shift;
    $sth_update_pdb->execute(
        $meta->{'exptl_method'},
        $meta->{'resolution'},
        defined $meta->{'creation_date'} ? $meta->{'creation_date'}->mysql_date : undef,
        defined $meta->{'last_mod_date'} ? $meta->{'last_mod_date'}->mysql_date : undef,
        $meta->{'rev_num'},
        $meta->{'id'}
    ) or croak $DBI::errstr;
}

sub insert_pdb_issue {
    my ($pdb_id, $level, $type, $text) = @_;
    $sth_insert_pdb_issue->execute(@_) or croak $DBI::errstr;
    print $level . ": " . $type . " - " . $text . "\n" unless $quiet;
}

sub insert_domain_meta {
    my $meta = shift;
    $sth_insert_domain->execute(
        $meta->{'dom_name'},
        $meta->{'pdb_id'},
        $meta->{'count_serial'},
        $meta->{'count_residues'}
    ) or croak $DBI::errstr;
}

sub insert_domain_issue {
    my ($dom_name, $level, $type, $text) = @_;
    $sth_insert_domain_issue->execute(@_) or croak $DBI::errstr;
```

```
    print $level . ": " . $type . " - " . $text . "\n" unless $quiet;
}

sub write_domain_csv {
    my $dom_name = shift;
    my $outpath = File::Spec->catfile($tmpdomaindir, $dom_name . '.csv');
    print "Writing $outpath\n" unless $quiet;
    open my $fh, '>', $outpath or die "Cannot open '$outpath' for writing: $!";
    $sth_select_atom_keep->execute;
    $csv->print ($fh, $sth_select_atom_keep->{NAME_lc});
    while (my $row = $sth_select_atom_keep->fetch) {
        $csv->print ($fh, $row);
    }
    $sth_select_atom_keep->finish;
    $fh->close();
}

sub insert_asym_unit_issue {
    my ($pdb_id, $pdb_chain, $level, $type, $text) = @_;
    $sth_insert_asym_unit_issue->execute(@_) or croak $DBI::errstr;
    print $level . ": " . $type . " - " . $text . "\n" unless $quiet;
}

sub write_asym_unit_csv {
    my ($pdb_id, $pdb_chain) = @_;
    my $outpath = File::Spec->catfile($tmpasymunitdir, $pdb_id . '_' . $pdb_chain  . '.csv');
    print "Writing $outpath\n" unless $quiet;
    open my $fh, '>', $outpath or die "Cannot open '$outpath' for writing: $!";
    $sth_select_asym_unit->execute( (($pdb_chain) x 3) );
    $csv->print ($fh, $sth_select_asym_unit->{NAME_lc});
    while (my $row = $sth_select_asym_unit->fetch) {
        $csv->print ($fh, $row);
    }
    $sth_select_asym_unit->finish;
    $fh->close();
}

sub compare_lists {
    my ($a, $b) = @_;
    my %seen;
    my @aonly;
    @seen{@{$b}} = ( );
    foreach my $item (@{$a}) {
        push(@aonly, $item) unless exists $seen{$item};
    }
    my @bonly;
    @seen{@{$a}} = ( );
    foreach my $item (@{$b}) {
```

```
        push(@bonly, $item) unless exists $seen{$item};
    }
    return(\@aonly, \@bonly);
}

sub mv_tmp_csv_files {
    my @files = glob File::Spec->catfile($tmpdomaindir, '*.csv');
    foreach my $file (@files) {
        print "Moving $file to $domaindir\n" unless $quiet;
        move($file, $domaindir) or croak "Could not move $file to $domaindir: $!\n";
    }
    @files = glob File::Spec->catfile($tmpasymunitdir, '*.csv');
    foreach my $file (@files) {
        print "Moving $file to $asymunitdir\n" unless $quiet;
        move($file, $asymunitdir) or croak "Could not move $file to $asymunitdir: $!\n";
    }
}

sub unlink_tmp_csv_files {
    my @files = glob File::Spec->catfile($tmpdomaindir, '*.csv');
    if(scalar(@files)) {
        print "Unlinking " . join(', ', @files) . "\n" unless $quiet;
        unlink @files or croak "Failed to unlink all files in $tmpdomaindir: $!\n";
    }
    @files = glob File::Spec->catfile($tmpasymunitdir, '*.csv');
    if(scalar(@files)) {
        print "Unlinking " . join(', ', @files) . "\n" unless $quiet;
        unlink @files or croak "Failed to unlink all files in $tmpasymunitdir: $!\n";
    }
}


=pod

=head1 NAME

mmcif_to_domain_landmarks.pl - extract domain ATOM data from PDBx/mmCIF files into csv files

=head1 SYNOPSIS

    mmcif_to_domain_landmarks.pl -i <pdb_file_dir> -o <csv_out_dir>

    mmcif_to_domain_landmarks.pl -h

    mmcif_to_domain_landmarks.pl -i ./mmCIF/ -o ./csv/

=head1 Commandline Options
```

```
=over 4

=item -h

Display this POD.

=item -q

Do not print any progress to STDOUT.

=item -i dir_path, --indir=dir_path

Path to the directory containing the PDBx/mmCIF files.

=item -o dir_path, --outdb=dir_path

Path to the directory where the output CSV files are to be written.

=back

=head1 DESCRIPTION

Extract ATOM coordinates data from Protein Data Bank PDBx/mmCIF files for all domains in the SCOP2 database.

The script was written to run once so there are no formal unit tests.
The script is supplied only as a record of the data extraction process.
Database integrity constraints are used where possible to validate the output.
Testing of the results was done manually after the run.

The code assumes that there exists a local instance of the L<scop2|http://scop2.mrc-lmb.cam.ac.uk/download.html> database with a C<
    scop2rw> user that can C<CREATE> and C<DROP> tables.
When run the script will prompt for the database password.

=head2 CSV files

The script creates and populates two sub directories within C<csv_out_dir> these are named C<asymetric_units> and C<domains>.
The C<domains> directory contain CSV files containing the atomic co-ordinates for each of the domains in the SCOP2 database, with the
    exception of domains referencing PDB files that have been marked as C<OBSOLETE> in the PDB.
The file name is the domain reference as given in the SCOP2 database (C<domains.dom_name>).

    $ ls -l domains
    -rw-r--r-- 1 tonyh tonyh  57876 Jan  7 09:10 CF-8000575-1SAYA.csv
    -rw-r--r-- 1 tonyh tonyh  28608 Jan  7 09:11 CF-8000764-2PWZA.csv
    -rw-r--r-- 1 tonyh tonyh  42865 Jan  7 09:09 CF-8000817-2JHFA.csv
    -rw-r--r-- 1 tonyh tonyh  27604 Jan  7 09:09 CF-8000825-1QORA.csv

    $ head domains/CF-8000575-1SAYA.csv
    id,serial,label_atom_id,label_alt_id,label_comp_id,cartn_x,cartn_y,cartn_z,occupancy,auth_asym_id,auth_seq_id
```

```
1,1,N,,MET,-12.817,34.708,32.066,1,A,1
2,1,CA,,MET,-11.663,35.505,32.484,1,A,1
3,1,C,,MET,-11.155,36.336,31.331,1,A,1
4,1,O,,MET,-11.533,36.107,30.171,1,A,1
5,1,CB,,MET,-10.552,34.586,32.994,1,A,1
```

The C<asymetric_units> directory contains CSV files containing the complete set of atomic co-ordinates for every asymetric unit referenced by the SCOP2 domains.

```
$ ls -l asymetric_units
total 94744
-rw-r--r-- 1 tonyh tonyh 110105 Jan  7 09:52 1A4I_B.csv
-rw-r--r-- 1 tonyh tonyh  44098 Jan  7 09:31 1A5J_A.csv
-rw-r--r-- 1 tonyh tonyh 120022 Jan  7 09:51 1A5Z_A.csv
-rw-r--r-- 1 tonyh tonyh  88068 Jan  7 09:27 1A8L_A.csv

$ head asymetric_units/1A4I_B.csv
id,label_atom_id,label_alt_id,label_comp_id,label_asym_id,label_seq_id,cartn_x,cartn_y,cartn_z,occupancy,auth_asym_id,auth_seq_id,
    pdbx_pdb_model_num
2160,N,,ALA,B,2,-2.875,17.033,57.446,1,B,2,1
2161,CA,,ALA,B,2,-3.679,17.481,56.266,1,B,2,1
2162,C,,ALA,B,2,-2.742,17.664,55.077,1,B,2,1
2163,O,,ALA,B,2,-2.582,18.769,54.567,1,B,2,1
2164,CB,,ALA,B,2,-4.435,18.761,56.575,1,B,2,1
```

These CSV data files can be read into C<R> using code something like

```
atom_data <- read.delim('domains/CF-8000575-1SAYA.csv', header=TRUE, sep=',', stringsAsFactors = FALSE, row.names = 1)
```

which gives a C<data.frame> of the form

```
> head(atom_data, 12)
   serial label_atom_id label_alt_id label_comp_id cartn_x cartn_y cartn_z occupancy auth_asym_id auth_seq_id
1       1             N           NA           MET -12.817  34.708  32.066         1            A           1
2       1            CA           NA           MET -11.663  35.505  32.484         1            A           1
3       1             C           NA           MET -11.155  36.336  31.331         1            A           1
4       1             O           NA           MET -11.533  36.107  30.171         1            A           1
5       1            CB           NA           MET -10.552  34.586  32.994         1            A           1
6       1            CG           NA           MET  -9.186  34.949  32.513         1            A           1
7       1            SD           NA           MET  -7.936  34.291  33.604         1            A           1
8       1            CE           NA           MET  -8.908  33.216  34.674         1            A           1
9       1             N           NA           GLU -10.311  37.308  31.630         1            A           2
10      1            CA           NA           GLU  -9.728  38.123  30.555         1            A           2
11      1             C           NA           GLU  -8.412  37.497  30.198         1            A           2
12      1             O           NA           GLU  -7.578  37.245  31.069         1            A           2
```

and hence the co-ordinates of just the backbone atoms could be obtained something like this

```
    > backbone_atoms <- atom_data[atom_data$label_atom_id %in% c('CA', 'C', 'N', 'O'), c('cartn_x', 'cartn_y', 'cartn_z')]

    > head(backbone_atoms)
       cartn_x cartn_y cartn_z
    1   -12.817  34.708  32.066
    2   -11.663  35.505  32.484
    3   -11.155  36.336  31.331
    4   -11.533  36.107  30.171
    9   -10.311  37.308  31.630
    10   -9.728  38.123  30.555
```

and a 3D plot to check the data looks OK

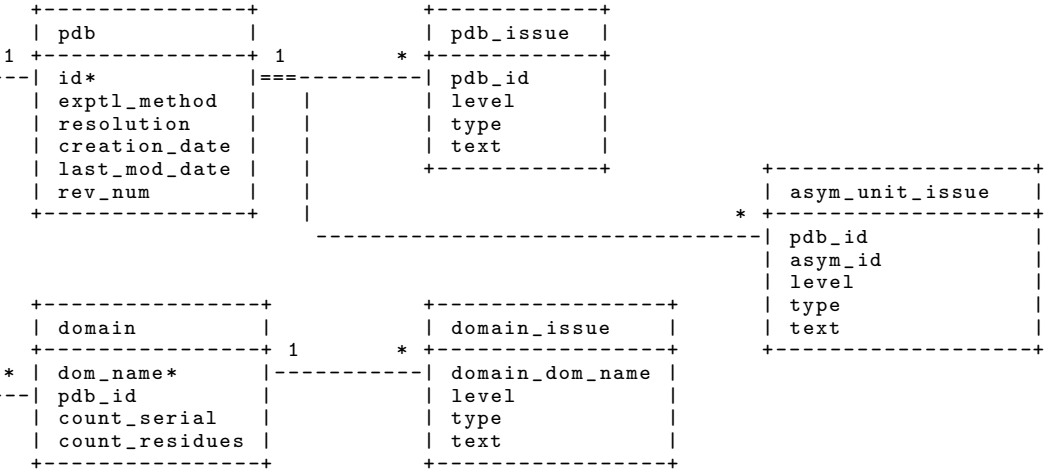
```
    library(rgl)
    plot3d(backbone_atoms, type = 's', col = 'darkgreen', radius = 0.4)
```

or plot all atoms, coloring sidechain atoms differently

```
    library(rgl)
    cols <- ifelse(atom_data$label_atom_id %in% c('CA', 'C', 'N', 'O'), 'darkgreen', 'gray')
    plot3d(atom_data[ , c('cartn_x', 'cartn_y', 'cartn_z')], type = 's', col = cols, radius = 0.4)
```

=head2 Database tables

The script also creates the following database tables in the C<scop2> schema



The C<pdb> and C<domain> tables contain metadata about the full protein entries and specific domains; derived from the PSB mmCIF files.

The quality of the data in the PDB is very variable.
The C<pdb_issue>, C<domain_issue> and C<asym_unit_issue> tables contain details of issues relating to the set of C<ATOM> records for each of the structures.
The issues are rated with one of two possible levels C<ERROR> or C<WARN>.

C<ERROR> indicates that the script was unable to process the the C<ATOM> records for the structure.
For records with an C<ERROR> the associated CSV files are NOT created.

pdb_id	level	type	text
2FB1	ERROR	OBSOLETE	missing mmCIF file
1D5Y	ERROR	UNEXPECTED_ON_ATOM_INSERT	DBD::mysql::st execute failed: Bad label_atom_id: >>O5'<<
2EC9	ERROR	UNEXPECTED_ON_ATOM_KEEP_INSERT	DOMAIN CF-8003777-2EC9H - DBD::mysql::db do failed: Duplicate entry 'N-60-H' for key 'label_atom_id'

C<WARN> indicates either an error in the data where the data is still understandable, or when there are data validation issues as indicated by the PDB validation reports.

pdb_id	level	type	text
3OL4	WARN	MODIFIED_POST_SCOP2	mmCIF file last modified 2015-04-22
3DBH	WARN	VALIDATION	36 A (torsion), 38 A (torsion), 39 A (torsion), 63 A (torsion), 65 A (torsion), 66 A (torsion), 79 A <snip>

domain_dom_name	level	type	text

```
    | CF-8002341-1MIJA | WARN  | AUTH_SEQ_ID_MISSING | 1314, 1315, 1316, 1317, 1318, 1319, 1320, 1321, 1322, 1323, 1324, 1325, 1326
                                    |
    | SP-8002542-1JT6B | WARN  | VALIDATION          | 4 (torsion), 18 (torsion), 21 (close_contact), 33 (torsion), 41 (torsion), 44 (
        torsion), 89 (torsion <snip> |
    +------------------+-------+---------------------+------------------------------------------------------------------------------------------------------------


    +--------+---------+-------+------------+-----------------------------------------------------------------------------------------------------------------------

    | pdb_id | asym_id | level | type       | text
                                                                                                                  |
    +--------+---------+-------+------------+-----------------------------------------------------------------------------------------------------------------------

    | 1OFC   | X       | WARN  | VALIDATION | 734 (rmsd_angle), 751 (peptide_omega), 752 (peptide_omega), 756 (torsion), 794 (chiral),
        795 (torsio <snip> |
    +--------+---------+-------+------------+-----------------------------------------------------------------------------------------------------------------------

=head1 See Also

C<fetch_pdb_files.pl>

=cut
```

C.3 Fetch PDB files

```
#!/usr/bin/perl

use strict;
use warnings;

use Getopt::Long;
use File::Slurp;
use File::Path qw/make_path/;
use File::Spec;
use File::Temp qw(tempfile);
use POSIX;

use constant ERROR_FILE => 'FETCH_ERRORS_' . strftime("%Y%m%d%H%M%S",localtime) . '.txt';

my $help = 0;
my $outdir;
my $infile = '';
my $pdbids = '';
GetOptions(
    'help' => \$help,
    "outdir=s" => \$outdir,
    "infile=s" => \$infile,
    "pdbids=s" => \$pdbids,
);

if ($help) {
    require Pod::Usage;
    Pod::Usage::pod2usage(-verbose => 2);
}

die "You must provide either a list of PDB IDs on the command line or in a file (not both)" unless ( ($infile || $pdbids) && ! ($infile &
    & $pdbids) );
die "You must supply a directory path for the retrieved data files" unless($outdir);
make_path($outdir) unless ( -d $outdir );

my @pdbids;
if ($pdbids) {
    @pdbids = split(/,/,$pdbids);
}
else {
    @pdbids = read_file($infile,chomp=>1) ;
}
die "List of PDB IDs looks wrong" if ( grep { ! m/^[A-Za-z0-9]{4}$/ } @pdbids );
```

```
my @sources = map { File::Spec->catfile( lc(substr($_,1,2)), lc($_).'.cif.gz' ) } @pdbids;
my ($tmp_fh, $tmp_path) = tempfile();
foreach (@sources) { print $tmp_fh $_ ."\n" };
close $tmp_fh;

my $status = system("rsync -rlpt -v -z --delete --files-from=$tmp_path rsync.ebi.ac.uk::pub/databases/pdb/data/structures/divided/mmCIF/
    $outdir");

warn "rsync had errors: $?" unless $status == 0;

unlink $tmp_path or warn "Could not unlink $tmp_path: $!";


=pod

=head1 NAME

fetch_pdb_files.pl - fetch PDBx/mmCIF files

=head1 SYNOPSIS

    fetch_pdb_files.pl (-i <pdb_id_file> | -p <pdb_ids>) -o <fetched_file_dir> [-q]

    fetch_pdb_files.pl -h

    fetch_pdb_files.pl -i ./pdb_ids.txt -o ./mmCIF/
    fetch_pdb_files.pl -p 1a7w,1aab,1adz -o ./mmCIF/

=head1 DESCRIPTION

Fetches Protein Data Bank files from wwpdb.org and puts them in a specified directory.

This script uses C<rsync>, as described at L<http://www.wwpdb.org/downloads.html>, so if a file is already present in the fetched_files_
    dir and is up-to-date it will not be re-fetched.

The fetched files are gzipped PDBx/mmCIF files and are created in a one level deep directory structure within fetched_file_dir. The
    directory structure is as per ftp server at wwpdb.org and looks something like this:

   mmCIF/j2/2j2u.cif.gz
           /2j2z.cif.gz
   ...
   mmCIF/jm/1jm0.cif.gz
           /1jm1.cif.gz
           /1jm4.cif.gz
           /1jm6.cif.gz
   ...
```

```
Note if you want to fetch a single file it is probably easier to do something like:

    wget 'http://www.rcsb.org/pdb/files/1a7w.cif'

=head2 Commandline Options

You can only specify one of B<-i> and B<-p>.

=over 4

=item -h

Display this POD.

=item -i filepath, --infile=filepath

A text file containing the four character PDB IDs of the PDB files to be fetched; one PDB ID per line.

Example file content:

    1a7w
    1aab
    1adz

=item -p comma_seperated_PDB_IDs, --pdbids=comma_seperated_PDB_IDs

Specify multiple PDB IDs on the command line as a comma seperated list.

=item -o directory_path, --outdir=directory_path

Directory to store the retrieved PDB files.

If the folder does not exist the script will try to create it.

=back

=cut
```

C.4 SQL queries

C.4.1 Families with at least 10 members

Domains from families with at least 10 members that are associated to asymetric units of no greater than 250 residues. The results are given in Table 4.3.

```
SELECT
    ds.dom_name,
    CONCAT_WS('_', ds.pdb_code, ds.pdb_chain) AS asym_unit,
    ds.node,
    d.count_residues AS dom_res_count,
    au.count_residues AS asym_unit_res_count,
    IF(di.domain_dom_name IS NULL, FALSE, TRUE) as dom_val_warn,
    IF(aui.pdb_id IS NULL, FALSE, TRUE) as asym_unit_val_warn
FROM
    domains ds LEFT JOIN domain_issue di ON
        (ds.dom_name = di.domain_dom_name AND di.type = 'VALIDATION')
               LEFT JOIN asym_unit_issue aui ON
        (ds.pdb_code = aui.pdb_id AND ds.pdb_chain = aui.asym_id AND aui.type = 'VALIDATION'),
    asym_unit au,
    domain d
WHERE
    au.pdb_id = ds.pdb_code AND au.asym_id = ds.pdb_chain
    AND au.count_residues <= 250
    AND ds.serial = 1
    AND d.dom_name = ds.dom_name
    AND ds.node IN (SELECT
            node
        FROM
            domains
        WHERE
            LEFT(node, 2) = 'FA'
        GROUP BY node
        HAVING COUNT(*) >= 10)
ORDER BY ds.node
```

Bibliography

Chapter 1

Andreeva, A. et al. (Jan. 1, 2014). "SCOP2 prototype: a new approach to protein structure mining". In: *Nucleic Acids Research* 42 (D1), pp. D310–D314. issn: 0305-1048, 1362-4962. doi: 10.1093/nar/gkt1242.

Bookstein, F. L. (June 28, 1997). *Morphometric Tools for Landmark Data: Geometry and Biology*. Cambridge University Press. 459 pp. isbn: 978-0-521-58598-9.

Chambers, J. (Nov. 23, 2010). *Software for Data Analysis: Programming with R*. Softcover reprint of hardcover 1st ed. 2008 edition. New York, NY: Springer. 516 pp. isbn: 978-1-4419-2612-8.

Christiansen, T. et al. (Mar. 9, 2012). *Programming Perl: Unmatched power for text processing and scripting*. 4 edition. Beijing ; Sebastopol: O'Reilly Media. 1184 pp. isbn: 978-0-596-00492-7.

Csaba, G., F. Birzele, and R. Zimmer (Apr. 17, 2009). "Systematic comparison of SCOP and CATH: a new gold standard for protein structure analysis". In: *BMC Structural Biology* 9, p. 23. issn: 1472-6807. doi: 10.1186/1472-6807-9-23.

Dill, K. and S. Bromberg (Dec. 13, 2010). *Molecular Driving Forces: Statistical Thermodynamics in Biology, Chemistry, Physics, and Nanoscience*. 2nd edition. London ; New York: Garland Science. 778 pp. isbn: 978-0-8153-4430-8.

Dryden, I. L. and K. V. Mardia (1998). *Statistical shape analysis*. Chichester; New York: John Wiley & Sons. isbn: 0-585-26570-4.

– (July 8, 2016). *Statistical Shape Analysis: With Applications in R*. 2nd Ed. Wiley. 496 pp.

Fox, N. K., S. E. Brenner, and J. Chandonia (Nov. 1, 2015). "The value of protein structure classification information—Surveying the scientific literature". In: *Proteins: Structure, Function, and Bioinformatics* 83.11, pp. 2025–2038. issn: 1097-0134. doi: 10.1002/prot.24915.

Golub, G. H. (2013). *Matrix computations*. In collab. with C. F. Van Loan. 4th ed. Johns Hopkins studies in the mathematical sciences. Baltimore: The Johns Hopkins University Press. 756 pp. isbn: 978-1-4214-0794-4.

Green, P. J. and K. V. Mardia (June 1, 2006). "Bayesian alignment using hierarchical models, with applications in protein bioinformatics". In: *Biometrika* 93.2, pp. 235–254. issn: 0006-3444, 1464-3510. doi: 10.1093/biomet/93.2.235.

Hamelryck, T., K. V. Mardia, and J. Ferkinghoff-Borg, eds. (Mar. 24, 2012). *Bayesian Methods in Structural Bioinformatics*. Springer. 385 pp. isbn: 978-3-642-27224-0.

Harder, T. et al. (Feb. 15, 2012). "Fast large-scale clustering of protein structures using Gauss integrals". In: *Bioinformatics* 28.4, pp. 510–515. issn: 1367-4803. doi: 10.1093/bioinformatics/btr692.

Hasegawa, H. and L. Holm (June 1, 2009). "Advances and pitfalls of protein structural alignment". In: *Current Opinion in Structural Biology* 19.3, pp. 341–348. issn: 0959-440X. doi: 10.1016/j.sbi.2009.04.003.

Holland, T. A. et al. (Aug. 18, 2006). "Partitioning Protein Structures into Domains: Why Is it so Difficult?" In: *Journal of Molecular Biology* 361.3, pp. 562–590. issn: 0022-2836. doi: 10.1016/j.jmb.2006.05.060.

Horn, R. A. and C. R. Johnson (2012). *Matrix Analysis*. Cambridge University Press. 663 pp. isbn: 978-1-139-78888-5.

Hubbard, T. J. P. et al. (Jan. 1, 1999). "SCOP: a Structural Classification of Proteins database". In: *Nucleic Acids Research* 27.1, pp. 254–256. issn: 0305-1048, 1362-4962. doi: 10.1093/nar/27.1.254.

Kendall, D. G. (Mar. 1, 1984). "Shape Manifolds, Procrustean Metrics, and Complex Projective Spaces". In: *Bulletin of the London Mathematical Society* 16.2, pp. 81–121.

Kendall, D. G. et al. (1999). *Shape and Shape Theory*. John Wiley & Sons. isbn: 0-471-96823-4.

Al-Lazikani, B., E. E. Hill, and V. Morea (Jan. 1, 2008). "Protein Structure Prediction". In: *Bioinformatics*. Ed. by J. M. Keith. Methods in Molecular Biology 453. Humana Press, pp. 33–85.

Lesk, A. M. (Nov. 2013). *Introduction to Bioinformatics*. Oxford University Press. 395 pp. isbn: 978-0-19-965156-6.

Lupyan, D., A. Leo-Macias, and A. R. Ortiz (Aug. 1, 2005). "A new progressive-iterative algorithm for multiple structure alignment". In: *Bioinformatics* 21.15, pp. 3255–3263. issn: 1367-4803. doi: 10.1093/bioinformatics/bti527.

Papachristodoulou, D. K. et al. (2014). *Biochemistry and molecular biology*. 5th ed. Oxford: Oxford University Press. 591 pp. isbn: 978-0-19-960949-9.

Pavlopoulou, A. and I. Michalopoulos (May 23, 2011). "State-of-the-art bioinformatics protein structure prediction tools (Review)". In: *International Journal of Molecular Medicine*. issn: 1107-3756, 1791-244X. doi: 10.3892/ijmm.2011.705.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing.

Røgen, P. (2005). "Evaluating protein structure descriptors and tuning Gauss integral based descriptors". In: *Journal of Physics: Condensed Matter* 17.18, S1523. issn: 0953-8984. doi: 10.1088/0953-8984/17/18/010.

Schönemann, P. H. (Mar. 1966). "A generalized solution of the orthogonal procrustes problem". In: *Psychometrika* 31.1, pp. 1–10. issn: 0033-3123, 1860-0980. doi: 10.1007/BF02289451.

Shindyalov, I. N. and P. E. Bourne (Sept. 1, 1998). "Protein structure alignment by incremental combinatorial extension (CE) of the optimal path." In: *Protein Engineering, Design and Selection* 11.9, pp. 739–747. issn: 1741-0126. doi: 10.1093/protein/11.9.739.

Sillitoe, I. et al. (Jan. 28, 2015). "CATH: comprehensive structural and functional annotations for genome sequences". In: *Nucleic Acids Research* 43 (D1), pp. D376–D381. issn: 0305-1048. doi: 10.1093/nar/gku947.

Sippl, M. J. (Mar. 15, 2008). "On distance and similarity in fold space". In: *Bioinformatics* 24.6, pp. 872–873. issn: 1367-4803. doi: 10.1093/bioinformatics/btn040.

Small, C. (1996). *The statistical theory of shape*. Springer. isbn: 0-387-94729-9.

Theobald, D. L.. and D. S. Wuttke (Dec. 5, 2006). "Empirical Bayes hierarchical models for regularizing maximum likelihood estimation in the matrix Gaussian Procrustes problem". In: *Proceedings of the National Academy of Sciences* 103.49, pp. 18521–18527. issn: 0027-8424, 1091-6490. doi: 10.1073/pnas.0508445103.

Tramontano, A. (May 24, 2005). *The Ten Most Wanted Solutions in Protein Bioinformatics*. 1 edition. Boca Raton, FL: Chapman and Hall/CRC. 216 pp. isbn: 978-1-58488-491-0.

Ye, Y. and A. Godzik (Sept. 27, 2003). "Flexible structure alignment by chaining aligned fragment pairs allowing twists". In: *Bioinformatics* 19 (suppl_2), pp. ii246–ii255. issn: 1367-4803. doi: 10.1093/bioinformatics/btg1086.

Zhang, Y. and J. Skolnick (Jan. 1, 2005). "TM-align: a protein structure alignment algorithm based on the TM-score". In: *Nucleic Acids Research* 33.7, pp. 2302–2309. issn: 0305-1048. doi: 10.1093/nar/gki524.

Chapter 2

Best, D. J. and N. I. Fisher (1979). "Efficient Simulation of the von Mises Distribution". In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28.2, pp. 152–157. issn: 0035-9254. doi: 10.2307/2346732.

Chui, H. and A. Rangarajan (2000). "A feature registration framework using mixture models". In: *Proceedings IEEE Workshop on Mathematical Methods in Biomedical Image Analysis. MMBIA-2000 (Cat. No.PR00737)*. Proceedings IEEE Workshop on Mathematical Methods in Biomedical Image Analysis. MMBIA-2000 (Cat. No.PR00737), pp. 190–197. doi: 10.1109/MMBIA.2000.852377.

Dimitrov, D. et al. (Oct. 1, 2009). "Bounds on the quality of the PCA bounding boxes". In: *Computational Geometry*. Special Issue on the 23rd European Workshop on Computational Geometry 42.8, pp. 772–789. issn: 0925-7721. doi: 10.1016/j.comgeo.2008.02.007.

Dryden, I. L., J. D. Hirst, and J. L. Melville (Mar. 1, 2007). "Statistical Analysis of Unlabeled Point Sets: Comparing Molecules in Chemoinformatics". In: *Biometrics* 63.1, pp. 237–251. issn: 1541-0420. doi: 10.1111/j.1541-0420.2006.00622.x.

Gärtner, B. and S. Schönherr (1997). "Smallest Enclosing Ellipses – Fast and Exact". In: *Informatik*. Series B.

Gelman, A. et al. (Nov. 1, 2013). *Bayesian Data Analysis, Third Edition*. CRC Press. 677 pp. isbn: 978-1-4398-4095-5.

Green, P. J. (Dec. 1, 1995). "Reversible jump Markov chain Monte Carlo computation and Bayesian model determination". In: *Biometrika* 82.4, pp. 711–732. issn: 0006-3444. doi: 10.1093/biomet/82.4.711.

– (2003). "Trans-dimensional markov chain monte carlo". In: *Highly Structured Stochastic Systems*. Oxford Statistical Science Series. Oxford University Press, pp. 179–198.

Green, P. J. and K. V. Mardia (June 1, 2006). "Bayesian alignment using hierarchical models, with applications in protein bioinformatics". In: *Biometrika* 93.2, pp. 235–254. issn: 0006-3444, 1464-3510. doi: 10.1093/biomet/93.2.235.

Hannay, J. H. and J. F. Nye (2004). "Fibonacci numerical integration on a sphere". In: *Journal of Physics A: Mathematical and General* 37.48, p. 11591. issn: 0305-4470. doi: 10.1088/0305-4470/37/48/005.

Householder, A. S. (1953). *Principles of numerical analysis*. International series in pure and applied mathematics. New York ; London: McGraw-Hill. 274 pp.

Hubbard, T. J. P. et al. (Jan. 1, 1999). "SCOP: a Structural Classification of Proteins database". In: *Nucleic Acids Research* 27.1, pp. 254–256. issn: 0305-1048, 1362-4962. doi: 10.1093/nar/27.1.254.

Kent, J. T., K. V. Mardia, and C. C. Taylor (2010). *Exploring an EM interpretation of the Softassign algorithm for alignment problems*. Research report STAT10-03. University of Leeds.

Kuipers, J. B. (2002). *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace, and Virtual Reality*. Princeton University Press. 398 pp. isbn: 0-691-10298-8.

Mardia, K. V. and P. E. Jupp (2000). *Directional Statistics*. John Wiley & Sons. isbn: 0-471-95333-4.

Möller, Tomas and John F. Hughes (1999). "Efficiently building a matrix to rotate one vector to another". In: *Journal of graphics tools* 4.4, pp. 1–4.

O'Rourke, J. (June 1, 1985). "Finding minimal enclosing boxes". In: *International Journal of Computer & Information Sciences* 14.3, pp. 183–199. issn: 0091-7036, 1573-7640. doi: 10.1007/BF00991005.

Schmidler, S. C. (2007). "Fast Bayesian shape matching using geometric algorithms". In: *Baysian Statistics 8*. Proceedings of the Valencia / ISBA 8th World Meeting on Bayesian Statistics. Ed. by J. M. Bernardo et al. Vol. 8. Benidorm, Alicante, Spain.: Oxford University Press, pp. 471–490. isbn: 978-0-19-921465-5.

Chapter 3

Cormen, T. et al. (Aug. 20, 2009). *Introduction to Algorithms*. 3rd edition. Cambridge, Mass.: MIT Press. 1312 pp. isbn: 978-0-262-53305-8.

Cruickshank, D. W. J. (2006). "Coordinate uncertainty". In: *International Tables for Crystallography Volume F: Crystallography of biological macromolecules*. Ed. by M. G. Ross-

mann and E. Arnold. International Tables for Crystallography F. Springer Netherlands, pp. 403–418. doi: 10.1107/97809553602060000697.

Davies, J. R. et al. (Nov. 15, 2007). "The Poisson Index: a new probabilistic model for protein-ligand binding site similarity". In: *Bioinformatics* 23.22, pp. 3001–3008. issn: 1367-4803, 1460-2059. doi: 10.1093/bioinformatics/btm470.

Goodall, C. R. (1991). "Procrustes Methods in the Statistical Analysis of Shape". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 53.2, pp. 285–339. issn: 0035-9246.

Green, P. J. and K. V. Mardia (June 1, 2006). "Bayesian alignment using hierarchical models, with applications in protein bioinformatics". In: *Biometrika* 93.2, pp. 235–254. issn: 0006-3444, 1464-3510. doi: 10.1093/biomet/93.2.235.

Hayek, L. C. (1994). "Analysis of amphibian biodiversity data". In: *Measuring and Monitoring Biological Diversity: Standard Methods for Amphibians*. Ed. by W. R. Heyer et al. 1st. Biological Diversity Handbook Series. Smithsonian Institution Press, pp. 207–269. isbn: 1-56098-284-5.

Hubert, M. and E Vandervieren (2007). "An adjusted boxplot for skewed distributions". In: *Computational Statistics and Data Analysis* 52, pp. 5186–5201.

Kleywegt, G. J. (Oct. 24, 1997). "Validation of protein models from C_alpha coordinates alone". In: *Journal of Molecular Biology* 273.2, pp. 371–376. issn: 0022-2836. doi: 10.1006/jmbi.1997.1309.

Langron, S. P. and A. J. Collins (1985). "Perturbation Theory for Generalized Procrustes Analysis". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 47.2, pp. 277–284. issn: 0035-9246.

Maechler, M., V. Todorov, et al. (Dec. 8, 2016). *Basic Robust Statistics*. Version 0.92-7.

Rodriguez, A. and S. C. Schmidler (2014). "Bayesian protein structure alignment". In: *The annals of applied statistics* 8.4, pp. 2068–2095. issn: 1932-6157. doi: 10.1214/14-AOAS780.

Sibson, R. (1979). "Studies in the Robustness of Multidimensional Scaling: Perturbational Analysis of Classical Scaling". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 41.2, pp. 217–229. issn: 0035-9246.

Chapter 4

Andreeva, A. et al. (Jan. 1, 2014). "SCOP2 prototype: a new approach to protein structure mining". In: *Nucleic Acids Research* 42 (D1), pp. D310–D314. issn: 0305-1048, 1362-4962. doi: 10.1093/nar/gkt1242.

Ankerst, M. et al. (1999). "OPTICS: Ordering Points to Identify the Clustering Structure". In: *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*. SIGMOD '99. New York, NY, USA: ACM, pp. 49–60.

Bairoch, A. and R. Apweiler (Jan. 1, 2000). "The SWISS-PROT protein sequence database and its supplement TrEMBL in 2000". In: *Nucleic Acids Research* 28.1, pp. 45–48. issn: 0305-1048. doi: 10.1093/nar/28.1.45.

Berjanskii, M. et al. (July 1, 2012). "Resolution-by-proxy: a simple measure for assessing and comparing the overall quality of NMR protein structures". In: *Journal of Biomolecular NMR* 53.3, pp. 167–180. issn: 0925-2738, 1573-5001. doi: 10.1007/s10858-012-9637-2.

Berman, H. M., K. Henrick, and H. Nakamura (Dec. 2003). "Announcing the worldwide Protein Data Bank". In: *Nature Structural Biology* 10.12, pp. 980–980.

Date, C. J. (Apr. 27, 2012). *Database Design and Relational Theory: Normal Forms and All That Jazz*. 1 edition. Sebastopol, Calif: O'Reilly Media. 278 pp. isbn: 978-1-4493-2801-6.

Dryden, I. L. and K. V. Mardia (July 8, 2016). *Statistical Shape Analysis: With Applications in R*. 2nd Ed. Wiley. 496 pp.

Ester, M. et al. (1996). "A density-based algorithm for discovering clusters in large spatial databases with noise". In: *Proceedings of The Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*. AAAI Press, pp. 226–231.

Kleywegt, G. J. (Oct. 24, 1997). "Validation of protein models from C_alpha coordinates alone". In: *Journal of Molecular Biology* 273.2, pp. 371–376. issn: 0022-2836. doi: 10.1006/jmbi.1997.1309.

Mardia, K. V., V. B. Nyirongo, et al. (June 1, 2011). "Hierarchical Bayesian Modeling of Pharmacophores in Bioinformatics". In: *Biometrics* 67.2, pp. 611–619. issn: 1541-0420. doi: 10.1111/j.1541-0420.2010.01460.x.

Richardson, J. S. (1981). "The Anatomy and Taxonomy of Protein Structure". In: *Advances in Protein Chemistry*. Vol. 34. Academic Press, pp. 167–339.

Chapter 5

Adcock, J. et al. (Sept. 1, 2016). "Critical assessment of methods of protein structure prediction: Progress and new directions in round XI". In: *Proteins: Structure, Function, and Bioinformatics* 84, pp. 4–14. issn: 1097-0134. doi: 10.1002/prot.25064.

Gelman, A. et al. (Nov. 1, 2013). *Bayesian Data Analysis, Third Edition*. CRC Press. 677 pp. isbn: 978-1-4398-4095-5.

Green, P. J. and K. V. Mardia (June 1, 2006). "Bayesian alignment using hierarchical models, with applications in protein bioinformatics". In: *Biometrika* 93.2, pp. 235–254. issn: 0006-3444, 1464-3510. doi: 10.1093/biomet/93.2.235.

Mardia, K. V., V. B. Nyirongo, et al. (June 1, 2011). "Hierarchical Bayesian Modeling of Pharmacophores in Bioinformatics". In: *Biometrics* 67.2, pp. 611–619. issn: 1541-0420. doi: 10.1111/j.1541-0420.2010.01460.x.

Code

Adler, Daniel, Duncan Murdoch, et al. (2017). *rgl - 3D Visualization Using OpenGL*. R package version 0.98.1. url: https://CRAN.R-project.org/package=rgl.

Atkinson, Kevin (2011). *GNU Aspell*. Version 0.60.7-20110707. url: http://aspell.net/.

Bates, Douglas and Martin Maechler (2017). *Matrix - Sparse and Dense Matrix Classes and Methods*. R package version 1.2-11. url: https://CRAN.R-project.org/package=Matrix.

Bluhm, Wolfgang (2004). *STAR::Parser*. Package version 0.59. url: http://www.iucr.org/resources/cif/software/starparser.

Brand, H. Merijn (2017). *Text::CSV_XS - comma-separated values manipulation routines*. CPAN package version 1.33. url: http://search.cpan.org/perldoc?Text%3A%3ACSV_XS.

Bunce, Tim (2017). *DBI - Database independent interface for Perl*. CPAN package version 1.637. url: http://search.cpan.org/perldoc?DBI.

Collins, John (2017). *latexmk – Fully automated LATEX document generation*. Version 4.41. url: https://ctan.org/pkg/latexmk.

Dahl, David B. (2016). *xtable - Export Tables to LaTeX or HTML*. R package version 1.8-2. url: https://CRAN.R-project.org/package=xtable.

Dowle, Matt and Arun Srinivasan (2017). *data.table - Extension of 'data.frame'*. R package version 1.10.4. url: https://CRAN.R-project.org/package=data.table.

Dryden, I. L. (2017). *shapes - Statistical Shape Analysis*. R package version 1.2.1. url: https://CRAN.R-project.org/package=shapes.

Evans, Paul (2017). *List::Util - A selection of general-utility list subroutines*. CPAN package version 1.49. url: http://search.cpan.org/perldoc?List%3A%3AUtil.

Golden, David (2013). *File::Temp - return name and handle of a temporary file safely*. CPAN package version 0.2304. url: http://search.cpan.org/perldoc?File%3A%3ATemp.

Guttman, Uri (2011). *File::Slurp - Simple and Efficient Reading/Writing/Modifying of Complete Files*. CPAN package version 9999.19. url: http://search.cpan.org/perldoc?File%3A%3ASlurp.

Hahsler, Michael and Matthew Piekenbrock (2017). *dbscan - Density Based Clustering of Applications with Noise (DBSCAN) and Related Algorithms*. R package version 1.1-1. url: https://CRAN.R-project.org/package=dbscan.

Ihaka, Ross et al. (2016). *colorspace - Color Space Manipulation*. R package version 1.3.2. url: https://CRAN.R-project.org/package=colorspace.

Keenan, James E. (2017). *File::Path - Create or remove directory trees*. CPAN package version 2.15. url: http://search.cpan.org/perldoc?File%3A%3APath.

Lentin, Jamie and Anthony Hennessey (2017). *unittest - TAP-Compliant Unit Testing*. R package version 1.3-0. url: https://CRAN.R-project.org/package=unittest.

Maechler, Martin, Peter Rousseeuw, et al. (2016). *robustbase - Basic Robust Statistics*. R package version 0.92-7. url: https://CRAN.R-project.org/package=robustbase.

Marquess, Paul (2017). *IO::Uncompress::Gunzip - Read RFC 1952 files/buffers*. CPAN package version 2.074. url: http://search.cpan.org/perldoc?IO%3A%3AUncompress%3A%3AGunzip.

Müller, Steffen (2016). *Data::Dumper - stringified perl data structures, suitable for both printing and eval*. CPAN package version 2.161. url: http://search.cpan.org/perldoc?Data%3A%3ADumper.

Pauley, Marty (2008). *Time::Piece::MySQL - Adds MySQL-specific methods to Time::Piece*. CPAN package version 0.06. url: http://search.cpan.org/perldoc?Time%3A%3APiece%3A%3AMySQL.

R Special Interest Group on Databases (R-SIG-DB), Hadley Wickham, and Kirill Müller (2017). *DBI - R Database Interface*. R package version 0.7. url: https://CRAN.R-project.org/package=DBI.

Rehsack, Jens (2017). *List::MoreUtils - Provide the stuff missing in List::Util*. CPAN package version 0.426. url: http://search.cpan.org/perldoc?List%3A%3AMoreUtils.

Ripley, Brian et al. (2017). *MASS - Support Functions and Datasets for Venables and Ripley's MASS*. R package version 7.3-47. url: https://CRAN.R-project.org/package=MASS.

SIGNES, Ricardo (2015). *Carp - alternative warn and die for modules*. CPAN package version 1.38. url: http://search.cpan.org/perldoc?Carp.

– (2016). *File::Spec - portably perform operations on file names*. CPAN package version 3.62. url: http://search.cpan.org/perldoc?File%3A%3ASpec.

Smith, Samuel (2017). *Time::Piece - Object Oriented time objects*. CPAN package version 1.3202. url: http://search.cpan.org/perldoc?Time%3A%3APiece.

Stowe, Jonathan (2016). *Term::ReadKey - A perl module for simple terminal control*. CPAN package version 2.37. url: http://search.cpan.org/perldoc?Term%3A%3AReadKey.

Tange, O. (2011). *GNU Parallel - The Command-Line Power Tool*. Version 20161222. url: http://www.gnu.org/s/parallel.

Vromans, Johan (2017). *Getopt::Long - Extended processing of command line options*. CPAN package version 2.5. url: http://search.cpan.org/perldoc?Getopt%3A%3ALong.