STAMP

Structural Alignment of Multiple Proteins
Version 4.0
User Guide

By Robert B. Russell & Geoffrey J. Barton

When using this program please cite R. B. Russell & G. J. Barton, Multiple protein sequence alignment from tertiary structure comparison, *PROTEINS: Struct. Funct. Genet.*,**14**, 309–323, 1992, and this manual.

Robert B. Russell (RBR)	Geoffrey J. Barton (GJB)
Biomolecular Modelling Laboratory	Laboratory of Molecular Biophysics
Imperial Cancer Research Fund	The Rex Richards Building
Lincoln's Inn Fields, P.O. Box 123	South Parks Road
London, WC2A 3PX, U.K.	Oxford, OX1 3PG, U.K.
Tel: +44 171 269 3583	Tel: +44 865 275368
FAX: +44 171 269 3417	FAX: 44 865 510454
e-mail: russell@icrf.icnet.uk	e-mail gjb@bioch.ox.ac.uk
WWW: http://bonsai.lif.icnet.uk/	WWW: http://geoff.biop.ox.ac.uk/

Contents

Chapter 1

Introduction and Overview

1.1 Preface to Version 4.0

Much has changed with STAMP. The changes are all described in detail somewhere in this document. I list the major changes here:

1. STAMP programs now all accept command line arguments. Parameter files are no longer necessary, but one can still use them if prefered.

2. The scanning protocol has been improved; there are several new options, a new scoring system, and multiple means of sorting output.

3. Initial fitting via ROUGHFIT, and tree generation have been incorporated into the main program; the program MAKETREE is thus no longer needed, and is no longer included in the distribution.

4. Several new programs are now available in the package:

AVESTRUC — creates an average model for a protein structural family given a STAMP alignment.

GSTAMP — a program to create input for Kraulis' program Molscript, to display structurally similar/dissimilar regions within a family of aligned protein structures.

PICKFRAME — allows the selection of a particular domain to be the 'parent' in a superimposition (i.e. to have all other domains superimposed on top of it, leaving it untransformed). This is very useful if you wish to combine, say, similarites involving separate domains from the same protein.

POSTSTAMP — a method for reassessing STAMP alignments.

STAMP_CLEAN — cleans up un-reliable regions (i.e. shortens long gaps where there is no structural similarity). This is most useful when one wishes to display or analyse an alignment.

VER2HOR — converts the text alignment file output from STAMP to vertical format, for a quick analysis of the alignment (i.e. if you don't have ALSCRIPT)

CLUS2BLC, MSF2BLC — two little programs to convert CLUSTAL or MSF format to the BLOCK format used within certain STAMP programs. Useful particularly if you don't have AMPS.

All of these are described in the sections that follow.

5. I hope that several bugs have been removed. As I have said before, STAMP was my first program, and I suspect that other bugs are still to be found. Thanks to all of those users who reported bugs, and I hope you keep reporting them. As ever, I shall swallow my pride.

This is the third version of STAMP to be distributed; comments are always greatly appreciated. STAMP was developed on a SUN SPARCstation and most recently modified using a Silicon Graphics INDY. As a consequence, it may suffer from system specific problems when running on other types of machines. Please let me know if you have such problems.

1.2 Overview

STAMP is a package for the alignment of protein sequence based on three–dimensional (3D) structure. It provides not only multiple alignments and the corresponding 'best-fit' superimpositions, but also a systematic and reproducible method for assessing the quality of such alignments. It also provides a method for protein 3D structure data base scanning. In addition to structure comparison, the STAMP package provides input for programs to display and analyse protein sequence alignments and tertiary structures. Please note that, although STAMP outputs a sequence alignment, it is a program for 3D structures, and NOT sequences. If you are after a multiple sequence alignment for proteins of unknown 3D structure, stop reading now. Contact GJB for information about AMPS, which can be used to perform multiple sequence alignments.

Comparison of 3D structures is a complicated business, particularly if one wants to do unusual things (i.e. reverse a strand direction, swap two segments of a structure around, only consider equivalent structures of greater than 10 residues, etc.). Complicated things are possible with STAMP but as a consequence, the method is very detailed. Please be patient, and read this manual carefully.

Alternatively, if you only want to to fairly straightforward things, such as align a set of structures or search a database of structures for similarities, you can skip this entire chapter and go straight to the next one, which contains a few worked examples that should demonstrate how to use STAMP in a black-box way.

1.3 Background

The aim of this work was to provide a set of multiple sequence alignments derived from structure alone. These alignments have obvious uses which have been described elsewhere (Russell & Barton, 1992; Sali & Blundell, 1990). Numerous other means of deriving such alignments have been presented, but, at the time of the development of STAMP, only one had been applied to alignments of more than two sequences, and no systematic method for assessing the quality of the alignments had been provided. These, then, were the goals of this work.

At the heart of the method is the Rossmann and Argos (Rossmann *et al.*, 1975) equation for expressing the probability of equivalence of residue structural equivalence:

$$P_{ij} = exp\frac{d_{ij}^2}{-2 \times E_1^2} exp\frac{s_{ij}^2}{-2 \times E_2^2}$$

where d_{ij} is the distance between C_α atoms for residues i and j, and s_{ij} is a measure of the local main chain conformation. A detailed description of this equation, and how it has been applied to multiple structures is given in (Russell & Barton, 1992).

STAMP makes extensive use of the Smith Waterman (SW) algorithm (Smith & Waterman, 1981; Barton *et al.*, 1994; Sankoff & Kruskal, 1983). This is a widely used algorithm which allows fast determination of the best path through a matrix containing a numerical measure of the pairwise similarity of each position in one sequence to each position in another sequence. Within STAMP, these similarity values correspond to modified P_{ij} values (above).

The result of the SW algorithm applied to a matrix of modified P_{ij} values is a list of residue equivalences. From this list, which we may obtain a set of equivalenced C_α positions. These are used to obtain a best fit transormation and RMS deviation by a least squares method of (Kabsch, 1978; McLachlan, 1979). This transformation can be applied in the relevant way to yield two new sets of coordinates for which calculation (and correction) of P_{ij} values, the SW path finding and the least squares fitting may be repeated in an iterative fashion until the two sets of coordinates, and the corresponding alignment, converge on a single solution.

This strategy has proved sucessful in the generation of tertiary structure based multiple protein sequence alignment for a wide variety of diverse protein structural families (Russell & Barton, 1992; Russell & Barton, 1993*a*; Russell & Barton, 1993*b*; Russell & Barton, 1994; Russell, 1994). The method can accurately superimpose and obtain alignments for families of proteins as structurally diverse as the greek key β sandwich folds (e.g. immunoglobulin domains, CD4, PapD chaperonin, azurin, superoxide dismutase, actinotaxin, prealbumin, etc.), the aspartic proteinase N and C terminal lobes, the Rossmann fold domains, the globin folds (including phycocyanins and colicins), and many others.

It is important to remember that this method assumes overall topological similarity, and will not, without explicit intervention, be able to superimpose/align structures with common secondary structures in similar orientations, but different connectivity or topologies (such as the different types of four helix bundle proteins: up-down-up-down with up-up-down-down).

Two measures of alignment confidence are provided (Russell & Barton, 1992)

1. A structural similarity Score (S_c) is defined in order that overall alignment quality and structural similarity may be compared across a wide range of protein structural families. These are defined below.

2. A measure of individual residue accuracy P'_{ij} is defined in order that residue equivalences may be normalised with respect to both the number of structures in an alignment and the length of the structures being aligned.

Alignments having a structural similarity Score S_c between 5.5 and 9.8 imply a high degree of structural similarity and almost always suggest a functional and/or evolutionary relationship.. Values between 2.5 and 5.5 correspond to more distantly related structures, and do not always imply a functional or evolutionary relationship. Values less than 2.0 generally indicate little overall structural similarity.

Stretches of three or more aligned positions with P'_{ij} values greater than 6.0 generally correspond to genuine topological equivalences, values between 4.0 and 6.0 are equivalent $> 50\%$ of the time, and values less than 4.0 are generally not equivalent. Stretches of residues having $P'_{ij} > 6.0$ generally correspond to regions of conserved secondary structure within a family of structures being compared. For multiple alignments, an alterntaive and more effective way of assessing residue-by-residue equivalence is provided in POSTSTAMP (see below).

Both of these measures are refered to repeatedly below. For a more detailed description of their derivation please refer to (Russell & Barton, 1992). In addition, RMSD is used to refer to the root mean square deviation between atoms selected for a fit. The CUTOFF refers the lowest allowable P'_{ij} for the program to use a particular pair of residues in a fit (called 'C' in (Russell & Barton, 1992)).

1.4 A brief description of the package

What follows is a brief overview of each application of STAMP. A detailed description of each of these can be found in later sections.

1.4.1 Initial superimposition

The structure comparison algorithm of Rossmann and Argos (Rossmann *et al.*, 1974), which is the method used by STAMP, requires that the protein structures being compared are approximately superimposed initially. If not then structural similarity may be undetected, and reliable superimpositions and alignments unattainable. This is a very important thing to remember of STAMP. If initial superimpositions do not yield high enough scores (i.e. $S_c < 2.0$) or if the structures are generally different, STAMP will warn you with "LOW SCORE – see the manual'.

The STAMP package provides three methods of arriving at an initial superimposition. The first of these is to make use of an alignment derived on the basis of sequence. The program ALIGNFIT requires that the sequences extracted from the PDB files (using the program PDBSEQ) are aligned vertically in AMPS block format (see format and examples below); one may use AMPS or another method of aligning sequences (provided one has first converted the alignment to the AMPS format; two programs to do so are included in the distribution: one converts CLUSTAL format, the other MSF format). The program compares all possible pairs of structures by performing a least squares fit on all equivalenced C_α atoms. Once all pairwise comparisons are compared, the program makes use of a tree to superimpose multiply all coordinates following the tree. Thus the final superimposition output is the best possible fit of the structure given the alignment. For an example where ALIGNFIT is used to provide an initial superimposition, refer to the alignment of the serine proteinases Chapter 2. AMPS can be obtained from GJB.

In instances when multiple sequence alignment is inaccurate, ALIGNFIT may still be used, though the initial superimpositions may not be accurate enough for STAMP to find structural similarity.

It is possible to skip the use of a multiple sequence alignment by using the ROUGHFIT option within STAMP. This option generates an initial alignment which simply consists of the sequences from each domain aligned from their N-terminal ends. This works well in cases where the lengths of the domains to be aligned are similar, and when they exhibit good structural similarity, but will often fail to provide a good starting point. See the alignment of the globin structures in Chapter 2 for an example of using ROUGHFIT.

By far the best way to arrive at initial superimpositions is to use the SCAN option within STAMP. One must merely select a domain with which to scan the other domains to be superimposed. One can obtain (by SCAN and after running SORTTRANS, and removing other redundancies) a set of superimpositions for all other domains onto the domain used to scan. This provides an excellent and accurate starting point from which to begin a multiple structure based alignment by STAMP. This works particularly well when structures are very diverse. For an example, see the alignment of the aspartyl proteinase N– and C–terminal lobes in Chapter 2.

1.4.2 Pairwise comparisons and alignments (PAIRWISE)

Given a suitable initial superimposition of structures, the best way to obtain a multiple alignment and superimposition of a diverse family of domains is to follow a hierarchy of similarity. This allows most similar domains to be compared/aligned first, and only makes comparisons/alignments between distantly related domains at a later time in the procedure.

Pairwise comparisons are an ideal way to obtain such a hierarchy. The PAIRWISE options in STAMP will result in all $N \times (N-1)/2$ comparisons being performed and will output a matrix of pairwise similarities. This is then be used to produce a dendrogram, or *tree*, from which multiple alignments and superimpositions may be generated.

1.4.3 Multiple alignment (TREEWISE)

Given the initial set of superimpositions, and a set of PAIRWISE similarity scores, the TREEWISE option will perform all alignments that are possible given a dendrgoram (generated by considering the PAIRIWISE scores). Statistics, transformations and alignments are output at each stage of the hierarchy so that a continuim of structure variation can be observed (ie. the output will get more and more structural varied as the program progresses).

Note that by default, STAMP performs both PAIRWISE and TREEWISE procedures together.

1.4.4 Structure database scanning (SCAN)

It is often desirable to compare a particular domain or protein structure to a database of known 3D structures in order that structurally similar proteins may be found. The PAIRWISE option within STAMP was a logical starting point from which protein structure database scans could be performed.

Given a single protein domain (a *query*) and a list of domains to which it is to be compared (a *database*), STAMP can be used to perform all possible comparisons of the query to the database structures. The initial superimposition problem is solved by attempting more than one initial fit with each database structure. This can be done in one of two ways, which I will call FAST and SLOW, for the obvious reasons.

In FAST mode, fits are performed by lying query sequence onto the database structure starting at every ith position, where i is an adjustable parameter usually set to five (i.e. the sequence is laid onto the 1st, 6th, 11th, etc. position). Diagramatically, this looks like:

```
Q=query, D=database

Fit 1  Q -------
       D -----------
Fit 2  Q  -------
       D -----------
Fit 3  Q   -------
       D -----------
      <etc.>
```

This approach is fine if the query is a single domain, and there is a strong similarity in the database structure. However, if similarity is weaker, or if the query is multi-domain (not always a good idea, I would recommend splitting the structure into domains first, though this may not always be possible), then SLOW mode will perform more fits (hence "SLOW") by sliding query and database sequences along each other like:

```
Q=query, D=database

Fit 1   Q -------
        D       -----------
Fit 2   Q -------
        D      -----------
Fit 3   Q -------
        D    -----------
      <etc.>
Fit N-2 Q        -------
        D -----------
Fit N-1 Q         -------
        D -----------
Fit N   Q          -------
        D -----------
```

In this approach, initial superimpositions are calulated using many more fractions of query and database structure, making detectiong of weak similarities more likely.

The residues that are equivalenced by either FAST or SLOW procedures are used to perform an initial fit, which is refined by the conformation-based and distance-based fit used during PAIRWISE/TREEWISE comparison of distantly related structures. If a high enough similarity score (S_c) is found after these three steps, then the transformation is saved for further analysis. The output from the SCAN routine is directly readable by STAMP so that once a list of domains similar to one's query is obtained, multiple alignment (ie. PAIRWISE and TREEWISE) may be performed.

The program PDBC can be used to generate a list of protein domains given a set of PDB identifier codes, and the program SORTTRANS can be used to sort the output from SCAN, and remove any redundancies.

The Sc values output during a SCAN differ slightly from those output during a PAIRWISE comparison. The correction introduced to correct the SW Score according to the length of the sequence lengths is removed. During multiple alignment the start and end points of the domains to be superimposed should be known, thus one can penalise all for all positions which are not involved in the alignment. During a scan, however, it is desireable to detect sub alignments of the two structures being compared (eg., N-terminal helix from query missing in database structure, a much longer database structure, etc.). Thus, the Sc for scanning may be defined in one of three ways (a=query, b=database, p=path, i=insertion, L=length):

Scheme 1

$$S_c = \left(\frac{S_p}{L_p}\right)\left(\frac{L_p - i_a}{L_a}\right)\left(\frac{L_p - i_b}{L_b}\right)$$

As for multiple structure alignment. As discussed, this is generally not the best way to compare a query to the database, since one would not usually wish to penalise insertions or omitted missing segments within the database structure (due to truncation values, etc.). However, this scheme may be useful if one is scanning a database of structures known to exhibit a particular fold (i.e., if one is merely after accurate superimpositions for a family of known structures; see Chapter 2).

Scheme 2

$$S_c = \left(\frac{S_p}{L_p}\right)\left(\frac{L_p - i_a}{L_p}\right)\left(\frac{L_p - i_b}{L_p}\right)$$

L_a and L_b have been replaced by L_p to removed any dependence on query or database structure length. The second two terms lower the score if gaps in the path are placed in query (a) or database structure (b). This avoids a consideration of length, but will allow short stretches structural equivalences to score highly.

Scheme 3

$$S_c = \left(\frac{S_p}{L_p}\right)\left(\frac{L_p - i_a}{L_a}\right)$$

Only penalises insertions in the query sequence. If a small fraction of the query sequence is in the actual path, then S_c drops. This scheme is most useful if one only wants similarities to the entire protein under consideration, since it penalises any omissions from the query structure.

Scheme 4

$$S_c = \left(\frac{S_p}{L_p}\right)\left(\frac{L_p - i_b}{L_b}\right)$$

The opposite of 3. Only penalises insertions in the database sequence. If a small fraction of the database sequence is in the actual path, then S_c drops. This scheme may be useful if one is scanning with a collection of secondary structure elements, since gaps are to be expected within the query (i.e., since the loops have been omitted).

Scheme 5

$$S_c = \left(\frac{S_p}{L_p}\right)$$

Raw score, no length requirement, will report even short alignments between similar sub–structures. This scheme may be useful for the search for short stretches of structural similarity, such as supersecondary structures.

Scheme 6

$$S_c = \left(\frac{S_p}{L_a}\right)\left(\frac{L_a - i_a}{L_a}\right)$$

Vaguely similar to Scheme 3, but this only scores hits favourably if the involve a significant fraction of the query structure (i.e. similarities only containing part of the query will not stand out). This is useful when one is comparing a particular domain to a database and is not interested in local similarities. This is the default for scanning.

For the most part, all of these scoring schemes will yield similar numbers for very similar structures. However, when more distantly related structures are compared, it becomes more useful to use a scheme specific to the particular problem (i.e., whether one wishes to scan with secondary structures only, when one is after only very similar structures, etc.).

Schemes are specified by the parameter SCANSCORE (see below). If you are confused, or haven't thought about this at all, just use the defaults. I do.

1.4.5 Displaying STAMP output (VER2HOR, DSTAMP, GSTAMP)

In order that the output from STAMP may be displayed in a more attractive manner, a program was developed to translate the vertical, block-file format of STAMP multiple alignments into input for GJB's program ALSCRIPT, which can then be used to display the alignments in Postscript. Details of DSTAMP and how to obtain ALSCRIPT are given

CHAPTER VI. Contact Geoff Barton for a copy of ALSCRIPT (worth having, even for other sequence alignments). The program deterimes reliable regions given a set of criteria, and highlights sequence and secondary structure accordingly.

Another program (VER2HOR) can display STAMP alignments in a horizontal text format for quick viewing (i.e. it takes the vertical format of STAMP output, and changes it to an easy to see horizontal format). This is particularly useful if you don't have ALSCRIPT.

STAMP multiple alignments are always associated with particular superimpositions. It is often useful to show such superimpositions graphically. GSTAMP provides input for Per Kraulis' program MOLSCRIPT. Most structual biology labs have at least one copy of this program, and it is available from Per Kraulis. One simple needs to run TRANSFORM (see below) on a STAMP alignment file, and then run GSTAMP on the same file, followed by separate runs of MOLSCRIPT, to produce separate Postscript files for each aligned structure, with structually equivalent regions shown as ribbons, the rest as C_α trace.

1.5 The programs contained within the package

STAMP consists of the main program (usually refered to as STAMP) and thirteen sub-programs. Briefly, the programs are:

STAMP	Main program, does PAIRWISE, tree construction, TREEWISE and SCAN modes.
ALIGNFIT	Given a list of domains and a multiple sequence alignment outputs an initial transformation.
PDBC	Finds and reports information about PDB files given a four (PDB) code and/or chain identifier.
TRANSFORM	Given a list of transformations, outputs the corresponding set of coordinates.
SORTTRANS	Sorts the output from SCAN, and removes repeated transformations.
PDBSEQ	Given a list of domains, extracts the corresponding sequences from the PDB files.
VER2HOR	Given a STAMP alignment file, outputs an easy to read text version of the alignment for quick analysis.
DSTAMP	Given a STAMP alignment file, outputs commands for GJB's program ALSCRIPT (alignment to Postscript).
GSTAMP	Given a STAMP alignment file, outputs commands for Per Kraulis' MOLSCRIPT program.
AVESTRUC	Given a STAMP alignment file, generates an average set of main chain or C alpha coordinates for the structural family.
POSTSTAMP	Reanalyses a STAMP alignment file (provides a more accurate set of equivalences for alignments of more than one structure).
PICKFRAME	Given a transformation, transforms all other domains onto another (specified by the user).
CLUS2BLC	Takes a sequence alignment in CLUSTAL format and converts it to AMPS format for use during the program ALIGNFIT.
MSF2BLC	Takes a sequence alignment in MSF format and converts it to AMPS format for use during the program ALIGNFIT.

Chapter 2

Worked examples

The examples described below show how to apply STAMP to particular problems. A description of the Input and Output and a summary of all parameters appear in Chapter 4.

All example output files may be found in the directory examples/ within the directory where STAMP is installed. There are four sub-directories in the examples directory corresponding to each of the four protein structure familes discussed in the examples below (s_prot/, ac_prot/, ig/, globin/).

Before begining you must ensure a few things:

1. That you have set the environment variable STAMPDIR to the name of the directory containing the various STAMP defaults files. This directory is called defs/ within the directory where STAMP is installed.

2. Ensure that you have a copy of the PDB, and that you edit the file STAMPDIR/pdb.directories (see below) to tell the program where the PDB files might be found (it is OK if they are stored in more than one place, or with different extensions). Given a PDB code, these programs search through the various directories until an appropriate file is found.

3. (optional, but worthwhile). Get a copy of DSSP and run it on the PDB structures used in the examples below (and indeed any others you wish to analyse). Edit the file STAMPDIR/dssp.directories to tell the program where to find them (in the same manner as STAMPDIR/pdb.directories). I would recommend putting DSSP files somewhere central for all users to share (this saves having to run the program many times on the same PDB files).

2.1 Multiple alignment (PAIRWISE and TREEWISE)

Mammalian and Bacterial Serine Proteinases

(This example is discussed in Russell & Barton, (1992).)

Despite a pronounced functional similarity (a highly conserved catalytic triad), this family of proteins shows little overall sequence similarity. Indeed, sequence alignment methods generally fail to provide an accurate alignment of these protein sequences. In situations like these, STAMP can be used to provide an accuarate alignment of protein sequences based on a comparison of 3D structure. This can often reveal regions of weak sequence similarity that are not detectable during a comparison of sequence. The files for this example are in the directory examples/s_prot in the directory where you have installed STAMP.

A list of the domains is given in the file s_prot.domains. The output. Running PDB-SEQ:

```
pdbseq -f s_prot.domains > s_prot.seqs
```

produced the file s_prot.seqs, from which an AMPS multiple sequence alignment was produced, and stored in the file s_prot_amps.align. Running ALIGNFIT:

```
alignfit -f s_prot_amps.align -d s_prot.domains -out s_prot_alignfit.trans
```

should give the output:

```
ALIGNFIT R.B. Russell 1995
 Reading in block file...
 Blocfile read: Length: 261
 Reading in coordinate descriptions...
 Reading coordinates...
 Checking for inconsistencies...
 Doing pairwise comparisons...
 Doing treewise comparisons...
 ALIGNFIT done.
 Look in the file s_prot_alignfit.trans for output and details
```

The final transformation (called alignfit.trans if default ALIGNFIT settings are used) is in the file s_prot_alignfit.trans.

This provides an initial set of transformations for use by STAMP. To run STAMP type:

```
stamp -l s_prot_alignfit.trans -prefix s_prot
```

Should produce the following output:

```
STAMP Structural Alignment of Multiple Proteins
 by Robert B. Russell & Geoffrey J. Barton
 Please cite PROTEINS, v14, 309-323, 1992

    Sc = STAMP score, RMS = RMS deviation, Align = alignment length
    Len1, Len2 = length of domain, Nfit = residues fitted
    Secs = no. equivalent sec. strucs. Eq = no. equivalent residues
    %I = seq. identity, %S = sec. str. identity

    No.   Domain1  Domain2 Sc     RMS    Len1 Len2  Align NFit Eq. Secs.   %I    %S
Pair   1     4chaa     3est 7.68   1.12    239  240    241  207 205   20  36.67  79.58
Pair   2     4chaa     2ptn 7.74   1.01    239  223    233  200 199   20  41.42  82.01
Pair   3     4chaa     1ton 6.82   1.28    239  227    241  182 178   19  31.80  76.57
Pair   4     4chaa    3rp2a 7.40   1.15    239  224    234  194 187   18  30.96  75.31
Pair   5     4chaa   2pkaab 7.21   1.34    239  232    240  197 194   19  31.38  80.75
Pair   6     4chaa     1sgt 7.04   1.33    239  223    238  189 182   20  30.13  79.92
                                       <etc.>
Reading in matrix file s_prot.mat...
Doing cluster analysis...
Cluster:  1 (    2ptn  &    2pkaab ) Sc  8.50 RMS   1.08 Len 232 nfit 213
 See file s_prot.1 for the alignment and transformations
Cluster:  2 (    2sga  &    3sgbe ) Sc  8.36 RMS   0.62 Len 191 nfit 164
 See file s_prot.2 for the alignment and transformations
                             <etc.>
Cluster:  8 (    1sgt  &    4chaa     3est    3rp2a     1ton     2ptn   2pkaab )
Sc  7.58 RMS   1.14 Len 267 nfit 177
 See file s_prot.8 for the alignment and transformations
Cluster:  9 (    1sgt    4chaa     3est    3rp2a     1ton     2ptn   2pkaab
&   2alp     2sga    3sgbe ) Sc  4.79 RMS   1.86 Len 292 nfit 109
 See file s_prot.9 for the alignment and transformations
```

The various fields describe details of the pairwise and treewise comparisons: S_c, RMS deviation, the alignment length (Align), the length of each structure in residues (Len1, Len2), the number of atoms used in the RMS fit (Nfit), the number of equivalent secondary structure elements (Secs), and the number of equivalent residues (see above, Eq.).

STAMP will also produce several files:

s_prot.mat – a file containing the information used to derive the structural similarity tree (i.e. the output from the PAIRWISE) mode. This is an upper diagonal matrix containing the pairwise S_c values.

s_prot.N – a series of files containing transformations and alignments created by running the TREEWISE mode in STAMP. Each file corresponds to a *node* in the similarity tree (i.e. a *cluster*), where two groups of one or more structures have been combined to form an alignment and transformations. The higher the value of N the more structurally dissimilar the proteins contained in the file are. Highly similar structures are clustered (aligned/superimposed) at an early stage in the program, with more distantly related structures being clustered towards the end.

The top of each file contains the information needed to generate (using TRANSFORM, see below) superimposed coordinates (in STAMP domain format, see below). After these details, various details of the similarity (RMS deviation, S_c value, etc) are given. The bottom portion of the file contains the structural alignment in STAMP format. Positions not aligned with gaps contain information as to the degree of local structural similarity, such as the distance between (averaged) C_α atoms, and the P'_{ij} value.

Methods for displaying sequence alignments and structures are described below.

2.2 Alignment using an initial rough superimposition

This method avoids having to create an initial sequence alignment, and tends to work for homologous proteins, or those having very similar lengths despite no sequence similarity.

Globins

Since the globin sequences are of similar length an initial superimposition accurate enough to proceed with STAMP can be obtained by merely aligning the N-terminal ends of the sequences and using whatever equivalences result to obtain an initial superimposition. The command ROUGH (ROUGHFIT procedure) is used. In addition, an initial conformation based fit is performed in order that any inaccuracies in this initial superimposition may be corrected. See the directory examples/globins.
To run STAMP in this example, type:

```
stamp -l globin.domains -rough -n 2 -prefix globin
```

should produce the following on the standard output (ignoring the header):

```
Running roughfit.

    Sc = STAMP score, RMS = RMS deviation, Align = alignment length
    Len1, Len2 = length of domain, Nfit = residues fitted
    Secs = no. equivalent sec. strucs. Eq = no. equivalent residues
    %I = seq. identity, %S = sec. str. identity

    No.   Domain1  Domain2 Sc     RMS    Len1 Len2  Align NFit Eq. Secs.   %I    %S
Pair   1     2hhbb    2hhba 8.08   1.37    146  141   146  133 130    7  41.10  78.08
Pair   2     2hhbb     2lhb 7.01   1.49    146  149   150  125 124    7  24.16  77.18
Pair   3     2hhbb     4mbn 7.96   1.42    146  153   147  138 137    8  23.53  77.78
Pair   4     2hhbb     1ecd 6.79   2.10    146  136   143  122 114    7  17.12  76.71
Pair   5     2hhbb     1lh1 5.80   2.39    146  153   154  112 106    7  15.69  69.28
                                         <etc.>
Cluster:  5 (    1lh1  &     2lhb     1ecd     4mbn    2hhbb    2hhba )
Sc  7.83 RMS   2.45 Len 156 nfit 116
 See file globin.5 for the alignment and transformations
```

where the output and files are as described for the serine proteinase example above, with ‘s_prot’ replaced with ‘globin’.

-rough performs the initial superimpositions (ROUGHFIT) and -n 2 means that the conformation biased fit will be performed before the final fit. This conformation biased fit is usually necessary when the initial superimpositions are approximate.

ROUGHFIT will not always work. Note that in this example all the pairwise S_c values are above 5.6, suggesting strong structural similarity. If when using the ROUGHFIT option you find low S_c values (the program will cry out LOW SCORE – see the manual), this usually means that ROUGHFIT hasn't managed to generate a good enough starting superimposition, and you should try something else, such as is described in the next section.

2.3 Database Scanning

Database scanning within STAMP is unpublished, apart from a brief description in a figure legend (Matthews *et al.*, 1994), but it has been fairly well tested since version 2.0. Indeed, two novel similarities have resulted in publications (Russell & Barton, 1993*a*; Matthews *et al.*, 1994).

Immunglobulin domain

One example of a scan is given. The light chain variable domain of the immunoglobulin 2FB4 is used to scan a small database of other protein domains containing both a diverse collection of related folds (greek key folds, including azurin, superoxide dismutase, CD4, etc.), and completely unrelated folds (such as globins). See the directory examples/ig for this example.

The 2FB4 domain is described in 2fb4lv.domain. To scan this through the database type:

```
stamp -l 2fb4lv.domain -s -n 2 -slide 5 -prefix 2fb4lv_stamp -d some.domains  -cut
```

'-s' specifies the SCAN mode '-slide' describes how many residues to slide the query sequence (2fb4lv) along each sequence in the file some.domains to provide each initial fit (i.e. the sequence of 2fb4lv is layed on top of each database sequence at postions 1, 6, 11, etc.). '-cut' tells the program to cut down each domain read in from some.domains according to where the similarity is found. If it is not specified, the output will contain domain descriptors identical to those found in 'some.domains'. When one is comparing a single-domain query to a database structure having mulitple domains, it is desirable to do this. Try running it both ways (with and without -cut) and look at the output: you will see what I mean. (e.g. CHAIN A is converted to A 1 _ to A 60 _ in one descriptor in the SCAN output and A 120 _ to A 175 _ in another, since there are two repeats of the query domain in the database structure).

The above run may take a couple of minutes, and should write the following to the standard output (again, ignoring the header):

```
Results of scan will be written to file 2fb4lv_stamp.scan
Fits = no. of fits performed, Sc = STAMP score, RMS = RMS deviation
Align = alignment length, Fit = residues fitted, Eq = equivalent residues
Secs = no. equiv. secondary structures, %I = seq. identity, %S = sec. str. identity

       Domain1    Domain2  Fits  Sc       RMS Len1 Len2 Align Fit  Eq   Secs   %I    %S
Scan     2fb4lv    2fb4lc    1   3.460   2.730  111  105  127   50   41    7   9.01  49.55
Scan     2fb4lv     2fb4l    3   9.711   0.001  111  166  110  110  109   11  66.27  64.46
Scan     2fb4lv    1mcplv    1   7.430   1.292  111  113  115   95   93   11  44.25  79.65
Scan     2fb4lv    1mcphv    1   6.111   1.686  111  122  124   83   80   10  22.95  75.41
Scan     2fb4lv     1cmsC    0   1.382   2.236  111  148  159   20   13    2   4.73  18.92
Scan     2fb4lv      3cd4    3   4.376   2.059  111  166  114   57   55   10   7.83  29.52
Scan     2fb4lv     2hhbb    0   0.000 100.000  111  146    0    0   55    0   0.00   0.00
Scan     2fb4lv      3dpa    3   3.846   2.397  111  166  132   58   53    6   4.82  34.94
Scan     2fb4lv     3sgbe    0   0.922   2.939  111  166  184   15   12    3   2.41  16.27
Scan     2fb4lv      1acx    1   2.951   2.237  111  108  135   47   42    5   8.11  36.04
Scan     2fb4lv     2abxa    0   0.973   2.740  111   74  108   11    5    0   0.90  19.82
Scan     2fb4lv      1l01    0   0.000 100.000  111  164    0    0    5    0   0.00   0.00
Scan     2fb4lv     2azaa    1   3.354   2.765  111  129  131   41   32    6   7.75  49.61
Scan     2fb4lv      1rnt    0   1.146   2.313  111  104  155   19   18    4   3.60  25.23
Scan     2fb4lv     2sodo    1   2.535   2.471  111  151  159   38   29    7   6.62  42.38
Scan     2fb4lv     8rubs    1   2.366   2.137  111  123  143   36   30    5   2.44  33.33
Scan     2fb4lv      2pcy    1   3.102   2.064  111   99  120   39   33    5  12.61  42.34
Scan     2fb4lv     8atca    0   1.054   2.353  111  166  165   19   17    2   2.41  17.47
See the file 2fb4lv_stamp.scan
```

where all of the fields are as for the PAIRWISE mode, save for Fits, which indicates the number of fits that were saved to the file '2fb4lv_stamp.scan'. Note that for domain descriptors (see some.domains) containing two Ig type folds (e.g. 2fb4l, 1cd4, etc.) that more than one fit has been saved, since the search found both of the Ig type folds in each of these two proteins. Not also that 'Fits' is zero for several of the examples, indicating that the no similarity was found within these proteins. Where more than one Fit is output for a domain in the database, the best S_c, RMS etc. are reported.

2fbjlv_stamp.scan will contain all the transformations output during the scan. Several of these will be redundant, since it is possible for a particular match to be found twice. To remove repeated transformations, or those not considered interesting, one must run the program SORTTRANS on the output.

```
sorttrans -f 2fb4lv_stamp.scan -s Sc 2.5 > 2fb4lv_stamp.sorted
```

sorts the input file by S_c values, and leaves only those non-redundent domain descriptions having an $S_c \geq 2.5$. In practice, I tend to use a value of 2.0, and then sort through the output to look for interesting similarities.

```
sorttrans -f 2fb4lv_stamp.scan -s rms 1.5  > 2fb4lv_stamp.sorted
```

sorts the input file by RMSD values, and leaves only those domain descriptions having an RMSD ≤ 1.5 Å. Despite its predominance in the literature, RMSD is not a very good means of measuring structural similarity, since low RMSDs can usually be obtained for any two structures if one considers a small enough collection of residues.

```
sorttrans -f 2fb4lv_stamp.scan -s nfit 40 > 2fb4l_stamp.sorted
```

sorts the input file by the number of atoms used in the final fitting, and leaves only those domain descriptions where nfit ≥ 40.

```
sorttrans -f 2fb4lv.scan -s n_sec 6 > 2fb4lv_stamp.sorted
```

sorts the input file by the number of equivalent secondary structures, and leaves only those having 6 or more secondary structures equivalent.

Combinations of these can be used to select out interesting domains from a scan output. Probably the best combination involves Sc and nfit (ie. score and nfit), since large structures can give fortuitously large S_c values with very few fitted atoms.

The final output is in the file 2fb4lv_stamp.sorted. This is the result of the first example (ie. -s Sc 2.5). Note that several structures similar to the Ig type domain have been detected, and appear (according to S_c) in the order one might expect from knowledge of the 3D structures, sequences and functions of these proteins.

The output from scanning is totally compatable with the other modes of the program. Once you have performed a scan, and have sorted the 'hits' down to an interesting set, you can then use the output from scan as the input for a multiple alignment. E.g.,

```
transform -f 2fb4lv_stamp.sorted -g -o ig_like.pdb
```

will read in the files, transform the coordinates and save them to the file ig_like.pdb (with each chain labelled starting with a different letter). This program is explained in one of the next sections.

```
stamp -l 2fb4lv_stamp.sorted -prefix ig_like > ig_like.log
```

will read in the transformations, and run PAIRWISE and TREEWISE comparisons to generate a multiple alignment of these structures. The results of this run are in the examples/ig directory. Note that there are several 'LOW SCORE – read the manual' warnings in the output (stored in ig_like.log). Note that one would normally edit the output from a scan before performaing a multiple alignment (i.e. to include only those domains one wants to consider further).

2.4 Using scans as a starting point for multiple alignment

In certain instances initial fits based on multiple sequence alignment will be far from accurate, such that even an initial conformation based fit will not be able to correct the poor initial superposition, and even genuine structural homology will be missed. In these instances it is possible to make use of the SCAN option to provide a more accurate initial superimposition.

To do this one need only select one representative of the domains to be superimposed and use this domain in a sensiitve scan *of the other domains*. By applying the same techinques as used for the scan with the Ig light variable domain (above) one can arrive at a set of initial transformations consisting of the transformations of all other domains onto the domain which was used as a query for the scan.

Aspartic Proteinase Domains

An example of how such an initial superimposition might be obtained is shown by the alignment of the aspartly proteinase N and C terminal lobes (see directory examples/ac_prot):

The N–terminal domain of 1CMS (in the file 1cmsN.domain) can be used to scan a list of all aspartyl proteinase N– and C– terminal domains (ac_prot.domains):

```
stamp -l 1cmsN.domain -n 2 -s -slide 5 -d ac_prot.domains -prefix ac_prot
```

Should produce:

```
        Domain1     Domain2    Fits  Sc        RMS  Len1 Len2 Align Fit   Eq.  Secs     %I     %S
Scan      1cmsN       1cmsN     1   9.744    0.000  174  175  175  174  173    18  99.43  94.86
Scan      1cmsN       1cmsC     1   2.749    2.352  204  175  148   63   58    13   8.00  41.14
Scan      1cmsN       4apeN     1   7.854    1.314  181  175  178  155  153    15  27.53  80.90
Scan      1cmsN       4apeC     1   2.901    2.207  208  175  152   69   68    14   7.43  40.57
Scan      1cmsN       3appN     1   7.574    1.291  182  175  174  148  147    18  26.86  83.43
Scan      1cmsN       3appC     1   2.775    2.422  205  175  149   61   58    13   7.43  41.71
Scan      1cmsN       2aprN     1   8.246    1.115  177  175  178  160  157    15  32.02  83.15
Scan      1cmsN       2aprC     1   2.941    2.099  201  175  147   65   58    14   9.14  39.43
Scan      1cmsN       4pepN     1   8.901    1.020  173  175  174  169  168    15  56.57  89.71
Scan      1cmsN       4pepC     1   2.738    2.561  206  175  152   64   52    11   8.00  40.00
See the file ac_prot.scan
```

The file ac_prot.scan will contain all 10 domains superimposed onto 1cmsN. Note that we haven't run the program with the '-cut' option, since the file ac_prot.domains contains an assignment of domains (done by me using molecular graphics). Running SORTTRANS removes any redundancies:

```
sorttrans -f ac_prot.scan -s Sc 2.5 > ac_prot.sorted
```

and running stamp will generate the multiple alignment as described for the serine proteinase and globin examples above.

```
stamp -l ac_prot.sorted -prefix ac_prot
```

The output files from running all of these programs appear in the directory examples/ac_prot.

2.5 Protein domain databases

The program PDBC may be used to output a set of STAMP readable domain descriptions. Given a list of four letter brookhaven codes and an optional set of chains. This will only work if you have a suitable 'pdb.directories' file. See the chapter on installation for details on how to do this.

```
pdbc -d 2hhba >! globin_fold.domains
pdbc -d 2hhbb >> globin_fold.domains
pdbc -d 4mbn  >> globin_fold.domains
pdbc -d 1lh1  >> globin_fold.domains
pdbc -d 1cola >> globin_fold.domains
pdbc -d 1cpca >> globin_fold.domains
```

will produce the following output (ignoring comments, which are specified by a '%' in column 0):

```
/(PDB PATH)/pdb2hhb.ent 2hhba { CHAIN A }
/(PDB PATH)/pdb2hhb.ent 2hhbb { CHAIN B }
/(PDB PATH)/pdb4mbn.ent 4mbn { ALL }
/(PDB PATH)/pdb1lh1.ent 1lh1 { ALL }
/(PDB PATH)/pdb1col.ent 1cola { CHAIN A }
```

Where (PDB PATH) denotes the location of the relevant PDB file on your system. Note that your PDB files may be called (code).pdb instead, or may follow some other convention. This is OK, see Chapter 5 (installation) for details as to setting this up.

Note that there doesn't need to be a filename in the domain file. One can merely leave it as 'Unknown' or some other string (i.e. not empty spaces), and the programs will try and find where the file corresonding to the four letter code is one your system. In other words, the files given in this distribution should work on your system, provided that you have all the PDB files.

Sensitive STAMP database comparisons can take a long time. For this reason we have compared the current PDB database to itself based on sequence, and clustered the data such that only one member of each sequence family is in our domain database. We have also split these structures into domains using author definitions. This database, even when a high degree of sequence similarity is required for clustering, reduces the size of the brookhaven database drastically (from over 4500 independent chains down to just under 1000 protein domains). It is probably sensible to scan this database, which contains just one representative of each sequence family, then if something interesting is found one can scan other structures related to the representative by sequence. A copy of this representative database is in the file defs/brookhaven_subset.domains.

Note that PDBC can be used to probe information about a PDB entry by using the '-q' option. Try it and see. This is a good test of whether STAMP has been set up properly on your system.

2.6 Generating transformed coordinates

The program TRANSFORM can be used with any file containing domain descriptions to output a set of PDB format files for display or further analysis. For example, if the transformed PDB files for the globin structural alignment are desired, then it is only necessary to type:

```
transform -f globin.5
```

should write the following to the standard output:

```
TRANSFORM R.B. Russell, 1995
 Using PDB files
 Files will not include heteroatoms
 Files will not include waters
 Domain   1,    1lh1 => to 1lh1.pdb
 Domain   2,    2lhb => to 2lhb.pdb
 Domain   3,    1ecd => to 1ecd.pdb
 Domain   4,    4mbn => to 4mbn.pdb
 Domain   5,   2hhbb => to 2hhbb.pdb
 Domain   6,   2hhba => to 2hhba.pdb
```

To get a set of PDB format files containing the superimposed coordinates. Running the program as shown above will produce one PDB file for each domain identifier. If one wishes to look at the superimposed structures *together* (in the same file), then the option -g (i.e. graphics) can be used:

```
transform -f globin.5 -g -o globins.pdb
```

should write the following:

```
TRANSFORM R.B. Russell, 1995
 Using PDB files
 Files will not include heteroatoms
 Files will not include waters
 All coordinates will be in file globins.pdb
 Domain   1,    1lh1 => to globins.pdb (chain A)
 Domain   2,    2lhb => to globins.pdb (chain B)
 Domain   3,    1ecd => to globins.pdb (chain C)
 Domain   4,    4mbn => to globins.pdb (chain D)
 Domain   5,   2hhbb => to globins.pdb (chain E)
 Domain   6,   2hhba => to globins.pdb (chain F)
```

This options puts transformed coordinates for each domain into one file (specified by -o, in this example it is globins.pdb). Each domain will be labelled sequentially with a different chain identifier (i.e. A, B, C, etc.). Note that only 'globins.pdb' is included in the example directory.

Be default, TRANSFORM does not include heteroatoms in the output. If you wish heteroatoms to be included, then add -het to the transform command. If you wish waters to be included in the file add -hoh. Note that heteroatoms/waters are sometimes included that fall outside the range of your domain descriptor. This may seem silly, but it is difficult to deterine which heteroatoms are associated with which residues given PDB format.

2.7 Generating averaged coordinates

It may also be useful to have a set of *averaged* coordinates derived from a protein structural family. This makes it possible to see what portions of the structure are common to all members of the family (i.e. the common core). The program AVESTRUC takes the output from STAMP (i.e. an aligned family of protein structures), and generates a PDB file containing averaged coordinates for the common core as identified by STAMP. For example, to generate the averaged coordinates for the aspartic proteinase domains one needs to type:

```
avestruc -f ac_prot.8 -o ac_prot_ave.pdb
```

The file ac_prot_ave.pdb will contain a set of averaged C_α atoms taken by averaging the coordinates for those positions within the file ac_prot.8 that are found to be structurally equivalent. To obtain a poly Alanine set of coordinates (i.e. including main chain and C_β coordinates), type:

```
avestruc -f ac_prot.8 -o ac_prot_ave.pdb -polyA
```

Note that this will only work if all main chain atoms are found in the file (i.e. it won't work if the PDB files contain only C_α atoms).

2.8 Displaying/processing the output

2.8.1 POSTSTAMP

There is something inherrently wrong with the way STAMP assigns equivalences within multiple alignments. It considers an average set of C_α coordinates and uses an average set of probabilities to derive equivalences when more than two structures are involved, and as a consequence, it appears to go wrong (sometimes only) during this process. Usually this is only when very distantly related proteins are being considered. A fix to this problem is to consider each pair of structures within the alignment separately, and to re-calculate the *raw* Rossmann and Argos probabilities. One need then define positions as structurally equivalent when *all* pairs of structures have a P_{ij} value larger than a cutoff at a particular residue position.

For example, for ten structures, there are $(10 \times 9/2) = 45$ pairs. For a position to be structurally equivalent across all members of the family, P_{ij} should be ≥ 0.5 for all 45 pairs.

POSTSTAMP does just this. It adds two new STAMP format fields to a STAMP alignment file: one tells whether the above is true (1) or false (0) for each position (i.e. is each position structurally equivalent across all members of the family); the second tells how many pairwise comparison have P_{ij} greater than or equal to the cutoff (e.g. 0.5).

For example,

```
poststamp globin.5 0.5
```

Creates a file globin.5.post, containing the above data for a P_{ij} value of 0.5.

2.8.2 STAMP_CLEAN

When aligning more than one structure, STAMP will usually create alignments that are fairly meaningless within regions that are not structurally equivalent across all structures. Such regions may have meaning for particular sub-families of structures, but for the purposes of display, are nonsensical. STAMP_CLEAN is a useful program that takes a STAMP alignment file and 'cleans up' such gaps. To run the program, for example (using the POSTSTAMP output file generated above):

```
stamp_clean globin.5.post 3 > globin.5.clean
```

will create a file globin.5.clean where all gaps not lying within structurally equivalent regions, and having fewer than 3 aligned residues in a row (i.e blocks where all sequences are not aligned with gap) are shortened to their minimum length.

2.8.3 Displaying text alignments

The program VER2HOR takes a STAMP alignment file and outputs a horizonal text format. For example, to display the globin alignment, one needs to type:

```
ver2hor -f globin.5.clean
```

to give (see examples/globin/globin.5.ver2hor):

```
VER2HOR R.B. Russell, 1995
 Prints STAMP alignments in horizontal format
  for quick viewing
 Reading Alignment...
 Blocfile read: Length: 163
 Getting STAMP information...
 6 STAMP fields read in for 163 positions
 Processing the alignment...
 Output:
 Very reliable => Pij' >=6 for stretches of >=3
 Less reliable => Pij' >=4.5 for stretches of >=3
 Post reliable => All Pij' > stamp_post parameter for stretches >=3

Number               10        20        30        40        50
     1lh1            gaLTESQAALVKSSWEEfnanipKHTHRFFILVLEIAPAAKDLFSFLkg
     2lhb    pivdtgsvapLSAAEKTKIRSAWAPvystyeTSGVDILVKFFTSTPAAQEFFPKFkg
     1ecd              LSADQISTVQASFDKv    kGDPVGILYAVFKADPSIMAKFTQFag
     4mbn             vLSEGEWQLVLHVWAKveadvaGHGQDILIRLFKSHPETLEKFDRFkh
    2hhbb            vhLTPEEKSAVTALWGKvn  vdEVGGEALGRLLVVYPWTQRFFESFgd
    2hhba             vLSPADKTNVKAAWGKvgahagEYGAEALERMFLSFPTTKTYFPHF

1lh1_dssp            ----HHHHHHHHHHHHHhhtthhHHHHHHHHHHHHH-GGGGGG-TTTtt
2lhb_dssp    ---sss------HHHHHHHHHHHHHhhhthhHHHHHHHHHHHHH-GGGGGG-GGGtt
1ecd_dssp              --HHHHHHHHHHHHHTt    tT-HHHHHHHHHHH-HHHHTT-TTTtt
4mbn_dssp             --HHHHHHHHHHHHHGg--hhhHHHHHHHHHHHHHHHHHGGGG----s
2hhbb_dssp           ----HHHHHHHHHHHTT--  hhHHHHHHHHHHHHHSGGGGGG-GGG--
2hhba_dssp            ---HHHHHHHHHHHHHhggghhHHHHHHHHHHHHH-GGGGGG-TTS

Very similar ----------1111111111111110----0111111111111111111111111--
Less similar ----------1111111111111110----0111111111111111111111111--
Post similar ----------1111111111111110----0111111111111111111111111--

<etc.>
```

The sequences are displayed, as are the DSSP secondary structures and three measures of similarity (explained at the top of the output). One can suppress the displaying of secondary structures by the option '-sec false', or can display a summary of the secondary structures (an average) by typing '-secsum true'. Try these and see. The column width can be modified by using '-columns <value>'. The remaning parameters are as for DSTAMP (see next section).

2.8.4 Pretty Alignments via ALSCRIPT

DSTAMP generates input files for GJBs ALSCRIPT program. Given a STAMP alignment file, DSTAMP can be run to create a fairly pretty alignment. Detailed descriptions of the parameters are given below. As a quick example, using the globin example,

```
dstamp -f globin.5.clean -prefix globin_align
```

will create files:

```
globin_align_als.com
globin_align_als.bloc
```

The first is a set of ALSCRIPT commands and the second is a modified alignment file. To get a pretty Postscript alignment, one needs to run alscript:

```
alscript globin_align_als.com
```

The file globin_align_als.ps will be created, and is previewable or printable on a Postscript printer. And is shown in Figure 1.

By default, residues occuring within structurally equivalent regions will be bold-faced, boxed and in upper- case, with other regions being lower-case, non-bold and in a smaller font. All secondary structure assignments will be shown below. It is possible to modify the output format (paramaters are described in a Chapter 4).

Figure 1 Globin alignment as discussed in the text.

2.8.5 Pretty Structures via MOLSCRIPT

GSTAMP can be used to display the structurally equivalences found by STAMP. It works by creating an input file for MOLSCRIPT (Kraulis, 1991) (contact Per Kraulis to obtain a copy).

As for DSTAMP, a detailed description of parameters is given later. Here is a quick example, using the first globin alignment (i.e. containing only two structures).

First one needs to generate transformed PDB coordinates using the program TRANSFORM:

```
transform -f globin.5.clean
```

This will create 2 PDB files with coordinates superimposed: 2hhbb.pdb and 2hhba.pdb.

```
gstamp -f globin.5.clean
```

This reads in the six structures and the alignment and outputs six molscript files called (domain identifier).molscript.

One must then run molscript on each of these files that one wants to display. For illustration, we will run two very distantly related globins:

```
molscript < 1lh1.molscript > 1lh1.ps
molscript < 2hhba.molscript > 2hhba.ps
```

To give the two postscript files are shown in Figure 2.

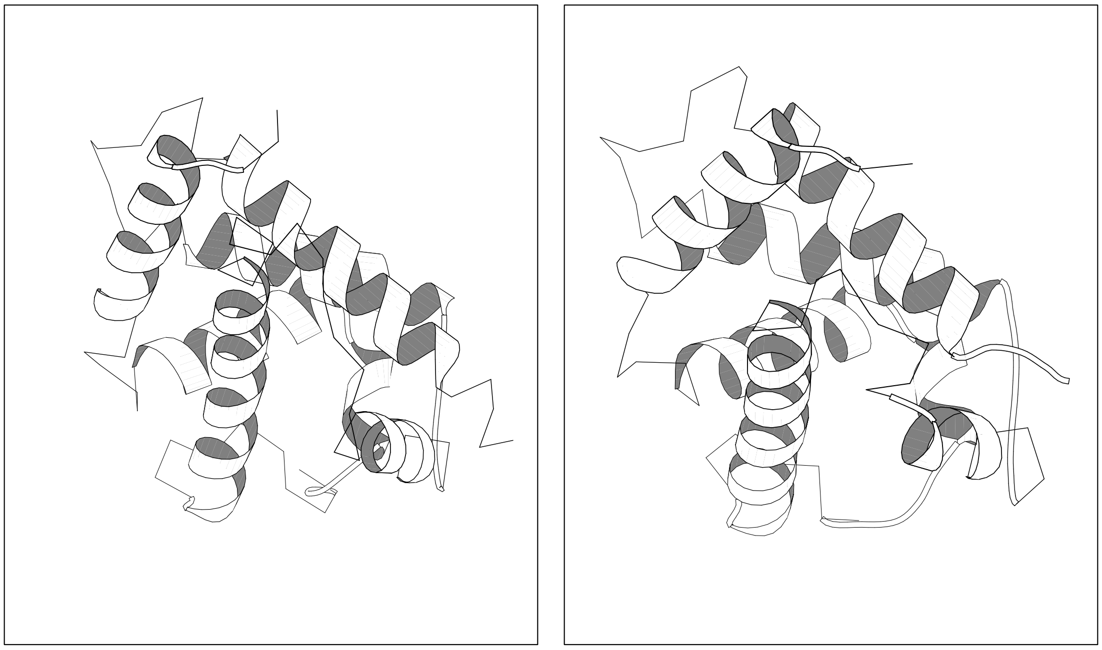

Figure 2 Superimpositions of globin 1lh1 (left) and 2hhba (right).

By default, GSTAMP will show equivalent helix, strand and coil residues as MOLSCRIPT α helix, β strand and coil, with un-equivalent regions being shown as C_α trace.

At best, GSTAMP will give only a starting point for further refinement. Invariably, one will need to modify the orientation of the image for the best view, and probably need to tweak the assignments of helix and strand to look clear; MOLSCRIPT will not work, for example, if one has very short β strands.

Chapter 3

Input and Output format for all programs

3.1 Describing domain structures

Every entry in a STAMP input file is called a 'domain'. This word is a bit of a misnomer, since these things needn't be single domains (though it is usually best to do structure comparisons at the domain level).

The problem of defining domains such that a wide variety of possibilities may be used (e.g. all the coordinates in a PDB file, one chain, bits of one chain, two chains, one chain and bits of another, etc) is solved by defining a domain by: 1) a file, 2) an identifier, and 3) a list of 'objects', from the file, to be included in the domain. An object is defined as a run of C_α coordinates, and a domain may contain more than one object.

Domains are stored in STAMP in files which may contain one or more of such domain definitions.

The format of these files must be as follows:

```
<file name> <identifier label> { <objects> }
```

or,

```
<file name> <identifier label> { <objects> [RETURN]
   R11 R12 R13   V1
   R21 R22 R23   V2
   R31 R32 R33   V3 }
```

<file name> is the full name (including path) of the PDB file in which the coordinate information is to be found. If you don't know the precise location of the file, then just

call it UNK or something (i.e. not a blank), and the programs should be able to find the appropriate PDB file using the identifier (if one can be found on your system), e.g. /usr/people/jack/pdb4mbn.ent

<identifier label> is a short name to be used by the program. eg. 4mbn1

If secondary structures are to be found by the program, then the first four letters of the identifier label should be the PDB code should correspond to the prefix used in your PDB/DSSP naming system. There should be no duplication of these, to allow for self comparison. It should contain the brookhaven four letter code first and anything else afterwards.
<objects> are coordinate descriptions, and may be one of three types:

```
1. ALL
```

all C_α 's from the file.

```
2. CHAIN X
```

only C_α 's labeled as chain X.

```
3. <chain1> <number1> <insert1> to <chain2> <number2> <insert2>
e.g. B 20 _ to B 67 P
```

only C_α 's between (and including) the two full brookhaven
residue names (chain, number, insertion code; the '_' character denotes a space)

N.B. THERE MUST BE AT LEAST ONE SPACE BETWEEN THE VARIOUS FIELDS.
Combinations of these are allowed within one domain, e.g. ' CHAIN A B 1 _ to B 65 _ '

R11 → R33 and V1 → V3 are a rotation matrix and translation vector, respectively.

Thus, a full description of three domains might look something like this:

```
/data/newpdb/pdb/pdb1ton.ent 1ton { ALL
0.9876 0.34 0.543  19.23
1.0  2.34   0.98473332  1.0
0.023  0.94 4.345     20.0 }
/data/newpdb/pdb/pdb2kai.ent 2kai_Kallikrien { CHAIN X CHAIN Y }
/data/newpdb/pdb/pdb3sgb.ent 3sgbe_SGprotease { E 20 _ to E 160 P
1.0 0.0 0.0   0.0
0.0 1.0 0.0   0.0
0.0 0.0 1.0   0.0 }
```

Note the spaces. There must be spaces separating each keyword or datum to be read, even between the braces. For example:

```
/data/newpdb/pdb/pdb3sgb.ent 3sgb_protease{E 20 _ to E 160P}
```

would not be allowed.

In the second domain (Kallikrien) the transformation will be set equal to the identity matrix with a translation of zero, since none has been supplied.

The domains must be listed at the start of a file (ie. nothing must come before them in a file), but anything may come afterwards, provided that it contains no braces (ie. { or }) unless they are on lines containing '%' in the first column.

It is possible to *reverse* the direction of an object in a domain description. For example, if one has two objects, one can reverse the direction of one or more of these by placing the word "REVERSE" in front of the object, e.g.:

```
/data/newpdb/pdb/pdb4mbn.ent { REVERSE _ 1 _ to _ 20 _ _ 21 _ to _ 120 _ }
```

3.2 Transformations

Transformations, which may or may not be included in the domain definition given above are in the sense:

```
Xnew    | R11 R12 R13 | Xold   V1
Ynew  = | R21 R22 R23 | Yold + V2
Ynew    | R31 R32 R33 | Zold   V3
```

or

```
Xnew = (R11*Xold + R12*Yold + R13*Zold) + V1
Ynew = (R21*Xold + R22*Yold + R23*Zold) + V2
Znew = (R31*Xold + R32*Yold + R33*Zold) + V3
```

If initial transformations are obtained in some other way (eg. those taken from a PDB file) they may be passed to STAMP if they are in the above format. As far as I can make out, this is the standard used in the PDB, but one can never be sure.

If no transformation is given, then the domain is assigned a unity rotation matrix and zero translation vector.

3.3 Sequence format

When necessary, STAMP programs read sequence information in NBRF (PIR) format. For example, user defined secondary structure assignment might be supplied in a file that looks like:

```
>Tonin
Tonin secondary structure  Author's assignments
----EEEEE-----EEEEEE-- <etc.> --HHHH---*
>Kallikrien
Kallikrien secondary structure -- visual inspection
----EEEEEEEE---E-EEEEE--- <etc.> --GGHHHH---*
>SGprotease
S. Griseus protease secondary structure.
----EEEEE---EEEE-EEEEEEEE--- <etc.> --GGHGHG---*
```

This is essentially NBRF (PIR) format. Note the position of the asterix. Comments must be limited to the single line between the >identifier and the start of the sequence string.

3.4 Multiple alignment format

STAMP alignment output consists first of a list of domain descriptions and relevant transformations. After this an alignment may or may not be output.

Multiple alignments are displayed as follows (see STAMPDIR/examples/globin_stamp_trans.6):

```
/data/newpdb/pdb/pdb1lh1.ent 1lh1 { ALL
   1.00000    0.00000    0.00000         0.00000
   0.00000    1.00000    0.00000         0.00000
   0.00000    0.00000    1.00000         0.00000  }
/data/newpdb/pdb/pdb2hhb.ent 2hhba { CHAIN A
   0.71639    0.34414    0.60691        19.45435
                  <Etc.>
  -0.31092   -0.94263    0.12159        68.85890  }


Alignment Score  Sc = 7.665619
Alignment length Lp = 156
RMS deviation after fitting on 116 atoms =  2.434597
Secondary structures are from DSSP

>1lh1    (cluster A) sequence
>2hhba   (cluster B) sequence
>2hhbb   (cluster B) sequence
>4mbn    (cluster B) sequence
>1ecd    (cluster B) sequence
>2lhb    (cluster B) sequence
>space
>1lh1_dssp  (cluster A) secondary structure from DSSP
>2hhba_dssp  (cluster B) secondary structure from DSSP
>2hhbb_dssp  (cluster B) secondary structure from DSSP
>4mbn_dssp   (cluster B) secondary structure from DSSP
>1ecd_dssp   (cluster B) secondary structure from DSSP
>2lhb_dssp   (cluster B) secondary structure from DSSP
#T -- '1' = used in the final fit
#P -- averaged Pij
#A -- distance between averaged CA atoms in angtroms
#G -- $P_{ij}{\prime}$ value
ABBBBB ABBBBB use  Pij      Distance $P_{ij}{\prime}$
* iteration 1
     P
     I
     V
```

```
     D
     T
     G
     S
     V
G V  A - -  -
AVHV P ---- -
LLLLLL ------ 1  0.50337  1.90006   6.98400
TSTSSS ------ 1  0.49631  2.00483   6.88900
EPPEAA HHHHHH 1  0.55533  1.89926   7.68300
SAEGDA HHHHHH 1  0.60834  1.80863   8.39600
QDEEQE HHHHHH 1  0.70134  1.64212   9.64700
AKKWIK HHHHHH 1  0.75434  1.52204  10.36000
ATSQST HHHHHH 1  0.75137  1.51092  10.32000
LNALTK HHHHHH 1  0.80831  1.36142  11.08600
VVVVVI HHHHHH 1  0.85737  1.21626  11.74600
KKTLQR HHHHHH 1  0.83537  1.27448  11.45000

                      <Etc.>

ITNKGI HHHHHH 1  0.85737  1.04393  11.74600
VVADML HHHHHH 1  0.84332  1.11847  11.55700
ILLIIL HHHHHH 1  0.81232  1.20349  11.14000
KTAAFR HTHHHH 1  0.80035  1.22529  10.97900
KSHASS HTTHHT 1  0.73137  1.29476  10.05100
EKKKKA HTTHHT 1  0.60031  1.66495   8.28800
M  Y   H  H
D  K   H  H
D  E   H  H
A  L   H  H
*
```

The '>' and '#' characters tell the routines that read alignments what is to be contained in each field. A '>' character denotes a character string which is to be displayed vertically, and a '#' character denotes a string of numbers to be displayed separated by spaces. Thus in the above example we have 13 character strings vertically (6 amino acid sequences, 1 string of spaces and 6 DSSP assignments) and 6 numeric fields (corresponding to various details from STAMP) specified. The actual alignment will be contained within '*' characters as shown. Accordingly, no occurance of '>', '#' and '*' characters should occur outside of these contexts.

The As and Bs just above the '*' symble refer to the members of the two cluster (branches) which are brought together during this alignment.

Briefly, the numeric fields are:

#T 1 or 0, 1 shows those residues used to determine the fit of the two sets of structures.

#P averaged Rossmann and Argos Pij value

#A distance between averaged C_α atoms

#G corrected Pij value ($P_{ij'}$)

Note that the program POSTSTAMP adds two new fields:

#B 1 if all pairiwse $P_{ij} \geq$ the user defined minimum, 0 otherwise

#R the total number of pairwise comparisons having $P_{ij} \geq$ the cutoff out of $N \times (N-1)/2$

3.5 Output from SCANS

Output from STAMP scans consists of a list of domains and a corresponding set of scores, lengths and other numbers that can be used to sort and understand the output.

The format is as follows (see examples/1cmsN_stamp_scan.trans):

```
% Output from STAMP scanning routine
%
% Domain 1cmsN was used to scan the domain database:
%  ac_prot.domains
% 2 fits were performed
% Fit 1 E1= 20.000, E2=  3.800, CUT=  1.000
% Fit 2 E1=  3.800, E2=  3.800, CUT=  4.500
% Approximate fits (alignment from N-termini) were performed
%   at every 5 residue of the database sequences
% Transformations were output for Sc=  2.000
%
% Domain used to scan
# Sc= 10.000 RMS=  0.01  Len= 999 nfit= 999 Seqid= 100.00 Secid= 100.00 q_len=  175 d_len=  175
          n_sec= 100 n_equiv 999 fit_pos= _  0 _
/disk3/pdb/pdb1cms.ent 1cmsN { _ 1 _ to _ 175 _  }
# Sc=   9.744 RMS=   0.000 len=  174 nfit=  174 seq_id= 99.43 sec_id= 94.86 q_len=  175 d_len=  175
         n_sec=   18 n_equiv=  173 fit_pos= _   1 _
/disk3/pdb/pdb1cms.ent 1cmsN_1 {  _ 1 _ to _ 175 _
   1.00000    0.00000    0.00000         0.00000
   0.00000    1.00000    0.00000         0.00000
   0.00000    0.00000    1.00000         0.00000  }
# Sc=   2.749 RMS=   2.352 len=  204 nfit=   63 seq_id=  8.00 sec_id= 41.14 q_len=  175 d_len=  148
        n_sec=   13 n_equiv=   58 fit_pos= _ 176 _
/disk3/pdb/pdb1cms.ent 1cmsC_1 {  _ 176 _ to _ 323 _
  -0.98340   -0.10624   -0.14708        36.75176
<etc.>
```

(note that the lines begining by '#' symbols have been wrapped here) '%' denotes a comment, and '#' denotes numbers corresponding to the domain description described below (both will be ignored by all programs except for SORTTRANS, which uses the '#' fields to sort and interpret the data.

'Sc' is the STAMP Score for the comparison of the query to each database sequence. 'RMS' is the RMS difference between equivalenced atoms, 'len' is the alignment length, 'nfit' is the number of atoms used during the final fit of the two domains, 'seq_id' and 'sec_id' are the sequence and secondary structure identities, 'q_len' and 'd_len' are the lengths of the query and database structure (in residues), 'n_sec' is the number of equivalenced secondary structures, and 'n_equiv' are the number of residues found within stretches of 3 or more having $P'_{ij} \geq 6$. These fields are used during any run of SORTTRANS to sort and remove redundant/poor superimpositions. 'fit_pos' is the brookhaven numbering of the position in the database sequence to which the query's N-terminal end was aligned for the initial fit. The transformation supplied is that for the superimposition of the database structures onto the query.

3.6 Output to standard output or log file

STAMP now keeps fairly quiet during its running, updating the user only after a pairwise/treewise/scan comparison has been compeleted. You can get lots of other output by using the -V (verbose) option. If you want a lot of output to be written to a file instead of the standard output, you can use -V in conjunction with -logfile <file name>. I would go into a detailed description of this output, but I am getting tired.

Chapter 4

Summary of STAMP parameters

4.1 Main program (STAMP)

The format for running STAMP is:

```
stamp -l <listfile> -s -o <output file> -P <parameter file>
      -n <1 or 2 fits> -d <database file for scans>
      -slide <slide value>
      -pen1 <gap penatly 1> -pen2 <gap pentalty 2>
      -prefix <output file prefix>
      -V
      -rough
      -cut
      -<parameter> <value>
```

If you have old STAMP parameter files, they can be read by using the command stamp -P <parameter file>. This means that the old file can be read in exactly the same way as for version 2.0.

In general, all commands can be specified by -<parameter> <value>. For example, '-first_pairpen 0.5'. However, I have made some abbreviations for frequently used commands, these are:

```
-l <list file>           same as -listfile <list file>
-o <output file>         same as -logfile <output file>
-n <1 or 2>              same as -npass <1 or 2>
-pen1 <gap penalty 1>    same as -first_pairpen <gap penalty 1>
-pen2 <gap penalty 2>    same as -second_pairpen <gap pentalty 2>
-prefix <output prefix>  same as -transprefix <output prefix>
-s                       same as -scan true
-d <database file>       same as -database <database file>
-slide <slide parameter> same as -scanslide <slide parameter>
-cut                     same as -co true
-rough                   same as -roughfit true
```

Default parameters are always looked for in the file STAMPDIR/stamp.defaults. You can personalise this is as you like, but I would recommend using the defaults, unless you have a thorough understnading of the method. The values described below were essentially chosen to mimick the successful and well-tested parameters (Russell & Barton, 1992).

I would recommend using the command line parameters. The commands, and their arguments are given below. The command line parameters are case insensitive. To use a parameter one need only type '-<parameter> <value>' or use on of the short forms listed above.

STAMP can also be supplied with a parameter file. Parameters in a parameter file can be supplied in the format:

```
<Parameter> <Value> <Optional Comments> [return]
```

eg.

```
PAIRWISE Yes   Perform pairwise calculations
E1 3.8
E2 3.8
CUTOFF 4.5
```

Since the program is written in C, the input is read in an open format. Generally, data are expected to be separated by spaces or return characters. The number and position of spaces, tabs and returns generally should not matter with the exception of PDB format, which is read as the fixed format described in the brookhaven documentation.

The possible parameters are listed below. Strings, characters, floats and integers are as expected (though strings may not contain spaces). Boolean variables may be set by any of the following:

```
TRUE  == TRUE, True, T, true, Yes, YES, yes, Y, 1
FALSE == FALSE, False, F, false, No, NO, n, 0
```

LOGFILE <string>
This is the file into which the log is to be written. If 'stdout' is supplied then the information is written to the standard output.
Default LOGFILE = stdout

LISTFILE <string> (or '-l <string>' or '-f <string>')
This is the name of a file that contains the location and description of the domains to be analysed and, if desired, an initial transformation.
Default LISTFILE = domain.list

SEC <integer>
This must be set to 0 (no secondary structure assignment) or to one of the following values:

SEC = 1 Kabsch and Sander's DSSP output. This program, which calculates secondary structure based on hydrogen bonding criteria (Kabsch & Sander, 1983) is available from the EMBL fileserver.
SEC = 2 Richards and Kundrot DEFINE output. This program, which calculates secondary structure based on local distance criteria (Richards & Kundrot, 1988), is available from Fred Richards. The required format is from an adapted version of the program written by RBR. If you are interested in making use of these definitions, you must acquire the program, then refer to RBR for details on how to interpret the output (ie. it is a bit finicky).
SEC = 3 Secondary structure summary format. A string of residue by residue secondary structure assignments for each domain is to be read in from SECFILE in the format specified in the previous chapter.

Note that it is not possible to mix assignments. This is probably not a very realistic thing to do anyway, since assignments can differ substantially. If you really want to do this, then the only possible way is to set SEC = 3, and define each secondary structure independently in SECFILE.
Default SEC = 1 (for DSSP).

SECFILE <string>
The file from which user specified secondary structure assignments are to be read (ie. SEC = 3 only).
Default SECFILE = stamp.sec

PAIRWISE <boolean>
If TRUE, then pairwise comparisons are to be performed for each possible pair of domains described in LISTFILE. A matrix of pairwise (S_c) scores will be output (to MATFILE).
Default PAIRWISE = TRUE

N.B. Many of the following parameters also apply to TREEWISE and SCAN comparisons. For clarity they are discussed here in the PAIRWISE comparison context.

NPASS <1 or 2> (or '-n <1 or 2>')
Whether one or two fits are to be performed. The idea is that the initial fit can be used with a conformation biased set of parameters to improve the initial fit prior to fitting using distance and conformation parameters. The parameters described below are called 'first_' and 'second_' accordingly. When NPASS = 1, then only the 'second_' (or unprefixed) parameters are used. Default NPASS = 1

SW <0 or 1>
If set to 0, then the entire M x N matrix will be calculated and used during the Smith Waterman path finding routine. If set to 1, then a corner cutting routine will be used (to save time). Note that corner cutting will nullify many of the parameters specified in (Russell & Barton, 1992), and is only recommended for SCAN mode. Accordingly, corner

cutting parameters are specified below (after SCAN).

PAIRPEN <float> (or '-pen1 <float>'/ '-pen2 <float>')
(first_PAIRPEN)
(second_PAIRPEN)
Smith-Waterman gap penalty to be used during the fitting. second_PAIRPEN and PAIRPEN are equivalent. (PAIRPEN is also relevant to treewise fitting)
Defaults PAIRPEN = second_PAIRPEN = 0.0 first_PAIRPEN = 0.0

E1 <float>
E2 <float>
(first_E1,first_E2)
(second_E2,second_E2)

Rossmann and Argos parameters to be used during the fitting. Rossmann and Argos suggested that E1 = E2 = 3.8 lead to good superimpositions, and further suggested that E1 = 20.0 and E2 = 3.8 would relax the distance requirement, and allow poor initial superimpositions to be improved. The defaults are defined accordingly.
Defaults:
E1 = second_E1 = 3.8
E2 = second_E2 = 3.8
first_E1 = 20.0
first_E2 = 3.8

I would not recommend modifying these parameters, since I really don't know what changing them will do. If it ain't broke, don't fix it as my father would say.

NA <float>
NB <float>
NASD <float>
NBSD <float>
NSD <float>
NMEAN <float>

Parameters used to define $P_{ij'}$ and S_c values. These are defined in (Russell & Barton, 1992). I wouldn't change these.

Defaults:
NA = -0.9497
NB = 0.6859
NASD = -0.4743
NBSD = 0.01522

NMEAN = 0.02
NSD = 0.1

CUTOFF <float>
(first_CUTOFF)
(second_CUTOFF)
This is the minimum $P_{ij}\prime$ value allowed for atoms to be used for a least squares fit. Equivalences above this value will be used to determine a transformation and RMS deviation.
Defaults:
CUTOFF = second_CUTOFF = 4.5
first_CUTOFF = 1.0

PAIRALIGN <boolean>
If true, then each final pairwise alignment will be output to the log file.
Default PAIRALIGN = FALSE

COLUMNS <integer>
Number of sequence positions to be displayed per line when either PAIRALIGN, SCANALIN or TREEALIGN is set to TRUE.
Default COLUMNS = 80

SCORETOL <float>
This is the percent Sc difference that will result in convergence being reached. In other words, if $100 \times abs|S_c - S_{c,old}|/S_{c,old} \leq$ SCORETOL then the fitting will be considered done.
Default SCORETOL = 1.0

MAXPITER <integer>
The maximum number of iterations allowed during the pairwise comparisons. This prevents a particular fit, which jumps between two values rather than converging, from lasting indefinitely.
Default MAXPITER = 10

MATFILE <string>
This is the file which contains an upper diagonal matrix consisting of the pairwise Scores (either 1/RMS, or Sc) for each comparison. It may then be used to derive a tree, if desired, for treewise analysis.
Default MATFILE = <stamp_prefix>.mat

ROUGHFIT <boolean> (or '-rough' to set to TRUE)
If set to TRUE, then an initial rough superimposition will be performed by aligning the

N-terminal ends of the sequences and fitting on whatever atoms this process equivalences. Probably this is too crude for structures that differ quite a bit, but if they are very similar, one can use this to avoid having to perform a multiple sequence alignment.

TREEWISE <boolean>
If TRUE, then a treewise comparison is performed by following a derived hierarchy. Reads in the matrix file specified (either created by PAIRWISE or some other method), derives a tree (dendrogram), and does a tree-based alignment.
Default TREEWISE = TRUE

TREEPEN <float>
(first_TREEPEN)
(second_TREEPEN)
Value subtracted from the $P_{ij'}$ matrix at positions where a residue is to be aligned with a gap. For details see (Russell & Barton, 1992).
Defaults TREEPEN = second_TREEPEN = 0.0 first_TREEPEN = 0.0

MAXTITER <int>
As for MAXPITER, but applied to the treewise case.
Default MAXPITER = 10

TREEALIGN <boolean>
As for PAIRALIGN, only for treewise comparisons.
Default TREEALIGN = TRUE

STAMPPREFIX <string> (or '-prefix <string>')
This is the name of the family of files that will be produced from a multiple alignment. The files will be named STAMPPREFIX.<N>, where N is the number of the cluster after which the alignment has been derived. There are always one fewer clusters than their are domains being compared.
Default STAMPPREFIX = 'stamp_trans'

SCAN <boolean> (or simply '-s' to set true)
If TRUE, then SCAN mode is selected. TREEWISE and PAIRWISE are set to FALSE. The first domain described in LISTFILE (the query) is used to scan all the domains listed in DATABASE. The parameters for scanning are described below. The output of a SCAN run appears in the file called STAMPPREFIX.scan.
Default SCAN = FALSE

DATABASE <string> (or -d <string>)
The list of domains to be compared with the query during a scan.
Default DATABASE = domain.database

MAXSITER <int>
As for MAXPITER and MAXTITER, but for scanning. Equivalent within the program to MAXPITER.
Default MAXSITER = 10

SCANALIGN <boolean>
As for PAIRALIGN and TREEALIGN, but for scanning. Equivalent within the program to MAXPITER.
Default SCANALIGN = FALSE

SCANSCORE <integer>
Specifies how the Sc value is to be calculated. This depends on the particular application. The values are described in the first chapter.

As a general rule of thumb, use SCANSCORE=6 for large database scans, when you are scanning with a small domain, and wishing to find all examples of this domain – even within large structures. Use SCANSCORE=1 when you wish to obtain a set of transformations for a set of domains which you know are similar (and have defined fairly precisely as domains rather than the larger structure that they may be a part of).
Default SCANSCORE = 6

SKIPAHEAD <boolean>
If set to TRUE, then the program will skip over all hits. In other words, if a similarity is found with a particular starting fit position, then the next fit position will be the last residue of the similar region. This is not always desireable, since there can be more than one hit within repetetive structures, such as α/β barrels.
Default SKIPAHED = TRUE

SCANCUT <float>
If SCANMODE = 1, then Sc must be >= SCANCUT in order for a transformation to be output.
Default SCANCUT = 2.0

SCANSLIDE <integer> (or '-slide <integer>') This is the number of residues that a query sequence is 'slid' along a database sequence to derive each initial superimposition. Initially, the N–terminus of the query is aligned to the 1st residue of the databse, once this fit has been performed and refined, and tested for good structural similarity, the N–terminus is aligned with the 1+<SCANSLIDE>th position, and the process repeated until the end of the database sequence has been reached.
Default SCANSLIDE = 5

SCANTRUNC <boolean>
If TRUE, then sequences from DATABASE that are more than SCANTRUNCFACTOR x

the length of the query sequence are truncated to this size. This saves a lot of CPU time, as comparisons between things that are vastly different in size are largely meaningless. Moreover, since most scans will be done with discrete domains, then this allows separate domains in large proteins to be compared to the query separately.
Default SCANTRUNC = TRUE

SCANTRUNCFACTOR <float>
The largest size of sequence which may be compared to the query sequence (expressed as SCANTRUNCFACTOR x query sequence length). Structures in the DATABASE that are larger than this will be truncated to this size if SCANTRUNC = TRUE.
Default SCANTRUNCFACTOR = 2.0

SLOWSCAN <boolean>
If set to TRUE, then the SLOW method of getting the initial fits for scanning will be used (See chapter 1).
Default SLOWSCAN = FALSE

MIN_FRAC <float>
This is the minimum ratio of database length/query length to be allowed. In other words, if a database structure is too small (ie. if databaselength/query length < MIN_FRAC), then the comparison will be skipped. Whether to use this or not depends on whether or not one is interested in sub alignments where only a part of the query structure is used. The default implies that all comparisons will be performed.
Default MIN_FRAC = 0.001

SECSCREEN <boolean>
If TRUE, then an initial comparison between query and DATABASE secondary structure assignments (if available) is performed. A secondary structure distance is defined by:

$$D_{sec} = \sqrt{(\|Q_h - D_h\|^2 + \|Q_b - D_b\|^2)}$$

where Q_h and Q_b are the percent of Helix and Beta structure in the query, and D_h and D_b are the same for the database sequence. If Dist is larger than a threshold (SECSCREEN-MAX) then the comparison will be ignored.
Default SECSCREEN = true

SECSCREENMAX <float>
This is the maximum value of Dist (above) tolerated. If Dist is larger than SECSCREEN-MAX then the comparison is ignored. For screening to be effective, it is important that secondary structure assignments are accurate (preferably done using the same program).
Default SECSCREENMAX = 60.0 (this is very lenient; 40 is usually safe)

CCFACTOR <float>
Corner cutting factor. This is approximately the maximum number of gaps to be tolerated in any pairwise comparison. Only used if SW = 1. For a more detailed explanation, refer to (Sankoff & Kruskal, 1983) (pp 279 – 281).
Default CCFACTOR = 30.0

CCADD <boolean>
If TRUE, then the difference between query and database sequence lengths will be added to CCFACTOR. Probably this is only realistic when SCANTRUNC is set TRUE.
Default CCADD = FALSE

PRECISION <integer>
Since STAMP works as much as possible with integers, this is what all floating point values are multiplied by during conversion. A value of 1000 has never presented us with any problems.
Default PRECISION = 1000

MAX_SEQ_LEN <integer>
The maximum length of alignment tolerated. The program ought to inform you when this value is surpassed.
Default MAX_SEQ_LEN = 1500

4.2 Summary of parameters for other programs

4.2.1 PDB checker (PDBC)

This is a simple program which looks for the location of a four letter PDB code (using the list of directories, prefixes and suffixes supplied in the file ./pdb.directories or if this does not exist STAMPDIR/pdb.directories) There are several options:

```
pdbc -q <four letter code>
```

will mearly report useful information (number of atoms, the occurance of HETATM, resolution, etc.) about each chain found in the PDB file which corresponds to the four letter code supplied.

```
pdbc -d <four letter code>[<chains to be considered>]
```

this outputs a domain description (or more than one if more than one chain is given. Sequential use of this program can be used to create a list of domains for use in scanning.

```
pdbc -m <four letter code>
```

this will just report the location of PDB and DSSP files. Good for a quick test of whether PDB codes can be found in the files specified in STAMPDIR.

Output is to standard output.

4.2.2 PDBSEQ

This program takes a list of protein domains (ie. a LISTFILE) and outputs a series of sequences derived from the described PDB files. The format is:

```
pdbseq -f <domain file> [-min <val> -max <val> -foramt <fasta>]
```

'-min/max <val>' specify the minimum/maximum sequence length to be output. If the length of a sequence is less than min or greater than max, the sequence will be skipped (useful particularly if one wants to ignore very short PDB sequence, such as peptide inhibitors, etc.). The output is in NBRF (PIR) format, and is written to the standard output. Using '-format <fasta> will make the output as FASTA format.

4.2.3 ALIGNFIT

ALIGNFIT takes a multiple sequence alignment of proteins of known 3D structures and uses it to superimpose them. It requires two files: an AMPS multiple sequence alignment (block format), and a domain description file. An optional parameter file may be supplied; if none is given the program simply uses default parameters.

The format is:

```
alignfit -a <AMPS file> -d <domain file>
   (-P <optional parm file> -<parameter> <value>)
```

-P can be used to read in an old ALIGNFIT parameter file (version 3.0 and earlier) The possible parameters, and their defaults are (names are case insensitive):

PAIRWISE <boolean>
If TRUE, then pairwise comparisons will be performed to derive a matrix (MATFILE).
Default PAIRWISE = TRUE

TREEWISE <boolean>
If TRUE, then treewise comparisons will be performed to derive a final transformation.
Default TREEWISE = TRUE

MATFILE <string>
The file into which the results of PAIRWISE are output.
Default MATFILE = alignfit.mat

MAX_SEQ_LEN <integer>
The maximum length of alignment to be tolerated.
Default is 3000

For most purposes, the default parameters should suffice. Not that CLUS2BLC and MSF2BLC convert CLUSTAL and MSF formats to block format, so that one can use alignments created using other programs (e.g. PILEUP, etc.) as a starting point for superimposition.

4.2.4 VER2HOR

This program provides a horizontal alignment given a STAMP alignment file (i.e. a text alignment written to the standard output). The format is:

```
ver2hor -f <stamp alignment file> [ -columns <width> ]
```

'-columns' specifies the number of columns to be used in the alignment output. This program is explained by example in the Worked Examples chapter. It also accepts most DSTAMP (see below) commands (i.e. those that are relevant to text output) from the command line.

4.2.5 DSTAMP

This program provides input for ALSCRIPT (Barton, 1993), GJB's program for the display of multiple sequence alignments. To get a copy of this program, refer to GJB at the above address.

The format is:

```
dstamp -f <STAMP alignment file> -prefix <output prefix>
   (-P <optional parm file> -<parameter> <value>)
```

where <parameter> is one of the many parameters described below. The new command line argument -P reads in parameter files, so if you have old DSTAMP files, they can still be read in this way.

The parameters for DSTAMP, and their defaults, are (parameter names are case insensitive):

TYPE <character> (or -c <character>)

The type of STAMP data to be used (ie. the first letter that occurs after the '#' characters in STAMP multiple alignment output). Default TYPE = 'G'

CUTOFF <float> (or -t <float>)
The minimum (or maximum in the case of RMS deviation) value to make a position considered as reliably aligned.
Default CUTOFF = 6.0

WINDOW <integer> (or -w <integer>)
The minimum length of a stetch of reliable regions to be allowed.
Default WINDOW = 3

SEC <boolean>
If TRUE, then display all secondary structure assignments.
Default SEC = TRUE

SECSUM <boolean>
If TRUE, then display a simple summary of secondary structure assignments rather than all of them. The 'SEC' options below should apply this summary.
Default SECSUM = FALSE

BOXSEQ <boolean>
If TRUE, then box reliably aligned sequences.
Default BOXSEQ = TRUE

BOXSEC <boolean>
As for BOXSEQ, but for secondary structure assignments.
Default BOXSEC = TRUE

BOLDSEQ <boolean>
If TRUE, then reliably aligned sequence positions will be in boldface.
Default BOLDSEQ = TRUE

BOLDSEC <boolean>
As for BOLDSEQ, but for secondary structure assignments.

CASESEQ <boolean>
If TRUE, then reliably aligned sequence positions will be in upper case, unreliably aligned in lower case.
Default CASESEQ = TRUE.

CASESEC <boolean>
As for CASESEQ, but for secondary structure assignments.

Default CASESEC = TRUE

SMALLSEQ <boolean>
If TRUE, then reliably aligned sequence positions will be large, unreliably aligned positions will be small.
Default SMALLSEQ = TRUE

SMALLSEC <boolean>
As for SMALLSEQ, but for secondary structure assignments.
Default SMALLSEC = TRUE

POINTSIZE <float>
The pointsize from which the entire alignment will be sized.
Default POINTSIZE = 8.0

VERBOSE <boolean>
If TRUE, then write a lot of comments to the standard output.
Default VERBOSE = FALSE

The output is a new alignment (block) file and an ALSCRIPT command file.

4.2.6 SORTTRANS

This program takes the output from a scan, and cleans and sorts the output. It removes repeated transformations by a simple least squares comparison of the matrices and vectors for those transformations which have the same identifier.

The format is:

```
sorttrans -f <scan output file> -s <keyword> <cutoff> [-t -i]
```

-f reads output from STAMP scanning, -s tells the program how to sort the output. The keyword tells which method to use. There are 8 possible keywords:

```
Sc              sort by Sc
rms             RMS deviaition
nfit            number of fitted atoms
len             alignment length
frac            nfit/len
q_frac          nfit/q_len (q_len = length of query structure)
d_frac          nfit/d_len (d_len = length of database structure)
n_sec           number of equivalent secondary structure elements
seq_id          percent  sequence identity
sec_id          percent secondary structure identity
```

sorted transformations are written to the standard output.
The option -t ==> ignore identifiers. This means that domains will be ignored if they have similar transformations and the same filename (i.e. assumes that if the same transformation is applied to the same file, we are dealing with the same superimposition).

4.2.7 TRANSFORM

This program takes a transformation file, either from ALIGNFIT, STAMP, or SORTRANS and outputs a series of PDB format files containing the specified coordinates transformed as specified in the given file.

The format is:

```
transform -f <transformation file> [ -g -het -hoh -o <output file> ]
```

options:

'-het' Include hetero atoms. Hetero atoms are normally not included in the output.
'-hoh' Include waters.

'-g' Graphics output. This mode puts all transformed coordinates into a single PDB file, and labels the chains for domains sequentially (after their order in the transformation file) with A, B, C.. etc. This allows fast analysis of the structures graphically (i.e. using Rasmol) since one need only colour each chain a different colour to see the superimposition. The default file for writing the coordinates using this mode is 'all.pdb', but this can be changed (see below).

'-o <output file>' When using '-g', this option allows the specification of a file to contain the transformed coordinates. The default is 'all.pdb'

The PDB files will be named <identifier>.pdb (except when running using the '-g' option).

4.2.8 PICKFRAME

It is often the case that one wishes a particular protein structure to be the 'parent' of the superimposition, i.e. the structure that is un-transformed. Accordingly, the program PICKFRAME allows one to select a particular reference frame for a particular domain identifier. Given a transformation file and an identifier, the program will set the selected identifier's transformation to the unit matrix and zero vector, and transform the other structures accordingly. This is useful if one wishes to combine different transformation files (i.e. if a multidomain protein has two domains, with each being similar to a separate domain).

The format is:

```
pickframe -f <transformation file> -i <domain identifier>
```

The output will be to the standard output (i.e. one need just pipe the results into a file).

This program is very useful if one wishes to superimpose STAMP results for two different domains from the same protein. Since one can just make all transformations relative to the PDB file containing the two domains, and then combine the output into one transformation file.

4.2.9 AVESTRUC

For various reasons, it is often useful to derive 'average' structures (i.e. for homology modelling, molecular replacement search objects, etc.). STAMP output provides an obvious starting point for obtaining an average structure. AVESTRUC reads in a STAMP alignment file, and generates another PDB file containing averaged coordaintes (either as C alpha or as a polyalanine structure).

The format is:

```
avestruc -f <STAMP alignment file>
      [ -polyA -c <STAMP char> -t <threshold> -w <window> -aligned ]
```

'-f' specifies the file to be considered. Note that this MUST BE a STAMP alignment file, containing both transformations and a sequence alignment. It will not work on transformation files lacking sequence alignment data or STAMP data.

'-polyA' generate polyalanine model, the default is a C alpha model

'-c <STAMP char>' '-t <threshold>' '-w <window>'
these three parameters tell the program how to define structurally equivalent residues. 'STAMP char' is the label of the STAMP field specified by the '#' character in the alignment file. 'threshold' is the minimum (or maximum in the case of RMS deviation) value of the specified STAMP parameter tolerated, and 'window' tells the minimum number of residues over which this must be true for structural equivalence. This is less complicated than it sounds.

The default is as described in (Russell & Barton, 1992):
STAMP char = 'G' (i.e. $P_{ij'}$)
threshold = 6.0
window = 3
(i.e. stretches of three or more residues having $P_{ij'} > 6.0$ are considered equivalent)

'-alignment' this flag negates the effect of the previous three flags, and defines positions as equivalent if all structures are present at a position (i.e. positions not containing any gaps are deamed equivalent). Probably this is useful only for highly similar 3D structures.

4.2.10 GSTAMP

Like DSTAMP, this program takes STAMP output and translates it in to input for another program, namely Per Kraulis' program MOLSCRIPT. The program allows one to create multiple molscript files (i.e. one for each structure in the STAMP alignment file), or a single molscript file for an average structure. Appropriate PDB files for these alternatives must be generated by using TRANSFORM and AVESTRUC, respectively, prior to running MOLSCRIPT.

When multiple structure are considered, structurally equivalent regions (specified as for AVESTRUC) are shown as MOLSCRIPT helix, strand or coil. Non-structurally equivalent regions are shown as C_α trace. For an example of how this looks, see Figure 1 (Russell & Barton, 1994) or Figure 1 in (Russell, 1994).

The rest is up to you. Once MOLSCRIPT input files have been generated, they can be modified to suit your particular display needs (i.e. using colour, etc.).

The format is:

```
gstamp -f <STAMP alignment file>
   [ -c <STAMP char> -t <threshold> -w <window> -aligned -a -cons ]
```

-f, -c, -t, -w and -aligned is as for AVESTRUC and DSTAMP.

-a specifies that an average structure is to be used.

-cons specifies how the secondary structures are to be define in the MOLSCRIPT files. By default, structures are displayed as helix or strand only if *all* structures are helix or strand at the positions. '- cons' means that structures are displayed as helix or strand if the majority of structures are helix or strand at the positions. In both cases, the remaining structures are drawn as 'coil'.

BUG: sometimes GSTAMP will output single residue strands for Molscript input. It is therefore necessary to modify the Molscript output to correct the odd mistake (single residue strands produce funny pictures in my version of MOLSCRIPT — try it and see).

4.2.11 STAMP_CLEAN

This program allows you to tidy up gaps that are not meaningful in the context of a multiple sequence alignment derived from structure. In other words, regions that are not similar across all members of a structural family can be 'cleaned' to remove isolated residues aligned in the middle of nowhere. Note that one doesn't always want to do this (since the sub-alignments can be meaningful). I just find this useful if I am preparing a figure for publication or something.

The format is:

```
stamp_clean <stamp alignment file> <minimum segment length> > <output file>
```

The <minimum segment length> is the minimum number of residues that is to be considered significant. I always use 3, since this means that short stretches of 1-2 residues that are surround by gaps (i.e. in any sequence) are 'cleaned'. Try it and see what I mean.

4.2.12 Converting alignment formats (MSF2BLC, CLUS2BLC)

Two command line driven programs can be used to convert MSF or CLUSTAL NBRF-PIR alignment format to the AMPS block format used by STAMP. One need only give these programs the name of the MSF/CLUSTAL alignment file and the name of a new file to store the converted alignment.

Chapter 5

Installation

5.1 Compiling/running

STAMP was developed on a Sun SPARCstation, and consequently may encounter difficulties running on other systems. UNIX systems on which I have installed STAMP include Silicon Graphics Indigo and Evans and Sutherland ESV systems.

Most of STAMP was written in C, thus STAMP requires an ansi-C compiler (e.g. gcc) for installation. STAMP requires fortran library routines.

STAMP should be received as a compressed tar file, so must be uncompressed and detarred to expose all files and directories.

On a SUN, Silicon Graphics or Dec-Alpha:

```
uncompress STAMP.tarfile.Z
tar xvf STAMP.tarfile
cd stamp.4.0
BUILD <sgi/sun/osf>
```

should work.

Note that there are several precompiled executables in the distribution. Files found in the directories bin/sgi, bin/sun and bin/osf store Silicon graphics (IRIX 5.2) Sun (SUNOS 4.1.3) and Dec–alpha (OSF/1 V3.0) respectively.

The directory src/ contains makefiles that I have made for sgi, sun and osf systems (e.g. src/makefile.osf). If you wish to compile it on a different machine, then copy one of these and make the appropriate adjustment.

Modifications may need to be made to the makefile to make it specific to your system.

FORLIB may need to be set to the file containing the appropriate fortran libraries (this differs from machine to machine). Typing 'make' in the directory into which these files have been copied should result in the creation of all the object files and executables. Once the executables are made, they will be put into the directory bin/<sgi/sun/osf>, and these can then either be included in your path name, or linked/copied to some central directory, such as /usr/local/bin.

5.2 Setting up STAMP files

To use STAMP, all users must define the environment variable STAMPDIR, and set it equal to the directory where you have done the installation. Users must also have STAMPDIR in their path. The logical thing is to make modifications to one's .cshrc file to this end.

STAMP reads PDB coordinate information and DSSP information in standard format. Thus, you really must have copies of PDB and DSSP output if you wish to make full use of the program (though DSSP is not, strictly speaking, required).

You must create files in the STAMP directory for reading by the program that are specific to your system. These files must be in STAMPDIR/pdb.directories. This file contains a description as to where possible PDB files may be found. The format is:

```
<directory>  <prefix>  <suffix> [RETURN]
```

For example:

```
/data/newpdb/pdb/           pdb .ent
/data/newpdb/prerelase/     pdb .ent
/usr/people/jack/extrapdb/  _ .pdb
./                          myprefix .pdb
```

A four letter code is meant to go between the suffix and prefix. For example, the file corresponding the PDB code 4mbn might be found in the file /data/newpdb/pdb/pdb4mbn.ent. A '_' character in the suffix or prefix field denotes no prefix or suffix. When a file is to correspond to a four letter code is to be found, STAMP routines will try each of the specifications in turn, and will use the first one found.

dssp.directories contains a description as to where possible DSSP files may be found. The format is as for pdb.directories, e.g.

```
/data/newdssp/  _ .dssp
/data/dssp/     _ .dssp
```

For example, the DSSP file for 4mbn might be found in the file /data/newdssp/4mbn.dssp

5.3 Getting other programs

There are several other programs that are useful to have when using STAMP:

DSSP – Definition of Secondary Structure in Proteins, Kabsch & Sander. Contact
Dr Chris Sander
E.M.B.L.
Meyerhofstr. 1
D-69117 Heidelberg
Germany
Tel: +49(6221)387361
Fax: +49(6221)387517
or see the WWW page: http://www.sander.embl-heidelberg.de/dssp/

DEFINE – Defines secondary structure in proteins, and can be used on PDB files containing only C_α atoms (DSSP won't). To obtain a copy, contact:
Dr Fred M. Richards
Department of Molecular Biophysics and Biochemistry
Yale University
P.O. Box 6666
260 Whitney Avenue
New Haven, CT 06511
USA

ALSCRIPT – displays alignments in PostScript format, contact GJB (see address above)
WWW page: http://geoff.biop.ox.ac.uk/

MOLSCRIPT – displays PDB structures in PostScript format, contact:
Dr Per Kraulis
Center for Structural Biochemistry
Karolinska Institute
Novum
S-141 57 Huddinge
SWEDEN
Tel: +46 8 608 9266
Fax: +46 8 608 9290
email: pjk@oyster.csb.ki.se
WWW page: http://www.unige.ch/crystal/prg-molscript.html

Chapter 6

Some of our studies involving STAMP

STAMP has been invovled in numerous published studies. Two novel similarities uncovered by STAMP have appeared in the literature: the similarity between the SH2 domain and domain II of E. coli biotin operon protein (Russell & Barton, 1993*a*); and the similarity between HIV matrix protein p17 and Interferon gamma (Matthews *et al.*, 1994).

STAMP has also aided several other investigations into protein structure. STAMP alignments have been used to determine the best accuracy of secondary structure prediction from multiple sequence alignment (Russell & Barton, 1993*b*). It has been used to investigate the conservation of various protein structural features across structural similar (but apparently non-homologous) proteins (Russell & Barton, 1994) and has been used for several investigations into protein domain structure (Russell, 1994; Sternberg *et al.* , 1995; Siddiqui & Barton, in preparation).

STAMP has also proved extremely useful when assessing the results of protein structure prediction by fold recognition (Russell *et al.* , 1995; Russell *et al.* submitted).

We hope you find it as useful as we have.

Bibliography

Barton, G. J. (1993). *Prot. Eng.* **6**, 37–40.

Barton, G. J., Barford, D. A. & Cohen, P. T. (1994). *European Journal of Biochemsitry,* **220**, 225–237.

Kabsch, W. (1978). *Acta crystallographica,* **A34**, 827.

Kabsch, W. & Sander, C. (1983). *Biopolymers,* **22**, 2577–2637.

Kraulis, P. J. (1991). *J. App. Cryst.* **24**, 964–950.

Matthews, S., Barlow, P., Boyd, J., Barton, G., Russell, R., Mills, H., Cunningham, M., Meyers, N., Burns, N., Clark, N., Kingsman, S., Kingsman, A. & Campbell, I. (1994). *Nature,* **370**, 666–668.

McLachlan, A. D. (1979). *J. Mol. Biol.* **128**, 49–79.

Richards, F. M. & Kundrot, C. E. (1988). *Proteins,* **3**, 71–84.

Rossmann, M. G., Liljas, A., Branden, C.-I. & Banaszak, L. J. (1975). *The Enzymes,* **11**, 61–102.

Rossmann, M. G., Moras, D. & Olsen, K. W. (1974). *Nature,* **250**, 194–199.

Russell, R. B. (1994). *Prot. Eng.* **7**, 1407–1410.

Russell, R. B. & Barton, G. J. (1992). *Proteins,* **14**, 309–323.

Russell, R. B. & Barton, G. J. (1993*a*). *Nature,* **364**, 765.

Russell, R. B. & Barton, G. J. (1993*b*). *J. Mol. Biol.* **234**, 951–957.

Russell, R. B. & Barton, G. J. (1994). *J. Mol. Biol.* **244**, 332–350.

Russell, R. B., Copley, R. R. & Barton, G. J. (1995). *Proc. 28th Hawaii. Int. Conf. Sys. Sci. IEEE Press,* **5**, 302–311.

Sali, A. & Blundell, T. L. (1990). *J. Mol. Biol.* **212**, 403–428.

Sankoff, D. & Kruskal, J. B., eds (1983). *Time warps, string edits, and macromolecules: The theory and practice of sequence comparison.* Addison-Wesley, Inc., Reading, Mass., USA.

Smith, T. F. & Waterman, M. S. (1981). *J. Mol. Biol.* **147**, 195–197.

Sternberg, M. J. E., Hegyi, H., Islam, S. A., Luo, J. & Russell, R. B. (1995). *Proceedings of the 3rd Annual Conference on Intelligent Systems for Molecular Biology,* , 376–383.