Sequence analysis of enzymes of the Shikimate pathway: development of a novel multiple sequence alignment algorithm.

by

Lachlan Hamilton Bell

A thesis presented for the Degree of Doctor of Philosophy

in

The Faculty of Science,
University of Glasgow.

Division of Biochemistry and Molecular Biology
Institute of Biomedical Life Sciences
University of Glasgow
Glasgow G12 8QQ

March 1996

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

Abstract

The possibility of homology modelling the shikimate pathway enzymes, 3-dehydroquinate synthase (e1), 3-dehydroquinase (e2), shikimate dehydrogenase (e3), shikimate kinase (e4) and 5-enolpyruvylshikimate 3-phosphate (EPSP) synthase (e5) is investigated. The sequences of these enzymes are analysed and the results found indicate that for four of these proteins, e1, e2, e3, and e5, no structural homologues exist. Developing a model structure by homology modelling is therefore not possible. For shikimate kinase, statistically significant alignments are found to two proteins with known structures, adenylate kinase and H-ras p21 protein. These are also judged to be biologically significant alignments. However, the alignments obtained show too little sequence identity to permit homology modelling based on primary sequence data alone.

An *ab initio* based methodology is next applied, with the initial step being careful evaluation of multiple sequence alignments of the shikimate pathway enzymes. Altering the parameters of the available multiple sequence alignment algorithms, produces a large range of differing alignments, with no objective way to choose a single alignment or construct a composite from the many produced for each shikimate pathway enzyme. This problem with obtaining a reliable alignment for the shikimate pathway enzyme will occur in other low sequence identity protein families, and is addressed by the development of a novel multiple sequence alignment method, Mix'n'Match.

Mix'n'Match is based on finding alternating Strongly Conserved Regions (SCRs) and Loosely Conserved Regions (LCRs) in the protein sequences. The SCRs are used as 'anchors' in the alignment and are calculated from analysis of several different multiple alignments, made using varying criteria. After divided the sequences into Strongly Conserved Regions (SCRs) and Loosely Conserved Regions (LCRs), the 'best' alignment for each LCR is chosen, independently of the other LCRs, from a selection of possibilities in the multiple alignments. To help make this choice for each LCR, the

secondary structure is predicted and shown alongside each different possible alignment. One advantage of this method over automatic, non-interactive, methods, is that the final alignment is not dependent on the choice of a single set of scoring parameters. Another is that, by allowing interactive choice and by taking account of secondary structural information, the final alignment is based more on biological, rather than mathematical factors. This method can produce better alignments than any of the initial automatic multiple alignment methods used. The SCRs identified by Mix'n'Match, are found to show good correlation with the actual secondary structural elements present in the enzyme families used to test the method.

Analysis of the Mix'n'Match alignment and consensus secondary structure predictions for shikimate kinase, suggest a closer match with the actual secondary structure of adenylate kinase, than is found between their amino acid sequences. These proteins appear to share functional, sequence and secondary structural homology. The proposal is made that a model structure of shikimate kinase, based on the structure of adenylate kinase, could be constructed using homology modelling techniques.

Table of Contents

List of Tables

List of Figures

Abbreviations

LCR	A Loosely Conserved Region in a multiple sequence alignment.
SCR	A Strongly Conserved Region in a multiple sequence alignment.
GCG	The University of Wisconsin Genetics Computer Group suite of sequence analysis programs.
CF	The Chou and Fasman secondary structure prediction algorithm.
GOR	The Garnier, Osguthorpe and Robson secondary structure prediction algorithm
GGBSM	The Gascuel and Golmard Basic Statistical Method for secondary structure prediction.
e1	The shikimate enzyme 3-dehydroquinate synthase.
e2	The shikimate enzyme 3-dehydroquinase.
e3	The shikimate enzyme shikimate dehydrogenase.
e4	The shikimate enzyme shikimate kinase.
e5	The shikimate enzyme 5-enolpyruvyl shikimate 3-phosphate (EPSP) synthase.

The IUPAC-IUB convention standard single letter code for amino acids is used.

Acknowledgements

I would like to thank my supervisor E. James Milner-White for his constant advice and support throughout the course of my three years of study, especially while I was writing my thesis. Thanks are also due for the help and advice given by my second supervisor, John Coggins. Their contributions are much appreciated. In addition, I wish to thank Ian Walker of Computing Services, University of Glasgow, who provided invaluable technical support.

Many thanks for constantly encouraging me to write up, go to my managers at work: Howard Shermann, Alan Bleasby and Duncan J. McGeoch.

My family, namely Joy, Christine, Tom, Martin, David, Karen, Mark, Kim, Jay, Caitlin, Ann, Walter and Morag together with my friends, particularly Robin for the competition to finish first and Nuala for inspiration, all deserve my thanks. I also wish to express gratitude to Hawkwind and Gong for serenading me through the process.

As the greatest source of support and encouragement throughout my studies, enabling me to devote the necessary time to this work, my love and thanks to my wife Clare.

Special thanks go to my proof readers, Christine and Clare. The English language would not be the same without you.

Finally, I would like to dedicate this work to my father, Robert. If he could have seen the end as well as the beginning.... in remembrance and love.

During the course of this work, the author was supported by a Science and Engineering Research Council studentship. Unless otherwise stated, all of the results described in this thesis were obtained by the author's own efforts.

Chapter 1 Introduction

1.1 The shikimate pathway - general introduction

The seven-step shikimate pathway synthesises chorismate from the carbohydrate precursors phosphoenol pyruvate and erythrose-4-phosphate [1-3] (figure 1.1). Chorismate is the common metabolic precursor of the aromatic amino acids, phenylalanine and tryptophan. It is also the precursor for many other aromatic compounds such as vitamins E and K, folic acid, ubiquinone, tetrahydrofolate, alkaloids, coumarins, lignin and various secondary metabolites [3, 4].

The shikimate pathway is present only in micro-organisms and plants, with other organisms requiring a supplementary dietary intake of aromatic amino acids. In higher plants it is a major biosynthetic pathway, with in excess of twenty percent of the carbon fixed by plants passing through it [5]. This has brought attention to this pathway as a potential target for pesticides and herbicides. The herbicide glyphosate, a specific inhibitor of the enzyme 5-enolpyruvylshikimate 3-phosphate (EPSP) synthase, is already of commercial importance [6-8].

The chemical actions of the seven enzymes of the shikimate pathway are the same in all plants and micro-organisms studied. However, the control and organisation of these enzymes shows a great deal of diversity between species [9].

In *Escherichia coli* the seven steps are separate and are catalysed by monofunctional proteins [10]. The genes of the seven enzymes are unlinked in the bacterial genome. In fungi and yeasts, the enzymes catalysing the five central steps of the pathway (e1-e5) occur on the *arom* complex. This consists of two identical pentafunctional polypeptide chains encoded by a single gene. The existence of this multifunctional enzyme has been shown in *Neurospora crassa* [11], *Euglena gracilis* [12], *Aspergillus nidulans* [13], *Schizosaccharomyces pombe* [14], and *Saccharomyces cerevisiae* [15].

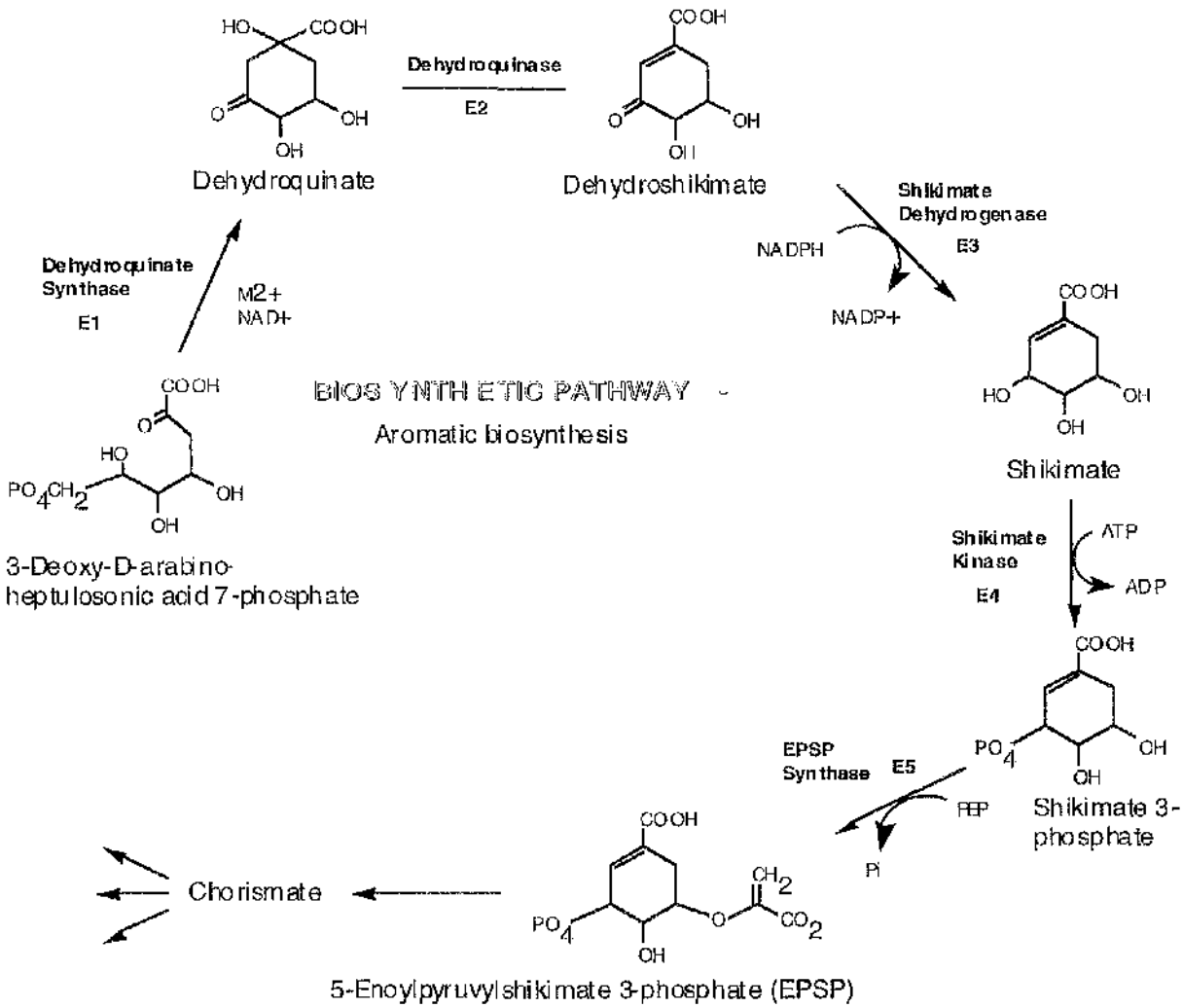


Figure 1.1
Reactions and enzymes of the second to sixth steps of the seven step biosynthetic shikimate pathway.

Studies on the enzymes of the shikimate pathway could provide answers and information relevant to a variety of research areas including the rational design of pesticides and herbicides. The principal information required for rational herbicide design is the structure and mechanism of the chosen enzyme. When this study was undertaken, no three dimensional structural information was available for any of the enzymes, however, the first structures are now appearing. EPSP synthase (e5) from *Escherichia coli* has been crystallised at 3 Å resolution [16] with further crystallisation studies being carried out on other enzymes of the shikimate pathway [17, 18].

A way to expand the knowledge base for the enzymes is to use computational sequence analysis and molecular modelling techniques. These are used to extract a range of information from primary sequence data and can potentially be used to predict a structure for the enzymes.

The rest of this introduction will present in more detail each of the enzymes comprising the shikimate pathway. A brief introduction to sequence analysis and molecular modelling methods will also be related and the objectives of this thesis will be outlined.

1.2 The shikimate pathway enzymes

The first step in the pathway is the condensation reaction between phosphoenol pyruvate and erythrose-4-phosphate. These are produced by the oxidation of glucose via the glycolytic and pentose phosphate pathways, respectively. The condensation reaction is catalysed by 3-deoxy-D-arabino-heptulosonate 7-phosphate synthase (DAHP synthase, e0, EC 4.1.2.15) and results in the loss of inorganic phosphate and the formation of the phosphorylated seven carbon keto sugar acid, 3-deoxy-D-arabino-heptulosonate 7-phosphate (DAHP). The enzyme requires a divalent cation, such as Co^{2+}, Mn^{2+}, or Mg^{2+}, for activity.

In *Escherichia coli*, three isoenzymes exist. The most active isoenzyme is a tetramer with a subunit molecular weight of 35 kD, with the other isoenzymes both dimeric with a subunit molecular weight of 39 kD.

As in many metabolic processes, the first enzyme of the shikimate pathway (DAHP synthase) is regulated and involved in the control of carbon flux. The pattern of regulation is complex and highly variable between species [3, 9]. An organism may posses from one to three isoenzymes, feedback inhibited, either by pathway end-products, pathway intermediates, a combination of the two or, exceptionally, the enzyme may be unregulated. Where an enzyme is responsive to more than one effector, they may act in a concerted, synergistic or cumulative manner.

The second reaction in the pathway is that catalysed by 3-dehydroquinate synthase (e1, EC 4.6.1.3) and involves the intramolecular exchange of the DAHP ring oxygen with the C7 carbon. This is accompanied by a reduction at C2 and an oxidation at C6. The enzyme requires Co^{2+} and NAD^{+} as cofactors and the reaction results in the release of inorganic phosphate and the production of 3-dehydroquinate (step 1, figure 1.1). In *Escherichia coli*, this enzyme is monomeric with a molecular weight of 39 kD [19, 20]. In fungi and yeast, this enzyme is one of the five catalytic components of the arom multifunctional enzyme [11, 21].

The third step in the pathway is the dehydration reaction which converts 3-dehydroquinate to 3-dehydroshikimate. This is catalysed by 3-dehydroquinase (e2, EC 4.2.1.10) and introduces the first double bond into the structure (step 2, figure 1.1) [22]. Chemical modification studies on the *Escherichia coli* enzyme identified a lysine and histidine residue at the active site [23, 24] with two methionines implicated as likely active site residues [25, 26].

The dehydroquinase enzyme exists in biosynthetic and catabolic forms. The biosynthetic 3-dehydroquinase is found in bacteria, plants, yeast and fungi while the catabolic dehydroquinase is found only in fungi such as *Neurospora crassa* and

Aspergillus nidulans. The biosynthetic enzyme is found as either dimers of subunit size 27 kD [27] or as a domain within a multifunctional protein corresponding to this size [15, 28]. The catabolic dehydroquinase is induced in response to quinic acid and is involved in the utilisation of quinic acid as carbon source for growth [29]. The catabolic enzyme forms large multimeric proteins consisting of identical monofunctional subunits of 16-18 kD [30, 31].

In fungi and yeast, 3-dehydroquinase is one of the five catalytic components of the arom multifunctional enzyme [11, 21] whereas in plants, 3-dehydroquinase and shikimate dehydrogenase (e3) share a single bifunctional polypeptide [28, 32, 33].

The fourth step in the pathway involves the transfer of hydrogen from NADPH to 3-dehydroshikimate to form shikimate (step 3, figure 1.1), and is catalysed by shikimate dehydrogenase (e3, EC 1.1.1.25). The enzyme is a monomer in *Escherichia coli* with a molecular weight of 32 kD [34]. The sequence contains a consensus nucleotide binding region which is thought to represent the NADPH binding site [35]. As mentioned above, in a variety of plants the activity resides with 3-dehydroquinase on a bifunctional protein and on the arom multifunctional protein in yeast and fungi [11, 15, 30].

The fifth step in the pathway involves the phosphorylation of shikimate to form shikimate-3-phosphate, using ATP, with the reaction being stimulated by Mg^{2+} (step 4, figure 1.1). This is catalysed by shikimate kinase (e4, EC 2.7.1.71). The enzyme is a monomer in *Escherichia coli* with a molecular weight of 19 kD. In fungi and yeast, the enzyme is one of the five catalytic components of the arom multifunctional enzyme [11, 21].

Escherichia coli has two isoenzymes, shikimate kinase I and shikimate kinase II. An examination of the kinetic properties indicated that shikimate kinase I may possess another function and only fortuitously catalyse this reaction. The Km of shikimate kinase II for shikimate is 200 μM, whereas that of shikimate kinase I is in excess of 5 mM [36, 37]. Shikimate kinase II is regulated by the aromatic amino acids which has

led to the hypothesis that an unknown metabolic pathway branches from the shikimate pathway at this point or, that in the evolutionary past, aromatic biosynthesis originated at this point [36].

The sixth step in the pathway involves shikimate 3-phosphate and phosphoenolpyruvate reacting to form 5-enolpyruvylshikimate 3-phosphate (EPSP) and inorganic phosphate catalysed by EPSP synthase (e5, EC 2.5.1.19) (step 5, figure 1.1). This enzyme has arguably been the main source of study in the pathway in terms of mechanism and structure, due to its inhibition by the broad spectrum herbicide, glyphosate [6, 7]. A number of amino acids have been implicated in the catalysis mechanism including arginine, cysteine, glutamate, histidine and lysine residues [3]. In fungi and yeast, this enzyme is one of the five catalytic components of the arom multifunctional enzyme [11, 21].

Chorismate synthase (e6, EC 4.6.1.4) catalyses the seventh and last reaction of the shikimate pathway, namely the trans 1,4-elimination of phosphate from EPSP to yield chorismate [38, 39]. The enzyme requires a reduced flavin cofactor, $FMNH_2$, to carry out the reaction. In *Escherichia coli* this enzyme is a tetramer with a subunit molecular weight of 38 kD.

1.3 Sequence analysis and molecular modelling

Molecular modelling is the process of predicting a protein's tertiary structure. An outline of some of the stages involved and the methods used is shown in figure 1.2. The first step is to search through amino acid sequence databases with the sequence of the protein of interest [40]. The next steps are dependent on the results of this procedure. If a homologous sequence is found, which has a known three dimensional structure, this structure can be used as the basis for building a model structure [40-42]. This approach is known as homology modelling and was first used in 1969 by Browne *et al.* [43] to model a structure of α-lactalbumin based on the known structure of lysozyme. If no

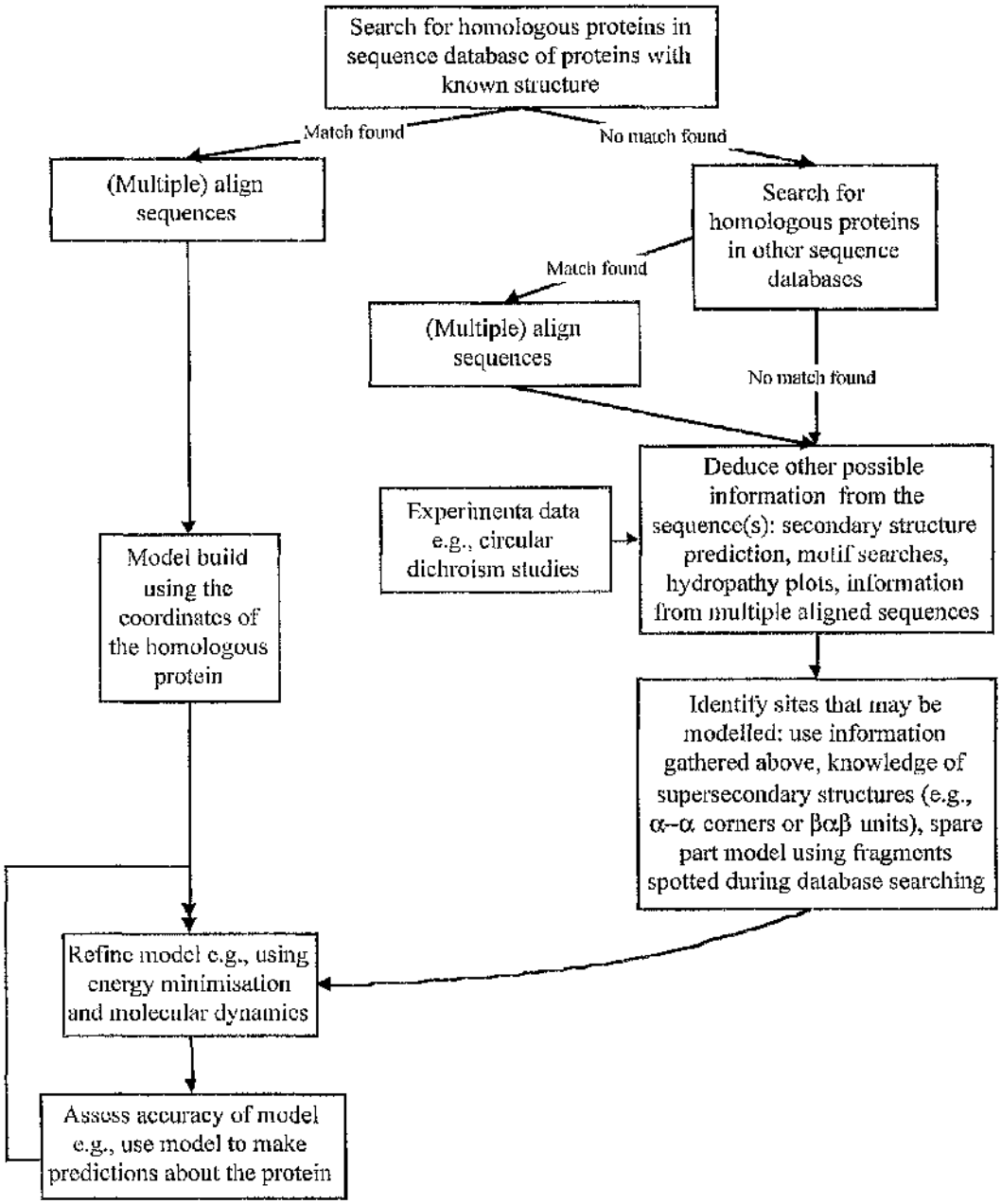


Figure 1.2

A flowchart detailing some of the procedures and methods used in homology and *ab initio* modelling of protein structures. The first stage in both procedures is sequence database searching. If a homologous sequence with a known structure is found, the left hand pathway for homology modelling is used, whereas if no homologous sequence is found, the right hand pathway for *ab initio* modelling is used.

homologue is found or no structural information is known for any homologues, *ab initio* prediction techniques can be used.

1.3.1 Homology modelling

The three dimensional structure of proteins is better conserved than the primary structure [40, 44-46] with structural homology being found between proteins even when limited primary sequence homology is found e.g., lactate dehydrogenase, glyceraldehyde 3-phosphate dehydrogenase and liver alcohol dehydrogenase have ninety four topologically equivalent residues but only three glycines and one aspartic acid are conserved between all the structures [47]. It follows that if homology is found between proteins at the primary structure level, it is likely that they may share a common tertiary structure. This knowledge is the basis of homology modelling.

In a protein family, the elements of secondary structure are usually found in comparable positions, with the main structural changes in the surface loops and coils or species specific secondary structural units [48-51]. Furthermore, most insertions and deletions occur in surface loops and coils rather than within secondary structural units [45, 52, 53]. Mutations that occur within the hydrophobic core can be compensated for by coordinated substitutions, acting to constrain any structural change [54, 55], or by shifts in the relative position of the secondary structural elements [56-58]. Chothia and Lesk quantified the structure/sequence relationship [59], showing that for distantly related proteins, sequence identity at the twenty percent level could mean that less than fifty percent of the molecules would have the same general fold. Increasing the percentage of sequence identity increases the percentage of the structures that share the same fold.

The stages involved in homology modelling start with aligning the sequences, with the intention of matching up topologically equivalent residues. The structure of the known sequence is then modified with the relevant insertions, deletions and mutations as

appropriate. The resulting model is refined using energy minimisation techniques to give a low energy conformation and remove any steric clashes [60-62].

The initial aligning stage is of critical importance, as any errors introduced at this stage will have potentially dramatic effects on the final model [63]. The sequence alignment generated in the database searching stage should be treated with caution as the algorithms used in database searching (described in more detail in chapter two) generally sacrifice sensitivity for speed. A better approach lies in using sensitive sequence aligning algorithms and, if possible, multiple sequence alignments [64]. These methods are described in more detail in chapter three.

When more than one structural homologue is found, two methods have been suggested for building a model structure. One approach is to generate a model structure from each of the known structures. These model structures can then be analysed and a preferred structure picked [65]. An alternative approach lies in constructing a composite three dimensional framework from all of the known tertiary structures, enabling the information from the structures to be used simultaneously [40, 59]. The sequence of the unknown structure is aligned and modelled to this framework. This approach has been shown to increase the accuracy of modelling in the case of sperm whale myoglobin [66].

The original structure's sequence is modified to that of the sequence of unknown structure by inserting, deleting and mutating residues. Two of the main tasks are then modelling loop structures and sidechain replacements. As the loops between secondary structural elements tend to be variable regions where insertions, deletions and mutations readily occur, these can be particularly difficult to model.

Studies on protein structure have identified and quantified a number of loop motifs. This information can be useful in the modelling process. The four residue β-turns have been widely studied [67-72]. The most commonly used classification is that of Richardson [69] which identifies six turn types, although a more recent and extensive

study, [70] proposed a new nomenclature based on the populated regions of a ϕ, ψ plot of the middle two residues in the turn. Sequence preferences for one of these turns have been identified. Systemic classifications have also been proposed for the β-hairpin loop, found between two anti-parallel β-strands [40, 72-77]. Efimov [77] lists the order of hydrophobic, hydrophilic and glycine residues that are expected for different types of hairpins and Sibanda *et. al.* [74] formulated rules based on loop length and sequence patterns that can give an indication of loop conformation in short loops. Another common motif is the β-bulge loop [69, 78], which can be thought of as being derived from the insertion of an extra residue in one of two adjacent strands of an anti-parallel β-sheet, although they can exist in the absence of β-sheet [79, 80]. Further examples of characterised loops include the paperclip loop [81, 82], commonly found at the C-terminal end of α helices, α-α corners [83] and hairpins [84], β-α-β loops [85] and omega loops [86].

A simple guide to modelling loops lies in first examining the sequences for characteristic sequence motifs, detailed in the literature cited above, which may enable a specific structure to be picked. Following this, the conformation of a loop of equivalent length in a homologous protein can be used [48], although this will not always give the correct structure. When no loop of equivalent length is found, other procedures include using the conformation from other proteins with similar supersecondary structure [76] which involves selecting a loop from a database of structures [87] according to a set of rules [88], or selecting loops based on geometrical features [89]. Failing this, *ab initio* techniques can be used to systematically search through loop conformations using energy minimisation [90, 91], molecular dynamics or distance geometry algorithms [92]. Such approaches are more likely to be accurate when the loop being modelled is short [93].

To model the replacement of sidechains, several approaches have been used including sampling all possible conformations [90], the use of the most probable torsion angles

[94, 95], rotomer libraries [96], or using the equivalent torsion angles of the sidechain being replaced [60, 97, 98].

Once the backbone model has been obtained and the sidechains replaced, energy calculations can then be used to remove steric clashes and allow local changes in structure [99]. A range of approaches have been suggested [40, 100] including constraining part of the structure from moving [60, 101], including surrounding water molecules in the model [102], and the use of molecular dynamics to explore a range of local conformations [91].

Once a model is constructed, an estimate should be made as to how correct it is. The relationship given by Chothia and Lesk [103] between the root mean squared (r.m.s.) difference in topologically equivalent C_α atoms and percentage sequence identity, can be used as a rough guide of expected r.m.s. deviation of the model structure from the x-ray crystal structure.

Analysing the model structure in the light of what is known about the structure of well defined proteins can also give useful information. For example, hydrophobic residues exposed to surface water molecules, or non hydrogen-bonded charged residues in the hydrophobic core are uncommon and their presence in the model may indicate that it is incorrect. Other indicators include whether β-strands are almost fully hydrogen bonded, the number of hydrophobic contacts, the ratio of polar to non-polar residues' solvent accessible surface [104, 105] and side chain packing [106]. The program PROCHECK [107] which uses a range of parameters from well defined proteins as described by Morris *et. al.* [108], together with 'ideal' bond lengths and angles, can also be used to assess the 'stereochemical quality' of a protein. A further useful check is whether or not the model can be used to make testable predictions [109].

Homology modelling can be a powerful technique, especially when there is high sequence identity (i.e., greater than fifty percent) between the sequences with the known and unknown structure. In such cases, the model may be expected to show differences

from the x-ray determined structure of the order of less than 1 Å for most of the protein [40, 46]. However, the large shifts in secondary structural elements observed in structures by Chothia and Lesk when the sequence identity is low [56-58, 103], are very hard to accommodate in this process.

1.3.2 *Ab initio* modelling

Ab initio modelling utilises computational sequence analysis techniques to predict information from the primary sequence of a protein. In combination with the knowledge obtained from studies on proteins with known three-dimensional structure, if enough useful information is gathered, it may be possible to predict a three-dimensional structure for part or all of the protein. Even if predicting a three-dimensional structure is not possible, the information gathered may prove useful, e.g., suggesting possible structural or catalytic features for the protein or for the design of further experiments. A variety of techniques are available for computational sequence analysis, including comparing and aligning sequences, motif analysis [110, 111] and prediction of various properties of the sequences e.g., hydrophobicity [112], hydrophobic moment [113], antigenicity [114], flexibility [115] and secondary structure.

Ab initio modelling is used when a database search reveals no structural homologues for the protein of interest. However, comparison and alignment of the sequences of the protein family members and the non structural homologous sequences that are found, can reveal much information. Details on the methods used in aligning are given in chapter three. Alignments of distantly related sequences can identify invariant residues essential to structure or function. Such sites can be analysed by mutagenesis studies [116, 117]. Variant residues may point to species dependent sites, useful for design of specific antibodies. Examination of the aligned sequences can reveal motifs that can be used to identify other family members [118] or define possible structural or functional units within the protein [119]. Where a large number of sequences are available for

aligning, approaches for predicting secondary and tertiary structure have been described using the pattern of sequence variation and conservation [120] or patterns of hydrophobicity [121]. Multiple alignments can also be used in various other roles including increasing the accuracy of secondary structure predictions [122-125], predicting surface residues [126], predicting active site residues [126-128] and evolutionary studies [129-131].

Using a protein's primary sequence, a variety of methods have been developed for predicting other aspects of the protein, however, by far the greatest amount of work has concentrated on predicting the secondary structure. The earliest secondary structure prediction methods, and still in wide spread use today, are those of Chou and Fasman (CF) [132-134] and Garnier, Osguthorpe and Robson (GOR) [135]. These use statistical analysis of the most likely conformation for residues together with empirical rules for creating a prediction and information theory respectively. Other methods include modifying the CF method by including data from other structural forms [136], updating the GOR method [137], empirically combining the CF method with stereochemical rules [138], neural networks [139-144], combining the CF method with structural class prediction [145], turn prediction combined with pattern recognition [146], using physical-chemical features of the residues [147, 148], combing different methods [124, 149], use of sequence similarities [124, 150], and hidden Markov methods [151]. Algorithms have also been described for predicting β-turns [152, 153] and turn residues [154]. The claims of accuracy of these secondary structure prediction methods vary from approximately fifty to eighty percent (detailed in references above and [155-157]), with the turn prediction methods of Cohen *et al.* [146, 154] claiming over ninety percent accuracy.

Methods have also been developed which allocate an amino acid sequence to one of four structural classes (high α content, high β content, mixed α + β content and disordered structures) [158-161] with a claimed accuracy of over eighty percent.

If a secondary structure is predicted, it may be possible to predict a tertiary structure, either using a topology prediction method [162-164] or by manually folding the protein using principles gathered from known protein structures [165]. Such principles include the packing of α-helices against a β-sheet [164], layering of side chains [166], turns in [167] and the conformation of [168] α/β barrel proteins, residue specific interactions in β-sheets [169] and classes of amphipathic helices [170]. The secondary structure predictions, which tend to be poor at correctly defining the start and end positions of secondary structural units [171], can also be refined using principles gathered from known protein structures, e.g., amino acid preferences for the N and C caps of α-helices [82, 172, 173] or other specific locations [174] and sequence fingerprints for specific secondary structures [175, 176].

1.4 Aims of the project

Information on the structure of the shikimate pathway enzymes is of great use in many research areas, including studies on enzyme mechanism and the rational design of pesticides and herbicides. When starting this project, the structural information on enzymes of the shikimate pathway was extremely limited, with no enzyme having a known three dimensional structure. One structure has since been solved, with work progressing on determining the x-ray structure of several more of these enzymes. An alternative approach to x-ray crystallography for gaining structural details, is the use of computational sequence analysis methods. Such methods may allow model structures for the shikimate pathway enzymes to be developed, potentially in a lot less time than a x-ray crystallography approach requires. Five of the shikimate pathway enzymes, e1 to e5, are chosen to carry out studies on.

The objectives of this work were therefore :-

(1) To investigate the possibility of homology modelling the shikimate enzymes e1 to e5.

(2) To investigate the possibility of *ab initio* modelling the shikimate enzymes e1 to e5.

(3) To determine weak links in the molecular modelling and sequence analysis process and develop methods if needed.

Chapter 2 Database analysis of the shikimate enzymes

2.1 Introduction to database analysis

Following the molecular modelling flowchart in the introduction (figure 1.2), once the enzyme sequences are known, the first step is to search the sequence databases [40]. The aim of database searches is to discover any already known sequences that show similarity to the sequences under investigation.

Similar sequences and the alignments they can produce, can accrue many benefits, as knowledge from an already characterised protein can be transferred to the homologue. The range of benefits are covered in the introduction and include highlighting candidate residues for site-directed mutagenesis probes [116, 117] and enhancing secondary structure and tertiary structure predictions [120, 124, 126]. Protein three-dimensional structures are conserved in evolution more than primary sequences and considerably more than DNA sequences [45, 103]. Therefore, if an unknown amino acid sequence matches one with a known three-dimensional structure, a testable tertiary model can be constructed [40, 42]. Predicting testable tertiary models was one of the aims of this thesis.

2.2 Search for enzyme sequences of the shikimate pathway

This thesis is concerned with five of the seven enzymes of the shikimate pathway (figure 1.1). These enzymes, labelled e1 to e5 carry out the second to sixth steps in this pathway. For each of the five enzymes, all the known amino acid sequences are collected. Searches are conducted on the major databases of known DNA and protein sequences [177] using the programs STRINGSEARCH from the GCG package[178] and DELPHOS, a part of the ISIS package [179] (table 2.1).

Program	Database searched	Database Version	Database Size (sequences)	Database Size (residues)
STRINGSEARCH	GenBank	73	78 608	101 008 486
"	EMBL	32	79 377	101 292 310
"	PIR	32.0	23 840	6 935 412
"	SwissProt	23.0	26 706	9 011 391
DELPHOS	OWL	20.2	29 874	10 875 829

Table 2.1

Showing the version number and size of the databases searched, for enzymes e1 to e5 of the shikimate pathway, for each of the programs used.

These programs search through the documentation of each sequence in the database for the presence of a user chosen text string or keyword. The DELPHOS program improves on the basic string matching approach taken by STRINGSEARCH by ignoring punctuation marks and by searching through a pre-generated index of the text in the database. These have the effect of greatly improving both the speed and sensitivity of the searching. Due to the possibility that a sequence's documentation may contain typographical errors, variations in any abbreviations used or differing punctuation (which would have the effect of causing false negatives in the search results), a variety of keywords are used with both programs to ensure that all known sequences of enzymes e1 to e5 are found.

For these five enzymes, a total of twenty five sequences are found from eleven different species: three sequences are known for enzyme e1; four sequences are known for enzyme e2; three sequences are known for enzyme e3; five sequences are known for enzyme e4; and ten sequences are known for enzyme e5. For each enzyme, the species that have been sequenced are shown in table 2.2. When referring to a specific enzyme, the number of the enzyme in the shikimate pathway and the accession number will be used. The accession number is a unique identifier given to each sequence by the database administrators. For example, the shikimate kinase from *E. coli* is referred to as e4-P08329.

2.3 Database searching methods

The ease with which sequences can be aligned depends on how closely related they are [180]. As the degree of identity in a protein family decreases, the core three-dimensional structure starts to differ by increasing amounts [103]. This is reflected by an increase in the proportion of insertions and deletions (collectively called indels), which makes aligning harder. Indeed, once the percentage of sequence identity drops below fifty percent, and especially lower than twenty five percent, (what Doolittle *et al.*, [181] refer to as the 'twilight zone' of sequence alignment) the

Table 2.2

Shikimate pathway enzyme sequences. Showing the known enzymes species, E. C. number, accession numbers - taken from the OWL database - and length for each sequence of the shikimate enzymes e1 to e5. The sequences of Aspergillus Nidulans and Saccharomyces Cerevisiae are pentafunctional polypeptides of length 1603 and 1588 residues respectively. The lengths given are the lengths of the individual enzymes along the sequence, taken from the annotations in the sequence databases. The sequence of Aspergillus Nidulans used differs from the sequence given in the databases according to a revised sequence for the Shikimate dehydrogenase (e3) enzyme received from Dr. A. Hawkins, Newcastle University (personal communication) and shown in appendix A. The sequence of pea EPSP sythnase was received from Prof. J. R. Coggins (personal communication) and is shown in appendix A.

Enzyme Name	Enzyme number	Accession Number	Species	Sequence Length (residues)
3-Dehydroquinate synthase E.C. 4.6.1.3	e1	P07639	Escherichia Coli	362
		P08566	Saccharomyces Cerevisiae	392
		P07547	Aspergillus Nidulans	384
3-Dehydroquinase E.C. 4.3.1.10	e2	P05194	Escherichia Coli	252
		P24670	Salmonella Typhi	252
		P08566	Saccharomyces Cerevisiae	233
		P07547	Aspergillus Nidulans	221
Shikimate dehydrogenase E.C. 1.1.1.25	e3	P15770	Escherichia Coli	272
		P08566	Saccharomyces Cerevisiae	283
		P07547	Aspergillus Nidulans	314
Shikimate kinase E.C. 2.7.1.71	e4	P08329	Escherichia Coli	174
		P10880	Erwinia Chrysanthemi	173
		Q00497	Lycopersicon Esculentum (tomato)	300
		P08566	Saccharomyces Cerevisiae	174
		P07547	Aspergillus Nidulans	194
5-enolpyruvylshikimate 3-phosphate (EPSP) synthase E.C. 2.5.1.19	e5	P07638	Escherichia Coli	427
		P07637	Salmonella Typhimurium	427
		P10748	Lycopersicon Esculentum (tomato)	520
		P11043	Petunia Hybrida	516
		P05466	Arabidopsis Thaliana (mouse-ear cress)	520
		Pea	Pea	447
		P12421	Bordetalla Pertussis	442
		P20691	Bacillus Subtilis	428
		P08566	Saccharomyces Cerevisiae	463
		P07547	Aspergillus Nidulans	447

importance of exactly what is meant by the sequence similarity becomes paramount. This will be explored in more detail later in this chapter when discussing the specific algorithms used in sequence alignment and database searches.

2.3.1 Methods used in pairwise sequence aligning

Database searches proceed by comparing the query sequence against every database sequence in turn. First, the main algorithms used in aligning two sequences together, are described. The evolutionary changes that can occur between two sequences include insertions, deletions and mutations, and any alignment method therefore should take these possibilities into account.

A large number of pairwise alignment algorithms have been described. These are detailed below, following an approximate chronological order. The majority of pairwise alignment methods can be roughly divided into two camps, depending on whether they compare single residues or lengths of residues at a time. Other methods have been described that utilise different approaches to sequence alignment to those given below, including methods based on fast Fourier transforms [182] and iterative statistical analysis of homologous subsequences [183]. These methods are not discussed in greater detail as they were unavailable for use and do not give any advantage over the methods discussed below.

2.3.1.1 Comparisons using single residues

A simple approach to sequence comparison is to lay one sequence along a matrix column and the other along the matrix row. A scoring matrix is then consulted which holds a score for the comparison of every residue against another and the appropriate score S(i,j) for the *i*th residue from sequence one and the *j*th residue from sequence two is inserted into every cell of this matrix. An example of a simple scoring matrix is the

identity matrix where identical residues score one and all others score zero. Diagonal runs of ones show up where there are identities between the sequences, and a visual examination can then be used to identify putative aligned regions.

An alternative to visual examination of the matrix is the dynamic programming method, developed by Needleman and Wunsch [184] (see figure 2.1 for illustration). This method constructs a matrix from the two sequences, as described above, the upper left-hand and lower right-hand corners of the matrix corresponding to the amino and carboxyl-termini of the amino acid sequences. The matrix elements are filled in according to the scoring matrix chosen. Then, starting from the carboxy-terminal corner of the matrix and proceeding row by row to the top left hand corner, the scores of each matrix element are replaced by the cumulative scores reached so far. The next step, which defines the alignment, starts at the highest score of the amino-terminal end and traces the path back through the matrix. The pathway that gives the highest cumulative score is the chosen alignment. If deviations are taken from a diagonal path, this is equivalent to inserting gaps. A penalty factor can be subtracted from the score for every gap used as a barrier to their insertion. This method was extended by the works of Seller [185] and Waterman *et al.* [186] who developed new, mathematically rigorous metrics to measure the distance between biological sequences, which could include insertions or deletions of arbitrary lengths. These metrics were incorporated into the Needleman and Wunsch iterative matrix method of calculation and ensure that the highest scoring alignment is found, although the value of the highest score is dependent on the scoring matrix used and what gap penalties are used. It can also be extended to align n sequences by carrying out the calculations in a n-dimensional matrix (see Murata [187] for a three-way alignment method).

This general method has had a variety of modifications and improvements made to it to enhance its usefulness. These include those made by Smith and Waterman [188] who modified this technique to find the highest scoring subsequence or 'local' alignment. This involves finding the highest scoring matrix element (which no longer has to start at

Figure 2.1

Needleman and Wunsch method for calculating the optimal alignment between two sequences. A matrix is constructed with one sequence laid along the columns and one along the rows, with their amino termini together. (**A**) Scores are inserted in the matrix elements according to how similar the residues at that position are - in this case a score of 1 for identity, otherwise 0 is given. (**B**) Starting from the carboxy termini (lower right) corner, and moving row-by-row, the cumulative scores are then inserted into the elements. (**C**) Starting from the amino termini corner, the pathway is traced out which gives the highest cumulative score. The pathway is shown highlighted. (**D**) This pathway gives the alignment.

(A)

	A	D	C	D	E	F	G
A	1	0	0	0	0	0	0
F	0	0	0	0	0	1	0
D	0	1	0	1	0	0	0
C	0	0	1	0	0	0	0
D	0	1	0	1	0	0	0
H	0	0	0	0	0	0	0
Y	0	0	0	0	0	0	0
F	0	0	0	0	0	1	0

(B)

	A	D	C	D	E	F	G
A	5	3	2	1	1	0	0
F	4	3	2	1	1	1	0
D	2	4	2	2	1	0	0
C	2	2	3	1	1	0	0
D	1	2	1	2	1	0	0
H	1	1	1	1	1	0	0
Y	1	1	1	1	1	0	0
F	0	0	0	0	0	1	0

(C)

	A	D	C	D	E	F	G
A	5	3	2	1	1	0	0
F	4	3	2	1	1	1	0
D	2	4	2	2	1	0	0
C	2	2	3	1	1	0	0
D	1	2	1	2	1	0	0
H	1	1	1	1	1	0	0
Y	1	1	1	1	1	0	0
F	0	0	0	0	0	1	0

(D)

A – D C D E – F G

A F D C D H Y F

the first row or column) and then tracing back the score. A weighting scheme (i.e. the combination of scoring matrix and gap penalties) was chosen so that no element of the matrix was below zero and the traceback stopped on reaching any element containing zero. This local alignment method has been refined with techniques which have allowed alternative local optimal alignments to be detected [189, 190] and improvements to be made in the speed of analysis [191], amongst others [192-195].

The treatment of gap scoring by these methods varies. They may have the same gap penalty for any gap length [196], concave scoring where the penalty increases in a non-linear fashion as the indels get longer [197], gaps at the ends of sequences not being scored [188], or non-affine gap penalties, introduced by Gotoh [198]. Non-affine gap penalties have a score for starting a gap and a second score which is dependent on the gap length. More recently, methods have been developed which enable the gap penalties to be varied along the sequence. This is useful if the secondary structure or other biological information is known so that areas that should not have gaps inserted, e.g., regions of secondary structure can have high penalties, and vice versa [199, 200].

An additional benefit of the Gotoh method's use of non-affine gap penalties is that it uses fewer steps and less computer memory than is required by previous methods. This is of importance as the dynamic programming methods are computationally expensive in terms of both computer memory and time. A variety of methods have been developed to deal with this [201-203]: only carrying out the calculations on parts of the matrix at a time, to reduce memory usage [204] and, to increase speed, only calculating the pathways through certain parts of the matrix [205-207].

2.3.1.2 Scoring matrices

Some form of scoring matrix is used in almost all pairwise aligning and database searching methods. A scoring matrix is used to quantify the mutational changes that occur between sequences and therefore plays an important role in any algorithm. An

example of a scoring matrix, the identity matrix, is shown in chapter 2.3.1.1, and a large number of other scoring matrices have been described. These include matrices based on; the "genetic distance" of the amino acids in the genetic code (i.e., the minimum number of base changes required to convert one residue to the other), conservation of various amino acid properties [208, 209], alternative amino acids found at corresponding positions in aligned sequence families [210], the observed occurrence of amino acids in protein secondary structural elements [150], observed replacements in superimposed three-dimensional structures of related proteins [211], hydropathy [95, 212], and combinations of the above [213]. Indeed, any quantitative scale of residue's attributes can be turned into a suitable scoring matrix [208, 212]. Kidera [214] and Nakai [215] on analysing 188 and 222 published indices respectively, show that the similarity between indices can be used to group the indices into four major clusters. The difference in indices between each cluster can be large, with indices from different clusters showing large variations in defining which residues are grouped as 'similar' [211].

Amongst the most widely used matrix is the PAM (Point Accepted Mutation) matrix of Dayhoff [216-218] which is based on data from mutation studies. Each matrix element is calculated by multiplying the conditional probability that residue A is replaced with residue B by the probability that residue A is replaced with anything. The matrix is normalised so that the probability that there is no change is ninety nine out of one hundred, corresponding to one accepted amino acid substitution per hundred residues. The final matrix then gives the probability that residue A is replaced by B in one unit of evolutionary time. The conditional probabilities are calculated from the observed frequencies of exchanges between the amino acids in groups of closely related, aligned sequences (greater than eighty five percent identity). The probability that residue A is replaced, the relative mutability, is calculated as the ratio of the total number of times the amino acid was observed to change, to the total exposure to change. For each pair of sequences, the exposure to change is calculated for each amino acid as the frequency of the occurrence of that amino acid multiplied by the total number of all amino acid

changes observed for that sequence pair per hundred sites. These values are summed to give the total exposure to change. Dayhoff and Eck's results from 1968 [218] were based on four hundred accepted point mutations while Dayhoff *et al.*'s 1978 results [216] used sixteen hundred accepted point mutations from seventy one groups of proteins. Little difference is found between the resulting matrices [212]. Jones *et al.*, [219] automated this methodology, enabling the calculation of the PAM matrix from up-to-date databases. In a comparison of the latest matrix, calculated from 2621 groups of proteins, to the 1978 matrix [216], only a few significant differences were seen and they almost all corresponded to exchanges that were observed only once by Dayhoff *et al.*.

In use, the PAM matrix is usually changed to use scaled logarithms for each matrix element to increase the speed of calculations [219]. Also, the 1-PAM matrix can be scaled to reflect larger intervals of evolutionary time by multiplying the matrix by itself, i.e., the 2-PAM matrix corresponds to the square of the 1-PAM matrix, the 3-PAM matrix with the cube, etc.. The commonest PAM matrix in use is the 250-PAM matrix.

2.3.1.3 Comparisons using lengths of sequences

A second approach to pairwise alignment uses regions of sequences of a given length such that each region of one sequence is compared to every other region of the other sequence. The sequences are laid out on the row and column of a matrix. A scoring matrix is employed and the score for each residue in a fragment against the residues in the other fragment are summed and inserted in the matrix cell at the middle position of the comparison. For example, if the fragment length was eleven, the summed score of comparing residues one to eleven in the first sequence against residues one to eleven in the second sequence would be inserted at cell (6,6) and against residues two to twelve in the second sequence inserted at cell (6,7), etc.. Replacing scores above a certain threshold with a dot and everything else with a blank produces a dot matrix or dotplot

(programs for doing this include COMPARE and DOTPLOT from the GCG package). Regions of similarity can be ascertained by visual examination of the dotplot. Diagonal lines, parallel to the main diagonal, trace out the best alignment and, if insertions and deletions have taken place, the lines are disjointed or broken. Another use of this method in sequence analysis is where the same sequence lies on both axes of the matrix, enabling internal sequence repeats to be identified.

Kubota *et al.*, [220, 221] improved this technique's sensitivity in detecting distance relationships, by scoring the residues using a correlation of many amino acid properties. Argos [213] extended and generalised this method with the scores for each fragment being calculated using both the Dayhoff matrix [216] and the correlation coefficients of five physiochemical properties (size, hydrophobicity, shape, strand and turn). These scores are scaled and summed together and placed along the entire length of the fragment. If part of a fragment has a higher score when used as part of another region, the higher score is kept. As a further refinement, the dotplot is recalculated using various length fragments and at each position in the matrix, the highest score from these different runs is used. The mean and standard deviation of all comparisons in the matrix are calculated and the scores replaced with their number of standard deviations above the matrix mean. Only scores above a threshold are displayed.

When the output is displayed, different symbols are used for different standard deviation ranges. This serves as an aid to visual examination enabling quick identification of significant aligned regions. An automated alignment of the two sequences is also possible which entails using a Needleman and Wunsch method [213, 222] to link up regions of high similarity already identified. Gaps are not allowed in the regions of high similarity, only in the areas joined by the Needleman and Wunsch method.

Although the fragment comparison methods are sensitive and useful for comparing distantly related sequences, they are computationally more expensive than the dynamic programming methods (Argos quotes an overnight run on a VAX computer being

enough time to compare one sequence against one hundred to one hundred and fifty sequences [213]).

2.3.2 Methods used in database searching

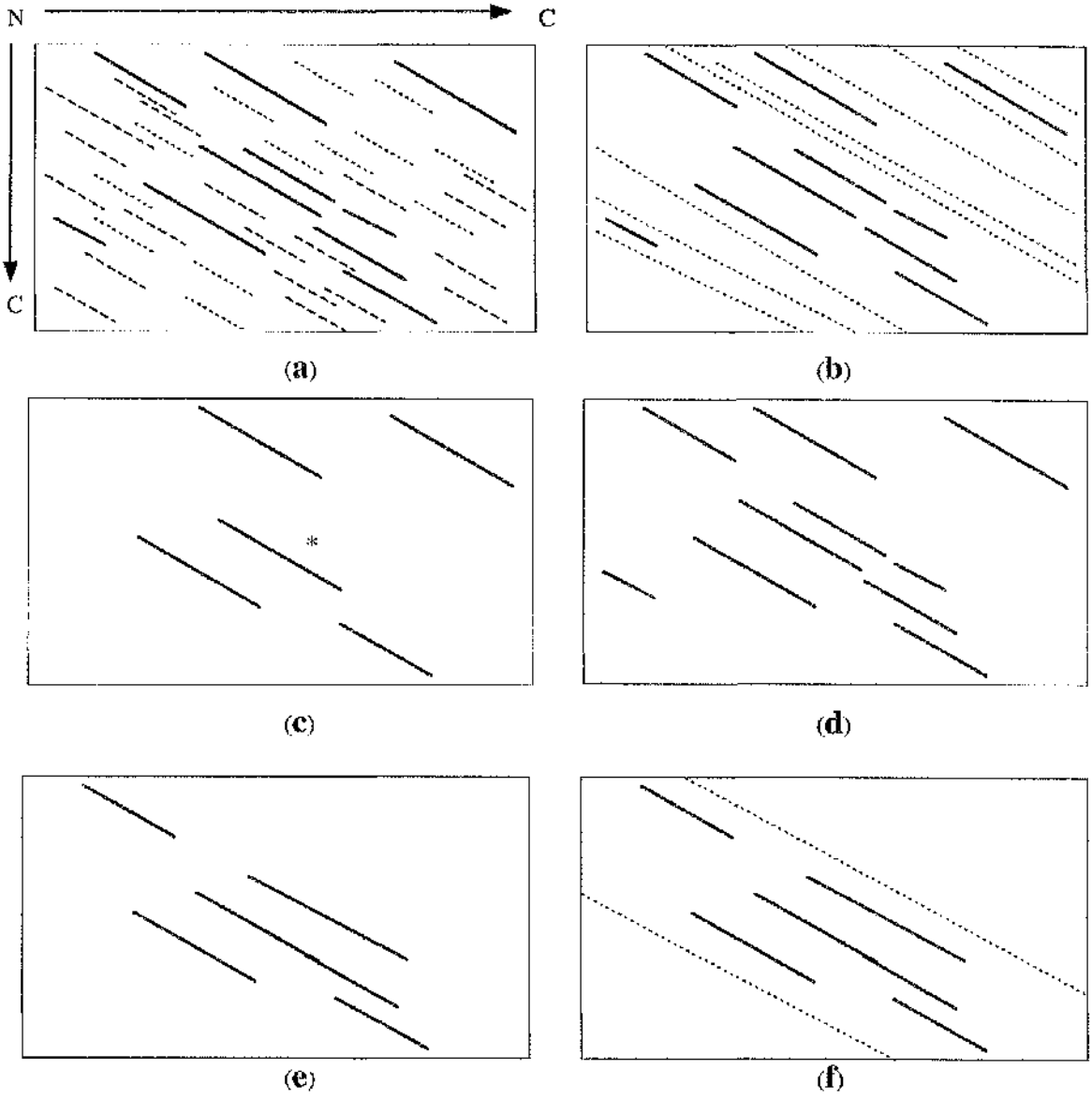
One practical consideration to be borne in mind when contemplating the use of the above methods in searching entire databases of sequences, is that of speed. The original dynamic programming algorithms have been superseded by faster ones (computational time has been reduced from, in the order of the cube of the average sequence length, to the square [196, 201]) and computers have greatly increased in power, however, they are still slow when used for database searches. This problem has been addressed in two ways; development of novel algorithms specifically designed for database searching which trade-off sensitivity and selectivity against speed, and conversion of pairwise and other alignment methods to run on multi-processor computers which utilise the speed gains inherent in their design.

2.3.2.1 Time efficient database searching algorithms

One of the earliest programs which was effectively used to search a database was that of Wilbur and Lipman [223]. This uses several techniques to increase the speed of sequence comparison while retaining sensitivity. Firstly, all identical matches between the sequences, of length k, are rapidly found using the 'lookup table' method of Dumas and Ninio [224] (see figure 2.2 A). Secondly, diagonals through the matrix of the two sequences are found where the number of k-tuple matches on that diagonal are a certain number of standard deviations above the mean of all diagonals with k-tuple matches. A window space of size w is then defined around the chosen diagonals (see figure 2.2 B). This has the effect of reducing the matrix space to be searched to find the alignment. The sequences are aligned and a score generated using a Needleman and Wunsch type algorithm where residues are given a score of +1 if they occur in window space and are

Figure 2.2

This figure illustrates how the fasta family of programs work. (**a**) The original method finds all *k*-tuple matches between the query sequence and the database sequence. (**b**) Then, diagonals with greater than a threshold number of matches on them are chosen and a window space, shown above by a red dotted line, defined around them. The final alignment is calculated in that space. (**c**) The FASTP method finds the *k*-tuple matches as in (a), but then choses the top 5 diagonals. These are rescored using the PAM-250 matrix and the highest score, shown above marked with an asterix, is kept to rank the hits in the database. Finally, optimal global alignments are carried out on the top database hits using the top diagonal and a narrow window space around them. (**d**) The FASTA method finds the *k*-tuple matches as in (a), but then choses the top 10 diagonals. (**e**) These are checked to see if any of the subsequences can be extended using an optimal aligning method. The method used permits gaps in the alignment and the scoring matrix to be changed. The scores from this step is kept to rank the database hits. (**f**) Finally, the extended subsequences and a narrow window space around it, shown above by a red dotted line, is used as the basis for calculating an optimal alignment.


N
C
C
(a)
(b)
(c)
(d)
(e)
(f)

part of a *k*-tuple match produced by that alignment. Gaps can be introduced to the alignment and each gap carries the same penalty, *g*. The raw score for each database sequence is dependent on the length of both the query sequence and database sequence used, so a correction factor is applied to each. Using this corrected score, the database sequences are ranked in order of their similarity to the query sequence. A statistical measure is also calculated from the corrected score which can be used as an approximate guide to the significance of the similarity between the query sequence and the database sequence. The variables *k*, *w* and *g* are all user-definable and setting their values is important both to the speed and sensitivity of the method. Indeed, if $k=1$ and *w* is large enough to include all diagonals, then this algorithm is equivalent to the Needleman and Wunsch algorithm with the same speed drawbacks. However, providing $k>1$ and *w* is of a modest size, for any given pairwise comparison, this method is much faster than those detailed above, and fast enough to enable searches of sequence databases to be made practicable. The time requirements of the method, for comparison of most sequences, depends on the sum of the sequence lengths rather than the product (e.g., doubling the length of the sequences doubles the comparison time rather than quadrupling it).

This method was improved upon by Lipman and Pearson [225] with their program FASTP which increased the speed and sensitivity of amino acid sequence comparisons. Identities of *k*-tuple length are again found by the above mentioned 'lookup table' method and the top five diagonal regions are chosen based on the number of matches and the distances between the matches. These regions are then rescored using the 250-PAM matrix (see figure 2.2 C). No insertions or deletions are allowed at this stage. Then, for each database sequence, the highest of these subsequence similarity scores is kept rather than the highest score generated over the whole diagonal. An 'optimised' score which does allow gaps is then generated for sequences with high initial scores. This applies a Needleman and Wunsch type algorithm to the initial alignment, and a narrow (sixteen residue) window space around it. Speed is increased by using a fast scoring scheme to get the ranking and only having to calculate a slow global alignment

for the top hits in the database, while sensitivity is increased by using a 250-PAM matrix rather than an identity matrix in the calculations.

SWEEP [226] is an extension of the FASTP method which can search the OWL database. Its major extensions are the use of user-definable scoring matrices and that the query sequence, once aligned with a homologous database entry, can then be segmented by user-specified criteria and these segments tested for homology.

Further refinements were made to FASTP by Pearson and Lipman [227] in the program FASTA. These increased the sensitivity of the method (detecting more distantly related sequences), with a small loss of selectivity (unrelated sequences get slightly increased scores) and little effect on speed. The top ten regions of highest similarity are found, rather than the five of FASTP (see figure 2.2 D). After rescoring by the 250-PAM matrix (although now any matrix can be used), instead of the top subsequence region being taken forward for ranking the database sequences, the subsequences are analysed to see if they can be joined up. Only subsequences with scores above a threshold are included in this step to keep the degradation in selectivity to a minimum (see figure 2.2 E). The joining procedure uses a form of optimal aligning and allows gaps to be inserted. The resultant score is used to rank the database sequences. So, rather than the score of an area of local similarity, as in FASTP, being used to rank the hits, ranking is now based on optimal alignments of areas of local similarity. The final step - applying the Needleman and Wunsch type algorithm to the initial alignment and a narrow window space around it - remains the same (see figure 2.2 F). Only small differences are expected between the final alignment in FASTA and the one used for ranking, whereas in FASTP, the optimised score can be greatly increased from the ranking score. These methods can now search through a database with computational time on the order of the sum of the sequence lengths rather than the product.

The STATSEARCH program [228, 229] for searching databases includes a measure of statistical significance. Only database sequences which compare with the query sequence at the one percent significance level are kept and alignments can then be

calculated using the Smith and Waterman [188] local similarity method. A dot matrix is constructed using a 'lookup table' approach and the statistic U is defined as the score of the maximum-scoring diagonal of the dot-matrix. A close approximation of the distribution of U for random sequences with the same composition and length is then used to assess the statistical significance of the observed value of U. This distribution of U is not calculated for every comparison made as this would be computationally exorbitant in time. Initially, using rounding-down of the sequence length and approximations of the sequence's composition, a 'lookup table' is consulted which holds one percent values to give a quick indication of the significance. If U exceeds this approximated one percent level, the actual sequence composition values are then used to recalculate U. If U still exceeds this new one percent level, the value of U is then calculated using the actual sequence length and compositions directly. This three step approach retains sequences that are clearly significant but enables a rapid elimination of most sequences that are insignificant and further consideration for those that are borderline. The risk of false negatives increases as the sequences increase in length, so STATSEARCH cuts up sequences into fragments. The analysis of these fragments is conducted in an overlapping manner to ensure that a similarity at the fragmentation point is not missed.

Another approach is the Basic Local Alignment Search Tool (BLAST) program of Altschul *et al.* [230]. This is based on finding Maximal Segment Pairs (MSP) which are defined as the highest score pair of identical length sequences chosen from two sequences and the scores are calculated using any scoring matrix. The statistical significance of these MSP scores can be estimated [231]. Each database sequence will contain a MSP, so a cut-off score S is set so that only sequences with a MSP score above this are recorded. S is set low enough so that all highly significant sequences as well as some borderline cases are found. The speed of this method lies in the way it eliminates sequences that are unlikely to exceed this score. The initial pass of the database is made looking for sequences that have small fragments that exceed another cut-off value, T. Any such hit is extended to see if it is contained within a region that

has a score exceeding S. T is empirically chosen to balance speed of execution and the risk of not finding some MSP's that do exceed S. The scanning for T scoring fragments is accomplished by first dividing the query sequence into small fragments, and for every fragment, every possible set of residues that match with this to give a score greater than T are calculated and stored. The database sequences are then rapidly scanned using the finite state machine method [232] to see if they contain one of these stored high scoring fragments. This fast searching for initial hits which can then be expanded as well as the use of a specially formatted, compressed database, results in this method being an order of magnitude faster than comparable methods.

An alternative method which emphasises sensitivity rather than speed in searching a database is that of profile analysis [233]. In this method, a group of aligned query sequences are converted into a position specific scoring table or profile. This profile is then compared to a database using dynamic programming methods. The profile lists, for each position in the query sequences, a score for every amino acid and an extra column for an insertions/deletions penalty. This penalty can be set low if there are gaps in the query sequences or at the user's request to encourage insertions at this position or high to discourage them. The score for every amino acid is calculated using the sum of the PAM matrix scores for every residue at that position multiplied by a weighting scheme that depends on the frequency of the residues. The scores between the query and database sequences are read from the profile rather than from a scoring matrix. The use of more than one sequence in the query enables a greater information content to be used in searching the database sequences. Also, each position and each sequence can be weighted enabling secondary structure or other known information to be incorporated into the tests. These have the effect of enhancing the sensitivity and selectivity of this method.

A further approach to speeding up database searching is taken by Higgins and Gouy [234]. If it is known that the search should only be undertaken on a subset of the database, their program extracts that subset for use by the searching method.

2.3.2.2 Searching on multi-processor computers

To speed up database searching, and enable vastly increased computing power to be applied to the problem, methods have been devised which use parallel processing computers. These are systems which have more than one central processing unit (CPU) enabling the query sequence to be simultaneously compared to many database sequences with a consequent increase in speed. This has made it feasible to search databases using rigorous, dynamic programming methods [235-239]. The sensitive and computer intensive fragment length based method of pairwise comparison has also been implemented on parallel computers [240].

Deshpande *et al.*, [237] implemented their program in a modular fashion which enabled the searching algorithm to be changed. A parallel version of the Smith and Waterman algorithm [188] and the FASTA [227] algorithm were developed. After many changes, which resulted in a sixteen fold speedup in their FASTA implementation, their final code on a sixteen CPU computer was only slightly faster than that of a single CPU MIPS workstation. Miller *et al.* [241], with their parallel implementation of FASTA, found that when the word size (*k*-tuple) was set to one (which can be the equivalent of a rigorous aligning method), the program speeded up as more processors were added. However, when the word size was set to two, i.e., the more normal, faster setting, the program soon ceased to speed up as processors were added. In both cases, the already high performance of the FASTA method, the speed with which the database could be read in and the communication overheads in their machines, limited the possible increases in speed that parallelisation could achieve. As algorithms for searching databases become faster, other computational resources than computing power can become the limiting factors.

Many of these methods use dedicated parallel processing hardware, where considerable effort must be spent on adopting the database searching code to take full advantage of the machine's architecture. Barton [242] developed a program which reduces the effort required to get the code running on a parallel system, by using a virtual parallel

machine composed of a network of linked, single CPU computers. These computers must run NFS (network file system), i.e., have a shared file system and have the 'rsh' command (most machines that run the UNIX operating system have this capability built in and such programs are available for IBM PC's). A main controlling program does much of the parallelisation and little has to be done to convert the database searching code from a single CPU version to enable it to work with this system. In practice, the main program splits the database into chunks and then sends commands to each computer to run a database search on its own local database chunk. The main program checks to see when each computer finishes its task and will then send out another chunk to be searched until the entire database is searched. Once all the searches are finished, the results can be merged. Barton, using a Smith and Waterman algorithm showed a 4.5 fold increase in speed when five computers were linked.

2.4 Which methods should be used

The large range of different methods described above all attempt to align one sequence against another and careful consideration should be applied to the choice of methods to use.

2.4.1 Global or local, optimal or sub-optimal

Methods can be described as 'global' (e.g., Needleman and Wunsch or Argos algorithms) or 'local' (e.g., Smith and Waterman, FASTA or BLAST algorithms) depending on whether they align all the sequences together or find the most similar subsequences. In searching a database for homologous sequences to be used for homology modelling, globally based methods should be used initially. The alignment along the entire length of the sequence is needed for the swap in three-dimensional co-ordinates that homology modelling entails. If no hits are found then local methods could be used to search the database for a protein which had homologous subsequences. This

may enable a 'spare-parts' approach to modelling to be carried out [87] or other information about the sequence may be gathered. The SWEEP program has the useful ability of being able to consider the query sequence in segments so that a more comprehensive search for such subsequences can be undertaken.

Argos's method may be the more sensitive of the two global methods. The benefits of this method are threefold: extended similarity (i.e., local neighbour interactions) counts toward what is regarded as similar; accommodation of the varying sizes of structural and functional motif in proteins (e.g., long invariant regions in interior beta strands or short invariant regions at catalytic sites) due to the rerunning of the calculations using different fragment lengths ; and use of the highest score along the entire length of the fragment emphasises the areas of similarity and reduces any noise in the matrix.

The methods can also be described as 'optimal' (e.g., the Needleman and Wunsch or Smith and Waterman algorithm) or 'sub-optimal' (e.g., the FASTA, STATSEARCH and BLAST algorithms amongst others [201, 206, 207]), depending on the rigour of their mathematical analysis. All of the methods quantify alignments using numerical scores, mostly derived from the gap penalties and scoring matrices used. 'Optimal' methods guarantee to find the alignment of the 2 sequences that has the highest score while 'sub-optimal' methods, more concerned with speed, find high scoring alignments.

Concentrating on the mathematical basis on which these methods work leads to the choice of optimal methods, which give the highest score. However, even when the highest scoring alignment is found, it has been noted that this does not necessarily agree with the real, biologically based alignment [63, 200, 243, 244]. The aim of sequence comparison is to get the residues to line up according to the superimposition of the three-dimensional structures, which is felt to be the most biologically meaningful type of alignment [48]. Unfortunately, the present level of biological knowledge of the processes involved in determining protein structure and/or function and of how primary structure relates to tertiary structure are not detailed enough to let us adequately assess the sequence characteristics important for these processes. This incomplete model of

sequence similarity limits all sequence comparisons algorithms, which therefore cannot be guaranteed to produce alignments equivalent to structural alignments.

Even with this limited model, the relationship of the primary to tertiary structure that is implicit in the present aligning algorithms is naive. The use of a single set of gap penalties and scoring matrix incorrectly assumes that the reasons and potential for indels and mutations are constant and singular throughout the length of the sequences. Both indels and mutations are dependent on structural and functional constraints that vary throughout the sequence length [48-51] e.g., indels are almost always found in surface loops and coils rather than within secondary structural units [52, 53] and a mutational change at one position may be compensated for by related changes at another position [54, 55, 120, 126]. Lesk *et al.* [199] and Barton and Sternberg [200] allow for the gap penalties to change according to a limited definition of secondary structure (defined as helix or strand, not helix or strand and last two residues of helix or strand) but not for any other reason e.g., knowledge of active sites where indels would be inappropriate. Most of the sub-optimal programs, e.g., FASTA, SWEEP and BLAST, do not permit gaps in the alignments in their initial phase where they rank the hits in the database. Argos [213], by using a mutation matrix and five other matrices based on residues' physiochemical properties, allows for more than just a single residue property to be examined when calculating a score. However, no existing method at present allows for the use of differing matrices along the sequence lengths or for a change at one position to have an effect on the probability of a change at another position.

The choice of which gap penalties and scoring matrix to use is made when running these programs, although some allow only limited change (e.g., FASTP only uses the 250-PAM matrix). The choice of which model of gap penalty to use, e.g., affine, non-affine, concave, constant, etc., is usually constrained, with each method using one model. Length dependent gap penalties (non-affine) have been advocated as optimal [196], however others have pointed out that they may not be always be optimal,

especially when large gaps are involved [187, 200]. The method mostly used is non-affine gap penalties, primarily because of the algorithmical speed enhancements that these allow [245]. Work carried out on which values to use for gap penalties has shown that no single value is best [222], and that all are dependent on the choice of scoring matrix [209]. The choice of gap penalties is important, as setting the penalties too low can allow alignments between unrelated sequences and setting them too high can disallow alignments between related sequences. The choice of which scoring matrix to use is also open to debate, with recommendations including matrices that take account of both genetic likelihood and structural similarities between the residues [209], the 250-PAM matrix [200, 212, 246] and a PAM matrix chosen on the evolutionary distance between the sequences [235]. It is generally agreed that the identity matrix is a poor choice. Overall, the choice made, for each of the above programs, is important to the final alignment produced [196, 247]. As the scoring schemes differ, so the highest scoring alignment they produce differs (see alignments in Henneke [248]). For Argos's program, the choice of fragment length and the threshold above which to count the fragment as showing similarity, are also critical factors.

Taking all these points into consideration clouds the decision of whether an optimal or sub-optimal alignment method produces the best alignment. It may be arguable that optimal methods should be used in preference to sub-optimal methods, as within the boundaries of their methodology, they do produce the highest rather than a high scoring alignment. However, it has been shown that most methods can work well when aligning members of a family of proteins with high identity with problems occurring only when the identity is low, when no automatic method may give satisfactory results [249]. The main consideration, however, when database searching, is one of speed. The sub-optimal methods are two orders of magnitude faster than optimal methods, so are preferred in general use.

2.5 Statistical significance of alignments

After searching a database, each method displays a user-definable number of top scoring alignments. Because of the nature of the algorithms involved, searching a database with a query sequence unrelated to anything in the database would still produce this list of top scoring alignments. Attempting to reduce this noise factor has led to the development of methods which attempt to quantify how good an alignment is and to check whether it could be expected to happen by chance.

One method [250], involves randomising one or both of the sequences and then realigning using the same scoring scheme Repeating this enables a mean and standard deviation of the alignment scores to be calculated. If the initial alignment score is a certain number of standard deviations above the mean, then it can be said to be a significant alignment. Critics of this method have suggested that it over-emphasises the significance [251]. Further criticisms have included the fact that the random sequences used are not a valid comparison. Residues in proteins do not occur at random but have preferences for certain dimer and trimer oligopeptides [252] and also certain structural based preferences, e.g., hydrophobic residues in every third or fourth position in α-helices. The scores achieved are also dependent on the length of the sequences used. Also, different researchers have found that significance lies at different numbers of standard deviations above the mean, with some recommending three [209] or four [213, 253] and others at least six [254]. The number of randomisations to use is given as twenty-five by Feng *et al*. [209], however Barton and Sternberg [254] recommend using at least sixty to achieve consistent results.

In an attempt to avoid these problems and also because calculating such statistics in a database search would be inconsistent with the need for speed, significance is sometimes expressed as *z* scores [223, 250]. These rely upon the assumption that a query sequence is randomly related to most sequences in the database enabling a random distribution to be obtained. The scores of the query sequence against the entire

database are used to calculate a mean and standard deviation. The z score can then be calculated by subtracting the mean from the alignment score and then dividing the result by the standard deviation. Values of three or more indicate a significant alignment [223, 250]. The distribution obtained, however, is only approximately random, thus z scores can only be used as a guide to significance. The main problems with this method occur if the composition of the query sequence is markedly different from the database average; the average score is lowered while sequences that share the composition have raised scores. Pearson and Lipman with their FASTA program note that this method produces further divergence from a random distribution thus weakening the usefulness of z scores. They recommend instead randomising the sequences (between 50 and 200 times) and using the highest score as a guide - if it approaches or surpasses the observed score then the similarity is not significant.

Collins *et al.* [235] use a similar approach where the lowest 97% scores from a database search are used to calculate the expected frequency of occurrence of any score. Their method calculates the probability that a given score would be obtained if the query sequence was used to search a identical size database of unrelated sequences.

STATSEARCH includes an explicit statistical approach to database searching with a test at the one percent significance level on every sequence it searches against. The distribution of the scores for random sequences of the same length and composition is used to assess the significance of the ungapped local alignment of the query and database sequences. Criticisms of this approach include: the alignment method does not allow indels; it has flaws inherent in the random composition of sequences approach; and a further problem in that the one percent significance level means that in a database of 100,000 sequences (present protein databases are approaching this size), 1000 sequences would be acceptable by chance.

The BLAST program includes a significance factor derived from the work Karlin and Altschul [231] carried out on the distribution of maximal segment pair (MSP) scores in random sequences. This work enables the approximate score an MSP should have to be

distinguishable from a chance similarity in the database to be calculated. Although only strictly applicable to random sequences, the Karlin and Altschul method shows that, when using a PAM matrix (certain others can also be used), high scoring chance MSPs should resemble MSPs that reflect real homology. This method has probably the most rigorous statistical approach, although again it applies only to sequences with no gaps.

When confronted with a long list of hits from a database search, a statistic to discern whether an alignment is biologically significant or not would be a useful tool. In practice none of the above methods fully qualify. If such a tool did exist, then the database searching methods would include it and would only present a list of significant hits. Although some of the methods are statistically rigorous, their statistical significance does not necessarily imply biological significance. The significance quoted a is useful aid but ultimately one must rely on the information that the alignment confers to the biologist rather than whether it is improbable or not.

2.6 A strategy for database searching

In consideration of the above, a general strategy towards searching for homologous sequences can be formulated. The initial step is to rapidly identify sequences with strong homology to the query sequence. To this end, a fast database searching method should be employed. At Glasgow, the available methods are FASTA and BLAST. As the databases differ in content and size, all available databases should be used to ensure a thorough search. The different possible methods do have differences in their sensitivity and selectivity - Argos *et al.* [255] generally rank BLAST as more sensitive than FASTA, so the best strategy is to search the databases with all available methods to ensure that no algorithmical quirk leads to any false negatives. Pearson and Lipman [227] recommend the use of the FASTA program instead of any other of their methods e.g., FASTP, due to its increased sensitivity, and this recommendation was employed. The gap penalties, scoring matrix and other parameters used should be varied, where

possible, to investigate any differences in the results obtained and to increase the chance of detecting distant homologies. Lipman and Pearson recommend starting a search with their program with a *k*-tuple of two and then reducing this to the more sensitive value of one [225]. It is likely that all methods find the top scoring, most similar sequences although they probably vary in what counts as a less well scoring alignment. With this in mind it is recommended to produce a large list of output and to study the medium hits on the list using any available biological information to determine whether an alignment is reasonable. As the shikimate enzymes each have more than one known sequence, another useful method is to multiply align the sequences and then generate a profile that can be used with the PROFILESEARCH program. This is a sensitive method for detecting distant relationships. It may also be desirable to search against DNA sequence databases to ensure that a possible homologue that has not yet been translated is not ignored. This possibility is remote though, as the protein sequence databases do include translations from DNA databases. Searching of DNA sequence databases using protein sequences can be done using the programs TBLASTN (of the BLAST suite) and FASTA.

If strong homologies are found, more sensitive, optimal methods can then be employed to see if the alignments produced can be extended. Methods include the GAP program, from the GCG package, which uses the Needleman and Wunsch algorithm and the Argos sequence fragment based approach. In the present analysis, the GAP program is used due to its availability.

Short regions (e.g., under ten residues long) with many identities should be treated with caution as their significance may be low. M^{c}Caldon and Argos [256] found that these matches could occur between clearly unrelated sequences, and it is also known that neither identical pentapeptides [117, 257] nor heptapeptides nor hexapeptides [258] necessarily share the same structure. However, such homologies could have a biological basis e.g., if the proteins shared an active site, so they should be checked.

If no particularly strong homologies are found, the top scores from a fast method should be analysed using the GAP program to check that the alignments cannot be extended into a more useful, longer alignment.

2.7 Results and discussion of database searches

To test for sequences of use in homology modelling, database searches are first carried out on release 8.0 of the NRL_3D [259]. This is a protein database of sequences with a known structure. This is converted from PIR format to GCG format using the GCG program DATASET [178]. The FASTA and BLAST programs are run against this database with each known sequence of the shikimate enzymes e1 to e5.

The FASTA program is run with two different gap penalty settings (table 2.3) and the more sensitive, *k*-tuple value of one. Default settings are used for other FASTA parameters. The number of matches between the query sequence and database sequences that are displayed is set to one hundred. The parameters of the BLAST program use their default settings. The maximum number of matches between the query sequence and database sequences that are displayed is set to two hundred and fifty.

Output from BLAST lists statistically significant matches between query sequence and database sequences. For seven of the enzyme sequences, no such matches are found. For the other sequences, matches are found which show moderate sequence identity, up to fifty eight percent, between the query and database sequence. These matches are between short subsequences of the query and database sequence, with the longest match being forty nine residues in length, corresponding to less than twenty percent of the query sequence length. A summary of the results found by the BLAST program are shown in table 2.4, with the full BLAST results for two e4 sequences, e4-P08329 and e4-P10880, listed in appendix B. The FASTA output lists the top one hundred matches found between query sequence and the database sequences, whether they are statistically significant or not. Varying the gap penalties used in the FASTA runs has

FASTA

parameter name	Set one	Set two
gap creation	12.0	5.0
gap extension	4.0	1.0
k-tuple	1	1

GAP

gap penalty name	Set one	Set two	Set three	Set four
gap creation	3.0	5.0	1.0	3.0
gap extension	0.1	0.5	0.1	0.5

Table 2.3

The parameters used for the alignment and database searching programs. FASTA, a database searching program, is run using two sets of gap penalties, with the *k*-tuple set to one (default setting is two). The pairwise alignment program, GAP, is run using four sets of gap penalties. The first set of gap penalties are the default values for the programs.

Enzyme	P08566	P07547	e1 P07639	e2 P05194	e2 P24670	e3 P15770	e4 P08329	e4 P10880	e4 Q00497
Significant hits	12	1	0	16	0	1	3	40	1
Number of identities	9	12	n/a	16	n/a	19	9	7	7
Hit length (residues)	16	34	n/a	49	n/a	34	24	12	14
Identity (%)	56	35	n/a	32	n/a	55	37	58	50

Enzyme	e5 P07638	e5 P07637	e5 P10748	e5 P11043	e5 P05466	e5 P12421	e5 P20691	e5 pea
Significant hits	0	0	0	0	0	4	6	2
Number of identities	n/a	n/a	n/a	n/a	n/a	13	9	14
Hit length (residues)	n/a	n/a	n/a	n/a	n/a	24	27	39
Identity (%)	n/a	n/a	n/a	n/a	n/a	54	33	35

Table 2.4

Results of searching the NRL_3D database with enzymes of the shikimate pathway using the BLAST program. The number of matches between query sequence and a database sequence, that BLAST reports as significant, are shown for each shikimate pathway enzyme sequence. For each enzyme, where significant matches are found in the database, the following values are shown for the match with the highest percentage identity: the number of identical residues in the match between the query and database sequence; the length of the match between the sequences; and the percentage identity score.

little effect on what database sequences are matched to the query sequence, however, a longer overlap is produced with the lower gap penalty setting. Examination of the data reveals that matches longer than forty five residues have a percentage sequence identity, ranging from fifteen to twenty five percent. Shorter matches can show higher percentage sequence identity values, with the highest being sixty three percent over a eleven residue match (e4-P08329 to porcine citrate synthase, nrl:1cts).

The reported matches between query and database sequences from BLAST, are too short to be of use in homology modelling, while FASTA does find longer matches but with percentage sequence identity values too low to be of use. The recommendation to use the GAP program to check the alignments produced by the database searching algorithms and to test if homologies can be extended into longer, more meaningful alignments, is followed.

Few enough matches are reported by BLAST to enable all of them to be tested. The results from FASTA are too numerous to enable this approach to be taken, so a selection of sequences is taken. Each sequence set is run through GAP four times, using different gap penalty settings (table 2.3). The statistical significance of the resulting alignments is calculated by taking the database sequence, changing the order of its residues at random, and re-running GAP. This process is repeated one hundred times. Two matches reported by BLAST show statistically significant results when GAP is used to extend the alignments, with the other matches scoring within one standard deviation of the mean of the random scores, or lower. The statistically significant alignments are between e4-P10880 and porcine adenylate kinase and e4-P10880 and human H-ras p21 protein (figure 2.3). Similarly, for FASTA, only the initial matches found between adenylate kinase or H-ras p21 protein and the sequences of enzyme e4 are shown to be statistically significant when complete alignments are generated using GAP (table 2.5).

Not all of the e4 sequences show statistically significant alignments: the score for e4-P07547 aligned against adenylate kinase is within two standard deviations of the

GAP of E4-P10880 x adenylate kinase

```
          .          .          .          .          .
  1 ......MTEPIFMVGARGCGKTTVGRELARALGYEFVDTDIFMQH..... 39
          ... ||:|||:.|:||.| . .:.. .||. :.|: ::.
  1 MEEKLKKSKIIFVVGGPGSGKGTQCEKIVQKYGYTHLSTGDLLRAEVSSG 50
          .          .          .          .          .
 40 .TSGMTVADVVAAEGWPGFR...RRESEALQAVATPNRVVATGGGMVLLE 85
    ..|. :.:::. :.:..:     . .:|: | .....:. .:|    :.
 51 SARGKMLSEIMEKCQLVPLETVLDMLRDAMVAKVDTSKGFLIDGYPREVK 100
          .          .          .          .          .
 86 QNRQFMRAHGTVVYLF...APAEELALRLQASPQAHQRPTLTGRPIAEEM 132
    |. :| |  | ...|:   |.:|.:. ||  ..:. .|.. .: .| . :
101 QGEEFERKIGQPTLLLYVDAGPETMTKRLLKRGETSGRVDDNEETIKKRL 150
          .          .          .          .
133 EAVLRERE...ALYQDVAHY.VVDATQPPAAI...VCELMQTMRLPAA 173
    |. .:. |   |:|:. :    |:|......:   ||. ::|::
151 ETYYKATEPVIAFYEKRGIVRKVNAEGSVDDVFSQVCTHLDTLK.... 194

          Quality:   74.0   Percent Identity: 20.710
Average quality based on 100 randomizations: 55.3 +/- 2.7
```

GAP of E4-P10880 x H-ras p21 protein

```
          .          .          .          .          .
  1 MTE.PIFMVGARGCGKTTVGREL.........ARALGYEFVDTDIFMQHT 40
    ||| .:.:|||||.|..:. :|          ...:: .: .  :: ..|
  1 MTEYKLVVVGARGVGKSALTIQLIQNHFVDEYDPTIEDSYRKQVVIDGET 50
          .          .          .          .          .
 41 SGMTVADVVAAEGWPGFR...RRESEALQAVATPNRVVATGGGMVLLEQN 87
    : :.: |..:.|::.::|    .|.:|:: .| . |.. . ::    . ||
 51 CLLDILDTAGQEEYSAMRDQYMRTGEGFLCVFAINNTKSFEDIHQYREQI 100
          .          .          .          .          .
 88 RQFMRAHGTVVYLFAPAEELALRLQASPQAHQRPTLTGRPIAEEMEAVLR 137
    :... ..:..: |.:   :|| |  .|.||:: :   | |.   :|.  :
101 KRVKDSDDVPMVLVGNKCDLAARTVESRQAQDLARSYGIPY...IETSAK 147
          .          .          .
138 EREALYQDVAHYVVDATQPPAAIVCELMQTMRLPAA 173
    .|::: :..  .| :  |.
148 TRQGVEDAFYTLVREIRQH..................  166

          Quality:   68.2   Percent Identity: 22.222
Average quality based on 100 randomizations: 50.6 +/- 2.9
```

Figure 2.3

Statistically significant alignments found during database searching. Alignments are carried out using the GAP program, with gap weight of 3.0, gap length of 0.1. In both alignments, the shikimate kinase sequence is uppermost. The statistics gathered from GAP are shown below each alignment. The adenylate kinase sequence is from porcine muscle (database reference nrl_3d:3adk), and the H-ras p21 protein sequence is from human (database reference nrl_3d:421p).

shikimate enzyme	adenylate kinase			H-ras p21		
	Score	Statistics	Identity (%)	Score	Statistics	Identity (%)
e4-P08329	75.9	56.3 ± 3.0	19	61.1	51.7 ± 3.1	19
e4-P10880	74.0	55.3 ± 2.7	21	68.2	50.8 ± 2.8	22
e4-Q00497	82.1	67.1 ± 3.1	23	56.2	56.9 ± 2.6	21
e4-P08566	76.4	68.2 ± 2.9	22	65.5	57.1 ± 2.7	21
e4-P07547	72.6	67.6 ± 2.9	21	59.9	56.8 ± 2.7	19

Table 2.5

Comparison of sequences of shikimate enzyme e4, against adenylate kinase and H-ras p21 protein. Comparisons are carried out using the GAP program, with gap weight of 3.0 and gap length of 0.1. The score for each alignment is shown, along with the mean and standard deviation of the alignment score. These statistics are calculated for each pair of sequences, by running GAP one hundred times with the randomised sequence of adenylate kinase or H-ras p21. The percentage sequence identity between the sequences is also shown. The adenylate kinase sequence is from porcine muscle (database reference nrl_3d:3adk), and the H-ras p21 protein sequence is from human (database reference nrl_3d:421p).

mean of the random scores and e4-Q00497 aligned against H-ras p21 protein has a lower score than the mean of the random scores. Yet, the homology between these proteins is an interesting result. All three, shikimate kinase, adenylate kinase and H-ras p21 protein, utilise adenosine triphosphate (ATP). Walker *et al.* [260] identified conserved patterns of residues in ATP binding loops, primarily the sequence G-X-X-X-X-G-K-(T) (where X is any amino acid and threonine is not conserved in all examples). This sequence motif is encompassed in the alignments of the matches found by BLAST and FASTA (appendix B). This common functionality and the presence of the ATP binding loop motif, suggest that the alignments of these proteins are biologically, as well as statistically, significant.

It is likely that the structure of the ATP binding loop is conserved between all these proteins, as it is between adenylate kinase and H-ras p21 protein [261], however, it is not necessarily the case that the rest of the structures will be conserved. The presence of identical residues throughout the alignments between shikimate kinase, adenylate kinase and H-ras p21 protein (figure 2.3), suggest that the structures may be conserved, as does the fact that the sequences are similar in size: adenylate kinase has one hundred and ninety four residues; H-ras p21 protein has one hundred and sixty six residues; and shikimate kinase sequences are roughly one hundred and seventy residues (table 2.2). The main drawback is the low sequence identity found between these sequences, which range from nineteen to twenty three percent. At this level, Chothia and Lesk [103] showed that large changes are seen in the structures of homologous proteins. Also, outside the ATP binding loop motif, there are no other strongly homologous regions, suggesting the structural homology may be confined to the ATP binding site alone. If homology modelling was to be undertaken, obtaining an accurate alignment of topologically equivalent residues would be problematic. No method is yet known, which can account for the large structural changes expected at this level of identity.

On balance, there is not enough information from the alignments of primary sequence alone, to warrant homology modelling shikimate kinase to either adenylate kinase or H-ras p21 protein.

Database searches are next carried out on the protein sequence databases, SWISSPROT and PIR. The FASTA and BLAST programs are run, with the settings as detailed above, with each known sequence of the shikimate enzymes e1 to e5. The purpose is to identify strongly homologous sequences, which can be added to alignments of the shikimate enzymes.

FASTA lists one hundred matches per query sequence and BLAST finds statistically significant matches for each enzyme sequence (table 2.6). Due to this large number of matches, the GAP program is run, using the settings detailed above, on selected sequences. Statistically significant alignments are obtained: between shikimate kinase (e4) sequences and other members of the adenylate kinase family, (e.g., GAP gives the alignment of e4-Q00497 to human adenylate kinase (swissprot:kad1_human) a score five standard deviations above the mean scores of one hundred randomisations); and between shikimate dehydrogenase (e3) and quinate repressor (swissprot:qa1s_neucr) and quinate dehydrogenase (swissprot:dhqa_neucr, swissprot:dhqa_aspni). The former relationship is expected, and leads on from the results obtained searching the NRL_3D database. The latter relationships have been reported [35, 119] and although the sequences show similarities, they are structurally different.

To ensure a thorough analysis, database searches are repeated with FASTA and TBLASTN, using the EMBL database of DNA sequences. Varying the database between SWISSPROT, PIR and EMBL finds little difference in the sequences reported as matching. This is an expected result as the difference in sequence content between the databases is minor. Altering the gap penalties for FASTA mostly influences the length of overlap in the matches between query and database sequence, and the order in which these matches are reported, rather than finding different sequences matching.

Enzyme	P08566	P07547	e1 P07639	e2 P05194	e2 P24670	e3 P15770	e4 P08329	e4 P10880	e4 Q00497
Significant hits	63	41	17	15	18	18	57	50	35
Enzyme	e5 P07638	e5 P07637	e5 P10748	e5 P11043	e5 P05466	e5 P12421	e5 P20691	e5 pea	
Significant hits	21	26	40	40	38	42	38	58	

Table 2.6

Results of searching the SWISSPROT database with enzymes of the shikimate pathway using the BLAST program. The number of matches between query sequence and a database sequence, that BLAST reports as significant, are shown for each shikimate pathway enzyme sequence.

The main finding from database searching is that strong homology is only found between the query sequence and the sequences of protein family members. Scores range from double to almost an order of magnitude greater, than matches to non family member sequences, e.g., e3-P07547 to e3-P08566 has a FASTA initn score of 451 while e3-P07547 to the highest scoring non family sequence, RNA replicase polyprotein (swissprot:polr_tymva), has an initn score of 124.

Chapter 3 Multiple alignment of the shikimate enzymes

3.1 The role of multiple sequence alignment

The next stage in investigating the structure of the shikimate enzymes is the extraction of information from the sequences. A useful procedure for this, and following the molecular modelling flowchart in figure 1.2, is to align the sequences judged to be similar after completing the database searches. In the case of the shikimate enzymes, only sequences belonging to the same enzyme were found to be similar and for each of the enzymes, more than two sequences are known, requiring the use of multiple sequence alignment methods. The aim of multiple sequence aligning is to generate alignments that would agree with those produced by superposition of the three-dimensional structures. These are felt to be the most biologically meaningful type of alignments [48].

A benefit of using multiple sequence alignment methods, is that the accuracy of alignments produced, measured by the number of residues correctly aligned according to a structural based alignment, is generally greater than if a pairwise alignment method is used [64, 254, 262]. Phylogenetic trees calculated from multiple alignments can also have more plausible topologies than trees derived from separate pairwise alignments [263]. An explanation for the increase in the accuracy of these methods when multiple sequences are used, is based on the knowledge that an alignment of homologous sequences will share approximately the same tertiary structure although with differing primary structures [103] - the homologous proteins will have differing approaches to achieving the same conformation. 'Averaging' over the sequences can be expected to reduce the importance of the 'noise' content any one sequence introduces and increases the importance of the common features.

3.2 Methods used in multiple sequence alignment

A number of differing approaches to multiple sequence aligning have been described. The approaches can be divided into two groups, depending on whether the sequences to be aligned are considered simultaneously or not. Simultaneously comparing all the residues in all the sequences is, as described below, a very computer intensive procedure. So far, this approach can only be applied to aligning a few, short sequences. To align simultaneously more, and longer sequences, methods have been developed which consider segments of the sequences at a time and generate an overall alignment from analysis of the alignment of the segments. Another method used involves summation of pairwise dotplots to produce a dotplot for the multiple sequences. The non-simultaneous approach is based on progressively pairwise aligning the sequences, building up the results into the final multiple alignment. These methods are described in more detail below.

3.2.1 Simultaneously aligning sequences

As previously detailed in chapter 2.3.1, a large body of work exists on the problem of pairwise sequence alignment. The dynamic programming method of Needleman and Wunsch [184], although primarily used for pairwise aligning is not limited to considering only two sequences at a time and can be used for simultaneously aligning multiple sequences. When dealing with more than two sequences though, a problem lies in the extensive computer requirements of the method - for three sequences, time requirements are proportional to the fifth power of the number of residues. Murata *et al.* [264] describe an extension to the method, applicable for three sequences, which reduced the time requirements of this approach at the expense of increased memory requirements. The increased memory requirement limits this method to peptides or proteins under two hundred residues in length. The Murata *et al.* method produces the highest scoring alignment using its scoring scheme, which included the use of the same

gap penalty regardless of whether the gap occurs in a single sequence or as an overlapping gap in two sequences and not scoring for pairs of matched residues opposite a gap. Including the scores of such pairwise matches can result in a higher scoring alignment [265] and the method was refined to include such pairwise scoring [187]. The size of sequences that could be compared was also increased to 900 residues in length by limiting the search space through the matrix. This technique of reducing the number of cells in the matrix which are examined - used in database searching algorithms to increase speed, and here to increase the number of sequences that can be aligned - has been shown to be capable of still producing the highest scoring, optimal alignment [266]. A dynamic programming method that uses such an approach, enabling four or five sequences to be optimally aligned, has been described [267, 268]. This method allows for further heuristic reductions to be made to the search space, permitting up to six to eight sequences to be aligned, although the alignment may no longer be optimal. The boundaries of the matrix areas to be searched are calculated from initial pairwise comparisons of the sequences. However, even with four or five sequences, this method may be impractical due to excessive time requirements, if the pairwise alignments are inconsistent enough to require a large search space.

This limitation in the number of sequences that can be aligned is relevant in the case of aligning the shikimate enzymes, where ten sequences are known for the enzyme 5-enolpyruvylshikimate 3-phosphate synthase (EPSP synthase or e5). These multiple alignment dynamic programming methods share the flaws already discussed for the pairwise algorithms, namely their simplified ties to the presently incomplete model of evolution. Further considerations also apply in what score the algorithms seek to minimise or maximise and their treatment of gaps. With two sequences, dynamic methods can use the similarity or difference between the sequences for scoring. This can be extended to multiple sequences by summation of the pairwise similarity/difference scores. This will tend to maximise the number of alignment positions where most or all of the sequences agree. An alternative approach exists where the multiple alignment is used to produce an evolutionary tree. Ancestor

sequences are calculated and the sum of the pairwise alignments of these ancestor sequences are used for scoring. This will bias the alignment towards minimising the overall number of mutations from the ancestor sequences. The different definitions of score can produce different alignments even when other parameters are kept the same (e.g., see alignments in [268]). The gap problem in multiple aligning lies in the rise in possible types of gaps and overhangs that can occur and how these are scored. The types of gaps and overhangs that can occur in aligning two and three sequences are illustrated in table 3.1. Whereas for pairwise aligning, the gap topology can be explained by a single insertion or deletion event, with three sequences this is not always the case. A useful ability missing in these methods, would be to take the evolutionary basis of the gap topology (i.e., whether they are explicable by a single or multiple insertion or deletion event) into consideration when calculating the gap penalty of the alignment. This complexity in the evolutionary model underlying the gap topology increases as more sequences are compared. The possible ways gaps can be scored for pairwise alignments have been described above and include constant, concave and length dependent penalties. With multiple alignments, the possibilities greatly increase and methods include: summation of the number of gaps in all pairwise comparisons; a column which has any gaps adding to the gap score; and a single score for a gap where the first and last column with no gaps, define the start and end of the gap (this is covered in more detail in Altschul and Lipman [245]). No method has been demonstrated to produce 'better' alignments, although some have clearer ties to a biological basis.

Another approach to simultaneous alignment of sequences uses sub-sequences or 'segments'; either by the search for such segments in the sequences ,or by analysis of the sequences in segments. Sobel and Martinez [269] describe a method that searches sequences to find common sub-sequences, greater than a user specified length. These segments are then joined to define the multiple alignment with gaps inserted to enable the segments to match up. The method pieces together the segments that give the highest score, where score is defined as the sum of the length of the matching segments

Caused by a single indel event	Alignments of two sequences	Alignments of three sequences
Yes	..GG-G.. GG.. ..GTWG.. GTWG..	..GG-G.. ..GG-G.. GG.. GG.. ..GTWG.. ..GT-G.. WG.. GTWG.. ..TTYG.. ..TTYG.. TYTC.. TYTG..
No		..GG-G.. GG.. ..G--G.. TWG.. ..TTYG.. TTYG..

Table 3.1

Possible types of gaps and overhangs that can occur between two and three sequences. The possibility that a single insertion or deletion (indel) event in an ancestral protein could account for such, is shown. Gaps are shown by '-' and the continuation of the sequences by '. .'. The overhangs shown can be at the amino or carboxy terminus.

minus a gap penalty. The fact that only identical segments that are present in all the sequences are matched, limits this method's ability when aligning distantly related sequences where such events may be rare. Other drawbacks include having to pre-specify the length of segments to be matched and the use of the heuristic scoring and gap placement schemes.

Johnson and Doolittle [265] employ a window based approach to build up the multiple alignment from progressive comparisons of segments. A window is moved along the sequences one residue at a time, with the best alignment from each window determining the alignment of one residue from each of the sequences. The best segment alignment is calculated from summation of the pairwise scores in the alignment minus a gap penalty. This method does not always find the overall highest scoring alignment. To reduce the computational requirements, limits are introduced into the range of comparisons any segment may participate in, however, the use of this method with more than four sequences is impractical. The window length chosen determines the maximum length of gap which can be introduced and which parts of a sequence are compared to any other. With sequences with long insertion or deletion events or that are distantly related, the choice of window length will become very important.

To enable more sequences to be aligned, Bacon and Anderson [208] describe a method which greatly reduces the number of comparisons made. The best segment alignments, generated from aligning two of the sequences are stored. These stored segment alignments are then compared in turn against the other sequences with the stored alignments being replaced after each comparison with the new best segment alignments. Although this method enables more sequences to be compared, as the number of sequences increase, the alignments produced become more dependent on the order of the sequence comparison - if a region of homology is weak in the first sequences, it may not be picked up in later sequences where the homology is strong. The segment alignments generated do not allow for gaps, which is of concern if distantly related sequences are being aligned.

The method used by Santibanez and Rohde [270] to search for identical segments in all or some of the sequences is based on the fast *k*-tuple algorithm of Wilbur and Lipman [271]. Matching segments present in all sequences are used to form an initial multiple alignment. The other segments found are then added to this initial alignment. The order of adding segments depends of the number of sequences they are found in, with those found in the most sequences being added first. With distantly related sequences, so few hits may be found as to invalidate the initial framework alignment found, leading to a poor overall alignment.

Vingron and Argos [262] divide the alignment procedure into two steps depending on how closely related the sequences are. Closely related sequences are aligned by consideration of *k*-tuple matches amongst the sequences with the regions in between matches being aligned by the profile analysis method [233], a modified dynamic programming method which uses information from all sequences being compared. Such groups of sequences are then aligned, two at a time, using the profile analysis method. This is not a strictly simultaneous aligning method, except for closely related sequences, although it does provide information from many sequences to be considered at a time. An advantage is that a virtually unlimited number of sequences can be aligned although there is a limit of ten sequences in any single group.

Chappey *et al.* [272] use an extension of a fast method for finding repeats in sequences [273] to build up a database of identical segments present in at least two of the sequences. These segments are then aligned together, starting with the highest scoring segments, to build up a multiple alignment. The segments can be scored on their frequency of occurrence, their length or a function of frequency and length. They recommend changing the scoring scheme depending on how related the sequences are as the schemes reflect different possibilities. The correct setting of this parameter can be seen to be important as changing the scoring scheme can alter the alignments.

Deperieux and Feyntman [274] extend the segment comparison method to include matches between non-identical residues. Each residue can be characterised by a number

of variables (e.g., physiochemical properties of the residues), and regions in the sequences which have similar properties in all sequences are found. The search for such matches employs a window based approach with threshold values for similarity being determined by the user. The matches found are then pieced together into a multiple sequence alignment, starting with the highest scoring set of segments as determined by one of a possible two methods. Gaps are introduced into the areas between matching segments to enable them to align. The matches automatically chosen for the multiple alignments may not be the best available and the user is recommended to critically evaluate the choices made. The risk of inappropriate matches being chosen increases with the length and number of sequences. The method most likely to find good choices for the multiple alignments is computer intensive with several hours of computer time needed to align three sequences. A useful aspect of the method is its delineation of regions it predicts to be structurally conserved.

The limitation that matches must involve identities or be present in all the sequences are removed in the MACAW program [249]. This uses pairwise comparison of all sequences to find segments that match above a user-specified threshold. The sequences are then searched to see if these segments can be found in other sequences. Gaps are not allowed in the segments used for matching. A heuristic approach is used to determine which segments are reported to the user, although the user can specify that segments must be present in at least a certain number of sequences. The segments can have their limits altered - the method does not guarantee finding the 'best' segments - and they can be linked to produce a multiple alignment. These operations are carried out using a graphical user interface. The multiple alignment is produced by adding gaps into the regions in between the segments.

The evolutionary changes that can be present in a family of sequences are mutations, insertions and deletions, and programs attempting to align sequences should be able to take these into account. Segment comparison methods that only look for identical segments are thus ignoring mutational events. Homologous regions which contain

mutations are ignored by such programs, lessening the validity of any multiple alignment produced. This problem is exacerbated with distantly related sequences where homologous regions are more likely to contain mutations, although it will still be present with closely related sequences. For example, Dayhoff *et al.* used closely related sequences with greater than eighty-five percent identity to calculate the PAM matrix [216]. The regions used to calculate the accepted mutations, aligned by 'eye', would be ignored by such programs, thus raising the possibility that alignments so produced could be worse than manually created alignments. Another drawback with some of the segment searching programs is their requirement that segments found should be present in all sequences, with the consequence that a region conserved in all sequences bar one would be treated as a non-homologous region. With distantly related sequences, the possibility that a region is not conserved in all sequences is increased, which may in turn lead to a poor multiple alignment. It is also an area of concern how these methods turn the sets of segments deemed homologous, into a complete multiple alignment. For most of the methods, gaps are placed with the single criterion of enabling the segments to match up. This is a biologically unrealistic method of dealing with insertion and deletion events. Some of the methods do have the useful ability of enabling the user to identify similar regions of the sequences. In cases where the sequences are so divergent as to make their aligning problematic, useful information can still be extracted from the sequences with the use of such methods.

A further approach to simultaneous aligning is based on extensions to the dotplot method, previously described in the pairwise aligning chapter 2.3.1.3. Dotplots place the sequences to be compared along the rows and columns of a matrix with scores between the residues being inserted into the respective cells. With multiple sequences, true simultaneous dotplots would require a n dimensional matrix, where n is the number of sequences. This rapidly becomes impractical as the number of sequences grows, as well as generating hard to visualise results. Current methods therefore rely on pairwise dotplots being carried out between the sequences, with the results of all of the resulting dotplots being summed in some way. The dotplots produced cannot simply be

superimposed due to the insertions and deletions that may occur between the various sequence pairs. One method to deal with this was described by Vihinen [275, 276] who used one of the sequences as a reference that the others are aligned against. Dotplots are produced using a window, with the score threshold used to determine similarity for that window being chosen by the user. The cells at equivalent positions in all of the dotplots are then summed and if they are higher than a second score threshold, again chosen by the user, they are added to a final dotplot. The choice of which sequence is picked as the reference sequence may have bearing on the final result because regions where insertions and deletions occur are not included in the dotplots to ensure consistency. In distantly related sequences, where the patterns of insertions and deletions is likely to be more variable between the sequences, more of the sequences would not be included in the dotplots. This method will also be sensitive to the parameters chosen for the dotplots and for the initial pairwise alignments, carried out using the Needleman and Wunsch algorithm.

Vingron and Argos [277] use a process of calculating the dotplots between all sequences with a modification of matrix multiplication to sum the dotplots together. Intermediate 'estimated' matrices are calculated which, when overlapped with the actual dotplots found, are used to delineate consistently aligned segments. Two methods can be used for this delineation; points are placed in the final dotplot if they are found in all dotplots or only in some. The dotplots can be calculated either using a method which includes 'sub-optimal' and 'optimal' points [244] or according to the 'sensitive' method previously described by Argos [213] where multiple window lengths and differing scoring matrices are used. The choice of significant segments can be interpreted automatically or with user interaction. The automatic method may sometimes result in too few points being found in the final dotplot, especially if the sequences are distantly related, thus necessitating user interaction. The segments are then joined together using the same method as described in their previous paper [262] which employs profile analysis to place the gaps. As well as being used for multiple sequence aligning, this method can be used to identify similar regions of the sequences.

3.2.2 Extension to pairwise aligning methods

The other main approach to multiple sequence aligning is based on pairwise aligning the sequences followed by compilation of these alignments into a composite. The main basis of the pairwise aligning methods has been previously dealt with (see chapter 2.3.1.1). A number of methods have been described which are detailed below following an approximate chronological order.

The MULTAN program [278, 279] proceeds by aligning each sequence against a consensus sequence. Initially, one of the sequences is chosen at random to act as a consensus sequence. During alignment a new consensus sequence is generated at each position by using the residue which gives the highest summed match when compared to the two residues being compared or a gap is inserted if the score is not greater than a user specified threshold value. Gaps inserted into the consensus sequence are inserted into all sequences bar the one being aligned This process repeats until no change is seen in the consensus sequence.

The approach taken by Barton and Sternberg [254] aligns the first two sequences, then the third is aligned against the alignment of sequences one and two, the fourth against the alignment of sequences one, two and three, and so on, until all the sequences have been aligned. Iteration is then performed to produce a final alignment. In this method, the most similar sequences are aligned first with the least similar sequences being aligned last. The score of the alignment, divided by the length of the shortest sequence, is used to generate the sequence ranking score. The alignments are carried out using the global, optimal, Needleman and Wunsch algorithm. The score for aligning a sequence against a number of sequences is calculated by taking the average score between all the residues being compared, which ensures that information from all sequences is included. The iteration step involves realigning each sequence against the complete alignment minus that sequence. These iterative steps can be conducted until no further change is seen in the alignment. A similar approach is used in the ALIGN program of

Taylor [280], except that the multiple alignment is produced at the end of the pairwise comparison stage - the iterative stage is missed out.

Henneke's [248] variation of Barton and Sternberg's method enables the user to weight residues, biasing the alignment towards matching these residues. Where the secondary structure is known for any of the sequences, that information can also be included with the use of different gap penalties for placing a gap against regions in the middle or end or not in a secondary structure. It is noted that the use of biasing the alignment towards aligning specific residues can lead to improved alignments although the correct choice of scoring parameters (gap penalties and scoring matrix) can have a bigger effect.

Feng and Doolittle [130, 263] use pairwise alignments between all the sequences to determine an approximate phylogenetic tree. The order of progressive pairwise aligning follows the order as defined in the tree diagram, so closely related sequences are aligned followed by progressive aligning of such pre-aligned clusters or of more distantly related sequences. The alignments use the Needleman and Wunsch algorithm. When adding a sequence to an already aligned pair of sequences, two possible alignments are scored and the highest scoring alignment is used. For example, when aligning Sequence C to the already aligned pair A and B, the score produced is compared to the alignment score produced between sequence A to aligned pair B and C.

A similar approach to that of Feng and Doolittle is used by the program PILEUP [178] except that when aligning a sequence to a cluster, the two possible ways of aligning are not used. PILEUP extends the method by enabling any scoring matrix to be chosen by the user .

The initial step of finding out how the sequences are related to each other is normally the slowest step in these methods and becomes increasingly so as the number of sequences rise. Higgins and Sharp [281] developed the CLUSTAL program, based on Feng and Doolittle's method [263], which increases the speed at which the multiple alignment can be calculated. Their approach lies in replacing the optimal Needleman

and Wunsch algorithm with the faster, sub-optimal Wilbur and Lipman algorithm [271] in both calculating the initial ranking pairwise alignments and for the progressive pairwise aligning. The Wilbur and Lipman algorithm is modified so that it can accommodate conservative substitutions and a slight degree of mismatch. These slow the method down compared to the original Wilbur and Lipman method but improve the sensitivity of the method. Compared to using the optimal pairwise aligning method, this method sacrifices sensitivity for speed and this may limit its applicability for aligning distantly related sequences, although the authors argue that it will generate a good initial approximation. The method was refined [282] to address the lack of sensitivity by replacing the Wilbur and Lipman algorithm used in the progressive pairwise aligning stage with a fast but optimal pairwise alignment method based on the method of Myers and Miller [203]. The score for aligning a sequence against a number of sequences is calculated by taking the average score between all the residues being compared thus ensuring that information from all sequences is included. This algorithm was recoded from FORTRAN to C in the ALIEN program [283] with an extension, allowing varying scoring matrices to be used.

An approach which does not use a pre-calculated ordering of the sequences is described by Berger and Munson [284]. The sequences are divided into two groups and these groups are pairwise aligned to each other. The alignment produced is used as a starting point for another round of this aligning and the process is repeated until no change is seen in the alignments. This has the advantage that the final alignment produced is not dependent on the order that the sequences are presented, and gaps can be removed from sequences thus reducing the possibility of an early misplaced gap being propagated through to the final alignment. The alignment algorithm is an extension to the Needleman and Wunsch algorithm which enables two groups of sequences to be treated as two sequences, however, it does not align regions that contain gaps, only scores the alignment for positions without any gaps in them and treats any number of contiguous gaps between aligned positions as a single gap. These extensions act to limit the

effectiveness of this algorithm to sequences that are closely related and of similar length.

Smith and Smith [285] use pairwise alignments between all the sequences to determine an approximate phylogenetic tree. The sequences are then aligned according to the order defined in the tree. At each stage a consensus sequence is defined and the progressive alignments are carried out on these consensus sequences. The final multiple alignment replaces the consensus residues with the actual residues in each sequence. The alignments use the Needleman and Wunsch algorithm.

A benefit of this pairwise extension approach, compared to the methods based on simultaneous aligning, is that the number of sequences that can be aligned is virtually unlimited. Also, they tend to be much faster. However, another consequence for most of these methods is that once a gap is inserted into an alignment, it is perpetuated through to the final multiple alignment, even if, when other sequences are added to the alignment, a better alignment would result on its removal. This is most likely to happen if the order of the pairwise sequence alignments is random or a sequence is aligned to a distant relative before being aligned to closely related sequences. Thus, the order in which the sequences are aligned can play a crucial role in determining the final alignment. Most of the methods therefore, first attempt to rank the sequences in order of similarity before the multiple alignment stage. This is commonly done by carrying out pairwise comparisons of the sequences against every other with the scores being used to rank the sequences. It is argued that this approach places reliance on the alignments generated between closely related sequences, and that it would be improper to remove a gap that was placed when aligned to closely related sequences, when the alignment to a distant relative would suggest doing so. An inherent drawback with these methods is the same as previously described for pairwise aligning methods: their simplified ties to the presently incomplete model of evolution. Any errors introduced by such methods may escalate as more sequences, which are less related, are added at each stage.

3.3 A strategy for multiple aligning

The above methods describe a variety of approaches, along with their various refinements, to the problem of multiple sequence aligning. In each of these approaches some limitations have been identified, however, the main drawback, applicable to all the approaches, is that changing the parameters used for aligning can change the alignment. For each method, changing the parameters can produce equally 'optimal' though radically different alignments. In an attempt to pre-define parameters that would produce the best alignment, Henneke *et al.* [64] investigated whether the parameters that produce the best pairwise alignments of the sequences produce the best multiple alignments. The results were negative. In common with pairwise aligning, no single set of parameters produce the 'best' alignment for all proteins, however, compared to pairwise methods, the number of parameters that can be altered with multiple alignment methods is usually increased. Even with these limitations, such methods can work well for aligning members of a family of proteins with high identity [267, 281, 282, 284]. The problems occur when identity is low, when it is often found that no automatic methods gives a satisfactory alignment [249].

In consideration of the above, a general strategy towards searching for multiple aligning can be formulated. For closely related sequences, an automatic method can be used, followed by visual inspection. Different parameters can be used, as well as using more than one method, to ensure that no algorithmical quirk is inadvertently biasing the alignment. For more distantly related sequences, more than one method should be used with varying scoring schemes, followed by close visual inspection. Various editors, designed to handle alignments of sequence data, have been developed which can aid in the process of manual aligning [178, 286-290].

3.4 Results and discussion using methods and strategy detailed above

The strategy outlined above for multiple aligning varies depending on how related the sequences are, however, the relationship between sequences is determined after aligning. To resolve this for the shikimate enzymes, an initial estimate is made. Due to the disparity in lengths between the sequences of some of the shikimate enzymes (e.g., the thirty residue difference between e1-P07639 and e1-P08566), it was decided to treat them as distantly related sequences for the purposes of aligning. If, after aligning, the sequences are found to be closely related, the extra computational resources that have been used in calculating alignments using more than one method are not wasted as the extra alignments can contribute to the final visual examination stage. The methods available for aligning on the local computer are PILEUP [178] and ALIEN [283]. In line with the recommendation to use many alignment methods, the multiple alignment method of Vingron and Argos [262] was also obtained but difficulty was encountered in getting this to compile and run correctly on the local computer.

The sequences of the pentafunctional arom complexes from *Aspergillus nidulans* and *Saccharomyces cerevisiae* are split into their individual enzymes and these are aligned against the other sequences for each of the five shikimate enzymes, as listed in table 2.2. The pentafunctional complexes are split so that the length of the individual sequences are increased, by an average of twenty residues, over those listed in table 2.2. This is to ensure that the alignments are obtained over the entire length of the sequences and to check that the limits given in the documentation for the complexes is correct.

Following the recommendations above, the shikimate enzymes are aligned using both the ALIEN and PILEUP programs, run with a variety of scoring schemes. When aligning proteins, ALIEN has seven possible parameters that can be altered: four for the pairwise aligning stage; two for the multiple aligning stage; and one which is set depending on whether the sequences are dissimilar or not in length. Two pairwise aligning parameters are altered. These penalise gaps (Wilbur Gap), and increase the area

of comparison in the pairwise matrix (Diagonal). The two pairwise aligning parameters that are not altered have default values which produce rigorous alignments. The two multiple aligning parameters are the penalties for creating and extending gaps (Fixed Gap and Floating Gap respectively). For the shikimate enzymes, although the sequence lengths vary, it was felt that they do not qualify as being dissimilar, so the setting is left as recommended for sequences of similar length. PILEUP has two parameters that affect the alignments produced. These are the penalties for creating and extending gaps (GapWeight and GapLengthWeight respectively).

ALIEN is run with nine different sets of parameters and PILEUP with eleven sets of parameters. The values chosen for the parameters, listed in table 3.2, are picked to give a spread of values, ranging from making gap insertions easy to hard. For the ALIEN alignments, six different scoring matrices are also used. Four matrices are based on Dayhoff's PAM matrix and two structure/genetic matrices: (i) PAM, log odds form of the mutation data matrix for 250 PAM [216]; (ii) DEF, a log odds form of the mutation data matrix for 250 PAM, modified so there are no negative scores and "alters some scores to arguably more reasonable levels" (quote taken from the user documentation supplied with ALIEN); (iii) DAYHOFF, a log odds form of the mutation data matrix for 250 PAM as rescaled and used by Gribskov and Burgess [291]; (iv) MD, a log odds form of the mutation data matrix for 250 PAM, modified so Y:R=0, W:F=2, W:Y=2 and Y:Y=12; (v) BACON, a structure/genetics matrix [208]; and (vi) SG, a structure/genetics matrix [209]. The matrices are listed in appendix C. A total of sixty-five alignments are generated for each of the five shikimate enzyme families. The results obtained from the alignments are summarised in table 3.3. For the alignment produced for each shikimate enzyme family by each scoring set (matrix and gap penalty set), results are given for: the total number of gaps in the alignment; the total number of insertions, regardless of length; the number of identical residues; the percentage sequence identity (this uses the definition of percentage identity as given by Feng *et al.* [209] - the number of identical residues divided by the length of shortest sequence); and the length in residues of the multiple alignment.

Name of gap penalty	Gap penalty set 1	2	3	4	5	6	7	8	9	10	11
ALIEN											
Wilbur Gap	3	3	3	3	5	8	8	8	2	-	-
Diagonal	10	15	10	20	10	20	20	20	8	-	-
Fixed Gap	10	10	20	15	10	10	15	20	10	-	-
Floating Gap	10	10	20	15	10	10	15	20	10	-	-
PILEUP											
GapWeight	3.0	3.0	3.0	3.0	2.0	2.0	2.0	2.0	4.0	4.0	4.0
GapLengthWeight	0.1	0.5	1.0	1.5	0.1	0.5	1.0	1.5	0.1	0.5	1.0

Table 3.2

Scoring parameters used to generate the different alignments. (**ALIEN**) The names of the gap penalties used by ALIEN and the values used in each of nine different sets of penalties. ALIEN is run using each of these sets with six similarity matrices, producing fifty four differing alignments. (**PILEUP**) The names of the gap penalties used by PILEUP and the values used in the eleven sets of gap penalties. Pileup is run using the default matrix, producing eleven alignments. The default value for each parameter for each method is shown in gap penalty set one.

Table 3.3

Results of sixty five alignments of shikimate enzymes e1 to e5. Alignments are produced using the ALIEN and PILEUP algorithms. Five quantitative measures, **a** to **e**, of the alignments generated are listed. The gap sets are numbered as the gap penalty sets given in table 3.2. The matrices used for ALIEN alignments are as described in the text and shown in appendix C. The alignments produced by PILEUP use the rescaled 250 PAMs matrix, labelled *DAYHOFF* above. For the alignment produced by each scoring set (matrix and gap penalty set), results are given for: (**a**) the total number of gaps in the alignment; (**b**) the total number of insertions, regardless of length; (**c**) the number of identical residues; (**d**) the percentage sequence identity; and (**e**) the length in residues of the multiple alignment. The highest values for results **a** to **d** are shaded. The sequences used for each enzyme are as shown in table 2.2. The lengths for the enzymes are given in table 3.2, except for the *Aspergillus nidulans* and *Saccharomyces cerevisiae* sequences where the lengths for e1 to e5 are; 400, 261, 291, 210, 489 , 410, 256, 289, 220 and 480, respectively.

Shikimate Enzyme c1 - results from ALIEN algorithm

Matrix	Gap set	1	2	3	4	5	6	7	8	9
PAM	a	82	82	76	76	82	82	76	76	82
	b	21	21	14	15	21	21	15	14	21
	c	115	115	109	109	115	115	109	109	115
	d	31.8	31.8	30.1	30.1	31.8	31.8	30.1	30.1	31.8
	e	418	418	416	416	418	418	416	416	418
DEF	a	76	76	76	76	76	76	76	76	76
	b	16	16	11	12	16	16	12	11	16
	c	112	112	104	104	112	112	104	104	112
	d	30.9	30.9	28.7	28.7	30.9	30.9	28.7	28.7	30.9
	e	416	416	416	416	416	416	416	416	416
BACON	a	76	76	76	76	76	76	76	76	76
	b	15	15	13	14	15	15	14	13	15
	c	111	111	110	110	111	111	110	110	111
	d	30.7	30.7	30.4	30.4	30.7	30.7	30.4	30.4	30.7
	e	416	416	416	416	416	416	416	416	416
DAYHOFF	a	91	91	76	76	91	91	76	76	91
	b	35	35	21	23	35	35	23	21	35
	c	116	116	115	114	116	116	114	115	116
	d	32.0	32.0	31.8	31.5	32.0	32.0	31.5	31.8	32.0
	e	421	421	416	416	421	421	416	416	421
SG	a	76	76	73	76	76	76	76	73	76
	b	14	14	9	11	14	14	11	9	14
	c	109	109	90	104	109	109	104	90	109
	d	30.1	30.1	24.9	28.7	30.1	30.1	28.7	24.9	30.1
	e	416	416	415	416	416	416	416	415	416
MD	a	76	76	76	76	76	76	76	76	76
	b	18	18	14	15	18	18	15	14	18
	c	113	113	109	109	113	113	109	109	113
	d	31.2	31.2	30.1	30.1	31.2	31.2	30.1	30.1	31.2
	e	416	416	416	416	416	416	416	416	416

Shikimate Enzyme e1 - results from PILEUP algorithm

Gap Set	1	2	3	4	5	6	7	8	9	10	11
a	109	76	79	79	85	82	79	79	88	76	79
b	18	14	14	14	24	16	15	15	14	14	14
c	106	103	98	98	110	96	98	98	102	103	98
d	29.3	28.5	27.1	27.1	30.4	26.5	27.1	27.1	28	28.5	27.1
e	427	416	417	417	419	418	417	417	420	416	417

Shikimate Enzyme e2 - results from ALIEN algorithm

Matrix	Gap Set	1	2	3	4	5	6	7	8	9
PAM	a	67	67	47	51	67	67	51	47	67
	b	27	27	15	17	27	27	17	15	27
	c	37	37	28	30	37	37	30	28	37
	d	14.7	14.7	11.1	11.9	14.7	14.7	11.9	11.1	14.7
	e	272	272	267	268	272	272	268	267	272
DEF	a	51	51	27	39	51	51	39	27	51
	b	16	16	7	11	16	16	11	7	16
	c	30	30	10	20	30	30	20	10	30
	d	11.9	11.9	4.0	7.9	11.9	11.9	7.9	4.0	11.9
	e	268	268	262	265	268	268	265	262	268
BACON	a	51	51	27	39	51	51	39	27	51
	b	17	17	12	15	17	17	15	12	17
	c	30	30	10	20	30	30	20	10	30
	d	11.9	11.9	4.0	7.9	11.9	11.9	7.9	4.0	11.9
	e	268	268	262	265	268	268	265	262	268
DAYHOFF	a	107	107	63	67	107	107	67	63	107
	b	57	57	30	34	57	57	34	30	57
	c	41	41	38	40	41	41	40	38	41
	d	16.3	16.3	15.1	15.9	16.3	16.3	15.9	15.1	16.3
	e	282	282	271	272	282	282	272	271	282
SG	a	39	39	27	27	39	39	27	27	39
	b	16	16	7	7	16	16	7	7	16
	c	20	20	10	10	20	20	10	10	20
	d	7.9	7.9	4.0	4.0	7.9	7.9	4.0	4.0	7.9
	e	265	265	262	262	265	265	262	262	265
MD	a	63	63	43	51	63	63	51	43	63
	b	27	27	12	16	27	27	16	12	27
	c	36	36	25	30	36	36	30	25	36
	d	14.3	14.3	9.9	11.9	14.3	14.3	11.9	9.9	14.3
	e	271	271	266	268	271	271	268	266	271

Shikimate Enzyme e2 - results from PILEUP algorithm

Gap Set	1	2	3	4	5	6	7	8	9	10	11
a	40	40	36	36	40	40	40	36	40	36	36
b	27	27	21	22	29	29	29	22	26	22	22
c	37	37	34	34	37	37	37	34	37	34	34
d	14.7	14.7	13.5	13.5	14.7	14.7	14.7	13.5	14.7	13.5	13.5
e	272	272	270	270	272	272	272	270	272	270	270

Shikimate Enzyme e3 - results from ALIEN algorithm

Matrix	Gap Set	1	2	3	4	5	6	7	8	9
PAM	a	87	87	66	66	87	87	66	66	87
	b	24	24	14	16	24	24	16	14	24
	c	46	46	43	43	46	46	43	43	46
	d	16.9	16.9	15.8	15.8	16.9	16.9	15.8	15.8	16.9
	e	313	313	306	306	313	313	306	306	313
DEF	a	66	66	21	45	66	66	45	21	66
	b	15	15	6	11	15	15	11	6	15
	c	43	43	13	37	43	43	37	13	43
	d	15.8	15.8	3.7	13.6	15.8	15.8	13.6	3.7	15.8
	e	306	306	291	299	306	306	299	291	306
BACON	a	57	57	24	45	57	57	45	24	57
	b	16	16	8	11	16	16	11	8	16
	c	38	38	13	37	38	38	37	13	38
	d	14.0	14.0	4.8	13.6	14.0	14.0	13.6	4.8	14.0
	e	303	303	292	299	303	303	299	292	303
DAYHOFF	a	96	96	57	78	96	96	78	57	96
	b	42	42	21	29	42	42	29	21	42
	c	46	46	40	44	46	46	44	40	46
	d	16.9	16.9	14.7	16.2	16.9	16.9	16.2	14.7	16.9
	e	316	316	303	310	316	316	310	303	316
SG	a	45	45	21	21	45	45	21	21	45
	b	13	13	4	5	13	13	5	4	13
	c	38	38	10	12	38	38	12	10	38
	d	14.0	14.0	3.7	4.4	14.0	14.0	4.4	3.7	14.0
	e	299	299	291	291	299	299	291	291	299
MD	a	75	75	45	66	75	75	66	45	75
	b	21	21	12	16	21	21	16	12	21
	c	45	45	37	43	45	45	43	37	45
	d	16.5	16.5	13.6	15.8	16.5	16.5	15.8	13.6	16.5
	e	309	309	299	306	309	309	306	299	309

Shikimate Enzyme e3 - results from PILEUP algorithm

Gap Set	1	2	3	4	5	6	7	8	9	10	11
a	37	37	36	30	37	37	37	31	37	37	30
b	17	17	16	10	21	21	25	16	17	17	10
c	41	43	40	34	45	45	44	42	40	43	34
d	23.7	24.9	23.1	19.7	26.0	26.0	25.4	24.3	23.1	24.9	19.7
e	309	309	308	302	309	309	309	303	309	309	302

Shikimate Enzyme e4 - results from ALIEN algorithm

Matrix	Gap Set	1	2	3	4	5	6	7	8	9
PAM	a	448	448	508	448	508	448	448	508	508
	b	39	39	22	25	39	39	25	22	3915
	c	15	15	12	12	15	15	12	12	15
	d	6.9	6.9	8.7	6.9	8.7	6.9	6.9	8.7	8.7
	e	317	317	305	305	317	317	305	305	317
DEF	a	448	448	423	423	448	448	423	423	448
	b	25	25	15	21	25	25	21	15	25
	c	12	12	4	4	12	12	4	4	12
	d	6.9	6.9	2.3	2.3	6.9	6.9	2.3	2.3	6.9
	e	305	305	300	300	305	305	300	300	305
BACON	a	423	423	423	423	423	423	423	423	423
	b	29	29	15	25	29	29	25	15	29
	c	3	3	2	3	3	3	3	2	3
	d	1.7	1.7	1.2	1.7	1.7	1.7	1.7	1.2	1.7
	e	300	300	300	300	300	300	300	300	300
DAYHOFF	a	508	508	453	468	508	508	468	453	508
	b	77	77	35	50	77	77	50	35	77
	c	18	18	15	15	18	18	15	15	18
	d	10.4	10.4	8.7	8.7	10.4	10.4	8.7	8.7	10.4
	e	317	317	306	309	317	317	309	306	317
SG	a	423	423	423	423	423	423	423	423	423
	b	22	22	17	18	22	22	18	17	22
	c	1	1	1	1	1	1	1	1	1
	d	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
	e	300	300	300	300	300	300	300	300	300
MD	a	468	468	423	448	468	468	448	423	468
	b	36	36	16	24	36	36	24	16	36
	c	14	14	1	12	14	14	12	1	14
	d	8.1	8.1	0.6	6.9	8.1	8.1	6.9	0.6	8.1
	e	309	309	300	305	309	309	305	300	309

Shikimate Enzyme e4 - results from PILEUP algorithm

Gap Set	1	2	3	4	5	6	7	8	9	10	11
a	488	483	483	483	508	488	488	488	483	483	483
b	28	26	26	26	37	31	30	30	28	26	26
c	16	14	14	14	17	16	16	14	15	14	14
d	9.2	8.1	8.1	8.1	9.8	9.2	9.2	8.1	8.7	8.1	8.1
e	313	312	312	312	336	313	313	313	313	312	312

Shikimate Enzyme e5 - results from ALIEN algorithm

Matrix	Gap Set	1	2	3	4	5	6	7	8	9
PAM	a	990	990	610	910	990	990	910	610	990
	b	151	151	61	95	151	151	95	61	151
	c	54	54	2	45	54	54	45	2	54
	d	12.6	12.6	0.5	10.5	12.6	12.6	10.5	0.5	12.6
	e	566	566	528	558	566	566	558	528	566
DEF	a	910	910	610	610	910	910	610	610	910
	b	106	106	58	64	106	106	64	58	106
	c	47	47	4	4	47	47	4	4	47
	d	11.0	11.0	0.9	0.9	11.0	11.0	0.9	0.9	11.0
	e	558	558	528	528	558	558	528	528	558
BACON	a	910	910	610	610	910	910	610	610	910
	b	121	121	66	77	121	121	77	66	121
	c	46	46	1	2	46	46	2	1	46
	d	10.8	10.8	0.2	0.5	10.8	10.8	0.5	0.2	10.8
	e	558	558	528	528	558	558	528	528	558
DAYHOFF	a	1070	1070	1000	960	1070	1070	960	1000	1070
	b	252	252	121	177	252	252	177	121	252
	c	54	54	50	52	54	54	52	50	54
	d	12.6	12.6	11.7	12.2	12.6	12.6	12.2	11.7	12.6
	e	574	574	563	567	574	574	567	563	574
SG	a	610	610	570	610	610	610	610	570	610
	b	74	74	50	66	74	74	66	50	74
	c	3	3	1	1	3	3	1	1	3
	d	0.7	0.7	0.2	0.2	0.7	0.7	0.2	0.2	0.7
	e	528	528	524	528	528	528	528	524	528
MD	a	910	910	610	610	910	910	610	610	910
	b	136	136	62	95	136	136	95	62	136
	c	52	52	2	45	52	52	45	2	52
	d	12.2	12.2	0.5	10.5	12.2	12.2	10.5	0.5	12.2
	e	563	563	528	558	563	563	558	528	563

Shikimate Enzyme e5 - results from PILEUP algorithm

Gap Set	1	2	3	4	5	6	7	8	9	10	11
a	907	853	835	799	979	871	853	835	853	835	799
b	139	122	106	78	151	144	133	119	124	104	87
c	51	51	47	41	50	51	52	48	50	52	43
d	11.9	11.9	11.0	9.6	11.7	11.9	12.2	11.2	11.7	12.2	10.1
e	570	564	562	558	578	566	564	562	564	562	558

Examining the results shows that for each of the enzyme families, changing the gap sets used in aligning can change the alignments produced. For example, when aligning enzyme e3 using PILEUP, gap set one produces an alignment with forty one identical residues and thirty seven gaps inserted, while gap set four produces an alignment with thirty four identical residues and thirty gaps inserted. This effect is seen in the alignments produced by both alignment methods. Changes are also observed in the ALIEN alignments produced when the gap sets are kept the same and the scoring matrix is altered. For example, when aligning enzyme e4 using gap set one, the DAYHOFF matrix produces eighteen identical residues whereas the SG matrix only finds one.

When looking at the effects of the scoring matrices used, it is seen that the DAYHOFF matrix consistently produces alignments with the highest values for the statistics shown (e.g., number of identical residues) and the SG matrix produces the lowest values. Higher values than seen in the DAYHOFF alignments for the total number of gaps inserted, are produced using the PILEUP method for enzymes e1 and e4 and equally high scores for some of the statistics are reached for enzymes e3 and e5 using the PAM matrix with gap sets one, two, five, six, and nine. Scores equally as low as produced by the SG matrix are also found for enzyme e2 using the DEF matrix and gap sets three and seven.

Examination of the DAYHOFF matrix shows that identical residues all receive a relatively high score of fifteen (e.g., compared with a range of two to twelve for MD and DEF) while mismatches receive high negative scores (up to minus ten). This may strongly bias the alignments towards aligning identical residues at the expense of inserting a large number of gaps. This is supported by the DAYHOFF matrix having the highest number of gaps inserted and the smallest average gap length, almost twice as short as next longest (see table 3.4), suggesting that a large number of small gaps are being included to force the alignment of identical residues. This effect of introducing large numbers of gaps to increase the number of identical residues is not seen in the PILEUP alignments where the DAYHOFF matrix is used, although for two of the

	PAM	DEF	BACON	DAYHOFF	SG	MD	PILEUP
Average Gap Length	11.5	17.9	14.6	6.6	19.2	13.0	17.4

Table 3.4

The average gap length in seven alignments of shikimate enzyme e4. The sequences used are as listed in table 2.2. The first six alignments are produced using the ALIEN program with default gap parameters (gap set one from table 3.2) while varying the scoring matrix used. The matrices are as described in the text and shown in appendix C. The results for the seventh alignment are shown in the column marked PILEUP. This shows the results for the PILEUP program with the default gap penalty (gap set one from table 3.2) and matrix.

enzymes, when the penalty for extending gaps is set to 0.1, PILEUP has the highest number of gaps inserted. This may be due to the gap penalties being better tuned to the matrix's scores - ALIEN's default matrix has a lower range of scores - or other implicit differences in the algorithms.

Examination of the SG matrix reveals that it has a small range of scores from zero to six, with an average score of 2.9. A consequence of this scoring of most residues as 'similar' with almost as high a score as identity, is that the alignment algorithms may score alignments with many similarities but few identities higher than alignments with more identities but fewer similarities. Another effect will be that to align 'similar' residues, fewer gaps need be inserted. These are borne out by the low results for the number of identical residues and gaps inserted, shown in table 3.3 and the highest average gap length of 19.2.

Looking at the effect of changing the gap penalties shows that different gap penalties can produce the same alignment. When using the ALIEN program, with all five enzymes, identical alignments are produced by gap set one, two, five, six, and nine, gap sets three and eight and gap sets four and seven. This may be due to the gap sets not having a large enough variation between the parameters in the sets that produce identical alignments. An alternative explanation is that this result is an artefact of the shikimate enzymes and the gap sets would produce differing alignments if used to align a different protein family. Similarly with PILEUP, for enzymes e1 to e4, up to nine unique alignments are produced by the eleven parameters with gap sets two and ten and gap sets four and eleven producing identical alignments. For enzyme e5, the eleven sets of gap parameters produce eleven different alignments. This illustrates that the gap sets are different enough to produce differing alignments although not for every protein family.

The range of values produced when changing the gap sets can be large or small and depends on both the protein family being aligned and the scoring matrix used. Similarly, the effect of changing the scoring matrix produces a range of results

dependent on the protein family and gap sets used. For example, for the enzyme e1, ALIEN alignments using the BACON matrix have a range in the number of identical residues of one, whereas for the SG matrix, a range of nineteen is seen. Keeping the same scoring matrix and comparing results from ALIEN alignments while changing the gap sets between the enzyme families shows that for the PAM matrix, enzymes e1 to e4 have a small range in the value of number of identical residues (between three and nine) while enzyme e5 shows a large range of fifty two in value and for the BACON matrix, enzyme e1 and e4 show a range of one in identical residue values while the three other enzymes have a large range (between twenty and forty five).

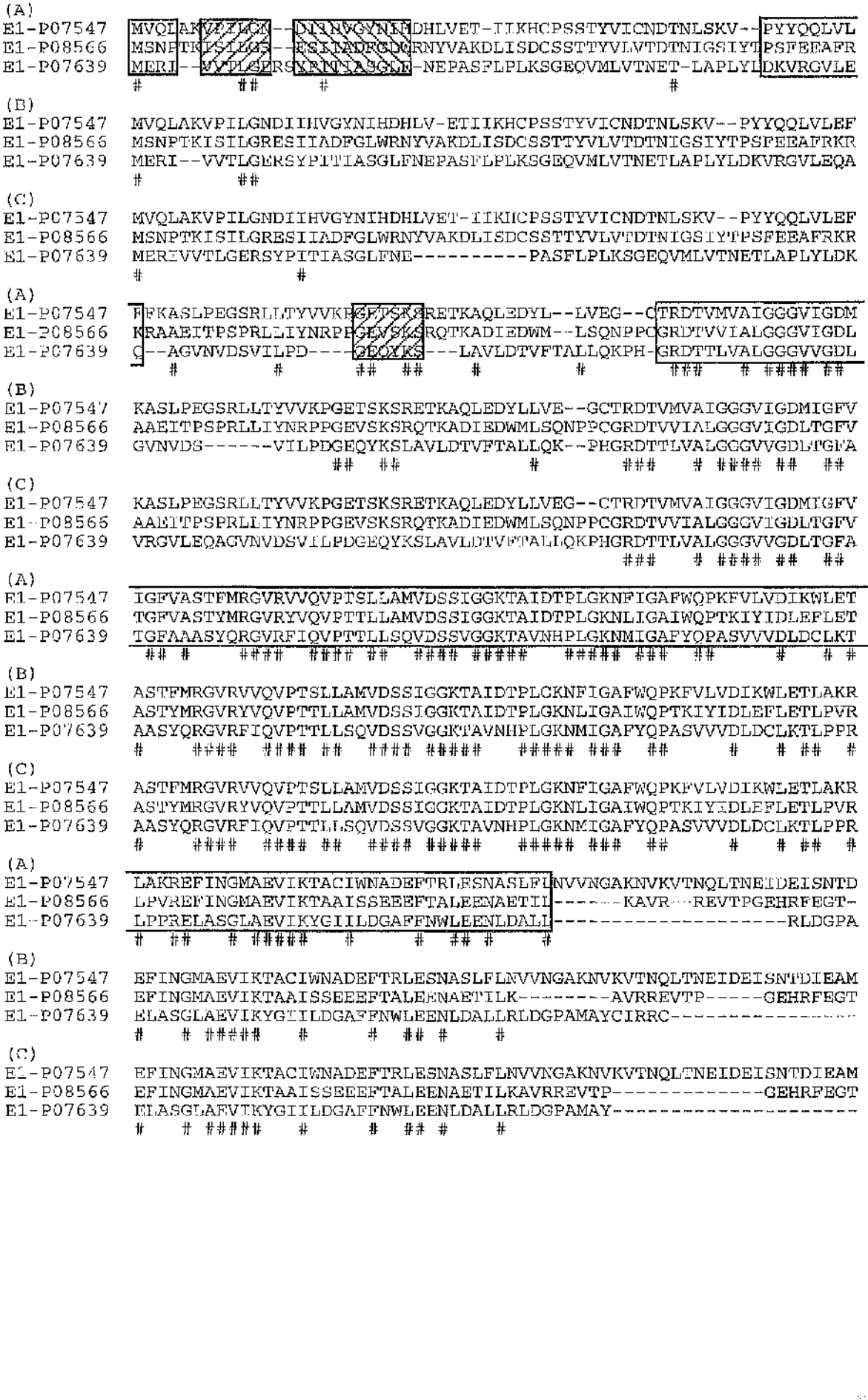
The above results demonstrate that producing alignments is dependent on changes to any of the inputs: algorithm, scoring matrix, gap penalties and protein sequences.

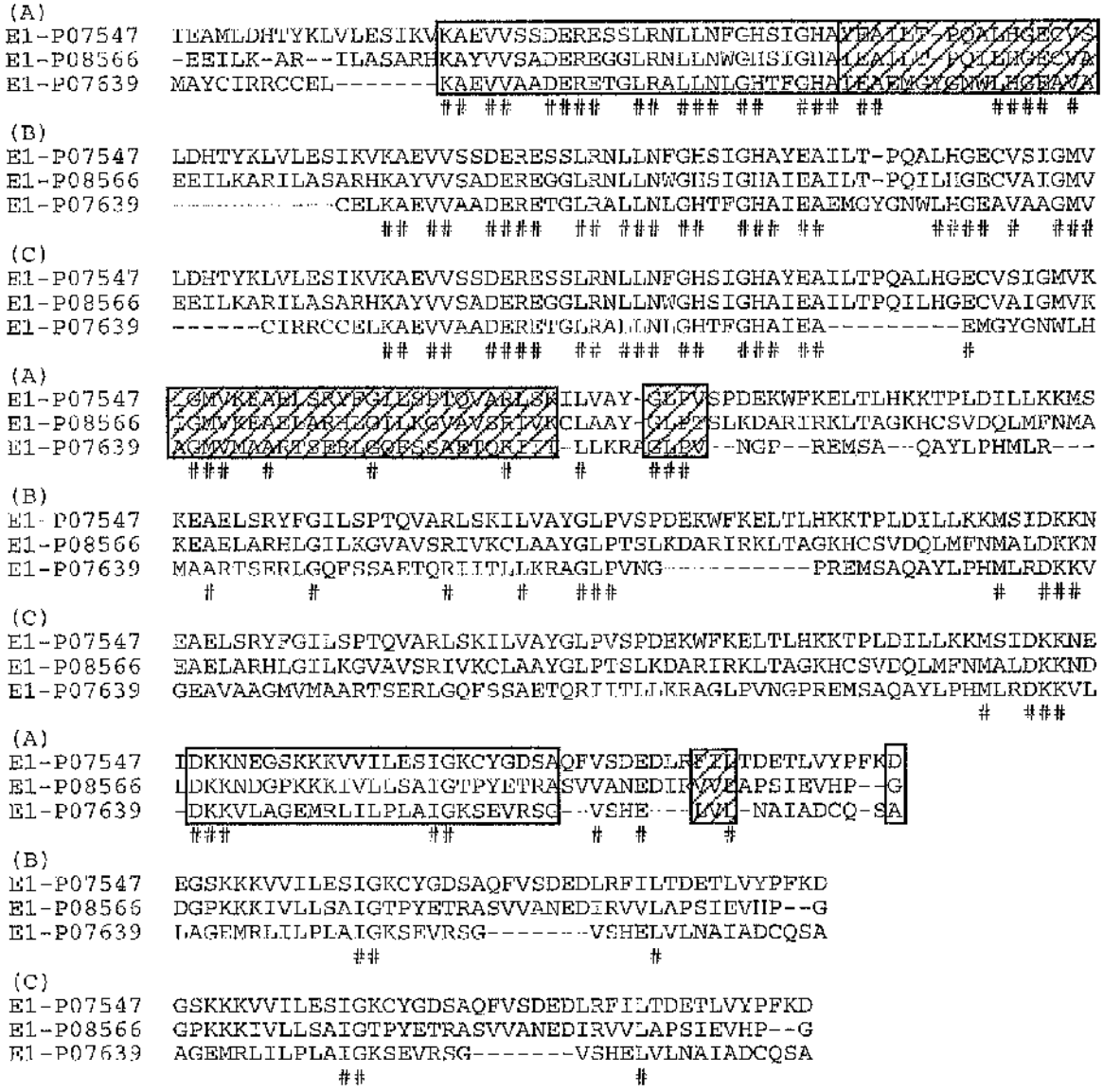
Attempts have previously been made to empirically define the best set of gap penalties to use for a scoring matrix when aligning [64, 209]. Such approaches would circumvent the variability problems. However, without *a priori* knowledge of the sequences being aligned - in Henneke's case, the three-dimensional structure for one of the sequences - the choice of which alignment value to prioritise (five are shown in table 3.3 from the many available) is unknown and such attempts fail. It may be tempting to pick one value (e.g., number of identical residues) and optimise for that, however, it has been shown that the quirks of a matrix may have a bias towards a value which invalidates the results and it is also likely that no single value will be sufficient to discriminate for the biological alignment or be applicable to all proteins.

The percentage identities of enzymes e2 to e5 are below the 'twilight zone' described by Doolittle [181] with enzyme e1 being below or just above it. This validates the previous assumption that the shikimate enzymes are distantly related. The next stage of aligning is therefore to critically inspect the alignments produced and try and derive a final consensus alignment. The large range in alignments produced for each enzyme (see tables 3.3 and 3.5) makes this a difficult task. To illustrate this, three alignments for enzyme e1 are shown in figure 3.1. Enzyme e1 should be the easiest, due to it's high

Figure 3.1

Three multiple alignments of shikimate enzyme e1. Identical residues are marked with a 'hash' below them. The alignments are carried out using the ALIEN algorithm with the following parameters. (**A**) Gap set six and matrix DAYHOFF are used, (**B**) gap set seven and matrix BACON are used and (**C**) gap set three and matrix SG are used. The gap sets are as shown in table 3.2 and the matrices are as described in the text. Regions conserved between the alignments are enclosed in a rectangle in alignment A. Regions of the alignments conserved in all three alignments are marked in a clear rectangle, areas conserved between alignment A and B are marked in a rectangle filled with '/' and areas conserved between alignment A and C are marked in a rectangle filled with '\'.


(A)
E1-P07547 MVQLAKVVFLCA--DINNVGYNHDHLVET-IIKHCPSSTYVICNDTNLSKV--PYYQQLVL
E1-P08566 MSNPTKISILGR--ESIIADFGLWRNYVAKDLISDCSSTTYVLVTDTNIGSIYTPSFEEAFR
E1-P07639 MERI--VVTLGERSYPITIASGLFNEPASFLPLKSGEQVMLVTNET-LAPLYLDKVRGVLE
(B)
E1-P07547 MVQLAKVPILGNDIIHVGYNIHDHLV-ETIIKHCPSSTYVICNDTNLSKV--PYYQQLVLEF
E1-P08566 MSNPTKISILGRESIIADFGLWRNYVAKDLISDCSSTTYVLVTDTNIGSIYTPSFEEAFRKR
E1-P07639 MERI--VVTLGERSYPITIASGLFNEPASFLPLKSGEQVMLVTNETLAPLYLDKVRGVLEQA
(C)
E1-P07547 MVQLAKVPILGNDIIHVGYNIHDHLVET-IIKHCPSSTYVICNDTNLSKV--PYYQQLVLEF
E1-P08566 MSNPTKISILGRESIIADFGLWRNYVAKDLISDCSSTTYVLVTDTNIGSIYTPSFEEAFRKR
E1-P07639 MERVVTLGERSYPITIASGLFNE----------PASFLPLKSGEQVMLVTNETLAPLYLDK
(A)
E1-P07547 FFKASLPEGSRLLTYVVKPGETSKSRETKAQLEDYL--LVEG--CTRDTVMVAIGGGVIGDM
E1-P08566 KRAAEITPSPRLLIYNRPPGEVSKSRQTKADIEDWM--LSQNPPCGRDTVVIALGGGVIGDL
E1-P07639 G--AGVNVDSVILPD----GEQYKS---LAVLDTVFTALLQKPH-GRDTTLVALGGGVVGDL
(B)
E1-P07547 KASLPEGSRLLTYVVKPGETSKSRETKAQLEDYLLVE--GCTRDTVMVAIGGGVIGDMIGFV
E1-P08566 AAEITPSPRLLIYNRPPGEVSKSRQTKADIEDWMLSQNPPCGRDTVVIALGGGVIGDLTGFV
E1-P07639 GVNVDS------VILPDGEQYKSLAVLDTVFTALLQK--PHGRDTTLVALGGGVVGDLTGFA
(C)
E1-P07547 KASLPEGSRLLTYVVKPGETSKSRETKAQLEDYLLVEG--CTRDTVMVAIGGGVIGDMIGFV
E1-P08566 AAEITPSPRLLIYNRPPGEVSKSRQTKADIEDWMLSQNPPCGRDTVVIALGGGVIGDLTGFV
E1-P07639 VRGVLEQAGVNVDSVILPDGEQYKSLAVLDTVFTALLQKPHGRDTTLVALGGGVVGDLTGFA
(A)
E1-P07547 IGFVASTFMRGVRVVQVPTSLLAMVDSSIGGKTAIDTPLGKNFIGAFWQPKFVLVDIKWLET
E1-P08566 TGFVASTYMRGVRYVQVPTTLLAMVDSSIGGKTAIDTPLGKNLIGAIWQPTKIYIDLEFLET
E1-P07639 TGFAAASYQRGVRFIQVPTTLLSQVDSSVGGKTAVNHPLGKNMIGAFYQPASVVVDLDCLKT
(B)
E1-P07547 ASTFMRGVRVVQVPTSLLAMVDSSIGGKTAIDTPLGKNFIGAFWQPKFVLVDIKWLETLAKR
E1-P08566 ASTYMRGVRYVQVPTTLLAMVDSSIGGKTAIDTPLGKNLIGAIWQPTKIYIDLEFLETLPVR
E1-P07639 AASYQRGVRFIQVPTTLLSQVDSSVGGKTAVNHPLGKNMIGAFYQPASVVVDLDCLKTLPPR
(C)
E1-P07547 ASTFMRGVRVVQVPTSLLAMVDSSIGGKTAIDTPLGKNFIGAFWQPKFVLVDIKWLETLAKR
E1-P08566 ASTYMRGVRYVQVPTTLLAMVDSSIGGKTAIDTPLGKNLIGAIWQPTKIYIDLEFLETLPVR
E1-P07639 AASYQRGVRFIQVPTTLLSQVDSSVGGKTAVNHPLGKNMIGAFYQPASVVVDLDCLKTLPPR
(A)
E1-P07547 LAKREFINGMAEVIKTACIWNADEFTRLESNASLFLNVVNGAKNVKVTNQLTNEIDEISNTD
E1-P08566 LPVREFINGMAEVIKTAAISSEEEFTALEENAETIL-------KAVR---REVTPGEHRFEGT-
E1-P07639 LPPRELASGLAEVIKYGIILDGAFFNWLEENLDALL----------RLDGPA
(B)
E1-P07547 EFINGMAEVIKTACIWNADEFTRLESNASLFLNVVNGAKNVKVTNQLTNEIDEISNTDIEAM
E1-P08566 EFINGMAEVIKTAAISSEEEFTALEENAETILK--------AVRREVTP-----GEHRFEGT
E1-P07639 ELASGLAEVIKYGIILDGAFFNWLEENLDALLRLDGPAMAYCIRRC----------
(C)
E1-P07547 EFINGMAEVIKTACIWNADEFTRLESNASLFLNVVNGAKNVKVTNQLTNEIDEISNTDIEAM
E1-P08566 EFINGMAEVIKTAAISSEEEFTALEENAETILKAVRREVTP----------GEHRFEGT
E1-P07639 ELASGLAEVIKYGIILDGAFFNWLEENLDALLRLDGPAMAY----------


(A)
E1-P07547 TEAMLDHTYKLVLESIKVKAEVVSSDERESSLRNLLNFGHSIGHAYEAILT-PQALHGECVS
E1-P08566 -EEILK-AR--ILASARHKAYVVSADEREGGLRNLLNWGHSIGHAIEAILT-PQILHGECVA
E1-P07639 MAYCIRRCCEL-------KAEVVAADERETGLRALLNLGHTFGHAIEAEMGYGNWLHGEAVA
(B)
E1-P07547 LDHTYKLVLESIKVKAEVVSSDERESSLRNLLNFGHSIGHAYEAILT-PQALHGECVSIGMV
E1-P08566 EEILKARILASARHKAYVVSADEREGGLRNLLNWGHSIGHAIEAILT-PQILHGECVAIGMV
E1-P07639 ----------CELKAEVVAADERETGLRALLNLGHTFGHAIEAEMGYGNWLHGEAVAAGMV
(C)
E1-P07547 LDHTYKLVLESIKVKAEVVSSDERESSLRNLLNFGHSIGHAYEAILTPQALHGECVSIGMVK
E1-P08566 EEILKARILASARHKAYVVSADEREGGLRNLLNWGHSIGHAIEAILTPQILHGECVAIGMVK
E1-P07639 ------CIRRCCELKAEVVAADERETGLRALLNLGHTFGHAIEA---------EMGYGNWLH
(A)
E1-P07547 AGMVKEAELSRYFGILSPTQVARLSKILVAY-GLPVSPDEKWFKELTLHKKTPLDILLKKMS
E1-P08566 AGMVKEAELARELGILKGVAVSRIVKCLAAY-GLPTSLKDARIRKLTAGKHCSVDQLMFNMA
E1-P07639 AGMVMAARTSERLGQFSSAETQRIITLLKRAGLPVNG--NGP--REMSA---QAYLPHMLR---
(B)
E1-P07547 KEAELSRYFGILSPTQVARLSKILVAYGLPVSPDEKWFKELTLHKKTPLDILLKKMSIDKKN
E1-P08566 KEAELARELGILKGVAVSRIVKCLAAYGLPTSLKDARIRKLTAGKHCSVDQLMFNMALDKKN
E1-P07639 MAARTSERLGQFSSAETQRIITLLKRAGLPVNG----------PREMSAQAYLPHMLRDKKV
(C)
E1-P07547 EAELSRYFGILSPTQVARLSKILVAYGLPVSPDEKWFKELTLHKKTPLDILLKKMSIDKKNE
E1-P08566 EAELARHLGILKGVAVSRIVKCLAAYGLPTSLKDARIRKLTAGKHCSVDQLMFNMALDKKND
E1-P07639 GEAVAAGMVMAARTSERLGQFSSAETQRIITLLKRAGLPVNGPREMSAQAYLPHMLRDKKVL
(A)
E1-P07547 IDKKNEGSKKKVVILESIGKCYGDSAQFVSDEDLRFILTDETLVYPFKD
E1-P08566 LDKKNDGPKKKIVLLSAIGTPYETRASVVANEDIRVVLAPSIEVHP--G
E1-P07639 -DKKVLAGEMRLILPLAIGKSEVRSG--VSHE---LVL-NAIADCQ-SA
(B)
E1-P07547 EGSKKKVVILESIGKCYGDSAQFVSDEDLRFILTDETLVYPFKD
E1-P08566 DGPKKKIVLLSAIGTPYETRASVVANEDIRVVLAPSIEVHP--G
E1-P07639 AGEMRLILPLAIGKSEVRSG-------VSHELVLNAIADCQSA
(C)
E1-P07547 GSKKKVVILESIGKCYGDSAQFVSDEDLRFILTDETLVYPFKD
E1-P08566 GPKKKIVLLSAIGTPYETRASVVANEDIRVVLAPSIEVHP--G
E1-P07639 AGEMRLILPLAIGKSEVRSG-------VSHELVLNAIADCQSA

	Shikimate enzymes				
Value	e1	e2	e3	e4	e5
a	73-109	27-107	21-96	423-603	570-1070
b	9-35	7-57	4-42	15-77	50-252
c	90-116	10-41	10-46	1-18	1-54
d	24.9-32.0	4.0-16.3	3.7-16.9	0.6-10.4	0.2-12.6
e	415-427	262-282	291-316	300-317	524-574

Table 3.5

The sixty five alignments generated for each shikimate enzyme produce a range of alignments. The range of results for five quantitative measures of the alignments generated are listed. The sixty five alignments for shikimate enzymes e1 to e5, are generated using the scoring parameters as listed in table 3.2 and the multiple alignment programs and matrices as listed in table 3.3. The five values listed, **a** to **e**, are as described in table 3.3: (**a**) the total number of gaps in the alignment; (**b**) the total number of insertions, regardless of length; (**c**) the number of identical residues; (**d**) the percentage identity (the number of identical residues divided by the length of shortest sequence); and (**e**) the length in residues of the multiple alignment.

percentage identity, relative to the other shikimate enzymes, and its low range of alignment values. The three alignments correspond to the alignments with the highest (one hundred and sixteen), a middle (one hundred and ten) and the lowest (ninety) number of identical residues. Only fifty eight percent of alignments A and B are identically aligned and, when alignment C is included, the length of the alignment that the three have in common decreases to forty two percent (table 3.6i). Greater than seventy-six percent of the alignment's identical residues are found in these common regions (table 3.6 ii). Thus, the majority of the identical residues, which act as aids while visually aligning, are concentrated in the minority of the sequence. Visual examination confirms the large variations found in these three alignments. Creating a consensus alignment from such alignments is a problematic and subjective procedure whose difficulty increases when the other alignments generated are also considered. The other four enzymes also show a large degree of disparity in their alignments, again leading to problems in generating an objective consensus alignment. In the case of the shikimate enzymes where the sequence identity is low, automatic aligning, even when followed by manual aligning, leads to unsatisfactory, subjective alignments. This is also likely to be the case with other low identity protein families.

(i)

	B	C	B & C
A	243 (58.4%)	185 (44.6%)	175 (42.2%)

(ii)

	A	B	C
Total number of identical residues	116	110	90
Number of identical residues found outside the common regions	32	26	6

Table 3.6

The length of the regions in common between three multiple alignments of enzyme e1 and the number of identical residues found in these regions. The alignments as shown labelled A, B and C in figure 3.1, are used. These alignments are generated using the ALIEN algorithm with three different sets of scoring parameters and represent the alignments found with the highest (116), a medium (110) and lowest (90) number of identical residues. (**i**) The length of the common region between alignment A and the other alignments is given in residues and the percentage of the alignment that the common region represents is shown in brackets. (**ii**) The total number of identities is shown for each of the three alignments as well as the number of residues found outside the regions common to all three alignments.

Chapter 4 Developing the Mix'n'Match method

4.1 Original development of novel method

Re-examining the premise of the aligning algorithms reveals that they do not use all of the information that can be brought to bear on the problem of aligning low identity proteins. An observation that can help is that, in a protein family, the sequences alternate between structurally conserved regions, which correspond to the framework of secondary structural units (e.g., α-helices and β-strands), and Loosely Conserved Regions (LCRs), which correspond to surface loops and coils or species specific secondary structural units. The structurally conserved regions show far more similarity than the LCRs [48-51]. Furthermore, indels are almost always found in surface loops and coils rather than within secondary structural units [52, 53]. A consequence of these observations for aligning sequences is that the LCRs are far more difficult to align correctly than the conserved regions.

Therefore, a better method of aligning than the above methods, which consider sequences in their entirety, would be to assess and define these two types of regions separately; the conserved regions to be used as 'anchors' and the LCRs between these considered independently.

Inspection of the alignments produced for the shikimate enzymes (e.g., see figure 3.1) shows that conserved regions are apparent. This is in agreement with Vingron and Argos's definition of conserved regions as "the regions unaffected by these parametric [gap penalties or similarity matrices] alterations" [244]. These can be used as the anchors for further alignment. Determining the real biological alignment for LCRs is probably indeterminate at the present time using automatic methods. A way around this is to generate a range of alignments and let the biologist examine them and determine, on the basis of their knowledge, which is best for that specific family of enzymes.

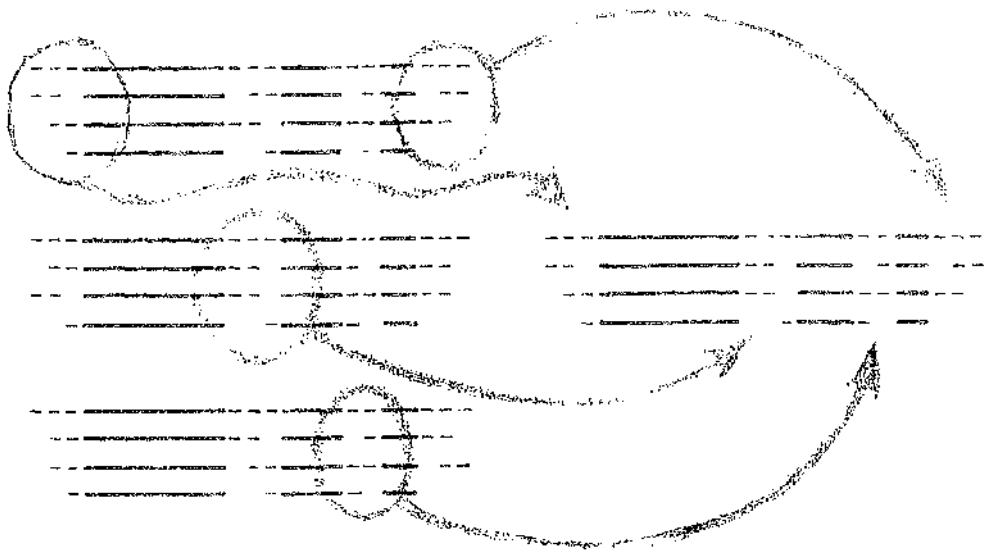
These observations led to the development of a novel multiple alignment procedure consisting of three steps. In the first step, several multiple alignments are generated using different gap penalties, scoring matrices and programs. In the second step, this set of multiple alignments is used to find the conserved regions, termed Strongly Conserved Regions (SCRs), and also to generate a range of different possible alignments for each LCR.

As each initial alignment in the set is mathematically near-optimal, each possible alignment for each LCR will be mathematically, if not biologically, reasonable. This leads to the final step, where the user picks the biologically 'best' alignment, for each LCR, from the range of possible alignments for that LCR. Many criteria could be used to help in this choice, and as an aid in choosing a 'best' alignment, secondary structure predictions are made for the proteins and displayed alongside each of the alignments in each set. These predictions are helpful because it is known that almost all indels occur in loops or 'coils', so LCR alignments that include indels among residues predicted to adopt secondary structures can be avoided. This is consistent with the recommendation of Barton and Sternberg [200], that alignment procedures include secondary structure information. The alignments for the SCRs and LCRs are then joined to produce a final alignment. This procedure has been named Mix'n'Match [247] after the interactive selection process used to pick LCRs from different alignments. The principle is illustrated in figure 4.1.

As previously described, several methods have been developed for finding SCRs in both pairwise and multiple alignments [244, 249, 274, 277]. Mix'n'Match uses a novel method for choosing SCRs which is advantageous in two respects: utilising a large range of parameters to define SCRs greatly reduces the adverse impact that varying parameters can have on the results; and it joins together the process of SCR and LCR determination into a single step, reducing computational overheads. Another benefit of the method, is that it elevates the importance of LCRs, the hardest to define yet biologically important regions of alignments, to a central role in the aligning procedure.

Figure 4.1

The principle of Mix'n'Match. Several multiple alignments are generated, which differ, because different parameters are used to generate them. The differences are all at the Loosely Conserved Regions (LCRs), drawn by fine lines, with none at the Strongly Conserved Regions (SCRs), drawn by thick lines. In Mix'n'Match, the SCRs are first defined and the LCRs are then considered independently. This allows LCRs from different alignments to be mixed to produce a better final alignment.



Initial
multiple alignments

Final
multiple alignment

4.2 Initial test of methods

A small scale study is undertaken using the already collected alignments for the shikimate enzymes to test the feasibility of the Mix'n'Match method. The study considers the number of LCRs and the number and lengths of SCRs found, as the number of multiple alignments used to delineate these regions increases. The delineation of the SCRs and LCRs from the multiple alignments is carried out manually. The available secondary structure prediction methods are also examined.

4.2.1 Delineation of SCRs and LCRs

The SCRs are defined first. Definition of the SCRs is carried out in a step-wise manner. The first two alignments are compared to produce a list of regions that are identically aligned. The next alignment is then compared with this list to produce a new list of identically aligned regions. This process is repeated until all alignments have been considered. This results in a list of regions that are identical in all alignments - the SCRs. Once the SCRs have been defined, the LCRs are the regions in between the SCRs and between the SCRs and the start or end of the alignments. The lengths of the SCRs can only decrease and the lengths of the LCRs increase, as the number of alignments considered increases.

A feature of this method is that if a set of parameters produce an alignment completely dissimilar to the others, then no SCRs are found. To alleviate this problem, the alignments are ranked in order of similarity and the most similar alignments are considered first. The ranking function chosen is the number of identities in the alignments. The number of identical residues in the sixty five alignments used show a spread of values (see figure 4.2), enabling its use as a ranking function.

The data collected for the study is shown in table 4.1. The percentage length of the SCRs as the number of alignments under consideration increases are plotted in figure

Figure 4.2

Showing the number of identical residues in the sixty five alignments produced for each of the five shikimate enzymes. The alignments are generated for each enzyme using the gap penalties as listed in table 3.2 and the matrices and multiple alignment programs as listed in table 3.3. The shikimate enzymes are labelled e1 to e5. Each column shows the sum of values for the enymes, e.g., there are eleven alignments of enzyme e4 and six of enzyme e5 with one identical residue

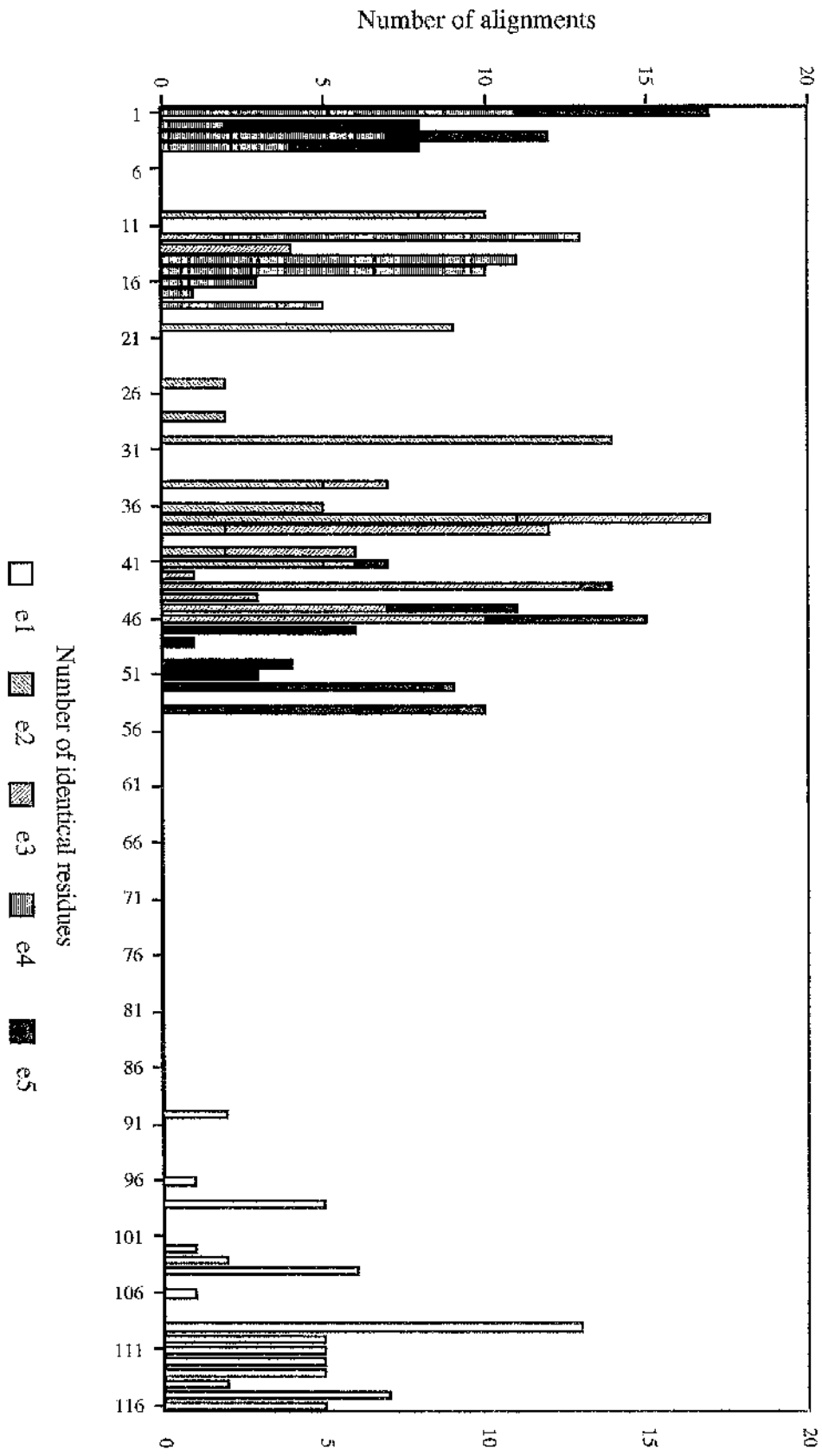

Number of alignments
0
5
10
15
20
1
6
11
16
21
26
31
36
41
46
51
56
61
66
71
76
81
86
91
96
101
106
111
116
Number of identical residues
e1
e2
e3
e4
e5

Table 4.1

Showing the results obtained using the Mix'n'Match method on sixty five alignments for the shikimate enzymes e1 to e5. The sixty five alignments are generated for each enzyme using the gap penalties as listed in table 3.2 and the matrices and multiple alignment programs as listed in table 3.3. They are ranked in order of their number of identities, highest first, and are then consecutively added together and the Mix'n'Match algorithm applied. The number of alignments used to calculate the data is shown in the column headed 'No of aligs'. The results are shown for the five shikimate enzymes, labelled e1 to e5. The data show is: (**SCR**) the number of Strongly Conserved Regions (SCRs) found in the alignments, (**LCR**) the number of Loosely Conserved Regions (LCRs) found in the alignments, (**LCR number**) the number of alignments that are found for all the different LCRS in the alignments, (**unique**) the number of unique alignments that are found for all the different LCRS in the alignments, (**SCR length**) the total length in residues of all the SCRs in the alignment, and (**% length**) the total length in residues of all the SCRs in the alignment as a percentage of the overall alignment length.

No of aligs	e1						e2						e3				e4						e5					
	scr	lcr	lcr number	unique	scr length	% length	scr	lcr	lcr number	unique	scr length	% length	scr	lcr	lcr number	unique	scr length	% length	scr	lcr	lcr number	unique	scr length	% length	scr	lcr	lcr number	unique	scr length	% length
2	1	0	0	0	421	100%	1	0	0	0	282	100%	1	0	0	0	316	100%	1	0	0	0	317	100%	1	0	0	0	566	100%
3	1	0	0	0	421	100%	1	0	0	0	282	100%	1	0	0	0	316	100%	1	0	0	0	317	100%	1	0	0	0	566	100%
4	1	0	0	0	421	100%	1	0	0	0	282	100%	1	0	0	0	316	100%	1	0	0	0	317	100%	1	0	0	0	566	100%
5	1	0	0	0	421	100%	1	0	0	0	282	100%	1	0	0	0	316	100%	1	0	0	0	317	100%	1	0	0	0	566	100%
6	9	9	54	18	271	64%	7	8	48	16	218	77%	8	9	54	18	190	60%	4	5	30	10	84	27%	9	10	60	20	263	46%
7	9	9	63	18	271	64%	7	8	56	16	218	77%	8	9	63	18	190	60%	4	5	35	12	84	27%	9	10	70	20	263	46%
8	9	9	72	18	271	64%	8	9	72	18	199	71%	8	9	72	18	190	60%	3	4	32	12	68	21%	9	10	80	20	263	46%
9	9	9	81	18	271	64%	8	9	81	18	199	71%	8	9	81	18	190	60%	3	4	36	12	68	21%	9	10	90	20	263	46%
10	9	9	90	18	271	64%	10	11	110	26	148	52%	8	9	90	18	190	60%	4	5	50	17	62	20%	9	10	100	20	263	46%
11	9	9	99	22	263	62%	10	11	121	26	148	52%	8	9	99	21	179	60%	4	5	55	17	62	20%	9	10	110	28	256	46%
12	9	9	108	22	263	62%	10	11	132	26	148	52%	8	9	108	21	179	57%	4	5	60	17	62	20%	9	10	120	28	256	45%
13	9	9	117	25	263	62%	10	11	143	26	148	52%	8	9	117	21	179	57%	4	5	65	17	62	20%	8	9	117	29	239	45%
14	9	9	126	25	263	62%	10	11	154	26	148	52%	8	9	126	21	179	57%	4	5	70	17	62	20%	8	9	126	29	239	42%
15	9	9	135	25	263	62%	10	11	165	30	146	52%	8	9	135	21	179	57%	4	5	75	18	62	20%	8	9	135	29	239	42%
16	9	9	144	25	263	62%	10	11	176	30	146	52%	8	9	144	26	179	57%	4	5	80	20	62	20%	8	9	144	29	239	42%
17	9	9	153	25	263	62%	10	11	187	30	146	52%	8	9	153	26	179	57%	4	5	85	20	62	20%	8	9	153	29	239	42%
18	9	9	162	25	263	62%	10	11	198	30	146	52%	8	9	162	29	179	57%	4	5	90	20	62	20%	8	9	162	35	236	42%
19	9	9	171	25	263	62%	10	11	209	30	146	52%	8	9	171	29	179	57%	4	5	95	20	62	20%	8	9	171	41	234	42%
20	9	9	180	29	229	54%	10	11	220	32	144	51%	6	7	140	24	155	57%	4	5	100	21	62	20%	8	9	180	46	234	41%
21	9	9	189	29	229	54%	10	11	231	32	144	51%	5	6	126	24	146	49%	4	5	105	21	62	20%	8	9	189	47	234	41%
22	9	9	198	29	229	54%	10	11	242	32	144	51%	5	6	132	24	146	46%	4	5	110	21	62	20%	8	9	198	50	234	41%
23	9	9	207	29	229	54%	10	11	253	32	144	51%	5	6	138	25	141	46%	4	5	115	21	62	20%	8	9	207	55	234	41%
24	9	9	216	29	229	54%	10	11	264	32	144	51%	5	6	144	25	141	45%	4	5	120	21	62	20%	8	9	216	55	234	41%
25	9	9	225	36	225	53%	10	11	275	32	144	51%	5	6	150	25	141	45%	4	5	125	22	62	20%	8	9	225	57	234	41%
26	9	9	234	36	225	53%	10	11	286	33	124	44%	5	6	156	25	141	45%	4	5	130	22	62	20%	8	9	234	58	234	41%
27	9	9	243	36	225	53%	10	11	297	33	124	44%	5	6	162	29	141	45%	4	5	135	22	62	20%	8	9	243	60	234	41%
28	9	9	252	36	225	53%	10	11	308	33	124	44%	5	6	168	29	141	45%	4	5	140	22	62	20%	8	9	252	66	194	41%
29	9	9	261	36	225	53%	10	11	319	33	124	44%	5	6	174	29	141	45%	4	5	145	22	62	20%	8	9	261	66	194	34%
30	8	8	240	35	221	52%	10	11	330	33	124	44%	5	6	180	29	141	45%	4	5	150	22	62	20%	8	9	270	66	194	34%
31	8	8	248	36	221	52%	10	11	341	35	113	40%	5	6	186	29	141	45%	3	4	124	21	47	15%	8	9	279	66	194	34%
32	8	8	256	36	221	52%	10	11	352	35	113	40%	5	6	192	30	141	45%	3	4	128	23	47	15%	8	9	288	66	194	34%
33	8	8	264	36	221	52%	9	10	330	35	109	38%	5	6	198	30	141	45%	3	4	132	23	47	15%	8	9	297	66	194	34%
34	7	8	272	39	217	52%	9	10	340	35	108	38%	5	6	204	31	141	45%	3	4	136	23	47	15%	8	9	306	74	192	34%
35	7	8	280	42	217	52%	9	10	350	35	108	38%	5	6	210	31	141	45%	3	4	140	26	46	15%	8	9	315	74	192	34%
36	7	8	288	43	217	52%	9	10	360	35	108	38%	5	6	216	32	141	45%	3	4	144	26	46	15%	8	9	324	74	192	34%
37	7	8	296	43	217	52%	9	10	370	35	108	38%	5	6	222	32	141	45%	3	4	148	26	46	15%	8	9	333	74	192	34%
38	7	8	304	43	217	52%	9	10	380	38	104	37%	5	6	228	33	141	45%	3	4	152	26	46	15%	8	9	342	74	192	34%
39	7	8	312	45	217	52%	9	10	390	38	104	37%	5	6	234	33	141	45%	3	4	156	26	46	15%	8	9	351	78	192	34%
40	7	8	320	45	217	52%	9	10	400	38	104	37%	5	6	240	37	141	45%	3	4	160	26	46	15%	8	9	360	78	192	34%

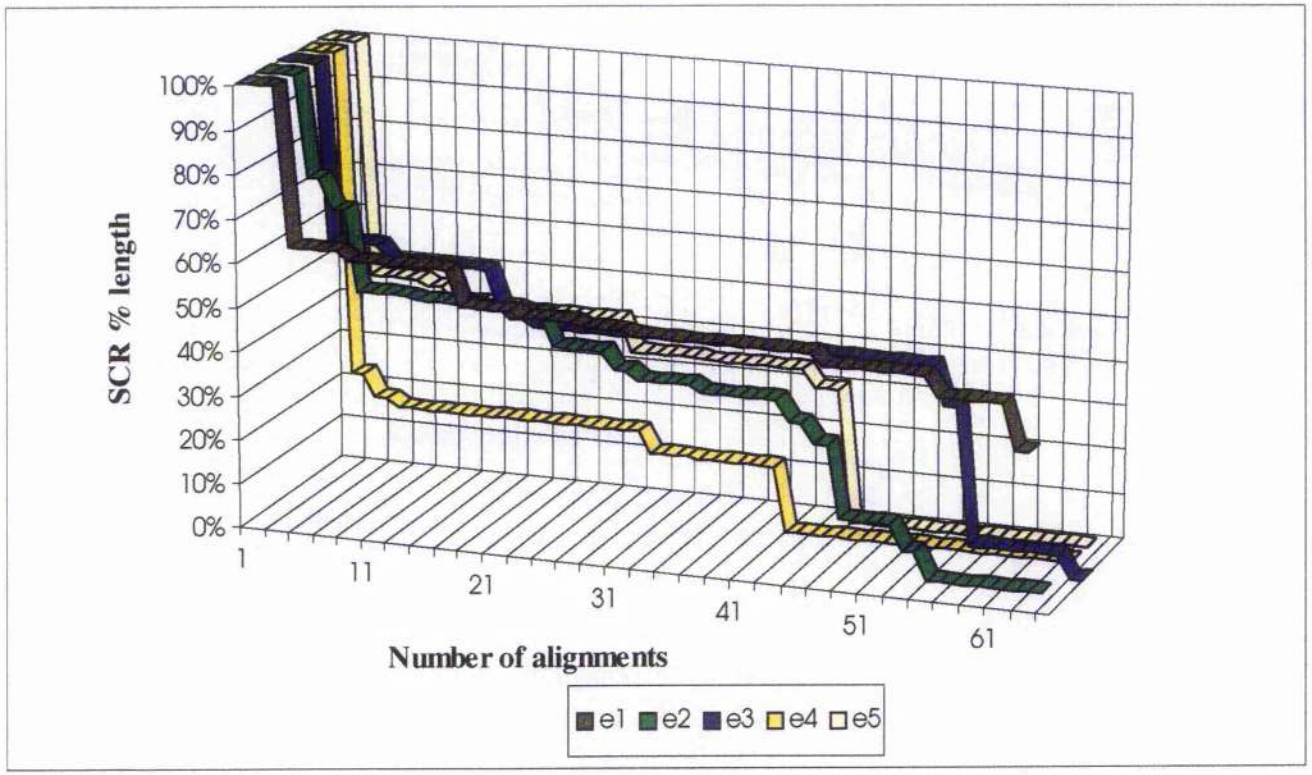
No of aligs	e1						e2						e3						e4						e5					
	scr	lcr	lcr number	uniq ue	scr length	% length	scr	lcr	lcr number	uniq ue	scr length	% length	scr	lcr	lcr number	uniq ue	scr length	% length	scr	lcr	lcr number	uniq ue	scr length	% length	scr	lcr	lcr number	uniq ue	scr length	% length
41	7	8	328	45	217	52%	9	10	410	38	104	37%	5	6	246	37	141	45%	3	4	164	26	46	15%	8	9	369	78	192	34%
42	7	8	336	45	217	52%	9	10	420	38	104	37%	5	6	252	37	141	45%	0	0	0	0	0	0%	8	9	378	78	192	34%
43	7	8	344	45	217	52%	9	10	430	38	104	37%	5	6	258	37	141	45%	0	0	0	0	0	0%	7	8	344	78	170	34%
44	7	8	352	45	217	52%	9	10	440	38	104	37%	5	6	264	37	141	45%	0	0	0	0	0	0%	7	8	352	78	170	30%
45	7	8	360	45	217	52%	8	9	405	38	91	32%	5	6	270	41	141	45%	0	0	0	0	0	0%	7	8	360	78	170	30%
46	7	8	368	45	217	52%	8	9	414	38	91	32%	5	6	276	41	141	45%	0	0	0	0	0	0%	0	0	0	0	0	30%
47	7	8	376	45	217	52%	7	8	376	38	79	28%	5	6	282	41	141	45%	0	0	0	0	0	0%	0	0	0	0	0	0%
48	7	8	384	48	217	52%	7	8	384	38	79	28%	5	6	288	41	141	45%	0	0	0	0	0	0%	0	0	0	0	0	0%
49	5	6	294	42	206	49%	3	4	196	29	33	12%	5	6	294	41	141	45%	0	0	0	0	0	0%	0	0	0	0	0	0%
50	5	6	300	42	206	49%	3	4	200	29	33	12%	5	6	300	43	140	45%	0	0	0	0	0	0%	0	0	0	0	0	0%
51	5	6	306	42	206	49%	3	4	204	29	33	12%	5	6	306	43	140	44%	0	0	0	0	0	0%	0	0	0	0	0	0%
52	5	6	312	42	206	49%	3	4	208	29	33	12%	5	6	312	44	140	44%	0	0	0	0	0	0%	0	0	0	0	0	0%
53	5	6	318	43	206	49%	3	4	212	29	33	12%	5	6	318	44	140	44%	0	0	0	0	0	0%	0	0	0	0	0	0%
54	5	6	324	43	206	49%	1	2	108	24	17	6%	5	6	324	45	140	44%	0	0	0	0	0	0%	0	0	0	0	0	0%
55	5	6	330	45	206	49%	1	2	110	24	17	6%	5	6	330	45	140	44%	0	0	0	0	0	0%	0	0	0	0	0	0%
56	5	6	336	45	206	49%	0	0	0	0	0	0%	4	5	280	44	113	44%	0	0	0	0	0	0%	0	0	0	0	0	0%
57	5	6	342	45	206	49%	0	0	0	0	0	0%	4	5	285	44	113	36%	0	0	0	0	0	0%	0	0	0	0	0	0%
58	4	5	290	43	187	44%	0	0	0	0	0	0%	1	2	116	32	16	36%	0	0	0	0	0	0%	0	0	0	0	0	0%
59	4	5	295	43	187	44%	0	0	0	0	0	0%	1	2	118	32	16	5%	0	0	0	0	0	0%	0	0	0	0	0	0%
60	4	5	300	43	187	44%	0	0	0	0	0	0%	1	2	120	33	16	5%	0	0	0	0	0	0%	0	0	0	0	0	0%
61	4	5	305	43	187	44%	0	0	0	0	0	0%	1	2	122	33	16	5%	0	0	0	0	0	0%	0	0	0	0	0	0%
62	4	5	310	43	187	44%	0	0	0	0	0	0%	1	2	124	35	16	5%	0	0	0	0	0	0%	0	0	0	0	0	0%
63	4	5	315	45	187	44%	0	0	0	0	0	0%	1	2	126	35	16	5%	0	0	0	0	0	0%	0	0	0	0	0	0%
64	2	3	192	43	145	34%	0	0	0	0	0	0%	1	2	128	36	16	5%	0	0	0	0	0	0%	0	0	0	0	0	0%
65	2	3	195	43	145	34%	0	0	0	0	0	0%	1	2	130	36	16	5%	0	0	0	0	0	0%	0	0	0	0	0	0%

4.3 for each of the five shikimate enzymes. The expected downward trend in the lengths of the SCRs is apparent. The first five alignments produce a single SCR with the same length as the alignments, indicating that these alignments are identical. When all sixty five alignments are considered, thirty four percent of enzyme e1 and five percent of enzyme e3 consist of SCRs. With enzymes e2, e4 and e5 however, no SCRs are found when all the enzymes are used. This point is reached with fifty six alignments for enzyme e2, forty two alignments for enzyme e4 and forty six alignments for enzyme e5. The SCR percentage length between the two extremes of considering two and sixty-five alignments show a sharp decline of between twenty three and eighty three percent once the sixth and first non-identical alignment is considered. The lengths of the SCRs then slowly decrease - the drop in percentage SCR length between considering ten and forty one alignments ranges from five to fifteen percent - until an extremely dissimilar alignment is considered, producing a second sharp decline of between fifteen and thirty one percent. In the case of enzymes e4 and e5, the second decline leads to no SCRs being found. For enzymes e2 and e3, short SCRs are still found, although for enzyme e2 these disappear after consideration of seven further alignments. For enzyme e1, the initial aligning does not produce an extremely dissimilar alignment and the second sharp decline is not observed.

After the sixth alignment, four of the enzymes (e1, e3, e4 and e5) show a decrease in the length of SCRs of less than half, before the second sharp decline or all the alignments are considered (e.g., the percentage SCRs length for enzyme e3 falls from sixty to thirty six percent). This indicates that as the number of identities reduces, the alignments of the conserved regions change only slightly. For enzyme e2, the comparable drop in length is from seventy seven to twenty eight percent, indicating that the alignments produced are more varied. The choice of cut-off point to stop considering alignments will have a more pronounced effect on the SCR length for e2 than on the other four enzymes. For enzyme e4, the SCRs found when considering six alignments constitute only twenty seven percent of the alignment. This is almost three times less than found for enzyme e3 and approximately two times less than the other three enzymes. The

Figure 4.3

Illustrating the variation in the percentage of alignment found in Strongly Conserved Regions (SCRs) as the number of alignments, used by the Mix'n'Match algorithm to find the SCRs, increases. Results are shown for the five shikimate enzymes, e1 to e5. Sixty five initial alignments are input to Mix'n'Match for each enzyme. These are generated using varying gap penalties (shown in table 3.2), scoring matrices (described in table 3.3 and listed in appendix C) and multiple alignment programs as described in the text. The alignments are ranked according to the number of identical residues, highest to lowest. The alignments are consecutively added together one at a time and the Mix'n'Match algorithm applied.


SCR % length
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1
11
21
31
41
51
61
Number of alignments
e1
e2
e3
e4
e5

percentage length of e4 drops to fifteen percent before the second sharp drop, which is approximately two times less than that found at the comparable point for the other enzymes. This illustrates that for any chosen cut-off point, the length of SCRs in an alignment can vary dramatically.

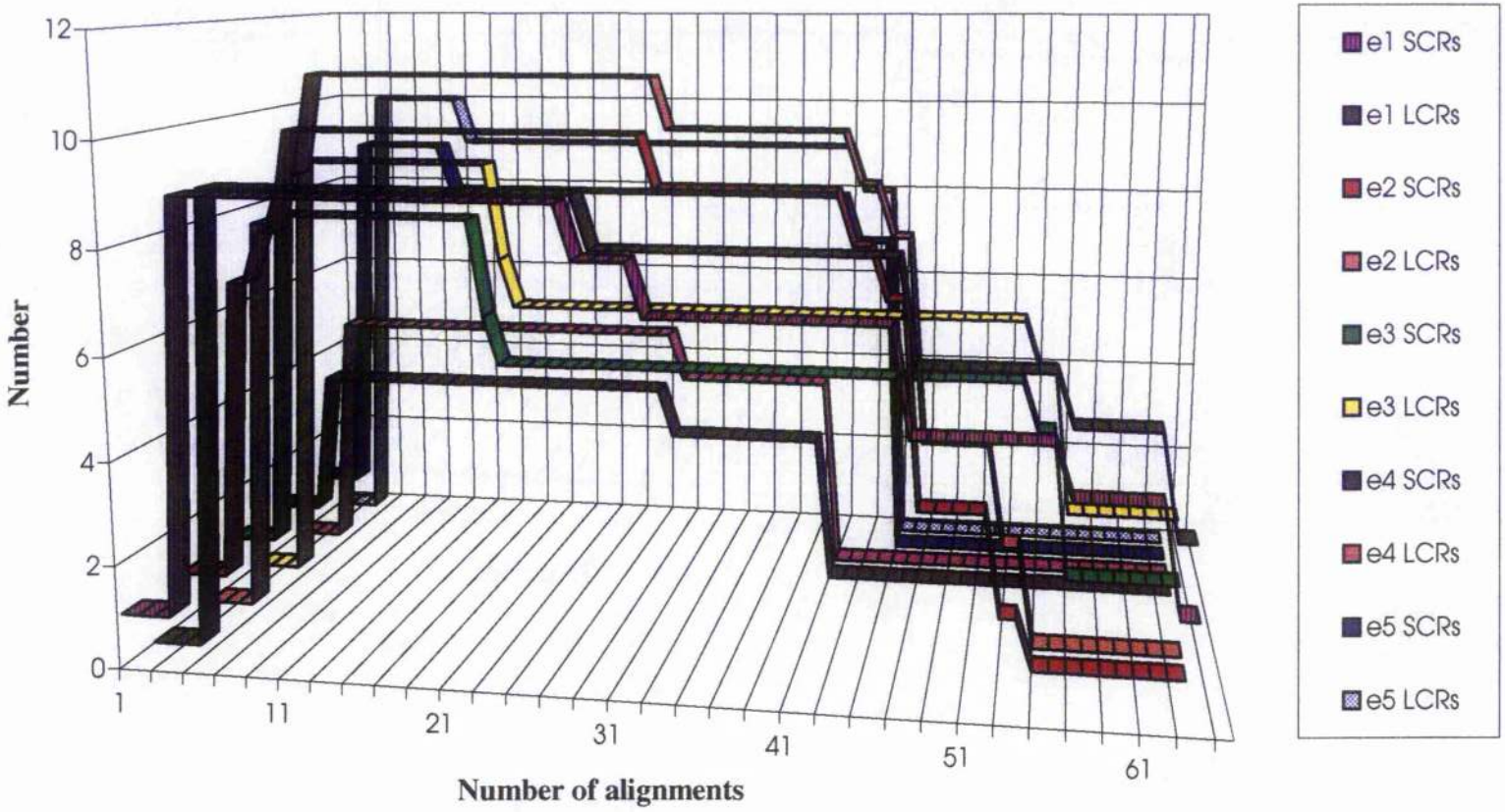
The initial number of SCRs for all of the enzymes is one with no LCRs being found. After the initial five alignments, the number of SCRs increases to between four and nine for the five enzymes. As more alignments are considered, for three of the enzymes (e1, e3 and e5) this number declines, with accompanying decreases in the length of the SCRs (see figure 4.4). The length of the SCRs can decline even when the number of SCRs remains the same, with removal of residues from the ends of the SCRs. For enzyme e2, the number of SCRs found increases from seven to a high of ten before decreasing, and for enzyme e4, the number varies between four and three. The overall length of the SCRs in residues, however, always declines. These effects are caused by a SCR being split into multiple SCRs when residues in the middle of the single SCR no longer qualify as conserved.

The number of LCRs in the alignment is equal to or one more or less than the number of SCRs. The case where both ends of the alignment are parts of SCRs results in the number of LCRs being one number lower than the number of SCRs. This is seen for the first five alignments, where the entire alignment is a single SCR. Equal numbers of LCRs and SCRs are seen when one end of the alignment is part of a SCR and the other end is part of a LCR. This occurs in the alignments of enzyme e1. When both ends of the alignment are LCRs, their number will be one greater than that of the SCRs. This is the most common case and is seen in all of the enzymes. This reflects the N and C sequence termini usually being on the exterior of proteins and loosely conserved.

Defining the SCRs automatically defines the LCRs. All alignments are then examined and the different alignments for each LCR are collected (see column marked 'LCR number' in table 4.1). The number found is the number of LCRs multiplied by the number of alignments under consideration. A more useful value is given when the

Figure 4.4

Illustrating the number of Strongly Conserved Regions (SCRs) and Loosely Conserved Regions (LCRs) found as the number of alignments, used by the Mix'n'Match algorithm to find the SCRs and LCRs, increases. Results are shown for the five shikimate enzymes, e1 to e5. Sixty five initial alignments are input to Mix'n'Match for each enzyme. These are generated using varying gap penalties (shown in table 3.2), scoring matrices (described in table 3.3 and listed in appendix C) and multiple alignment programs as described in the text. The alignments are ranked according to the number of identical residues, highest to lowest. The alignments are consecutively added together one at a time and the Mix'n'Match algorithm applied.


12
10
8
6
4
2
0
Number
1
11
21
31
41
51
61
Number of alignments
e1 SCRs
e1 LCRs
e2 SCRs
e2 LCRs
e3 SCRs
e3 LCRs
e4 SCRs
e4 LCRs
e5 SCRs
e5 LCRs

alignments found for each LCR are examined and duplicates removed (see column marked 'unique' in table 4.1). After considering the first six alignments, only two of which are unique, the average number of unique alignments per LCR is two, indicating that different alignments are found for each LCR. As the number of alignments considered increases, the average number of unique alignments found per LCR also increases, up to a maximum of eighteen unique alignments per LCR (found for enzyme e3).

A stepping effect in the percentage lengths of SCRs and the number of SCRs and LCRs found is noticeable where a number of consecutive alignments produce the same values (see figures 4.3 and 4.4). This is a result of the sixty five alignments used not producing sixty five different alignments. As detailed previously, different gap penalties and scoring matrices can produce identical alignments.

4.2.2 Testing the secondary structure prediction methods

The secondary structure prediction methods available on the University of Glasgow VMS system are the algorithms of Chou and Fasman (CF) [133, 134, 292] and Garnier, Osguthorpe and Robson (GOR) [135, 137] as implemented in the GCG programs PEPPLOT [293] and PEPTIDESTRUCTURE [178]. The accuracy of these methods has been the subject of studies [135, 149, 155-157] which indicate that their accuracy is in the order of fifty one [155] to seventy one percent [134] for the CF method and fifty three [156] to sixty three percent [137] for the GOR method.

4.2.2.1 Chou and Fasman, Garnier, Osguthorpe and Robson methods

The GCG programs encompass three different implementations of the GOR algorithm. One implementation (in PEPPLOT) uses decision constants while the others (one in PEPPLOT and one in PEPTIDESTRUCTURE) do not. Decision constants are a means

of biasing the predictions if the percentage of residues in α-helices and β-strands is known.

A comparison of these GCG implementations, as well as the CF algorithm, is conducted to examine their level of accuracy. Tests are carried out on a total of 2407 residues from nine proteins with known three dimensional structures. The structural information is taken from the Brookhaven structure database [294]. The methods are used to carry out three state predictions (α-helix, β-strand or other structure) and these are compared with the actual structure of the residues.

The two implementations of the GOR algorithm without decision constants differ in their accuracy of prediction in every case (see table 4.2). The PEPTIDESTRUCTURE program's implementation is more accurate for seven of the proteins by up to ten percent, with an overall accuracy two percent greater. The implementation of the GOR algorithm where decision constants are used is the most accurate, with an overall accuracy three or five percent higher than the other two implementations. This greater level of accuracy is mostly due to more accurate predictions, ranging from seven to twenty three percent better, for the three all β structure proteins. For the six α/β structure proteins, the use of decision constants has no perceptible benefit with a GOR implementation without decision constants producing more accurate predictions. The CF algorithm produces the best or equal best predictions for three of the proteins and is third most accurate overall. Although the difference in the overall level of accuracy between the CF and GOR implementations is small at five percent, the variation for individual proteins can be large, with a difference in accuracy of thirty three percent being recorded for the sipunculan worm retractor muscle myohaemerythrin protein. The accuracy of the methods lie at the lower end of the ranges noted above for these algorithms, ranging from fifty three to fifty eight percent.

Sequence name	Length (residues)	% alpha	% beta	% CF	% GOR	% GOR2	% GOR3
1tim	247	46	17	58	62	52	59
2sod	151	2	38	67	68	66	66
2mhr	118	70	0	43	59	53	76
2hhb	287	78	0	53	61	55	71
2mcp	443	48	4	60	54	63	70
3ldh	329	41	13	47	46	42	46
1etu	177	44	20	53	47	53	52
3adk	194	55	13	59	58	57	56
3grs	461	34	24	46	51	46	45
Totals	2407			54	55	53	58

Table 4.2

Showing the accuracy of the GCG secondary structure prediction programs, PEPPLOT and PEPTIDESTRUCTURE. The sequences are listed using their Brookhaven structure database names; chicken triose phosphate isomerase (1tim), bovine superoxide dismutase (2sod), sipunculan worm retractor muscle myohaemerythrin (2mhr), human haemeoglobin (2hhb), mouse immunoglobin fab (2mcp), dogfish lactate dehydrogenase (3ldh), eshcherichia coli elongation factor Tu (1etu), porcine adenylate kinase (3adk), and human glutathione reductase (3grs). The percentage of the sequences in α-helical and β-strand structure (labelled '% alpha' and '% beta', respectively) is taken from the Brookhaven files. The PEPTIDESTRUCTURE program is used to predict secondary structure using the methods of CF and GOR. The PEPPLOT program is used to predict the structure using the GOR method (GOR2) and the GOR method using decision constants (GOR3). The decision constants used are based on the predicted percentage of alpha and beta structure of the GOR2 method where no decision constants are used. The accuracy is the number of residues with the correctly predicted structure divided by the number of residues in the sequence.

4.2.2.2 The method of Cohen *et al*.

The prediction algorithm of Cohen *et al*. [146], with a claimed accuracy for predicting turn residues of ninety eight percent, is also examined. The algorithm starts by assigning turn residues using hydrophilicity and the ideal spacing of turns in a sequences. The regions in between the turns are then assigned secondary structures using a pattern recognition scheme. The program is available for UNIX computers, however, the available system runs the VMS operating system. A program to carry out the initial turn assigning step was developed in FORTRAN according to the details given in the reference (see appendix G). The turn generating algorithm searches the sequence for patterns of hydrophilic residues which define definite turn regions (T1). If the length between such turns is less than nineteen residues, no further turns are searched for. If the length is greater, weaker turn regions (T2 and T3) are looked for in an order dependent on the length. Eleven residues are defined as being hydrophilic for T1 regions (D, E, G, H, K, N, P, Q, R, S and T) with increases in the numbers for the weaker turns (T2 residues = the T1 residues plus Y and T3 residues = the T2 residues plus A). The patterns these residues must occur in is also defined.

The resulting code is run on a single test sequence (porcine adenylate kinase, 3adk from the Brookhaven structural database [294]) with a known three dimensional structure. Only the strongest turn predicting T1 regions are looked for and the known structure was classified into a three state prediction of α-helical, β-strand or turn residues. The accuracy in predicting turn residues is ninety one percent (see table 4.3) however, this coincides with a disturbingly high false positive rate of sixty two percent. Seventy seven percent of the test sequence is predicted as having a turn structure instead of the actual value of thirty two percent. To reduce over prediction, the code is modified with a filtering step not mentioned in the reference, so that a 'turn residue' can contribute to only a single pattern and final prediction (see appendix G and figure 4.5). Fifty four percent of the protein is now predicted as having turn structure, the level of false positives decreases to fifty one percent, however, there is also a drop to eighty one

Figure 4.5

The predicted turn residues according to the algorithm of Cohen *et al.*. The predictions are carried out on porcine adenylate kinase. The sequence is listed in the line labelled 'sequence' with the structure (as taken from the structure database entry) given in the line above. The residues classified as turn predicting are shown in the line 'T1/2 residues'. The patterns as laid out in the reference are then applied to the T1 and T2 residues and the predicted turn residues found are shown in the line labelled 'predicted'. The algorithm is modified with an additional filtering step during the pattern detecting stage with predicted turn residues found shown in the line labelled 'modified'. The filtering step is that once a residue is used in a pattern, it cannot be used as part of any other pattern.

```
Structure          aaaaaTS bbbbbb TTSSaaaaaaaaaaaT  bbbbaaaaaaaaaaaT aaaaaaaaa
Sequence           MEEKLKKSKIIFVVGGPGSGKGTQCEKIVQKYGYTHLSTGDLLRAEVSSGSARGKMLSEI
T1/2 Residues       TTT TTTT     TTTTTTTTTT TT  TTYTYTT TTTT  T T TTTT TTT  TT
Predicted           TTTTTTTT     TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
Modified                TTTT     TTTTTTTTTT      TTTTTTT TTTT      TTTT TTTTTTT

Structure          aTTT    aaaaaaaaaaaaaTTTTT S bbbbS  SSaaaaaaaaaaT   Sbbbbbb
Sequence           MEKGQLVPLETVLDMLRDAMVAKVDTSKGFLIDGYPREVKQGEEFERKIGQPTLLLYVDA
T1/2 Residues       TTTT  T TT  T  TT    T TTTTT   TTYTTT TTTTT TTT TTTT   Y T
Predicted          TTTTT                 TTTTTTT   TTTTTTTTTTTTTTTTTTTTT    TT
Modified            TTTT                  TTTTT   TTTTTT TTTTT     TTTT

Structure          aaaaaaaaaaaaaaaT     STTaaaaaaaaaaaTaaaaaaaTTTT bbbbb    aa
Sequence           GPETMTKRLLKRGETSGRVDDNEETIKKRLETYYKATEPVIAFYEKRGIVRKVNAEGSVD
T1/2 Residues      TTTT TTT  TTTTTTTT TTTTTT TTT TTYYT TTT    YTTTT  TT T TTT T
Predicted          TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT    TTTTT  TTTTTTTTTT
Modified           TTTT      TTTTTTTT TTTTTT TTTTTTT          TTTTT  TTTTTTTT

Structure          aaaaaaaaaaaT
Sequence           DVFSQVCTHLDTLK
T1/2 Residues      T  TT  TT TT T
Predicted          T      TTTTTTT
Modified               TTTTTTT
```

Legend for line labelled 'structure'

- a = α-helical residue
- b = β-strand residue
- T, S = turn residue

Legend for line labelled 'T1/2 Residues'

- T = belong to the T1 classification
- Y = belong to the T2 classification

Legend for lines labelled 'Predicted' and 'Modified'

- T = turn residue

	original method	modified method
Correctly predicted turn residues (%)	90.5	81.0
True positive predictions (%)	37.7	48.6
Number of predicted turn residues	151	105
Number of predicted turn residues (%)	77.8	54.1

Table 4.3

Showing the accuracy of predicting turn residues using the algorithm of Cohen *et al.* and a modified version of this algorithm. The modified algorithm consists of an additional filtering step which excludes any residues being used in more than one turn predicting pattern. The predictions are carried out on the 194 residues of porcine adenylate kinase. A turn residue in the adenylate kinase sequence is defined using three dimensional structural information, as a residue not in an α-helical or β-strand structure. The number of turn residues correctly predicted as such is shown as a percentage. The number of predicted turn residues which have a turn structure (true positives) is shown as a percentage. The number of predicted turn residues and the percentage of the sequence which this represents are also shown. The actual figure for adenylate kinase is sixty three turn residues, which represent thirty two and a half percent of the overall sequence.

percent in the accuracy of predicting turn residues (table 4.3). This method, although having a high level of accuracy in predicting turn residues, has such high levels of falsely predicting turn residues as to lead to the abandonment of its use.

Cohen *et al.* report that the longest length they see without a T1 region is seventy two residues whereas the longest for the adenylate kinase test sequence is seventeen residues. Indeed, no weaker T2 or T3 regions need be looked for. This seems to indicate that their program includes extra filtering steps or other means, not mentioned in the reference, which act to reduce the rate of over prediction. However, the eleven residues that Cohen *et al.* use as turn predictors compose almost sixty percent of protein residues (fifty eight percent in Chou and Fasman's study of twenty nine proteins [292] and fifty seven percent in the PIR database version 19.0 [118]). The over prediction of turn residues, seen in the test sequence, seems a likely consequence of the usage of such a large base of turn predicting residues.

4.2.2.3 Consensus methods

A means of increasing the accuracy of secondary structure predictions is by the use of consensus predictions, either by combined predictions from more than one method [124, 149, 156] or averaging predictions over an aligned family of sequences [122, 123, 127]. As more than one method is available, the increase in accuracy that combined predictions from different methods provide is studied to see if this would be an useful addition to the Mix'n'Match algorithm. Where the predictions agree, assigning the structural type is trivial, however, where the predictions differ, either no prediction is made or additional rules are used to define a type. As the latter requires complex, method specific rules to show an improvement in accuracy, only the former is studied.

The CF and GOR method with decision constants, both from the GCG suite, are used to predict a consensus secondary structure for a total of 2407 residues from nine proteins with known three dimensional structures. These two methods are picked because of

their different algorithmical natures, with the most accurate implementation of the GOR algorithm being chosen. The methods are used to carry out three state predictions (α-helical, β-strand or other structure). A consensus prediction is taken when the two methods agree on the secondary structural type for a residue. The results are then compared with the actual structure of the residues. An increase in prediction accuracy of eight to ten percent is seen (table 4.4) when compared to the accuracy of the original methods (see table 4.2). This figure is in close agreement with the level of accuracy seen in more extensive studies [124, 149]. However, this type of joint prediction has the drawback of not producing predictions for the entire protein sequence. Only fifty seven percent of the residues have a predicted structure which reduces the overall accuracy of this method to thirty seven percent. This is close to the level that random prediction achieves so the use of this method is abandoned.

The alternative consensus method averages the predictions over aligned protein families. The predictions can be modified with rules based on sequence conservation or smoothing over a moveable window. A study of the accuracy of this type of consensus prediction is not carried out for two reasons. Firstly, such a study requires a number of protein families, where all the sequences in a family have known three dimensional structures to enable the construction of robust alignments for comparisons. There are few of these families, limiting the usefulness of any results. Secondly, the main benefit expected from the use of consensus predictions is the reduction in volume of the secondary structure data presented to the user rather than the increase in accuracy. This is because the predictions methods are not accurate enough in their predictions and the increase in accuracy for consensus methods, as noted from previous studies, is small and does not raise the accuracy to levels where the predictions are reliable.

Sequence name	Length (residues)	residues predicted	residues correctly predicted	Consensus prediction accuracy (%)	Residues predicted (%)	Overall prediction accuracy (%)
1tim	247	143	94	66	58	38
2sod	151	84	70	83	56	46
2mhr	118	53	38	72	45	32
2hhb	287	167	123	74	58	43
2mcp	443	267	214	80	60	48
3ldh	329	204	105	51	62	32
1etu	177	108	64	59	61	36
3adk	194	90	68	76	46	35
3grs	461	249	122	49	54	26
Totals	2407	1365	898	66	57	37

Table 4.4

Showing the accuracy of consensus secondary structure prediction programs. The prediction methods used are the CF algorithm in the PEPTIDESTRUCTURE program and the GOR algorithm with decision constants in the PEPPLOT program. The decision constants used are based on the predicted percentage of α-helical and β-strand structure when no decision constants are used. The sequences are listed using their Brookhaven structure database names; chicken triose phosphate isomerase (1tim), bovine superoxide dismutase (2sod), sipunculan worm retractor muscle myohaemerythrin (2mhr), human haemeoglobin (2hhb), mouse immunoglobin fab (2mcp), dogfish lactate dehydrogenase (3ldh), eshcherichia coli elongation factor Tu (1etu), porcine adenylate kinase (3adk), and human glutathione reductase (3grs). A consensus prediction is taken when the two methods agree on the secondary structure type for a residue. The number of residues where the methods agree and the number of times where such predictions agree with the secondary structure as defined in the Brookhaven file are shown. The percentage accuracy, defined as the number of residues with the correctly predicted structure divided by the number of residues predicted is shown. The percentage of residues in the sequence which have a predicted secondary structure and the overall accuracy of the predictions, defined as the number of residues with the correctly predicted structure divided by the number of residues in the sequence are also shown.

4.2.3 Summary of refinements to the Mix'n'Match method

With a large number of alignments generated by altering the aligning parameters and with the definition of SCRs reading "the regions unaffected by parametric alterations", the above data demonstrates that SCRs are not always found when all the alignments are used. However, it also shows that in all cases, SCRs are found if less than all the alignments are considered. A cut-off point in the number of alignments to consider can therefore be used to stop the SCR defining process before the point of finding no SCRs is reached. The reason why no SCRs are found if all alignments are included in the defining process and thus the need for a cut-off point, is a consequence of the mathematical basis of the aligning algorithms, which are capable of misaligning homologous sequences or even aligning completely dissimilar sequences or. It is reasonable to exclude alignments with very poor alignment scores, indicating that the alignment is not biologically reasonable, from the SCR defining process. The definition of SCRs is therefore changed to read "the regions unaffected by parametric alterations *in all the alignments under consideration*" [247].

Ranking the alignments according to an alignment score and excluding the lowest scoring alignments is a necessary feature of this method. The ranking method chosen is the identity score of each alignment. Other scoring schemes could be used, however as no scheme has an objectively better discriminating ability than any other, identity is chosen due to the simplicity with which it can be implemented.

The cut-off point to use should not be too low or too high. In the results detailed above, using less than six or more than forty two alignments could produce SCRs consisting of either all the alignment or none of the alignment. A value of fifteen alignments to use is chosen as roughly midway between these boundaries. It should be noted that the other alignments are only discarded for the purposes of defining SCRs and are still utilised as possible sources of LCR alignments.

It is also considered possible that, as increasing the number of alignments under consideration reduces the size of SCRs found, a similar effect on the size of SCRs may be apparent with increasing number of sequences. The possibility should exist that only a subset of sequences are considered when delineating SCRs.

Initially two secondary structure prediction algorithms in the GCG suite of programs are used to generate four predictions for the sequences. A fifth prediction is added to Mix'n'Match [247] using the Gascuel and Golmard Basic Statistical Method (GGBSM) algorithm [295]. This is achieved by recoding into FORTRAN the PASCAL code outlined in the reference. This method has a claimed accuracy of fifty eight percent.

The use of five secondary structure predictions for each sequence in the alignments may lead to the user being presented with an overwhelming amount of data. The amount of secondary structure prediction information shown to the user was reduced by converting the predictions into consensus predictions. Two methods are available of which one is studied. This method is shown to increase the accuracy of predictions but as it is only applicable to a limited length of the sequence, is considered unsuitable. The second method, based on averaging the predictions made by a method over an aligned family of proteins, is added to the Mix'n'Match method [247]. The averaging method used is carried out on each method separately, with a consensus prediction from each method shown underneath the aligned sequences. The rules applied, in order of application, are: the predictions are turned into three state predictions (α-helical, β-strand and turn); no consensus prediction is made for a residue if there are only two or less predictions for that residue; and the predicted state in the majority is chosen as the consensus, except if there is more than one state with the highest number of votes in which case no consensus prediction is made. To highlight cases where the predictions are in strong agreement and may provide more accurate predictions, the consensus prediction uses a capital letter if all or all bar one predictions agree for a residue.

The use of the consensus method chosen allows the comparison of different prediction methods for each residue and a measure of whether each methods' prediction was near

unanimous for each residue in the alignment. The predictions are only included as a guide to the user, to supplement their own knowledge of the protein under consideration and are not meant to be used as the sole reason for the choice of LCRs.

Chapter 5 Programming the Mix'n'Match method

5.1 Steps in programming the Mix'n'Match method

Manually implementing the Mix'n'Match method is a laborious and error prone procedure. The appropriately gapped secondary structure predictions and consensus secondary structure predictions have to be lined up underneath each alignment. This requires the use of a multiple alignment editor. The GCG suite of programs available at Glasgow University contains the multiple alignment editing program LINEUP [178]. Alignments produced using the PILEUP program can be entered directly into LINEUP, however, output from the ALIEN program has to reformatted before use. The secondary structure predictions have to be entered one at a time into the editor for every alignment, with whatever gaps are present in the alignments manually inserted into the secondary structures. The consensus secondary structure predictions can then be manually calculated. The alignments are then printed from the editor and examined to determine the SCRs. After determining the SCRs, the alignments have to be re-examined to determine the LCRs. Once the LCRs are known, the user can choose an alignment for each LCR. Once both the SCRs and LCRs are known, these, along with the consensus secondary structure predictions, are entered into the multiple alignment editor to produce a final alignment. The complexity and number of the above steps make inadvertently erroneous data entry a very real possibility and manual usage of this method is too slow to be practical. The LINEUP editor has a limit of handling a maximum of thirty sequences, although fewer must be used if the secondary structure predictions are also entered.

As a result of the above, the decision was made to computerise the Mix'n'Match procedure. An initial version was programmed which computerised the data entry steps, removing the possibility of mistakes from this stage. The program requires two files as input, one containing all the alignments and the other containing the sequences. The

program predicts the secondary structure for each sequence and reads in the alignments. Then, for each alignment, it outputs, in a form ready for input into LINEUP, the alignment along with the correctly gapped secondary structure and consensus secondary structure predictions. The user runs the LINEUP program to produce printable output of the alignments with secondary structure predictions. The printed output is used to determine the SCRs and LCRs and a final alignment is entered into the editor to produce printable output. This version still requires considerable user intervention in delineating SCRs and LCRs and the number of sequences that can be aligned is still bound by LINEUP's limitations.

This version is extended to produce a second, final version containing routines for calculating SCRs and LCRs and capable of producing printable output without the use of an external program. This second version consists of two programs, Mix'n'MatchA and Mix'n'MatchB. Mix'n'MatchA requires the same two input files as before and the output is a list of the different alignments found for each LCR, together with their consensus secondary structure predictions, in a format suitable for printing. Once the user has chosen an alignment for each LCR, Mix'n'MatchB is run which joins the SCRs and the LCRs together producing a final alignment, suitable for printing. This version automates almost every step of the Mix'n'Match procedure and is essentially limitless in the number and length of sequences it can be applied to. The two programs consist of 2766 lines of FORTRAN code (table 5.1) contained in three source code files. The construction and testing of this second version is described below.

5.2 Implementation

The program is implemented in standard FORTRAN 77 on Glasgow University's central Digital VAX computer, running the VMS operating system. This system is accessible through a VT100 terminal on the departmental network, and is written using the VMS editor EDT. The choice of computer system to use is primarily dictated by the

	Mix'n'MatchA	Mix'n'MatchB	Common code	Totals
code	2354	138	274	2766
comments	1089	104	93	1286
Totals	3443	242	367	4052

Table 5.1

Showing the number of coding and comment lines in the programs Mix'n'MatchA and Mix'n'MatchB. Program code which these programs share, is in the source code file 'common'.

need to have close ties with the GCG suite of programs which run on VAX/VMS computers. The choice of programming language is dictated by the ease with which FORTRAN may be learnt when compared with alternatives like C or C++. FORTRAN also has more accessible string array handling capabilities when compared to C or C++. This is important for the Mix'n'Match programs where string arrays are extensively used.

5.3 Data structures

A variety of parameters which control how big the arrays which hold data can be, are set at the start of the program. These are: the maximum number of sequences in an alignment, the maximum number of alignments that can be read in, the maximum length of a sequence, the maximum number of SCRs that can be found, the maximum length of sequence names and the number of secondary structure predictions used. The initial values for these are shown in table 5.2. These values can be changed so that longer or more sequences can be examined, as long as the computer the program is running on has enough memory. If one of these parameters is exceeded during program execution, the program will stop with an error message indicating which value was exceeded and an explanation as to how to increase the parameter. If changes are made, these values must be changed in Mix'n'MatchB as well.

The main data structures are the two dimensional string arrays, '`arr_a`', '`arr_b`' and '`arr_c`'. These are used to store sequences, alignments or secondary structure predictions.

Mix'n'Match parameter	Variable Name	Value of Variable
Maximum number of sequences	MaxSeqNum	30
Maximum number of alignments	MaxAlisNum	100
Maximum length of sequences	MaxSeqLen	2000
Maximum number of SCRs that can be found	MaxNumSCR	100
Number of secondary structure prediction methods used	No_Preds	5
Maximum length of the sequence names	NamLen	11

Table 5.2

Showing the initial values for the main variables used in the programs Mix'n'MatchA and Mix'n'MatchB. These variables are used to set the size of the main arrays used to hold data.

5.4 Algorithms

The Mix'n'Match program undertakes many tasks during execution and the algorithms for three of the most important, calculating a consensus secondary structure prediction and delineation of the SCRs and LCRs, are described below.

5.4.1 Consensus secondary structure prediction

Calculation of the consensus secondary structure takes place in the module 'calc_cons'. It involves reading the secondary structure predictions from file into the string array 'arr_b'. Each alignment is read from file, one at a time, into string array 'arr_a'. The secondary structure predictions are copied into the array 'arr_c' and the appropriate gaps from the alignment in array 'arr_a' inserted into equivalent positions in array 'arr_c'. The consensus secondary structure can then be calculated from array 'arr_c'.

The alignments and secondary structure predictions are stored in fast access, 'unformatted' format files to minimise the time taken to read in the data. Information is stored in these files in the format, integer then character array, for each sequence. The value of the integer is the length of the character array, enabling the data to be read directly into an array.

The position of gaps in the alignment are calculated using the internal FORTRAN string search function 'index'. The start position of the string being searched is modified to be one greater than the position of the last gap found.

The consensus is calculated one column at a time, from the beginning to end of the alignment. Each row is examined in turn and the appropriate counter for structural type incremented. Three structural types are defined, α-helical, β-strand or turn residue, marked in the secondary structure predictions with the characters 'h', 'b' and 't' respectively. A prediction method may predict more than three states or use different characters for these, e.g., using the character 'a' to identify α-helical residues. These are

modified to the three structural type characters in the module 'second' which carries out the structure predictions and writes them to file, thus ensuring that the consensus prediction module only has valid input. This approach allows for extra prediction algorithms to be added without modification to the consensus prediction module.

Once the votes for each structural type are known, the algorithm applied is (in order of application): no consensus prediction is made for a residue if there are only two or less votes for that residue; the state with the majority of votes is chosen as the consensus, except if there is more than one state with the highest number of votes in which case no consensus prediction is made. Small letters are used in the consensus predictions except for the case where the predictions are in strong agreement and may provide more accurate predictions, where a capital letter is used if all or all bar one predictions agree for a residue.

The time spent calculating consensus predictions is dependent on: the number of alignments which have to be read in; the number of gaps in each alignment which have to be placed in array 'arr_c'; the number of sequences, which determines the number of votes to be counted in each column; and the length of the alignment which determines the number of consensus predictions to be made. It is expected that these are linear dependencies. The slowest step is likely to be the first, reading alignments in from file.

A possible improvement to this algorithm would be to reorder the program so that the consensus secondary structure predictions are carried out directly after reading in each alignment, which occurs in the module 'doalis'. At this stage, each alignment is already stored in a string array, so removing the need to read the alignments back from file, resulting in a speeding up of the program. However, the current implementation allows for a clearer program flow and easier maintenance, as the reading alignments and secondary structure prediction modules, are separate and distinct.

5.4.2 Delineation of SCRs

The SCRs are defined as "the regions in the alignments under consideration, unaffected by parametric alterations". Thus delineating the SCRs is a two step procedure: picking alignments to use; and then using these alignments to find regions identically aligned. A large degree of user freedom is programmed into the SCR finding module. Defining the SCRs can be carried out in three ways: automatically, where both the alignments to use in the calculation and the SCRs are found by the program; semi-automatically, where the user chooses which alignments to use, and the SCRs are found in these alignments automatically; and manually where the user defines the SCRs from a user chosen alignment. The algorithms for the automatically calculated sections are detailed below.

Automatically finding which alignments are used to define SCRs involves picking a number of alignments with the highest identity scores. The number of alignments to use is initially set at fifteen (see chapter 4.2.3) but can be increased or decreased by the user, with boundary checks on the value used so that it cannot be greater than the number of alignments read in or less than two. The algorithm used to pick alignments is a two step procedure.

The procedure uses the identity score for each alignment, previously determined in the alignment input module '`doalis`', and which are stored in the integer array '`idscores`'. The scores are indexed so that the score for the first alignment is stored in array position one, for the second alignment in array position two, etc.. The first step is carried out by the module '`idgen`' which searches through array '`idscores`' and places unique identity scores in descending order into the integer array '`idmaxs`'. This algorithm requires three comparisons for every alignment. The second step is carried out by the module '`auto`'. The top score in '`idmaxs`' is compared with the scores in '`idscores`' and all matches are entered in array '`AliToRd`'. This is repeated with the next score in array '`idmaxs`' until the number of alignments to use has been found. For

this second step, if there are x identity scores and y alignments and all alignments are to be used, up to xy comparisons have to be made.

A possible improvement to this algorithm would be to combine the two steps into one. As the highest identity score is found, the alignments with that score are recorded in array 'AliToRd', then those with the next highest identity scores until the number of alignments to use has been picked. With x identity scores and y alignments and all alignments to be used, only $3y$ comparisons instead of $3y + xy$ comparisons would have to be made.

Automatically calculating the SCRs involves finding the regions in the alignments that are aligned identically and is carried out using the module 'scr_fnd'. This module is run if the user chooses either the automatic or semi-automatic SCR calculation. Definition of the SCRs is carried out in a step-wise manner. The first alignment in array 'AliToRd' is read from file into array 'arr_a'. The other alignments listed in array 'AliToRd' are read from file, one at a time, into array 'arr_b' and the two arrays compared to find identical regions.

The first step is to reorder the alignments given in array 'AliToRd' into ascending numerical order using the module 'reorder'. This results in the array listing the alignments in the order in which they will be read in from file. A potentially large loss of speed would result if alignments are numbered out of order as the alignment file would have to be rewound before re-reading through the alignments to find the specified one to input. As detailed in chapter 5.4.1, the alignments are stored in a fast access, 'unformatted' format file.

After reading in an alignment into 'arr_b', the module 'test' is called to conduct the comparison between the two arrays. Comparisons are carried out on array 'arr_b' using a substring or window of 'arr_a'. The 'test' module walks this window along the alignment held in 'arr_a' in one column steps. The size of the window determines the minimum size of a SCR and is initially set to three residues, although it can be

altered by the user. At each new position, the 'test' module calls the module 'ident' which conducts the comparison. 'Ident' first compares the window of sequence one in 'arr_a' to find a match in sequence one of 'arr_b'. Only if a match is found are further comparisons made between the other rows in the arrays. If an identical region is found in the two arrays, a check is made to see if the window of 'arr_a' occurs at any other position in 'arr_b'. The number of times a match is found is passed back to the 'test' module. When no matches are found, the window is advanced one position and 'ident' called again. If one match is found, a hits string, 'seq_a' is marked with asterices at the equivalent position where the match occurred in 'arr_a'. This method is illustrated in figures 5.1. If more than one match is found, this may not be a SCR and the position is ambiguously defined (see figure 5.2). The match is marked as a hit in the hits string but a warning is printed on the screen recommending the user choose their own SCRs or increase the minimum width of SCRs. The window is moved along 'arr_a' and the 'ident' module called until the entire alignment has been compared against 'arr_b'.

A feature of the 'ident' module is that the substring from 'arr_a' is not compared to the entire alignment in 'arr_b'. The search through 'arr_b' uses a window of size plus or minus thirty residues from the position in the alignment of the window from 'arr_a'. This step is taken because the region in 'arr_a' is likely to occur in a roughly similar position in 'arr_b'. Thus searching through all of 'arr_b' is unnecessary and will result in slowing the program down.

After comparing the two alignments, the 'test' module passes back to the 'scr_fnd' module the hit string, 'seq_a'. This string contains a record of where identical regions are found. When the first two alignments are compared, the hits marked in this hits string are placed in a results string, 'seq_c'. For all following comparisons, a logical AND operation is conducted on the two strings with the results stored in 'seq_c'. Thus, after comparison of the first alignment read in against all the others, the places in the

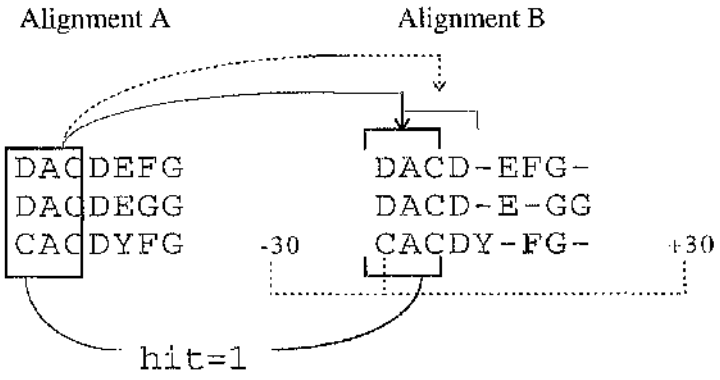


Figure 5.1

The algorithm for finding SCRs. The 'ident' module is used to find regions identically aligned in two alignments. Three columns of residues in alignment A are compared to three columns of residues in alignment B. The number of columns compared at a time is set by the user and determines the minimum possible width of strongly conserved regions. The position of column one in alignment A is used as the centre point of a window space which determines how much of alignment B is compared, shown above, not to scale, as the dotted line below alignment B. A window variable of plus or minus thirty is used with error checks to prevent this window exceeding the size of the alignment. In the above example where the first three columns of alignment A are being compared, positions one to seven of alignment B will be compared (shown diagrammatically above for two of the seven comparisons). Once the columns in the window of alignment B are compared, identical matches cause a variable, 'hit', initially set to zero, to be incremented by one. This variable can be used be the calling routine.

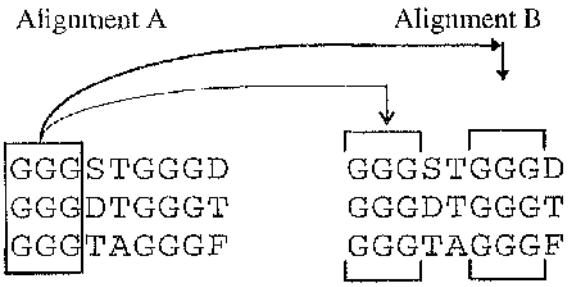


Figure 5.2

Illustrating how two or more matching regions can be found in an alignment. The region boxed in alignment A is identical to two regions in alignment B, shown by connecting lines. Distinguishing which match is the SCR is a non-trivial task.

first alignment where identical regions are found in all the others, are marked with asterices in the string 'seq_c'.

The module 'convert' assesses the position of each of the SCRs in 'seq_c', calls the module 'bugremove' to ensure it is a SCR and if it is, converts it into numerical format and stores it in the two dimensional integer array, 'scrPos'. These positions are found using the internal FORTRAN string search function 'index'. The way the above implementation of the SCR finding algorithm works can make it possible for SCRs to be defined when one of the sequences consists entirely of gaps. These may not be true SCRs (see figure 5.3). The module 'bugremove' identifies sequences which consist entirely of gaps. The other modules in the program which require knowledge about SCRs obtain this information from the array 'scrPos'.

The manual method of visually comparing two alignments, enables SCRs of any size from one residue upwards to be identified, with the only consideration being the amount of time the user must spend. Including such discrimination into an automatic algorithm is a difficult process. As the width of a SCR decreases, the possibility of its occurrence at more than one position in the alignment increases (see figure 5.2). The ability to discriminate which is the correct position for the SCR would be hard to code and expensive in computer time.

The algorithm implemented for finding SCRs requires three parameters: a minimum width of SCR, a window size within which alignments are compared and the number of sequences to use when comparing alignments. The minimum size of a SCR is initially three residues although it can be altered by the user. This parameter is used in the module 'test'. The window size within which alignments are compared is set to thirty residues and is used in the module 'ident'. As noted in chapter 4.2.3, it may be beneficial to be able to alter the number of sequences to use when comparing alignments. This is implemented at the start of module 'scr_fnd' and is initially set to ten. When the user chooses to delineate SCRs using a number of sequences less than are in the alignments, the module 'rand' is called which randomly picks which sequences

```
Alignment 1          ****
      Sequence a     aa-----pd
      Sequence b     cgcdefgpc
      Sequence c     d--d-fgpc

Alignment 2          ****
      Sequence a     a-----apd
      Sequence b     cgcdefgpc
      Sequence c     d--d-fgpc

Alignment 3          ****
      Sequence a     -------aapd
      Sequence b     cgcdefgpc--
      Sequence c     d--d-fgpc--
```

Figure 5.3

Illustrates a potential problem with the method used to choose SCRs. The method finds all regions that are identical between the different alignments looked at. In the three alignments shown above, the identical regions found are shown with a '*' above the appropriate columns. However, although the alignments in sequences *b* and *c* are equivalent, that of sequence *a* is not. This region should not be considered a SCR. These regions can be found by searching for regions which have one or more sequences entirely composed of gaps.

are used. Once selected, all calculations of SCRs and LCRs involve only these selected sequences.

5.4.3 Delineation of LCRs

Once the SCRs are defined, the LCRs can be found. This is a two step procedure. Firstly, every alignment for every LCR is found and written to file. Secondly, these are read back in and the unique alignments for each LCR are written to file. This file, which will be read by the Mix'n'MatchB program, is an 'unformatted' file which stores each alignment with its identifying details. A second version of this information is produced which is meant to be examined by the user and used to choose an alignment for each LCR. This is in a format suitable for printing and also includes consensus secondary structure prediction below each alignment.

Finding all alignments for all LCRs is carried out by the module '`fndlcr`'. The alignment used to calculate the SCRs is still held in the array '`arr_a`'. The file which stores all the alignments, output by the module '`doalis`', is opened and each alignment is read in turn into array '`arr_b`'. The presence in the alignment of each SCR is checked using the module '`ident`'. With this information, the presence of LCRs can be determined using an IF ... ENDIF construct, and those found written to file.

The '`uniq`' module finds the unique alignments for each LCR. The first step is, for each LCR, the alignments for that LCR are read in from the file generated in the previous module, and output to a temporary, 'unformatted' scratch file. The module '`uniq_lcrs`' is then called. This sets up a second temporary, 'unformatted' scratch file and assigns variable names which identify which of the two temporary files is to be read from and which written to. The first alignment is read from temporary file and output to the file '`lcr.data`' which stores the unique alignments - the first alignment read is always unique. All the other alignments are then read in, one at a time, and the '`ident`' module called to compare alignments. Any unique alignments are output to temporary

file. After comparing the alignments, the files to be read from and written to are swapped and the process is repeated until there are no more alignments to be read in the temporary file. This process is illustrated in figure 5.4. The procedure repeats until all the LCRs have been processed

The next module, called 'wrlcr1', outputs the unique alignments in a user readable format, together with extra information which includes the number of gaps and identical residues and the consensus secondary structure prediction. This is accomplished by reading in the data stored in the unique LCR file, 'lcr.data'. The number of gaps and identical residues is calculated from the LCR alignment and the appropriate consensus sequence read from file.

5.5 Optimisation of algorithms

Algorithms can be optimised for a variety of purposes, including speed of execution or memory usage. The benefits of optimising an algorithm are greater the more the algorithm is used. It has been pointed out above that improvements in algorithms can be made to some of the routines to increase the speed of program execution. Improvements in program speed are probably also possible for other modules, including the often called routines for finding SCRs and LCRs, for example, the module 'ident'.

Optimising the code has not been a major factor during the writing of this program. Producing easily modified, robust and easy to follow source code had priority. However, some optimisations are carried out. The need to capitalise alphabetic characters resulted in coding two modules, 'mkcapb' and 'mkcapc' which are fastest with small and long strings respectively. Two approaches to placing numbers held in an array into numerical order were explored for the module 'reorder', with the fastest chosen (see code in appendix H for details). To reduce the memory requirements of the program, where applicable, temporary results are written to files held on disk. To alleviate the speed decrease inherent in this approach, fast access, 'unformatted' style

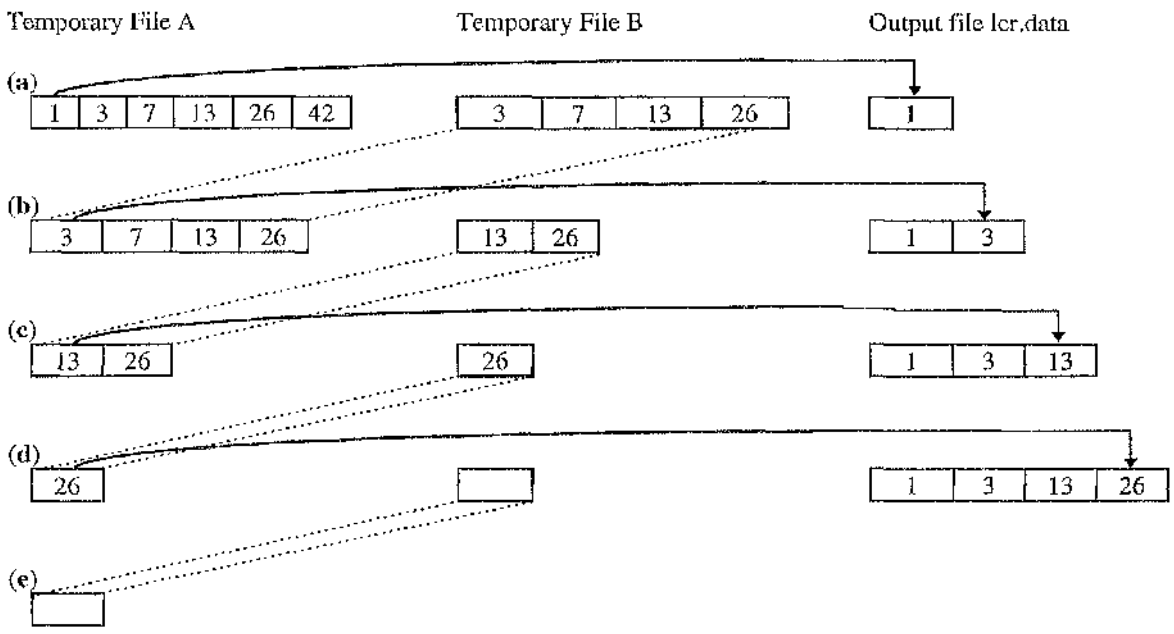


Figure 5.4

Illustrating the principle of the algorithm to find unique Loosely Conserved Regions (LCRs). The alignments found for each LCR are found using a previous module. In the example above, the LCR is found in alignments 1, 3, 7, 13, 26 and 42 and of these, alignments 1, 3, 13 and 26 are unique. (**a**) The previous module has stored these alignments in temporary file A. The first alignment in this file is output to file 'lcr.data'. The rest are tested against this first alignment and different alignments written to temporary file B. Alignment 42 is identical to alignment 1 so is not written. (**b**) Temporary files A and B are swapped (shown by the dotted lines), the first alignment in temporary file A, 3, is written to the output file 'lcr.data' and other alignments not identical to alignment 3 are written to temporary file B. (**c**) Temporary files A and B are swapped, the first alignment, 13, is written to the output file 'lcr.data' and other alignments not identical to alignment 13 are written to temporary file B. (**d**) After writing the first alignment in temporary file A to 'lcr.data', there are no further alignments to compare, leaving temporary file B empty. (**e**) After swapping temporary files A and B, file A is found to be empty so all the unique alignments have been found and output to file 'lcr.data'. The above process get now be repeated on the alignments found for another LCR.

files are used, with data formats chosen, which require the minimum of computation when reading from the file. An example is reading in the alignments in the module 'doalis'. Reading in the alignment data in the original ALIEN or PILEUP format requires extensive coding with multiple error checking. This data is output in a machine readable format so that other modules require few lines of code with only minimal error checks to read the data. A variety of general optimising techniques, for example, unrolling loops and moving as many calculations as possible outside loops, are also implemented.

5.6 Program development

The Mix'n'Match programs are constructed along a modular line with each task broken into separate modules. This allows for easier updating and debugging. The final structure of the programs Mix'n'MatchA and Mix'n'MatchB are shown in figures 5.5 and 5.6. The structure of the initial version of the Mix'n'Match algorithm is shown for comparison in figure 5.7. For the final version of the program, the structure of the reading alignments module is shown in figure 5.8, the secondary structure prediction and consensus secondary structure prediction modules in figure 5.9, the SCR calculating module in figure 5.10, the LCR calculating modules in figure 5.11, and the outputting of SCRs and LCRs modules in figure 5.12. To improve the legibility of these diagrams, the flow of data is not always shown.

Two modules were written with the help of Dr. Ian Walker, Glasgow University Computing Service's VMS System Manager, who supplied the VMS specific commands used to spawn a sub process (needed to call the GCG programs and incorporated in the module 'gcgcmds') and the external function call for generating random numbers (incorporated in the module 'rand').

The structure of Mix'n'MatchA and Mix'n'MatchB compared to the original version illustrate the extent of the modifications which took place during the course of

development. The modifications include: changes to the sequence and alignment input modules after the decision was made to increase the number of formats which could be used; extending the secondary structure prediction modules to be able to run the prediction programs as well as read in their predictions; addition of the GGBSM prediction method; and adding modules for finding and outputting SCRs and LCRs.

In the first version, only ALIEN format alignments could be read, using the module 'alien'. The code that tests for a valid alignment file is removed from this module, expanded to deal with PILEUP as well as ALIEN and incorporated into a separate module 'alis_type'. The 'doalis' module uses 'alis_type' to find out which module should be used to read in the next alignment from file. A module to read in PILEUP alignments, 'pileup_rd', is coded and the module to read in ALIEN alignments is recoded and renamed 'alien_rd'. As different alignment programs output their data using different gap characters and with the sequences ordered differently, modules 'changechar' and 'order_seqs' are constructed to unify the approach used. The routines for calculating the number of identities and gaps are generalised to cope with different formats and placed in separate modules. The module 'ali_wr' is written which outputs the alignment in a format easily read by other modules in the program.

To enable the GCG prediction programs to be run, the sequence files have to be present and for the GGBSM method, have to be read in. Modifications are made to the 'input' module to check for the presence of the sequence files, in either NBRF/PIR or GCG 'file of file names' formats. The module 'second' is expanded to call the modules 'rd_gcg' and 'rd_pir' which read in GCG and NBRF format sequence files, respectively. The module 'ggbsm' is constructed to carry out GGBSM predictions and is called from the module 'second'. To run the secondary structure prediction programs, the module 'gcgprd' is constructed. This is added to the module 'second' in an IF ... ENDIF statement which ensures it is accessed only when the predictions have not already been run.

An alternative way to incorporate GCG programs into a program is to use the GCG library calls. This allows closer integration of the GCG commands and the use of other GCG routines, e.g., for reading in PILEUP format alignments. A drawback is that the format of this library is not static and any changes to the library may require extensive modification to the user program. Using sub processes to call GCG programs should not require modifications when the GCG suite of programs is upgraded.

To increase the flexibility of the method, the modules '`altvar`' and '`changevar`' are written which enable the user to change most of the internal parameters. These include the width of the final output, the minimum width of an SCR and the number of alignments used to pick SCRs from. These modules are incorporated into the '`input`' and '`scrfnd`' modules. Extensive online help is also provided which explains what questions are asked by the program and why.

Testing of the modules is carried out individually and collectively through construction of appropriate driver routines and interactive examination of the code using the VMS debugger.

The source code for the Mix'n'Match programs is shown in appendices H, I and J.

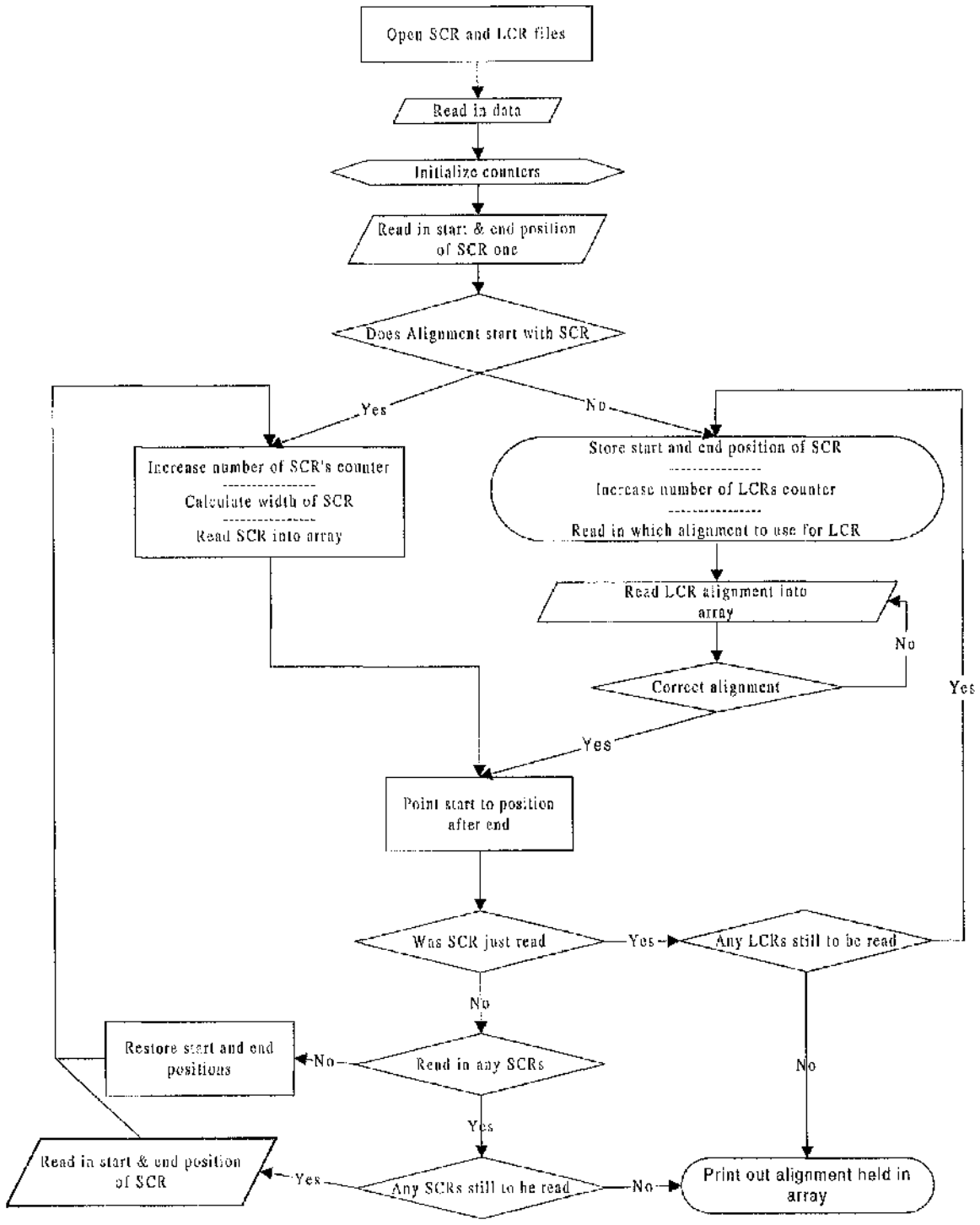


Figure 5.6

A flowchart of the Mix'n'MatchB program.

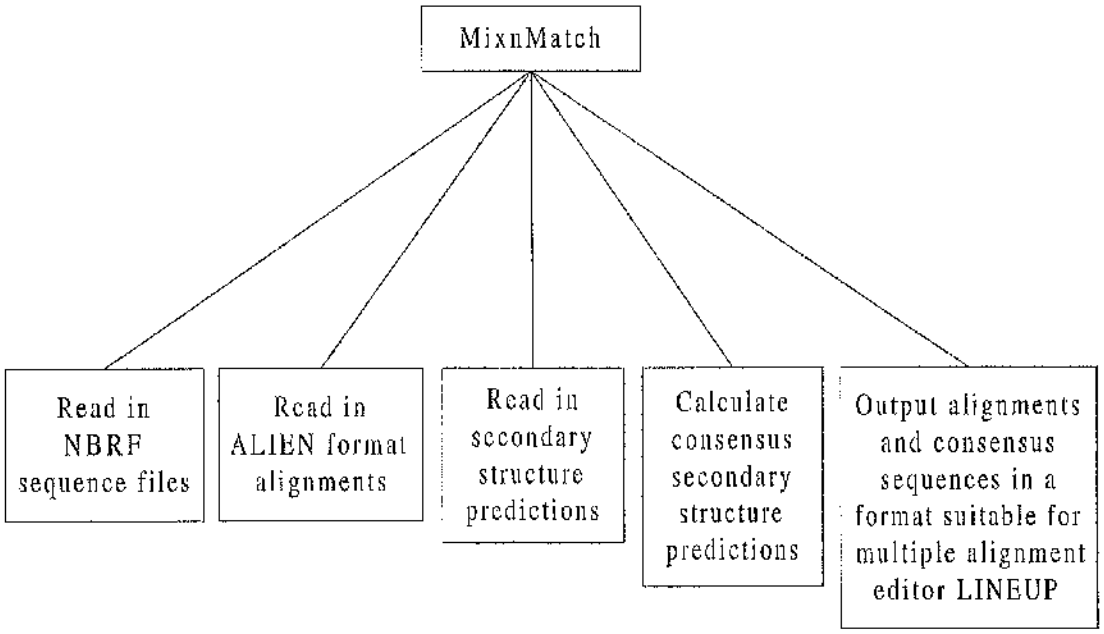


Figure 5.7

A structure diagram of the initial program developed for the Mix'n'Match algorithm.

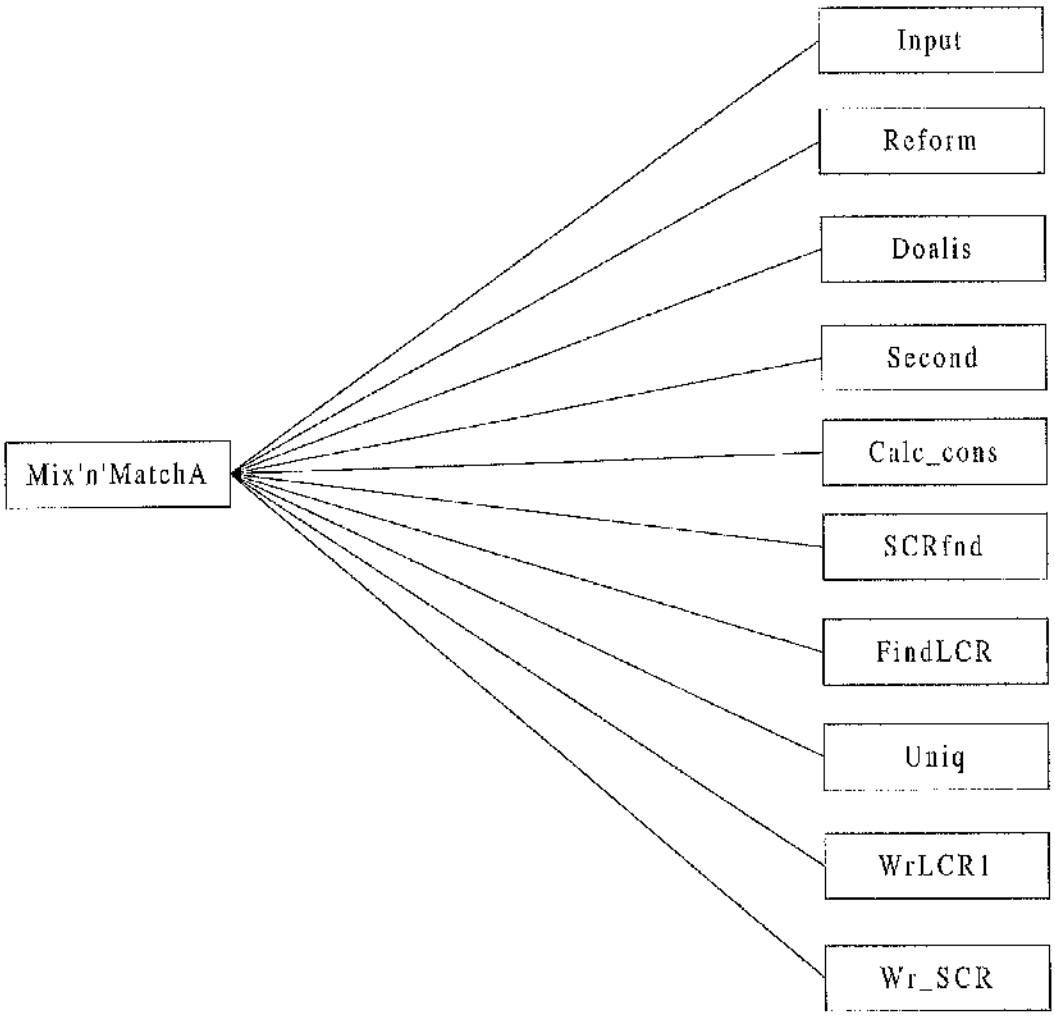


Figure 5.5

A structure diagram of the Mix'n'MatchA program.

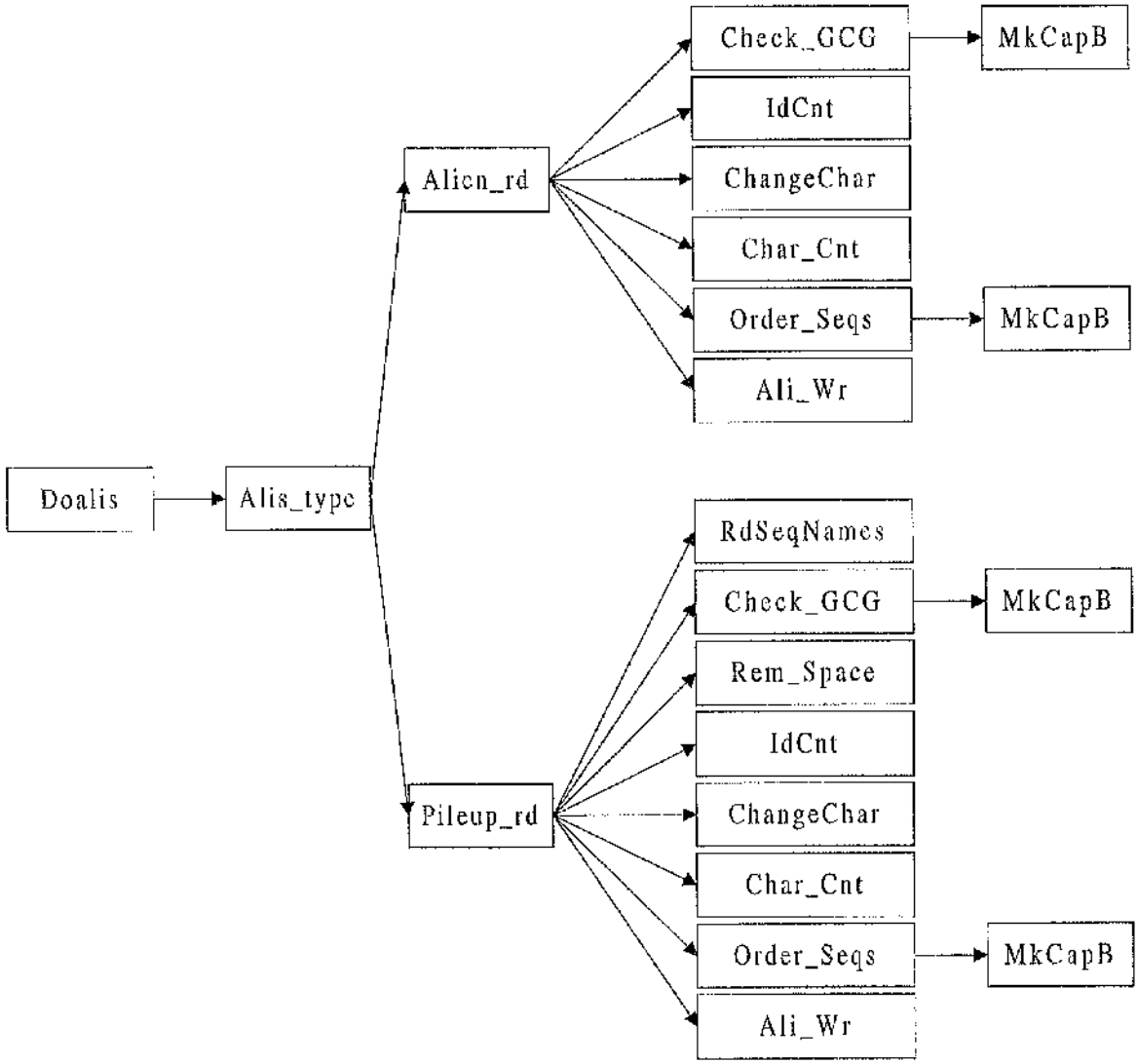


Figure 5.8

A structure diagram of the doalis module from the Mix'n'MatchA program.

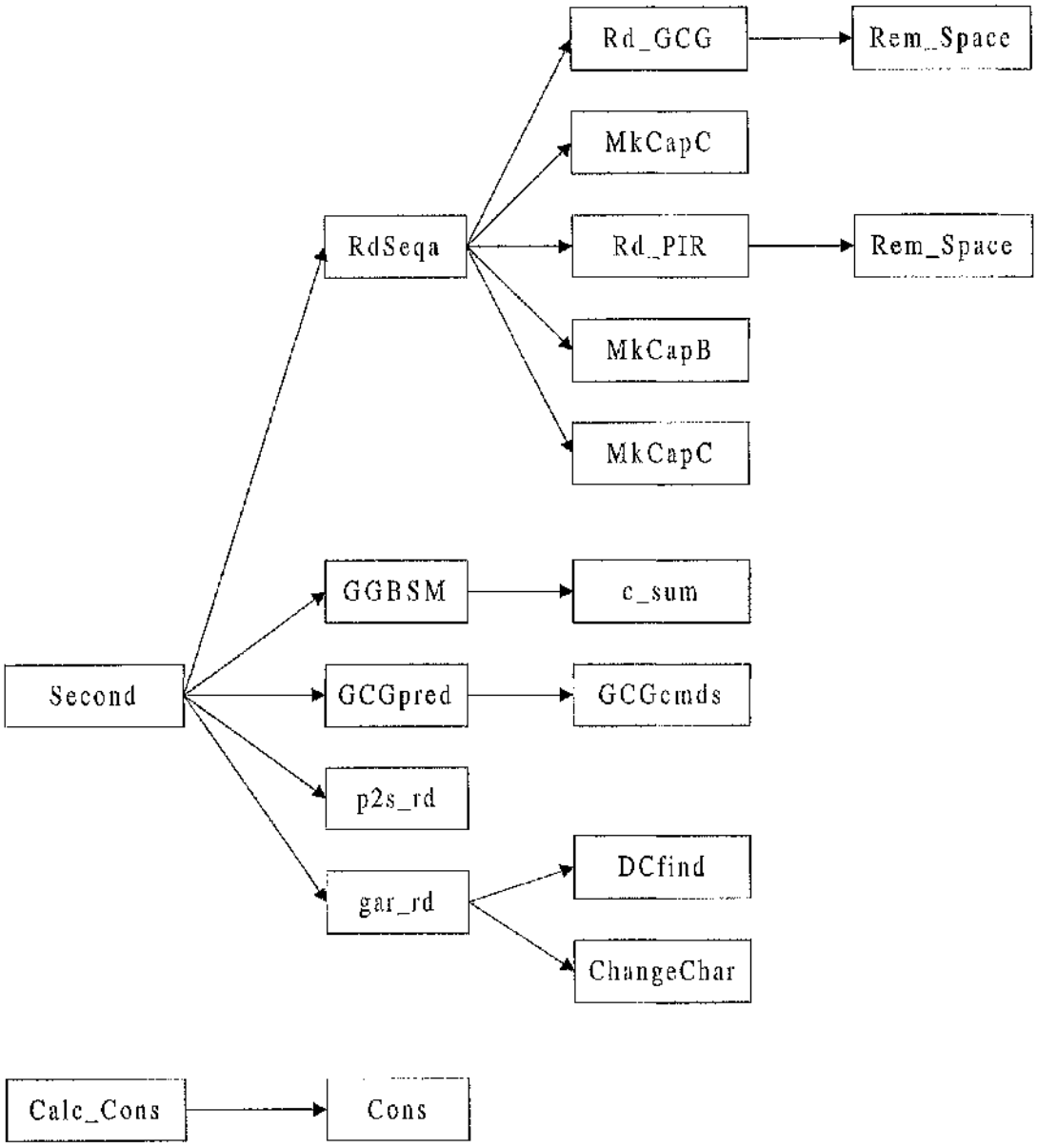


Figure 5.9

A structure diagram of the second and calc_cons modules.

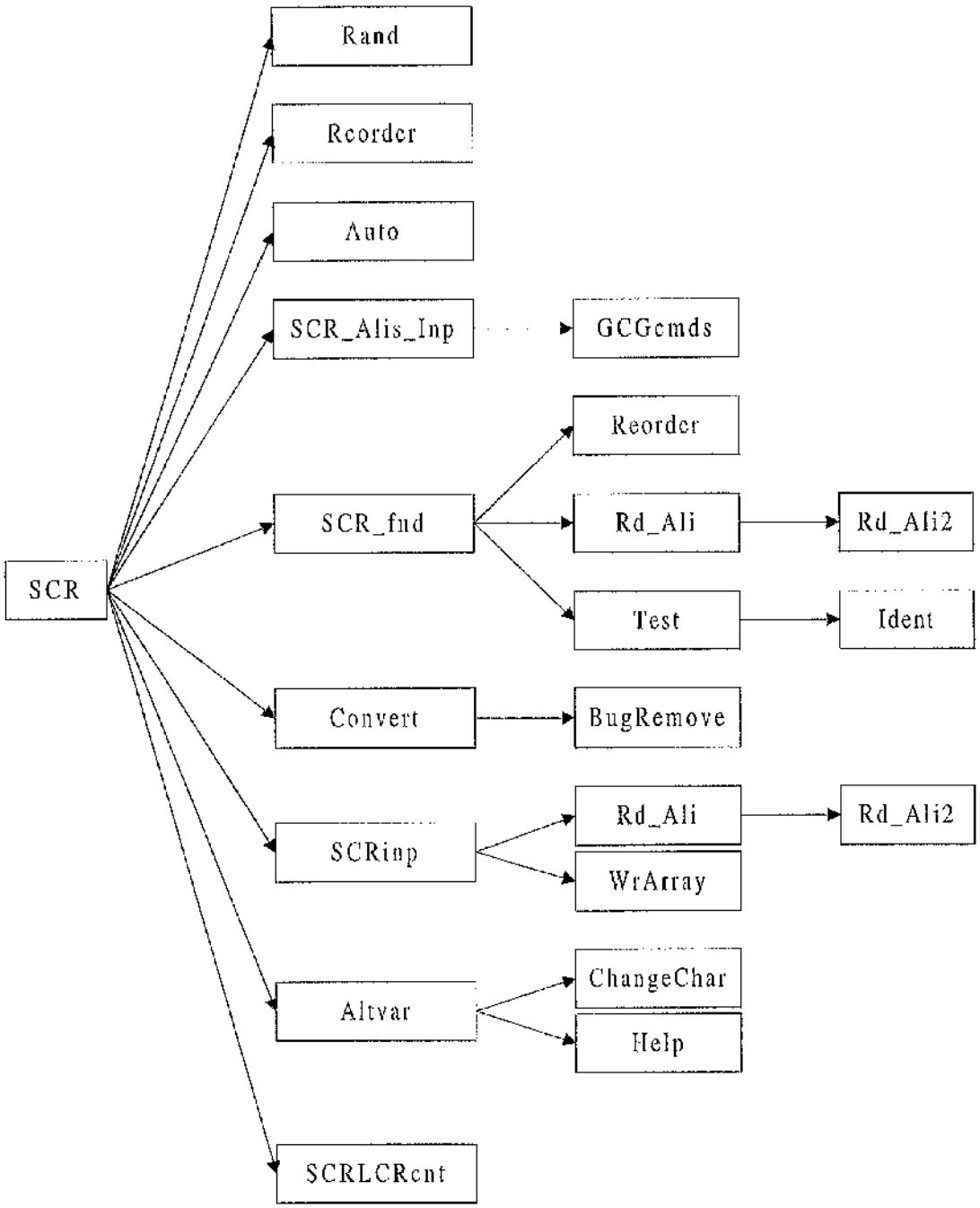


Figure 5.10

A structure diagram of the scr module.

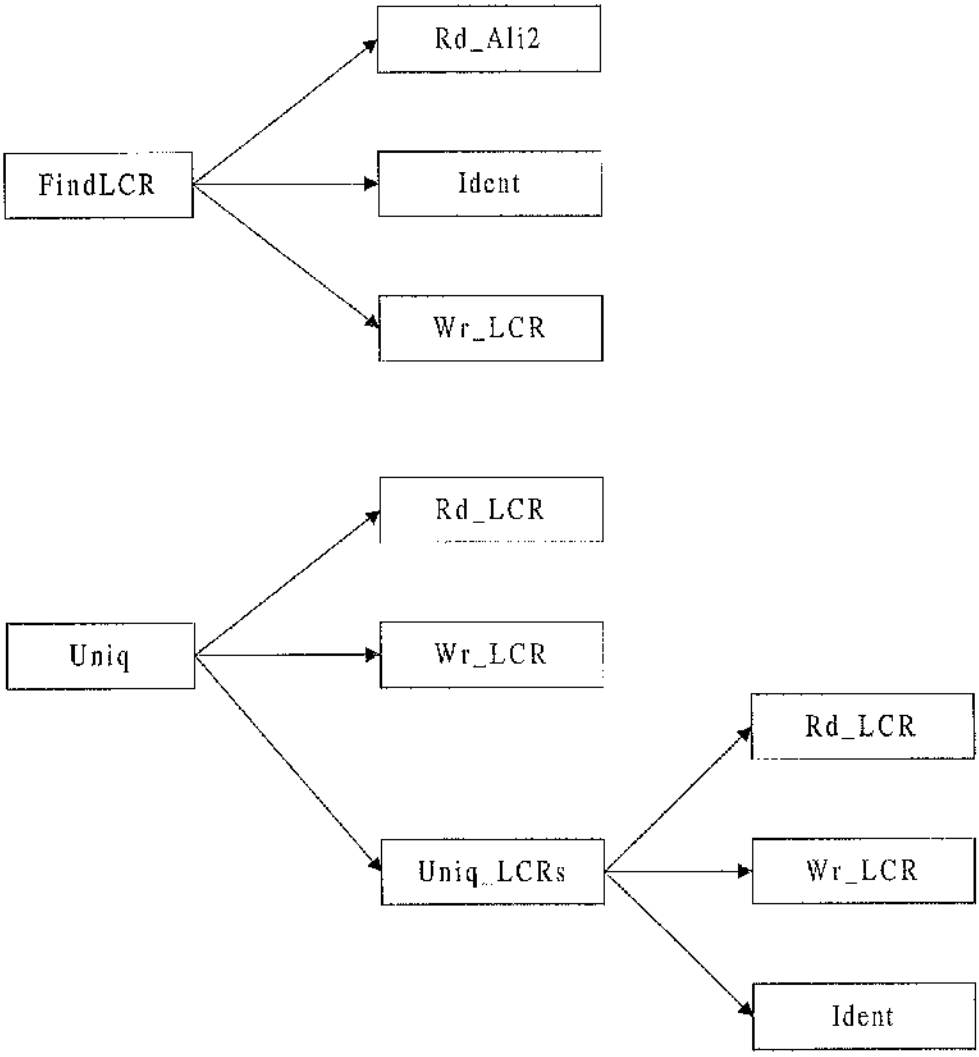


Figure 5.11

A structure diagram of the FindLCR amd Uniq modules.

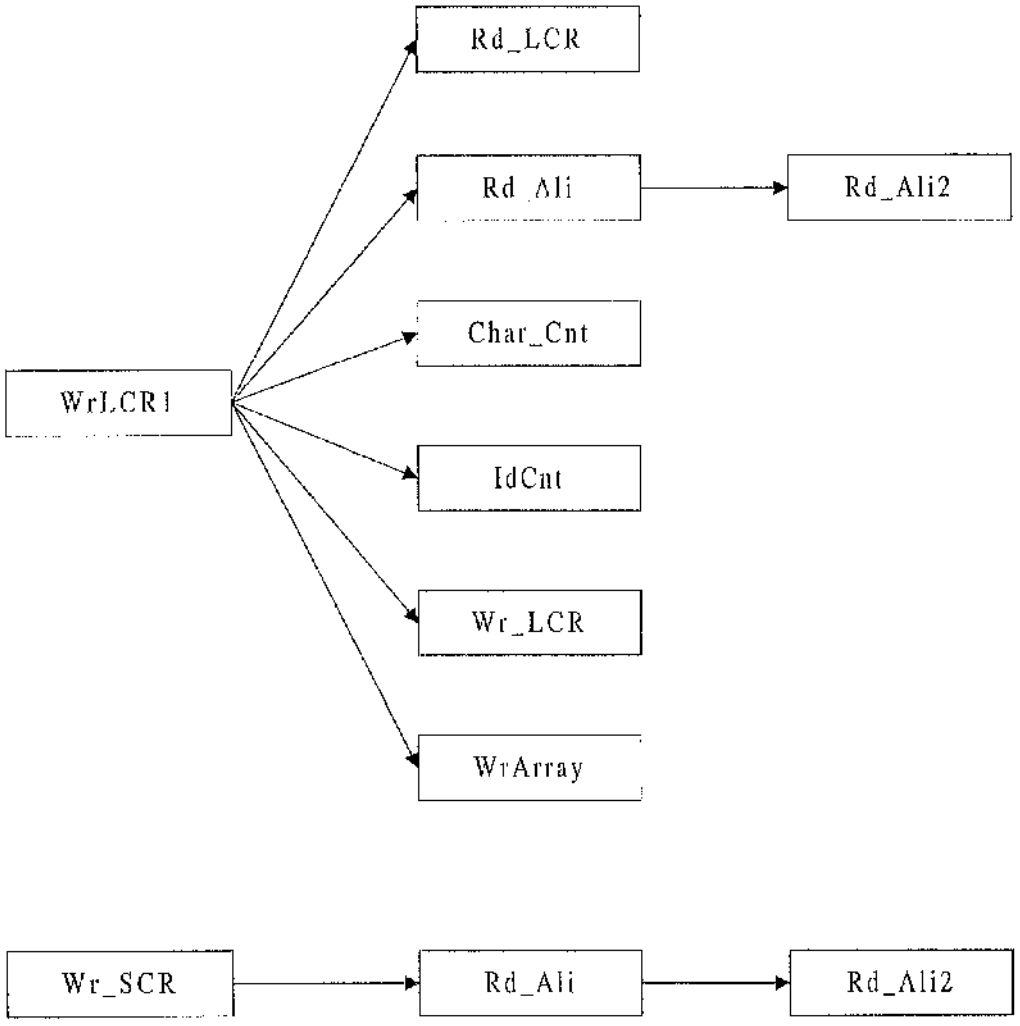


Figure 5.12

A structure diagram of the WrLCR1 and Wr_SCR modules.

Chapter 6 Mix'n'Match results

6.1 Control results for the Mix'n'Match method

The initial study (see chapter four) demonstrated that SCRs are found using the Mix'n'Match method. A more extensive test of the Mix'n'Match method is now undertaken. The best way to test the accuracy of a multiple alignment method is to compare the results produced to reference alignments obtained by three dimensional superimposition. The reference alignment of a protein family identifies the structurally equivalent residues that define the common tertiary fold. However, not every structurally equivalent residue is useful in testing accuracy. This is because, in structural terms, loops do not form part of the conserved core of a protein. Hence, only residues that are part of α-helices and β-strands are used as test regions.

Mix'n'Match is tested with two protein families, the globins and the serine proteases. These families have sequences with known three dimensional structures that have been used to generate reference alignments. For the globin family, seven test regions, as used by Barton and Sternberg [254] and taken from the reference alignment of Lesk and Chothia [58], are used. These regions correspond to conserved α-helices A, B, C, E, F, G and H. The seven globin sequences used are listed in table 6.1. For the serine protease family of sequences, twenty three test regions, identified by Greer [48, 49], are used. These regions correspond to structurally equivalent residues that form β-strands and an α-helix. The eleven serine protease sequences used are listed in table 6.2. The choice of these families allows Mix'n'Match alignments to be compared against both the reference alignments and also against the alignments of these sequences produced by other multiple alignment programs. The effectiveness of a Mix'n'Match alignment is estimated according to whether the alignments at the test regions are correct; to qualify, every residue has to be correctly aligned, with no exceptions.

Code	Protein	Species	Accession Number
GGLMS	Cyanohaemoglobin	Sea lamprey	P02208
HAHO	Haemoglobin α-chain	Horse	P01958
HAHU	Haemoglobin α-chain	Human	P01922
HBHO	Haemoglobin β-chain	Horse	P02062
HBHU	Haemoglobin β-chain	Human	P02023
LGHB	Leghaemoglobin	Lupinus luteus (root nodule)	A50249
MYWHP	Myoglobin	Sperm whale	P02185

Table 6.1

Listing the seven globin sequences used in aligning. The sequences are listed with their acession numbers.

Code	Protein	Species	Accession Number
EXBO	Factor Xa	Bovine	P00743
EZEC58	Mast Cell Protease	Rat	P00770
HPHB	Haptoglobin	Chimpanzee	M20760
KFBO	Factor IX	Bovine	P00741
PLHU	Plasminogen	Human	P00747
TBBO	Prothrombin	Bovine	J00041
N1CHG*	Chymotrypsin	Bovine	
N1SGT*	Trypsin	*Streptomyces griseus*	
N3EST*	Elastase	Porcine	
N2PKA*	Kallikrein	Porcine	
N1NTP*	Trypsin	Bovine	

Table 6.2

Listing the serine protease sequences using in aligning. The sequences are listed with their acession numbers. The ones marked with an asterix '*' are taken from the Brookhaven structural database and are listed using their database entry name.

As previously noted, the results from all multiple alignment programs depend to some extent on the values used for the program's parameters. The values that can be changed in Mix'n'Match are those used in choosing the SCRs. The effect of varying these values is investigated below. The alignments produced by Mix'n'Match are also dependent on the criteria used in the choice of 'best' alignment for each LCR. Although experienced users could use complex criteria for this choice, to mimic the results possible for non-experienced users, the simple criterion of choosing the alignment that has the fewest number of gaps in predicted secondary structural units is employed. In practice, it is recommended that structure prediction information is used only as a guide, and not as the sole criterion of choice, due to the inaccuracy inherent in any secondary structure prediction method.

6.2 Effects of varying internal Mix'n'Match parameters

Three parameters are used by Mix'n'Match when delineating SCRs: the minimum length of SCR that can be found; the length of window used when comparing two alignments; and the number of sequences used when comparing alignments. The eleven members of the serine protease family used in the reference alignment of Greer [48] are aligned (table 6.2) using the gap penalties, scoring matrices and multiple alignment programs described above for the shikimate pathway enzymes (see chapter 3.4 and tables 3.2 and 3.3), generating sixty five alignments.

To test the effect the minimum length of SCR has on the SCRs found, the Mix'n'Match programs are run on the serine protease and globin alignments while varying the minimum length of the SCRs from two to nine residues and two to six residues, respectively. The detection of SCRs is tested using between two and sixty five alignments (table 6.3 and figure 6.1). The expected result is that as the minimum width of SCR increases, smaller SCRs will be ignored, resulting in a reduction in the overall length of SCRs found. The data collected is in broad agreement with this.

Figure 6.1

Illustrating the effect that varying the minimum width a Strongly Conserved Region (SRC) can be, on the percentage of alignment found in SCRs as the number of alignments, used by the Mix'n'Match algorithm to find SCRs, increases. Sixty five initial alignments are input to Mix'n'Match, generated using varying gap penalties (shown in table 3.2), scoring matrices (described in table 3.3) and multiple alignment programs as described in the text. The alignments are of eleven serine proteases, as listed in table 6.2. The alignments are ranked according to the number of identical residues, highest to lowest. The alignments are consecutively added together one at a time and the Mix'n' Match algorithm applied. All Mix'n'Match parameters are set to their defaults except the minimum width a SCR can be, which is varied from two to nine residues.

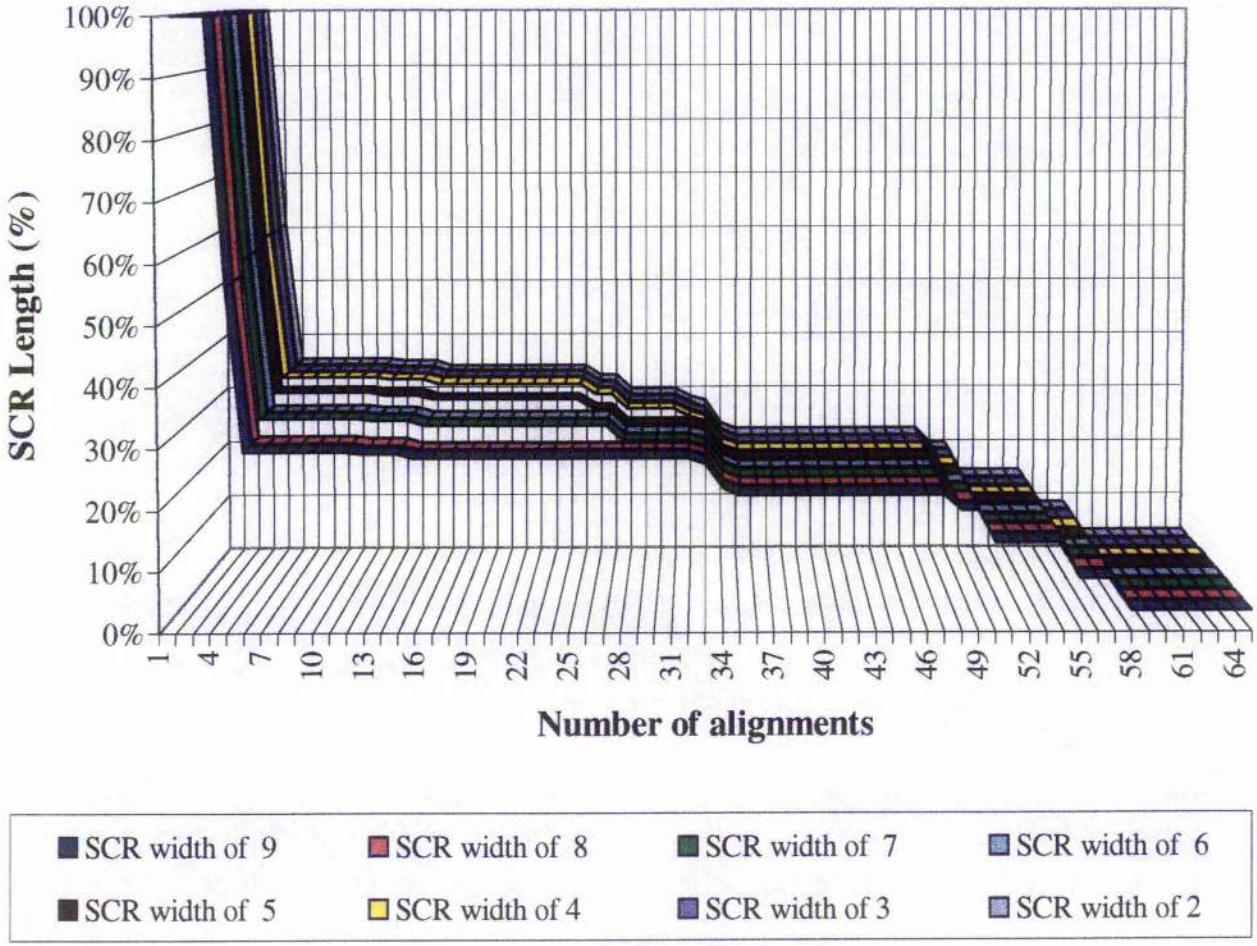

SCR Length (%)
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1
4
7
10
13
16
19
22
25
28
31
34
37
40
43
46
49
52
55
58
61
64
Number of alignments
SCR width of 9
SCR width of 8
SCR width of 7
SCR width of 6
SCR width of 5
SCR width of 4
SCR width of 3
SCR width of 2

Table 6.3

Showing the effect of varying the minimum width of SCRs on the length of SCRs found. The figure given is the total number of residues that occupy SCRs in the whole protein, i.e., at the top all or most residues are defined as SCRs whereas at the bottom, only a few or no residues are. Results are for (**A**) eleven serine proteases, as listed in table 6.2, and (**B**) seven globins, as listed in table 6.1. The number of alignments used to calculate the SCRs, range from two to sixty five alignments. Default settings are used for other parameters. The default parameter for minimum width of SCR is three residues, shown shaded above.

A) Effect of varying the minimum width of SCRs on the length of SCRs found for serine proteases.

No of Alignments	SCR width of 2		SCR width of 3		SCR width of 4		SCR width of 5		SCR width of 6		SCR width of 7		SCR width of 8		SCR width of 9	
	SCR length	SCR length (%)	SCR length	SCR length (%)	SCR length	SCR length (%)	SCR length	SCR length (%)	SCR length	SCR length (%)	SCR length	SCR length (%)	SCR length	SCR length (%)	SCR length	SCR length (%)
2	268	100	268	100	268	100	268	100	268	100	268	100	268	100	268	100
3	268	100	268	100	268	100	268	100	268	100	268	100	268	100	268	100
4	268	100	268	100	268	100	268	100	268	100	268	100	268	100	268	100
5	162	60	162	60	162	60	158	59	158	59	158	59	151	56	151	56
6	95	35	95	35	95	35	91	34	86	32	86	32	79	29	79	29
7	95	35	95	35	95	35	91	34	86	32	86	32	79	29	79	29
8	95	35	95	35	95	35	91	34	86	32	86	32	79	29	79	29
9	95	35	95	35	95	35	91	34	86	32	86	32	79	29	79	29
10	95	35	95	35	95	35	91	34	86	32	86	32	79	29	79	29
11	95	35	95	35	95	35	91	34	86	32	86	32	79	29	79	29
12	95	35	95	35	95	35	91	34	86	32	86	32	79	29	79	29
13	94	35	94	35	94	35	90	34	85	32	85	32	78	29	78	29
14	94	35	94	35	94	35	90	34	85	32	85	32	78	29	78	29
15	94	35	94	35	94	35	90	34	85	32	85	32	78	29	78	29
16	92	34	92	34	92	34	88	33	83	31	83	31	76	28	76	28
17	92	34	92	34	92	34	88	33	83	31	83	31	76	28	76	28
18	92	34	92	34	92	34	88	33	83	31	83	31	76	28	76	28
19	92	34	92	34	92	34	88	33	83	31	83	31	76	28	76	28
20	92	34	92	34	92	34	88	33	83	31	83	31	76	28	76	28
21	92	34	92	34	92	34	88	33	83	31	83	31	76	28	76	28
22	92	34	92	34	92	34	88	33	83	31	83	31	76	28	76	28
23	92	34	92	34	92	34	88	33	83	31	83	31	76	28	76	28
24	92	34	92	34	92	34	88	33	83	31	83	31	76	28	76	28
25	92	34	92	34	92	34	88	33	83	31	83	31	76	28	76	28
26	87	32	87	32	87	32	83	31	83	31	83	31	76	28	76	28
27	87	32	87	32	87	32	83	31	83	31	83	31	76	28	76	28
28	80	30	80	30	80	30	76	28	76	28	76	28	76	28	76	28
29	80	30	80	30	80	30	76	28	76	28	76	28	76	28	76	28
30	80	30	80	30	80	30	76	28	76	28	76	28	76	28	76	28
31	80	30	80	30	80	30	76	28	76	28	76	28	76	28	76	28
32	76	28	76	28	76	28	76	28	76	28	76	28	76	28	76	28
33	74	28	74	28	74	28	74	28	74	28	74	28	74	28	74	28
34	63	24	63	24	63	24	63	24	63	24	63	24	63	24	63	24
35	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22
36	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22
37	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22
38	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22
39	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22
40	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22
41	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22
42	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22
43	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22
44	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22
45	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22
46	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22
47	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22
48	53	20	53	20	53	20	53	20	53	20	53	20	53	20	53	20
49	53	20	53	20	53	20	53	20	53	20	53	20	53	20	53	20
50	39	15	39	15	39	15	39	15	39	15	39	15	39	15	39	15
51	39	15	39	15	39	15	39	15	39	15	39	15	39	15	39	15
52	39	15	39	15	39	15	39	15	39	15	39	15	39	15	39	15
53	39	15	39	15	39	15	39	15	39	15	39	15	39	15	39	15
54	39	15	39	15	39	15	39	15	39	15	39	15	39	15	39	15
55	23	9	23	9	23	9	23	9	23	9	23	9	23	9	23	9
56	23	9	23	9	23	9	23	9	23	9	23	9	23	9	23	9
57	23	9	23	9	23	9	23	9	23	9	23	9	23	9	23	9
58	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3
59	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3
60	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3
61	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3
62	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3
63	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3
64	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3
65	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3

B) Effect of varying the minimum width of SCRs on the length of SCRs found for globins.

No of Alignments	SCR width of 2		SCR width of 3		SCR width of 4		SCR width of 5		SCR width of 6	
	SCR length	SCR length (%)	SCR length	SCR length (%)	SCR length	SCR length (%)	SCR length	SCR length (%)	SCR length	SCR length (%)
2	156	93	156	93	156	93	156	93	156	93
3	156	93	156	93	156	93	156	93	156	93
4	156	93	156	93	156	93	156	93	156	93
5	156	93	156	93	156	93	156	93	156	93
6	156	93	156	93	156	93	156	93	156	93
7	156	93	156	93	156	93	156	93	156	93
8	156	93	156	93	156	93	156	93	156	93
9	119	71	119	71	119	71	119	71	119	71
10	95	57	95	57	95	57	95	57	95	57
11	95	57	95	57	95	57	95	57	95	57
12	95	57	95	57	95	57	95	57	95	57
13	95	57	95	57	95	57	95	57	95	57
14	95	57	95	57	95	57	95	57	95	57
15	95	57	95	57	95	57	95	57	95	57
16	80	48	80	48	80	48	80	48	80	48
17	80	48	80	48	80	48	80	48	80	48
18	80	48	80	48	80	48	80	48	80	48
19	80	48	80	48	80	48	80	48	80	48
20	80	48	80	48	80	48	80	48	80	48
21	80	48	80	48	80	48	80	48	80	48
22	80	48	80	48	80	48	80	48	80	48
23	80	48	80	48	80	48	80	48	80	48
24	80	48	80	48	80	48	80	48	80	48
25	63	38	63	38	63	38	63	38	63	38
26	63	38	63	38	63	38	63	38	63	38
27	39	23	39	23	39	23	39	23	39	23
28	39	23	39	23	39	23	39	23	39	23
29	39	23	39	23	39	23	39	23	39	23
30	39	23	39	23	39	23	39	23	39	23
31	39	23	39	23	39	23	39	23	39	23
32	39	23	39	23	39	23	39	23	39	23
33	39	23	39	23	39	23	39	23	39	23
34	39	23	39	23	39	23	39	23	39	23
35	39	23	39	23	39	23	39	23	39	23
36	39	23	39	23	39	23	39	23	39	23
37	39	23	39	23	39	23	39	23	39	23
38	39	23	39	23	39	23	39	23	39	23
39	39	23	39	23	39	23	39	23	39	23
40	39	23	39	23	39	23	39	23	39	23
41	2	1	0	0	0	0	0	0	0	0
42	2	1	0	0	0	0	0	0	0	0
43	2	1	0	0	0	0	0	0	0	0
44	2	1	0	0	0	0	0	0	0	0
45	0	0	0	0	0	0	0	0	0	0

Results found for the serine protease alignments (table 6.3A and figure 6.1), show that when up to four alignments are considered, all sizes of SCR width define one hundred percent of the alignment as a SCR. This indicates that the first four alignments are identical. Four unique sets of results are found for the eight different minimum widths used, with identical results found when the widths are: two, three and four; five; six and seven; and eight and nine. As the width increases between these four sets, the overall length of SCR found does decrease. However, the differences are apparent only when between five and thirty one alignments are considered. The same SCRs are found when between thirty two and sixty five alignments are considered, for all sizes of SCR width. This indicates that as the number of alignments considered increases, the prevalence of short SCRs decreases, until after thirty two alignments are considered, only SCRs of length nine or longer are left. The differences between the four sets of results are small, with three cumulative differences being found: the exclusion of a four mer SCR found in alignment thirty one differentiates the SCR widths of two, three and four from five; the exclusion of a five mer SCR found in alignment twenty five differentiates the SCR widths of five from six and seven; and the exclusion of a seven mer SCR found in alignment twenty seven differentiates the SCR widths of six and seven from eight and nine.

Results for the globin alignments (table 6.3B) show similar trends. Two unique sets of results are found for the five different minimum widths used. The same SCRs are found when between two and forty alignments are considered, for all sizes of SCR width. When forty one or more alignments are considered, no SCRs are found for minimum widths of three, four, five, or six. When the width is two, a single SCR of width two is found when between forty one and forty four alignments are considered.

The algorithm used to detect SCRs necessitates the use of a minimum SCR width. The minimum SCR width should be as short as possible so that short SCRs are detected, however, the likelihood of finding a SCR in two position, invalidating the SCR, is increased with small width. In the above runs, no SCRs are found in two positions. For

the serine proteases, widths of two, three or four fit the above criteria and for the globins, widths of two to six. A minimum SCR width of two could be chosen, but, to reduce the possibility of finding a SCR in two positions, the default minimum width is set to three residues.

To test the effect of varying the length of window used in the 'ident' module, used to compare alignments, the Mix'n'Match programs are run on the serine protease alignments while varying the window length from ten to eighty residues in steps of ten. Window sizes of twenty five and thirty five are also tested. The detection of SCRs is tested using between two and sixty five alignments (table 6.4 and figure 6.2).

A window size is used when comparing alignments to reduce the amount of time spent carrying out the comparisons. It may be expected that if the window size is too short, a SCR in one alignment would not be found in another if the numbering of the alignments varied too much. Additionally, if the window size is too long, the possibility of the SCR being found in more than one position may be increased. The results show that the same SCRs are found for all the window sizes tested and no duplicate SCRs are found. It is reasonable that any value from ten to eighty could be chosen as default with the provision that the shorter the value chosen, the faster a comparison between two alignments may be carried out. However, to decrease the possibility of either of the two error conditions pointed out above occurring, a midway value of thirty is chosen as the default window size.

To test the effect of varying the number of sequences used while finding SCRs, the Mix'n'Match programs are run on the alignments of the eleven serine protease sequences while varying the number of sequences used from six to eleven. The detection of SCRs is tested using between two and sixty five alignments (table 6.5 and figure 6.3).

The user is given the ability to change the number of sequences in an alignment that are tested. Once set, the same randomly picked sequences are used for all delineation of

Figure 6.2

Illustrating the effect that varying the window size used in the 'ident' module for comparing two alignments, has on the percentage of alignment found in Strongly Conserved Regions (SCRs) as the number of alignments, used by the Mix'n'Match algorithm to find SCRs, increases. Sixty five initial alignments are input to Mix'n'Match, generated using varying gap penalties (shown in table 3.2), scoring matrices (described in table 3.3) and multiple alignment programs as described in the text. The alignments are of eleven serine proteases, as listed in table 6.2. The alignments are ranked according to the number of identical residues, highest to lowest. The alignments are consecutively added together one at a time and the Mix'n' Match algorithm applied. All Mix'n'Match parameters are set to their defaults except the window size used in the 'ident' module for comparing two alignments which is varied between ten and eighty residues in steps of ten. For clarity, only four sets of results are shown.

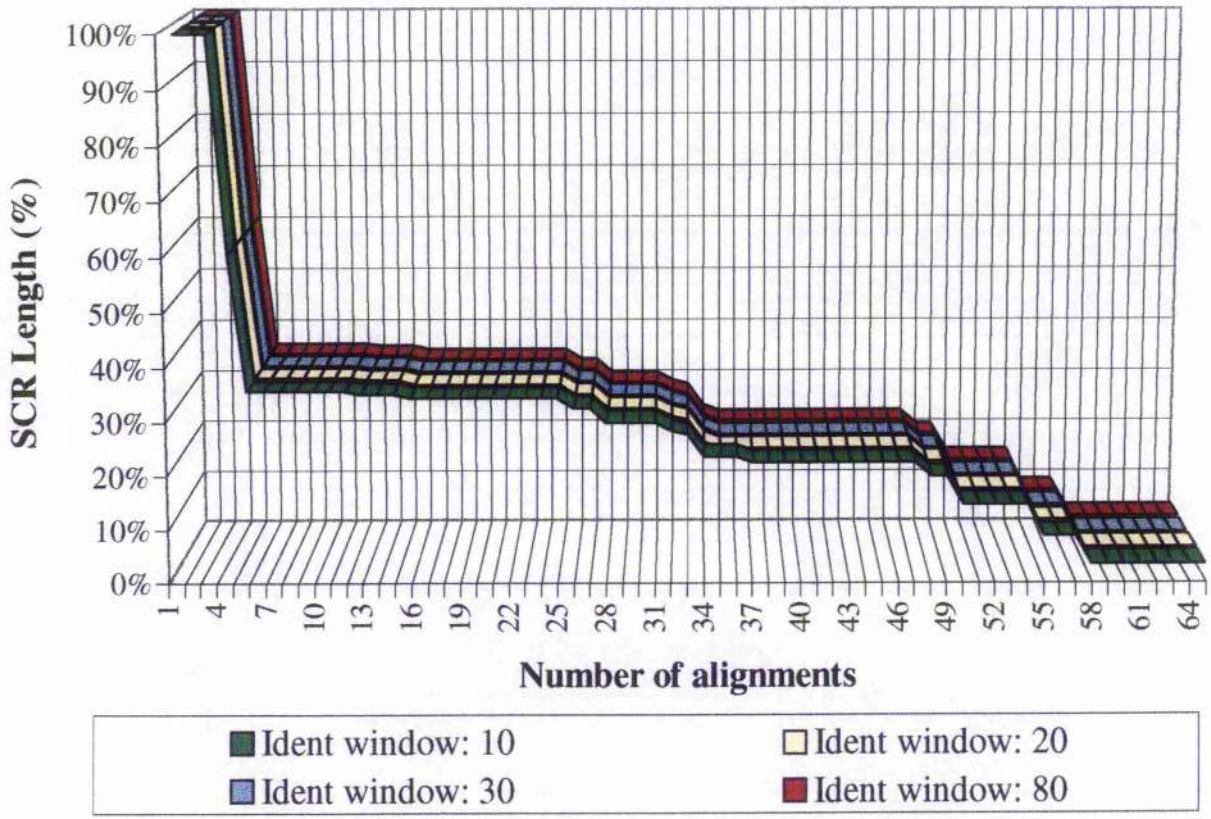

SCR Length (%)
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1
4
7
10
13
16
19
22
25
28
31
34
37
40
43
46
49
52
55
58
61
64
Number of alignments
Ident window: 10
Ident window: 20
Ident window: 30
Ident window: 80

Table 6.4

Showing the effect of varying the window size used in the 'ident' module for comparing two alignments with each other. The figure given is the total number of residues that occupy SCRs in the whole protein, i.e., at the top all or most residues are defined as SCRs whereas at the bottom, only a few or no residues are. The alignments are of eleven serine proteases, as listed in table 6.2. The number of alignments used to calculate the SCRs, range from two to sixty five alignments. Default settings are used for the other parameters. The default parameter for window size is plus or minus thirty residues, shown shaded above.

No of Alignments	Window size: 10		Window size: 20		Window size: 25		Window size: 30		Window size: 35		Window size: 40		Window size: 50		Window size: 60		Window size: 70		Window size: 80	
	SCR length	SCR length (%)	SCR length	SCR length (%)	SCR length	SCR length (%)	SCR length	SCR length (%)	SCR length	SCR length (%)	SCR length	SCR length (%)	SCR length	SCR length (%)	SCR length	SCR length (%)	SCR length	SCR length (%)	SCR length	SCR length (%)
2	268	100	268	100	268	100	268	100	268	100	268	100	268	100	268	100	268	100	268	100
3	268	100	268	100	268	100	268	100	268	100	268	100	268	100	268	100	268	100	268	100
4	268	100	268	100	268	100	268	100	268	100	268	100	268	100	268	100	268	100	268	100
5	162	60	162	60	162	60	162	60	162	60	162	60	162	60	162	60	162	60	162	60
6	95	35	95	35	95	35	95	35	95	35	95	35	95	35	95	35	95	35	95	35
7	95	35	95	35	95	35	95	35	95	35	95	35	95	35	95	35	95	35	95	35
8	95	35	95	35	95	35	95	35	95	35	95	35	95	35	95	35	95	35	95	35
9	95	35	95	35	95	35	95	35	95	35	95	35	95	35	95	35	95	35	95	35
10	95	35	95	35	95	35	95	35	95	35	95	35	95	35	95	35	95	35	95	35
11	95	35	95	35	95	35	95	35	95	35	95	35	95	35	95	35	95	35	95	35
12	95	35	95	35	95	35	95	35	95	35	95	35	95	35	95	35	95	35	95	35
13	94	35	94	35	94	35	94	35	94	35	94	35	94	35	94	35	94	35	94	35
14	94	35	94	35	94	35	94	35	94	35	94	35	94	35	94	35	94	35	94	35
15	94	35	94	35	94	35	94	35	94	35	94	35	94	35	94	35	94	35	94	35
16	92	34	92	34	92	34	92	34	92	34	92	34	92	34	92	34	92	34	92	34
17	92	34	92	34	92	34	92	34	92	34	92	34	92	34	92	34	92	34	92	34
18	92	34	92	34	92	34	92	34	92	34	92	34	92	34	92	34	92	34	92	34
19	92	34	92	34	92	34	92	34	92	34	92	34	92	34	92	34	92	34	92	34
20	92	34	92	34	92	34	92	34	92	34	92	34	92	34	92	34	92	34	92	34
21	92	34	92	34	92	34	92	34	92	34	92	34	92	34	92	34	92	34	92	34
22	92	34	92	34	92	34	92	34	92	34	92	34	92	34	92	34	92	34	92	34
23	92	34	92	34	92	34	92	34	92	34	92	34	92	34	92	34	92	34	92	34
24	92	34	92	34	92	34	92	34	92	34	92	34	92	34	92	34	92	34	92	34
25	92	34	92	34	92	34	92	34	92	34	92	34	92	34	92	34	92	34	92	34
26	87	32	87	32	87	32	87	32	87	32	87	32	87	32	87	32	87	32	87	32
27	87	32	87	32	87	32	87	32	87	32	87	32	87	32	87	32	87	32	87	32
28	80	30	80	30	80	30	80	30	80	30	80	30	80	30	80	30	80	30	80	30
29	80	30	80	30	80	30	80	30	80	30	80	30	80	30	80	30	80	30	80	30
30	80	30	80	30	80	30	80	30	80	30	80	30	80	30	80	30	80	30	80	30
31	80	30	80	30	80	30	80	30	80	30	80	30	80	30	80	30	80	30	80	30
32	76	28	76	28	76	28	76	28	76	28	76	28	76	28	76	28	76	28	76	28
33	74	28	74	28	74	28	74	28	74	28	74	28	74	28	74	28	74	28	74	28
34	63	24	63	24	63	24	63	24	63	24	63	24	63	24	63	24	63	24	63	24
35	63	24	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22
36	63	24	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22
37	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22
38	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22
39	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22
40	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22
41	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22
42	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22
43	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22
44	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22
45	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22
46	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22
47	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22	60	22
48	53	20	53	20	53	20	53	20	53	20	53	20	53	20	53	20	53	20	53	20
49	53	20	53	20	53	20	53	20	53	20	53	20	53	20	53	20	53	20	53	20
50	39	15	39	15	39	15	39	15	39	15	39	15	39	15	39	15	39	15	39	15
51	39	15	39	15	39	15	39	15	39	15	39	15	39	15	39	15	39	15	39	15
52	39	15	39	15	39	15	39	15	39	15	39	15	39	15	39	15	39	15	39	15
53	39	15	39	15	39	15	39	15	39	15	39	15	39	15	39	15	39	15	39	15
54	39	15	39	15	39	15	39	15	39	15	39	15	39	15	39	15	39	15	39	15
55	23	9	23	9	23	9	23	9	23	9	23	9	23	9	23	9	23	9	23	9
56	23	9	23	9	23	9	23	9	23	9	23	9	23	9	23	9	23	9	23	9
57	23	9	23	9	23	9	23	9	23	9	23	9	23	9	23	9	23	9	23	9
58	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3
59	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3
60	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3
61	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3
62	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3
63	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3
64	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3
65	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3	9	3

SCRs and LCRs. This choice is introduced because it is considered possible that as increasing the number of alignments under consideration reduces the size of SCRs found, a similar effect on the size of SCRs may be apparent with increasing number of sequences. The expectation is that reducing the number of sequences looked at while calculating SCRs will lead to an increase in the overall length of SCRs found.

In all cases, the first four alignments are seen as one hundred percent SCR. The same SCRs are found when between eight and eleven sequences are used to calculate the SCRs. When seven sequences are used, the overall SCR length is seven mer longer when six and seven alignments are considered, one mer longer when eight to twenty four alignments are considered, and four mer longer when thirty two to thirty six alignments are considered. Seven sequences give the same results as using between eight and eleven sequences when considering the other alignments i.e., between twenty six and thirty one and between thirty seven and sixty five. When six sequences are used, the overall length of SCRs found ranges from between seven to thirty three mer longer than those found when between eight and eleven sequences are used although the same results as all other cases are found when between sixty three and sixty five alignments are considered.

This increase in length of SCR found when six and seven sequences are used is as expected. However, the SCRs found are 'false' SCRs which do not exist when all sequences are used. Where possible, the number of sequences used should be the same as the number of sequences in the alignments. For the purposes of setting this parameter, a default value is ten is used.

The effect the module 'bugremove' has on the SCRs found is shown in table 6.6 and figure 6.4. This module is used to remove SCRs where one or more sequences in a putative SCR are composed entirely of gaps. The detection of SCRs by the Mix'n'Match programs is tested using between two and sixty five serine protease alignments, with and without this module. When five alignments are considered, compared to the data generated with the module, the absence of 'bugremove' finds two extra SCRs of total

Figure 6.3

Illustrating the effect that varying the number of sequences in the alignments that are used in calculating the Strongly Conserved Regions (SCRs), has on the percentage of alignment found in SCRs as the number of alignments, used by the Mix'n'Match algorithm to find SCRs, increases. Sixty five initial alignments are input to Mix'n'Match, generated using varying gap penalties (shown in table 3.2), scoring matrices (described in table 3.3) and multiple alignment programs as described in the text. The alignments are of eleven serine proteases, as listed in table 6.2. The alignments are ranked according to the number of identical residues, highest to lowest. The alignments are consecutively added together one at a time and the Mix'n' Match algorithm applied. All Mix'n'Match parameters are set to their defaults except for the number of sequences in the alignments that are used in calculating the SCRs, which is varied between six and eleven.

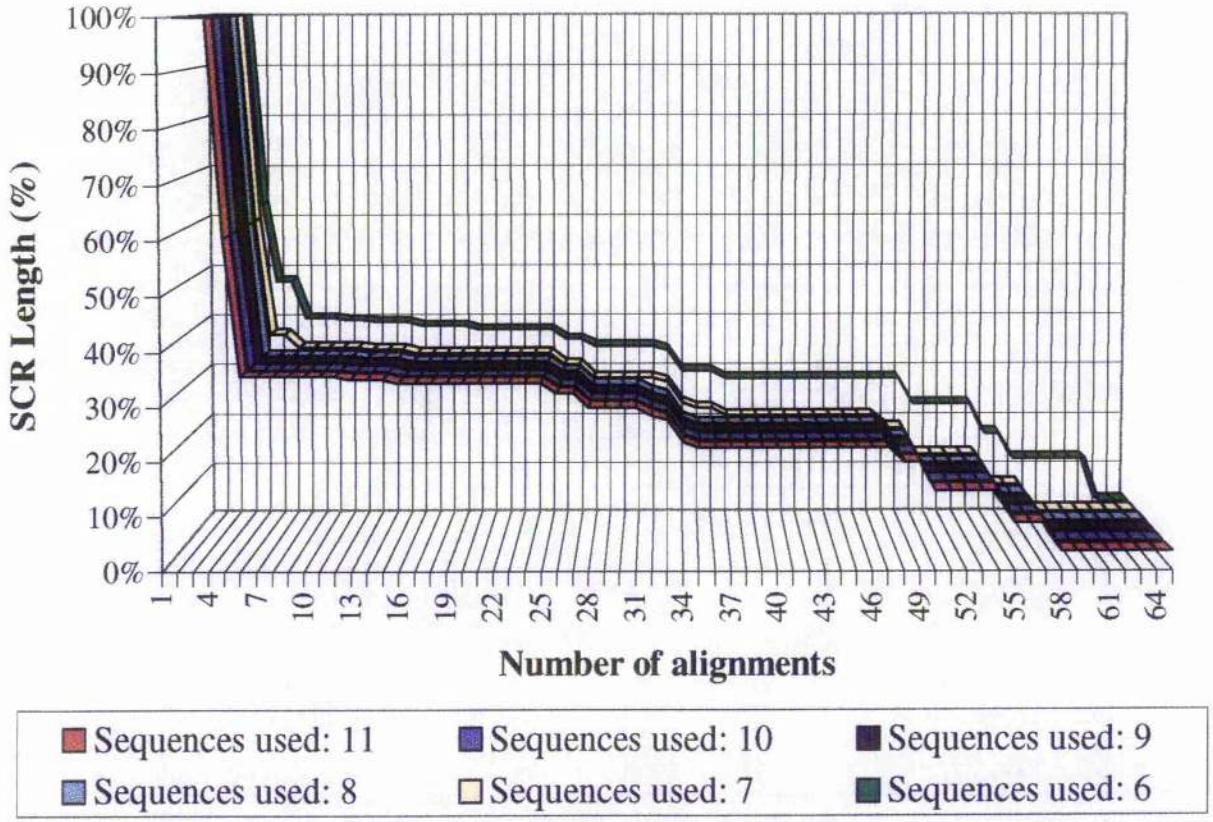

SCR Length (%)
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1
4
7
10
13
16
19
22
25
28
31
34
37
40
43
46
49
52
55
58
61
64
Number of alignments
Sequences used: 11
Sequences used: 10
Sequences used: 9
Sequences used: 8
Sequences used: 7
Sequences used: 6

Table 6.5

Showing the effect of varying the number of sequences to use when comparing two alignments with each other. The figure given is the total number of residues that occupy SCRs in the whole protein, i.e., at the top all or most residues are defined as SCRs whereas at the bottom, only a few or no residues are. The alignments are of eleven serine proteases, as listed in table 6.2. The number of alignments used to calculate the SCRs, range from two to sixty five alignments. Default settings are used for other parameters. The default parameter for number of sequences to use is ten, shown shaded above.

No of Alignments	Sequences used: 6		Sequences used: 7		Sequences used: 8		Sequences used: 9		Sequences used: 10		Sequences used: 11	
	SCR Length	SCR Length (%)	SCR Length	SCR Length (%)	SCR Length	SCR Length (%)	SCR Length	SCR Length (%)	SCR Length	SCR Length (%)	SCR Length	SCR Length (%)
2	268	100	268	100	268	100	268	100	268	100	268	100
3	268	100	268	100	268	100	268	100	268	100	268	100
4	268	100	268	100	268	100	268	100	268	100	268	100
5	169	63	162	60	162	60	162	60	162	60	162	60
6	128	48	103	38	95	35	95	35	95	35	95	35
7	128	48	103	38	95	35	95	35	95	35	95	35
8	108	40	96	36	95	35	95	35	95	35	95	35
9	108	40	96	36	95	35	95	35	95	35	95	35
10	108	40	96	36	95	35	95	35	95	35	95	35
11	107	40	96	36	95	35	95	35	95	35	95	35
12	107	40	96	36	95	35	95	35	95	35	95	35
13	106	40	95	35	94	35	94	35	94	35	94	35
14	106	40	95	35	94	35	94	35	94	35	94	35
15	106	40	95	35	94	35	94	35	94	35	94	35
16	104	39	93	35	92	34	92	34	92	34	92	34
17	104	39	93	35	92	34	92	34	92	34	92	34
18	104	39	93	35	92	34	92	34	92	34	92	34
19	104	39	93	35	92	34	92	34	92	34	92	34
20	102	38	93	35	92	34	92	34	92	34	92	34
21	102	38	93	35	92	34	92	34	92	34	92	34
22	102	38	93	35	92	34	92	34	92	34	92	34
23	102	38	93	35	92	34	92	34	92	34	92	34
24	102	38	93	35	92	34	92	34	92	34	92	34
25	102	38	93	35	92	34	92	34	92	34	92	34
26	97	36	87	32	87	32	87	32	87	32	87	32
27	97	36	87	32	87	32	87	32	87	32	87	32
28	93	35	80	30	80	30	80	30	80	30	80	30
29	93	35	80	30	80	30	80	30	80	30	80	30
30	93	35	80	30	80	30	80	30	80	30	80	30
31	93	35	80	30	80	30	80	30	80	30	80	30
32	93	35	80	30	76	28	76	28	76	28	76	28
33	91	34	78	29	74	28	74	28	74	28	74	28
34	80	30	67	25	63	24	63	24	63	24	63	24
35	80	30	64	24	60	22	60	22	60	22	60	22
36	80	30	64	24	60	22	60	22	60	22	60	22
37	76	28	60	22	60	22	60	22	60	22	60	22
38	76	28	60	22	60	22	60	22	60	22	60	22
39	76	28	60	22	60	22	60	22	60	22	60	22
40	76	28	60	22	60	22	60	22	60	22	60	22
41	76	28	60	22	60	22	60	22	60	22	60	22
42	76	28	60	22	60	22	60	22	60	22	60	22
43	76	28	60	22	60	22	60	22	60	22	60	22
44	76	28	60	22	60	22	60	22	60	22	60	22
45	76	28	60	22	60	22	60	22	60	22	60	22
46	76	28	60	22	60	22	60	22	60	22	60	22
47	76	28	60	22	60	22	60	22	60	22	60	22
48	76	28	53	20	53	20	53	20	53	20	53	20
49	76	28	53	20	53	20	53	20	53	20	53	20
50	62	23	39	15	39	15	39	15	39	15	39	15
51	62	23	39	15	39	15	39	15	39	15	39	15
52	62	23	39	15	39	15	39	15	39	15	39	15
53	62	23	39	15	39	15	39	15	39	15	39	15
54	62	23	39	15	39	15	39	15	39	15	39	15
55	46	17	23	9	23	9	23	9	23	9	23	9
56	46	17	23	9	23	9	23	9	23	9	23	9
57	32	12	23	9	23	9	23	9	23	9	23	9
58	32	12	9	3	9	3	9	3	9	3	9	3
59	32	12	9	3	9	3	9	3	9	3	9	3
60	32	12	9	3	9	3	9	3	9	3	9	3
61	32	12	9	3	9	3	9	3	9	3	9	3
62	32	12	9	3	9	3	9	3	9	3	9	3
63	9	3	9	3	9	3	9	3	9	3	9	3
64	9	3	9	3	9	3	9	3	9	3	9	3
65	9	3	9	3	9	3	9	3	9	3	9	3

Figure 6.4

Illustrating the effect that the presence or absence of the 'bugremove' has on the percentage of alignment found in Strongly Conserved Regions (SCRs) as the number of alignments, used by the Mix'n'Match algorithm to find SCRs, increases. Sixty five initial alignments are input to Mix'n'Match, generated using varying gap penalties (shown in table 3.2), scoring matrices (described in table 3.3) and multiple alignment programs as described in the text. The alignments are of eleven serine proteases, as listed in table 6.2. The alignments are ranked according to the number of identical residues, highest to lowest. The alignments are consecutively added together one at a time and the Mix'n' Match algorithm applied. All Mix'n'Match parameters are set to their defaults except the module is either enabled or disabled.

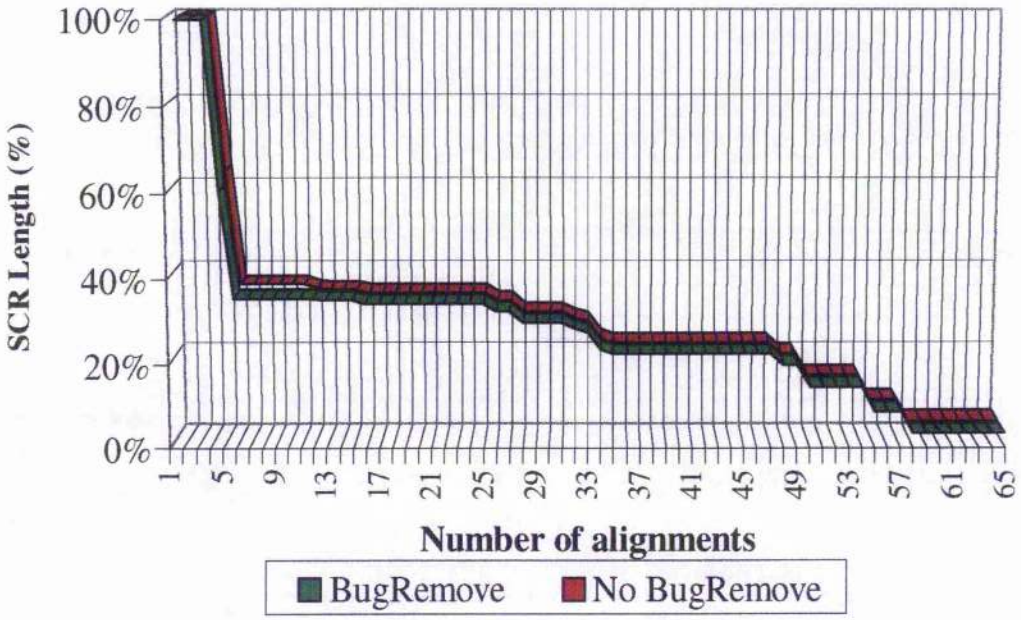

100%
80%
60%
40%
20%
0%
SCR Length (%)
1
5
9
13
17
21
25
29
33
37
41
45
49
53
57
61
65
Number of alignments
BugRemove
No BugRemove

Table 6.6

Showing the effect of whether or not the module 'bugremove' is enabled, on the length of Strongly Conserved Regions (SCRs) found. The figure given is the total number of residues that occupy SCRs in the whole protein, i.e., at the top all or most residues are defined as SCRs whereas at the bottom, only a few or no residues are. The alignments are of eleven serine proteases, as listed in table 6.2. The number of alignments used to calculate the SCRs, range from two to sixty five alignments. Default settings are used for other parameters. The module 'bugremove' is enabled by default, shown shaded above.

No of Alignments	BugRemove enabled		BugRemove disabled	
	SCR Length	SCR Length (%)	SCR Length	SCR Length (%)
2	268	100	268	100
3	268	100	268	100
4	268	100	268	100
5	162	60	170	63
6	95	35	98	37
7	95	35	98	37
8	95	35	98	37
9	95	35	98	37
10	95	35	98	37
11	95	35	98	37
12	95	35	95	35
13	94	35	94	35
14	94	35	94	35
15	94	35	94	35
16	92	34	92	34
17	92	34	92	34
18	92	34	92	34
19	92	34	92	34
20	92	34	92	34
21	92	34	92	34
22	92	34	92	34
23	92	34	92	34
24	92	34	92	34
25	92	34	92	34
26	87	32	87	32
27	87	32	87	32
28	80	30	80	30
29	80	30	80	30
30	80	30	80	30
31	80	30	80	30
32	76	28	76	28
33	74	28	74	28
34	63	24	63	24
35	60	22	60	22
36	60	22	60	22
37	60	22	60	22
38	60	22	60	22
39	60	22	60	22
40	60	22	60	22
41	60	22	60	22
42	60	22	60	22
43	60	22	60	22
44	60	22	60	22
45	60	22	60	22
46	60	22	60	22
47	60	22	60	22
48	53	20	53	20
49	53	20	53	20
50	39	15	39	15
51	39	15	39	15
52	39	15	39	15
53	39	15	39	15
54	39	15	39	15
55	23	9	23	9
56	23	9	23	9
57	23	9	23	9
58	9	3	9	3
59	9	3	9	3
60	9	3	9	3
61	9	3	9	3
62	9	3	9	3
63	9	3	9	3
64	9	3	9	3
65	9	3	9	3

length eight mer, when between six and eleven alignments are considered, adds an extra SCR of length three (figure 6.5), and when more than eleven alignments are considered, no difference in results are found. The feature that this module protects against, therefore seems to a rare event.

For the serine protease alignments, the overall trends in SCR length as the number of alignments under consideration increases, are the same as seen with the shikimate enzymes in chapter 3.4. A large drop in percentage length is seen once the first non-identical alignment is considered. The SCR length then gradually declines. The second sharp drop apparent with shikimate enzymes e2 to e5 is not present and, like shikimate enzymes e1 and e3, SCRs are found even when considering all sixty five alignments.

The cut-off point in the number of alignments to consider for delineating SCRs is picked after examination of the trend in SCR length as the number of alignments under consideration increases. A value is initially chosen using the shikimate enzyme data. The data for the serine proteases confirm that the choice is acceptable.

The above results show that similar SCRs are found over a broad range of the internal parameters.

6.3 Control alignments produced using default settings

Using the sixty five initial alignments generated for the globin and serine protease families, Mix'n'Match is used to produce final alignments. For the globin family, four LCRs must be picked and for the serine protease family, nine LCRs must be picked. For the serine protease family, the selection of alignments for each LCR is shown in appendix K. This is the output file '`print.me`' from the Mix'n'MatchA program. The choice of LCRs is shown in table 6.7. The criteria used is as discussed above.

Figure 6.6 shows the multiple alignment produced by Mix'n'Match for seven members of the globin family, where the proportion of identical residues in all seven sequences is

Figure 6.5

The Mix'n'Match alignment of eleven serine proteases. The SCRs found by the Mix'n'Match program when five alignments are considered are shown above the alignment, with the SCRs found by the Mix'n'Match program when between six and eleven alignments are considered are shown below the alignment. Extra SCRs found when the module 'bugremove' is absent, are shown by including asterices (*) in the SCRs. The default parameters of Mix'n'MatchA are used. The choice of alignment for each LCR is made randomly, by picking the first alignment for each LCR from the range available.

(A) Globin sequences

LCR Number	Possible choices	Alignment chosen
1	1, 12, 13, 30, 32, 40, 48	1
2	1, 12	1
3	1, 2, 12, 30	2
4	1, 7, 12, 13, 30, 31,	13

(B) Serine protease sequences

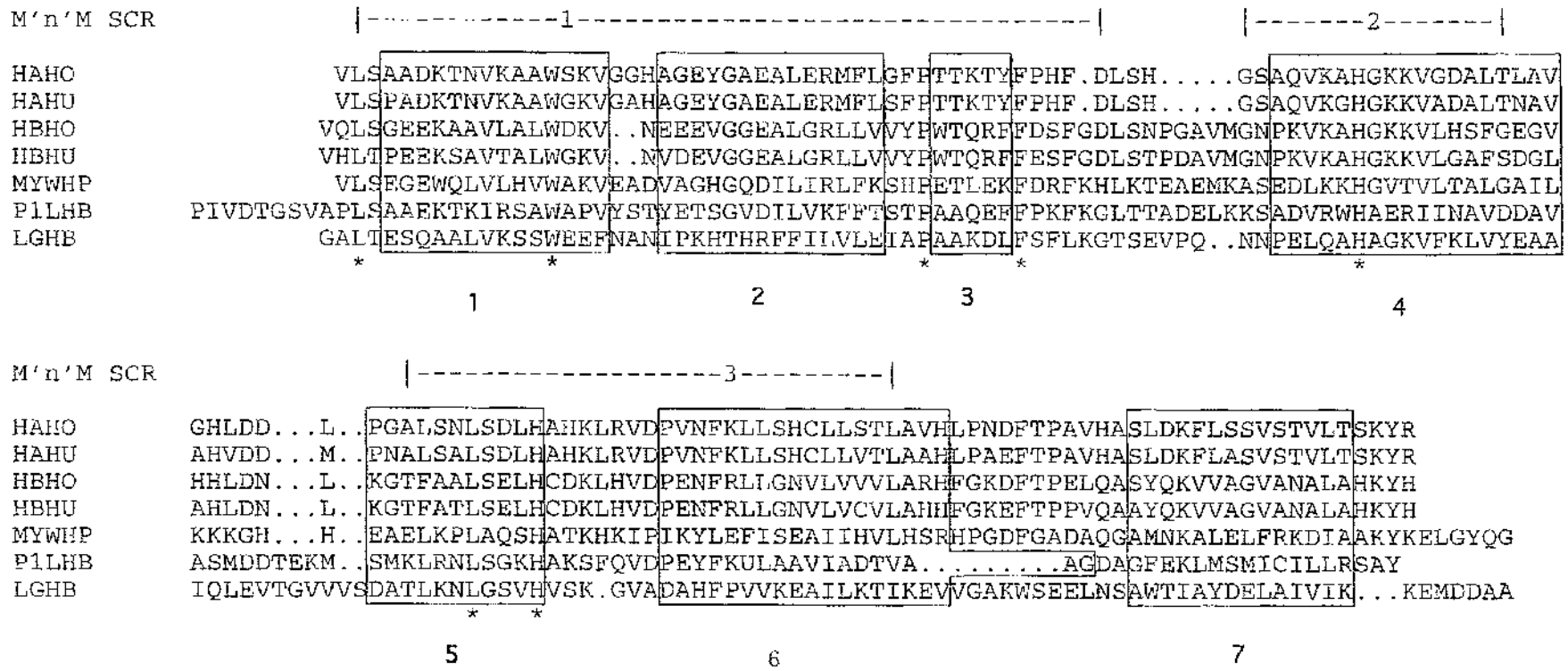
LCR Number	Possible choices	Alignment chosen
1	1,8,17,19,21,25,26,27,28,35,37,44,46,53,55,56	17
2	1,8,17,19,21,23,25,26,27,28,35,36,37,39,44,45,46,53,55,56,58,61,62,65	35
3	1,8,19,21,26,27,37,39,44,55,56	55
4	1,8,19,21,26,27,37,44,48,53,54,55,56	56
5	1,8,19,21,25,26,44,46,55,58	25
6	1,3,8,19,21,23,26,37,41,44,55,56	56
7	1,8,19,21,23,25,26,44,48,55,61,62	44
8	1,8,19,21,25,26,44,55,58	21
9	1,7,10,16,19,21,26,55	55

Table 6.7

Showing the possible choice of alignments for each LCR when aligning (**A**) seven globin sequences and (**B**) eleven serine protease sequences. Sixty five initial alignments are generated for the globin and serine protease families as input to Mix'n'Match and the possible choices refer to the initial alignment that the unique LCR alignment is found in. The actual choices for the serine protease alignments are listed in appendix K.

Figure 6.6

A Mix'n'Match multiple alignment of seven globins. The top row, labelled "M'n'M SCR", gives the three SCRs found by Mix'n'Match. The boxed regions numbered one to seven show the test regions. Bold numbers are used to indicate that the test regions is correctly aligned. These test regions are structurally equivalent residues from secondary structural units. '*' is placed under aligned, identical residues and '.' represents a gap.


M'n'M SCR
1
2
3
4
M'n'M SCR
3
5
6
7
HAHO horse haemoglobin a-chain
HBHO horse haemoglobin b-chain
HAHU human haemoglobin a-chain
HBHU human haemoglobin b-chain
MYWHP sperm whale myoglobin
P1LHB sea lamprey cyanohaemoglobin
LGHB lupin leghaemoglobin

only five percent. The boxes in figure 6.6 are drawn around the test regions and bold numbers are used to indicate that a test region is correctly aligned. Mix'n'Match produces an alignment in good agreement with the reference alignment of Lesk and Chothia [58]. Six of the seven test regions are correctly aligned, with test region six having two residues misaligned. When compared to the results of the automatic methods used to generate the initial alignments, we see that most sets of scoring parameters produce alignments worse than this, and none better (table 6.8a). Table 6.8b shows that each test region was correctly aligned in at least some of the initial alignments. This may lead to an expectation that Mix'n'Match should have correctly aligned all seven test regions, instead of producing an alignment with one region misaligned. This is not a fault of the criterion used for choosing the best alignment for each LCR, but of the initial alignments. Test regions six and seven are both part of one LCR and none of the initial alignments had both of these test regions correctly aligned. So Mix'n'Match has found the alignment with the highest number of correctly aligned test regions that it can, with the alignments used as input.

Figure 6.7 shows the alignment produced by Mix'n'Match of eleven members of the serine protease family, where the proportion of identical residues in these sequences is eight percent. The boxes in figure 6.7 are drawn around the test regions and bold numbers are used to indicate that a test region is correctly aligned. Mix'n'Match produces an alignment in good agreement with the reference alignment of Greer [48, 49]. Fourteen of the test regions are correctly aligned. When compared to the results of the automatic methods used to generate the initial alignments, we see that this final alignment is better than any of the initial alignments (table 6.8c). From table 6.8d, we see that sixteen test regions were found to be correctly aligned in at least some of the initial alignments. Therefore, it might in principle have been possible to produce an even better final alignment. The reason for correctly aligning fourteen rather than sixteen regions is that the main criterion in choosing the alignment for each LCR, relies on relatively inaccurate secondary structure prediction information. In addition, two of the regions are correctly aligned by only one out of the sixty five alignments and

Figure 6.7

A Mix'n'Match multiple alignment of eleven serine proteases. The serine protease sequences used in the alignment are as follows: bovine chymotrypsin (CHG), bovine trypsin (NTP), porcine elastase (ELA), bacterial trypsin (N1SGT), group specific protease (EZEC58), factor Xa (EXBO), haptoglobin heavy chain (HPHB), factor IXA (KFBO), human plasminogen (PLHU), plasmin β-chain (TBBO) and kallikrein (N2PKA). The top row, labelled "M'n'M SCR", gives the nine SCRs found by Mix'n'Match. The boxed regions numbered one to twenty three show the test regions. Bold numbers are used to indicate that the test regions is correctly aligned. These test regions are structurally equivalent residues from secondary structural units. '*' is placed under aligned, identical residues and '.' represents a gap.

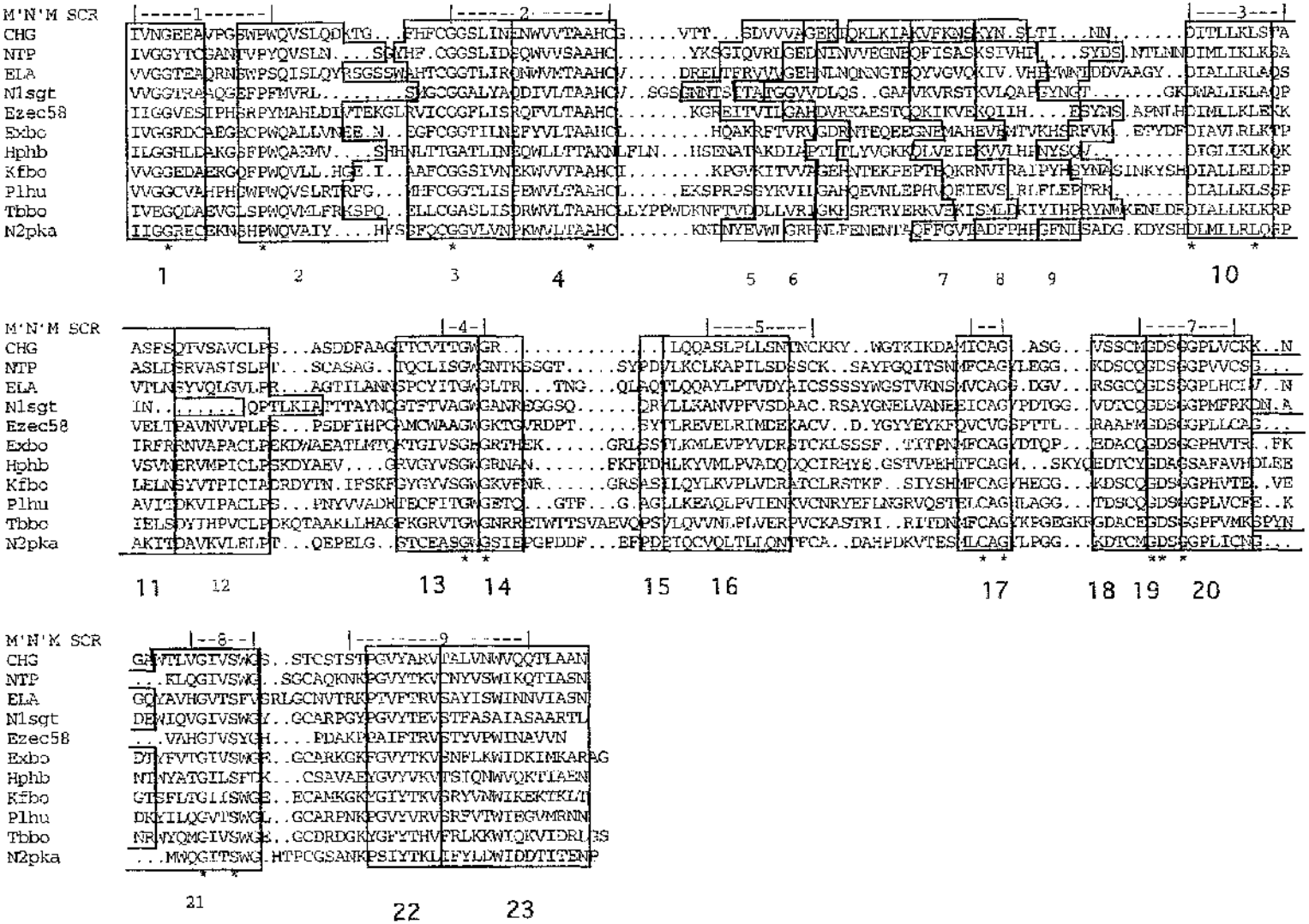

M'N'M SCR
CHG
NTP
ELA
N1sgt
Ezec58
Exbo
Hphb
Kfbo
Plhu
Tbbo
N2pka
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23

Table 6.8

Correctly aligned test regions in alignments of seven globins and eleven serine proteases. For each of the two protein families used to test the accuracy of Mix'n'Match, sixty five initial alignments are generated using ALIEN and PILEUP. For the seven globin sequences, seven regions are used to test the accuracy of these initial alignments. For the eleven serine proteases, twenty three test regions are used. A and C show how many of the initial alignments have a certain number of these regions correctly aligned in the globin and serine protease families, respectively. At the right is shown the numnber of test regions that are correct in the final alignment generated by Mix'n'Match. B and D show how often each of the individual test regions is correctly aligned for the globin and serine protease families, respectively.

A

	Number of test regions found correctly aligned (out of a possible seven)						Mix'n'Match
	0	1	3	4	5	6	6
Number of alignments with this number of correct regions	25	4	7	8	12	9	

B

	Test region						
	1	2	3	4	5	6	7
Number of alignments that correctly align the region (out of a possible 65)	32	40	42	29	20	3[b]	19[a]

C

	Number of test regions found correctly aligned (out of a possible twenty three)											Mix'n'Match
	1	2	4	5	6	7	8	9	10	12	13	14
Number of alignments with this number of correct regions	2	2	4	9	21	6	6	7	2	3	3	

D

	Test region																						
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Number of alignments that correctly align the region (out of a possible 65)	61	1[a]	0	40	0	0	0	0	0	53	43	0	25	9	8	15	12	7[b]	62	33	1[a]	58	11[b]

[a]These alignments are all produced using ALIEN

[b]These alignments are all produced using PILEUP

consequently, it would be too much to expect that the best possible alignment is always chosen for each LCR.

As well as comparing the alignments produced by Mix'n'Match to reference alignments, we compare them to the alignments produced by other alignment methods. The alignments used as comparisons are those given in the references for the methods. For the globin family, using the seven sequences listed in table 6.1, Barton and Sternberg ([254]; B & S) produce an alignment with one test region misaligned, Henneke [248] obtains an alignment with two test regions misaligned and Higgins and Sharp ([281]; CLUSTAL) generate an alignment with three test regions misaligned. For the serine protease family, using the eleven sequences listed in table 6.2, Henneke [248] produces an alignment with ten test regions correctly aligned. These results are summarised in table 6.9, together with the results from the initial alignment programs (ALIEN and PILEUP) used by Mix'n'Match and those of Mix'n'Match itself. For ALIEN and PILEUP, the number of correctly aligned test regions shown is taken from the best alignment out of those initially generated. Because the results of automatic methods are dependent on their initial scoring parameters, this gives an advantage over the results for the other three methods in table 6.9 (B & S, Henneke, CLUSTAL), generated from only one set of scoring parameters. Mix'n'Match, with its use of a large range of scoring parameters, produces alignments that are independent of any single scoring parameter. Also, prejudging which alignment is most correct is only possible when the correct alignment is already known. Mix'n'Match generates alignments as good as, or better than, the best of any of the other methods, without foreknowledge of the correct alignment.

Figures 6.8 and 6.9 display the SCRs found for the serine protease and globin families respectively, compared to the known structurally conserved regions. The SCRs found, as the number of alignments used to calculate SCRs varies, illustrate that the SCRs are in good agreement with the known structurally conserved regions over a broad range of values, including when the default value of fifteen alignments is used.

Figure 6.8

Comparison of the SCRs found by Mix'n'Match to the structurally conserved regions of eleven serine protease sequences. The different SCRs found by Mix'n'Match, as the number of alignments used to find SCRs increases, are shown. The serine protease sequences used are as listed in table 6.2. Default Mix'n'Match parameters are used when aligning. The twenty three structurally conserved regions, taken from the reference alignment of Greer, are shown numbered below the alignment although this is only a rough guide as not all are rectangular in shape. Above the alignments are shown the SCRs found by Mix'n'Match with the prefix 'scr number'. The number indicates how many alignments are used to find the SCRs shown e.g., the line starting 'SCR 5' shows that SCRs found when five alignments are considered. Where numbers of alignments used to find SCRs are not listed, the same results are obtained as the next lowest number shown, e.g., the SCRs found when thirty alignments are used to find SCRs, are the same as shown for the line starting 'SCR 28'.


scr 58
scr 55
scr 50
scr 48
scr 35
scr 34
scr 33
scr 32
scr 28
scr 26
scr 16
scr 13
scr 6
scr 5
N1chg IVNGEBAVPGSWPWQVSLQDKTG...PHFCGGSLINENWVVTAAHCG.....VTT...SDVVVAGEKIQKLKIAKVFKNSKYN.SLDI...NN.......DITLLKLSTA
N1ntp IVGGYTCGANTVPYQVSLN....SGYHF.CGGSLIDSQWVVSAAHC.......YKSGIQVRLGEDNINVVEGNEQFISASKSIVHP....SYDS.NTLNNDIMLIKLKSA
N3est VVGGTEAQRNSWPSQISLQYRSGSSWAHTCGGTLIRQNWVMTAAHCV.....DRELTFRVVVGEHNLNQNNGTEQYVGVQKIV.VHPYWNTDDVAAGY..DIALLRLAQS
N1sgt VVGGTRAAQGEFPFMVRL........SMGCGGALYAQDIVLTAAHCV..SGSGNNTSITATGGVVDLQS..GAAVKVRSTKVLQAPGYNGT.......GKDWALIKLAQP
Ezec58 IIGGVESIPHSRPYMAHLDIVTEKGLRVICGGFLISRQFVLTAAHC.......KGREITVILGAHDVRKAESTQQKIKVEKQIIH....ESYNS.APNLHDIMLLKLEKK
Exbo IVGGRDCAEGECPWQALLVNEE.N..EGFCGGTILNEFYVLTAAHCL........HQAKRFTVRVGDRNTEQEEGNEMAHEVEMTVKHSRFVK..ETYDFDIAVLRLKTP
Hphb ILGGHLDAKGSFPWQAKMV....SHNLTTGATLINEQWLLTTAKNLFLN..HSENATAKDIAPTLTLYVGKKQLVEIEKVVLHPNYSQV.........DIGLIKLKQK
Kfbo VVGGEDAERGQFPWQVLL.HGE.I..AAFCGGSIVNEKWVVTAAHCI........KPGVKITVVAGEHNTEKPEPTEQKRNVIRAIPYHSYNASINKYSHDIALLELDEP
P1hu VVCGCVAHPHSWPWQVSLRTRFG...MHFCGGTLISPEWVLTAAHCL.....EKSPRPSSYKVILGAHQEVNLEPHVQEIEVS.RLFLEPTRK.......DIALLKLSSP
Tbbo IVEGQDAEVGLSPWQVMLFRKSPQ..ELLCGASLISDRWVLTAAHCLLYPPWDKNFTVDDLLVRIGKHSRTRYERKVEKISMLDKIYIHPRYNWRENLDRDIALLKLKRP
N2pka IIGGRECEKNSHPWQVAIY....HYSSFQCGGVLVNPKWVLTAAHC.......KNDNYEVWLGRHKLFENENTAQFFGVTADFPHPGFNLSADG.KDYSHDLMLLRLQSP
11-] {--12--] [--13--][14] {[----16----} [17-] [18-][-][-20--]

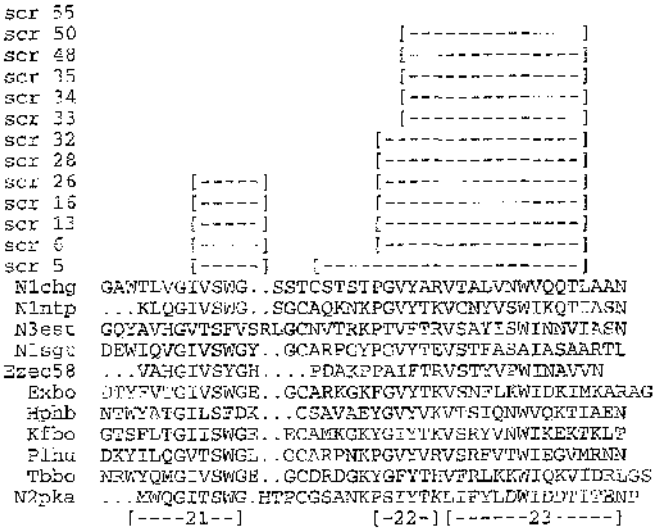

scr 55
scr 50
scr 48
scr 35
scr 34
scr 33
scr 32
scr 28
scr 26
scr 16
scr 13
scr 6
scr 5
N1chg GAWTLVGIVSWG..SSTCSTSTPGVYARVTALVNWVQQTLAAN
N1ntp ...KLQGIVSWG..SGCAQKNKPGVYTKVCNYVSWIKQTIASN
N3est GQYAVHGVTSFVSRLGCNVTRKPTVFTRVSAYISWINNVIASN
N1sgx DEWIQVGIVSWGY..GCARPGYPGVYTEVSTFASAIASAARTL
Ezec58 ...VAHGIVSYGH....PDAKPPAIFTRVSTYVPWINAVVN
Exbo DTYFVTGIVSWGE..GCARKGKFGVYTKVSNFLKWIDKIMKARAG
Hphb NTWYATGILSPDX...CSAVAEYGVYVKVTSIQNWVQKTIAEN
Kfbo GTSFLTGIISWGE..ECAMKGKYGIYTKVSRYVNWIKEKTKLP
P1hu DKYILQGVTSWGL..GCARPNKPGVYVRVSRFVTWIEGVMRNN
Tbbo NRWYQMGIVSWGE..GCDRDGKYGFYTHVFRLKKWIQKVIDRLGS
N2pka ...KWQGITSWG.HTPCGSANKPSIYTKLIFYLDWIDDTIQENP
[----21--] [-22-][------23-----]

Figure 6.9

Comparison of the SCRs found by Mix'n'Match to the structurally conserved regions of seven globin sequences. The different SCRs found by Mix'n'Match, as the number of alignments used to find SCRs increases, are shown. The globin sequences used are as listed in table 6.1. Default Mix'n'Match parameters are used when aligning. The structurally conserved regions, shown below the alignment, are as used by Barton and Sternberg and taken from the reference alignment of Lesk and Chothia and correspond to conserved α–helices A, B, C, E, F, G and H. For helix G, this is only a rough guide as it is not rectangular in shape. Above the alignments are shown the SCRs found by Mix'n'Match with the prefix 'scr number'. The number indicates how many alignments are used to find the SCRs shown e.g., the line starting 'SCR 10' shows that SCRs found when ten alignments are considered. Where numbers of alignments used to find SCRs are not listed, the same results are obtained as the next lowest number shown, e.g., the SCRs found when thirty alignments are used to find SCRs, are the same as shown for the line starting 'SCR 27'.


SCR 41
SCR 27
SCR 25
SCR 16
SCR 10
SCR 9
SCR 2
HAHO VLSAADKTNVKAAWSKVGGHAGEYGAEALERMFLGFPTTKTYFPHF.DLSH.....GSAQVKAHGKKVGDALTLAV
HAHU VLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHF.DLSH.....GSAQVKGHGKKVADALTNAV
HBHO VQLSGEEKAAVLALWDKV..NEEEVGGEALGRLLVVYPWTQRFFDSFGDLSNPGAVMGNPKVKAHGKKVLHSFGEGV
HBHU VHLTPEEKSAVTALWGKV..NVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNPKVKAHGKKVLGAFSDGL
MYWHP VLSEGEWQLVLHVWAKVEADVAGHGQDILIRLFKSHPETLEKFDRFKHLKTEAEMKASEDLKKHGVTVLTALGAIL
P1LHB PIVDTGSVAPLSAAEKTKIRSAWAPVYSTYETSGVDILVKFFTSTPAAQEFFPKFKGLTTADELKKSADVRWHAERIINAVDDAV
LGHB GALTESQAALVKSSWEEFNANIPKHTHRFFILVLEIAPAAKDLFSFLKGTSEVPQ..NNPELQAHAGKVFKLVYEAA
|------A------| |---------B----| |-C| |---------B------|
SCR 27
SCR 25
SCR 16
SCR 10
SCR 9
SCR 2
HAHO GHLDD...D..PGALSNLSDLHAHKLRVDPVNFKLLSHCLLSTLAVHLPNDFTPAVHASLDKFLSSVSTVLTSKYR
HAHJ AHVDD...M..PNALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAAHLPAEFTPAVHASLDKFLASVSTVLTSKYR
HBHO HHLDN...L..KGTFAALSELHCDKLHVDPENFRLLGNVLVVVLARHFGKDFTPELQASYQKVVAGVANALAHKYH
HBHJ AHLDN...L..KGTFATLSELHCDKLHVDPENFRLLGNVLVCVLAHHFGKEFTPPVQAAYQKVVAGVANALAHKYH
MYWHP KKKGH...H..EAELKPLAQSHATKHKIPIKYLEFISEAIIHVLHSRHPGDFGADAQGAMNKALELFRKDIAAKYKELGYQG
P1LHB ASMDDTEKM..SMKLRNLSGKHAKSFQVDPEYFKVLAAVIADTVA.........AGDAGFEKLMSMICILLRSAY
LGHB IQLEVTGVVVSDATLKNLGSVHVSK.GVADAHFPVVKEAILKTIKEVVGAKWSEELNSAWTIAYDELAIVIK...KEMDDAA
|---- F-----| |-------G-------| |------H-----|

Table 6.9

A comparison of the accuracy of different alignment methods. The table gives the number of test regions correctly aligned by different methods for seven globins and eleven serine proteases. The figures at the end of row gives the total number of test regions in these protein families. The methods used are those of Barton and Sternberg (B & S), Henneke (Henneke), Higgins and Sharp (CLUSTAL), the initial aligning methods ALIEN and PILEUP, and Mix'n'Match. The alignments of the globins and serine proteases are those given in the above references, except for ALIEN and PILEUP, where the figures show the results for the best alignment out of the sixty five initial alignments generated for Mix'n'Match.

Alignment Method / Protein Family	B & S	Henneke	CLUSTAL	ALIEN	PILEUP	Mix'n'Match	Total number of test regions
Globins	6	5	4	6[a] (6[b])	5[a] (5[b])	6	7
Serine Proteases	-	10	-	9[a] (8[b])	13[a] (10[b])	14	23
No. of alignments generated.	1	1	1	54	11	65	

[a]The number of correctly aligned test regions was taken from the best of the alignments generated. Such a choice is only possible knowing what the 'correct' alignment should be.

[b]The number of correctly aligned test regions obtained using the default parameters for ALIEN and PILEUP.

6.4 Mix'n'Match alignment of the shikimate enzymes

The sequences for shikimate enzymes e1 to e5, are aligned using Mix'n'Match. The settings used for the initial aligning programs, ALIEN and PILEUP, are as previously described in chapter 3.4. The SCRs are automatically chosen by Mix'n'Match with default settings used for all parameters.

The alignments found are shown in figure 6.10. SCRs are found in each shikimate pathway enzyme family (figure 6.10). For four of the enzymes, e1, e2, e3, and e5, eight or more SCRs are found and these are evenly spaced throughout the alignments. These SCRs comprise forty four or more percent of the alignments (table 6.10). The number of different alignments found for each LCR ranges from two to fourteen (table 6.11). This indicates that the alignments are comprised of alternating well conserved regions and less well conserved regions. This is consistent with what is known about the structure of protein families. For these low sequence identity protein families (table 6.10), the initial automatic aligning methods used, produce alignments that vary greatly with changes to parameters, and have no way of identifying reliably aligned regions. Creating a composite alignment from the range generated is a difficult task. The Mix'n'Match method simplifies the task of producing a final alignment, by only requiring alignment of the forty to sixty percent of the protein, identified as LCR. Following on from the results observed for the test enzyme families, where SCRs share good correlation with structurally conserved regions, it is anticipated that the forty to sixty percent of the protein sequences, defined as SCRs by Mix'n'Match, will correspond to conserved structural or functional regions in the structures of the enzymes. In agreement with this, the sequence motif identified by Bugg *et al.* [119], thought to have a role in substrate binding or to be near the active site, is located in a SCR, in the enzymes it occurs in, e1, e2 and e4. For proteins with this percentage sequence identity, the percentage of the molecules' structures which would have the same general fold, is similar to the percentage SCR lengths found for these four shikimate pathway enzymes [59]. Testing the validity of the alignments produced, and

Shikimate enzyme	e1	c2	e3	e4	e5
SCRs					
SCR length (residues)	259	142	179	61	252
SCR length (%)	61.9	52.0	57.9	19.5 (26.8[b])	44.5
Sequence identity					
Identical residues (residues)	112	31	30	14	41
Percentage identity (%)	26.8	11.3	9.7	4.5 (6.1[b])	7.2
Length of alignment (residues)	418	273	309	313 (228[a])	566

[a]The shikimate kinase sequence from tomato has an eighty five residue long N terminal sequence, not seen in the other family members. The length of the alignment, if this non homologous region is ignored, is shown in brackets.

[b]Numbers in brackets are the results of the percentage calculations using the smaller alignment length for shikimate kinase, e4 (see note [a] above).

Table 6.10

The total length of SCRs and number of identical residues found in Mix'n'Match alignments of shikimate pathway enzymes e1 to e5.

	The number of alignments found for each LCR.				
	Shikimate pathway enzyme				
LCR Number	e1	e2	e3	e4	e5
1	4 (1)	8 (10)	3 (16)	9 (7)	14 (19)
2	2 (6)	2 (6)	7 (45)	2 (10)	8 (23)
3	2 (1)	4 (10)	5 (16)	2 (6)	4 (11)
4	5 (25)	2 (1)	5 (21)	3 (6)	3 (1)
5	11 (39)	2 (6)	2 (1)	9 (7)	5 (1)
6	2 (6)	3 (17)	2 (6)	-	6 (18)
7	5 (6)	2 (10)	4 (6)	-	8 (1)
8	2 (6)	4 (6)	2 (6)	-	6 (20)
9	3 (34)	3 (1)	6 (21)	-	5 (39)
10	-	2 (6)	-	-	5 (18)
11	-	2 (6)	-	-	-
Total number of LCRs found	9	11	9	5	10

Table 6.11

The number of LCRs found by the Mix'n'Match program when aligning shikimate pathway enzymes e1 to e5. For each enzyme family, the number of possible alignments for each LCR found, is shown, together with the alignment number that was chosen for the final alignment of the enzyme family.

whether the SCRs are structural or functional regions, awaits the completion and release of the results from on-going x-ray crystallography studies on these enzymes.

For shikimate kinase, only four SCRs are found, concentrated in the N terminal half of the alignment. These four SCRs comprise only twenty seven percent of the alignment. This reflects the poorly conserved nature of the C terminus of e4, where no identical residues are found. Shikimate kinase also has the lowest overall percentage identity of the five shikimate enzymes studied, at six percent (table 6.10). The number of different alignments found for each LCR of e4 is low; two or three for the internal LCRs (numbers two to four), and nine for the N and C terminal LCRs. This indicates that the N terminus is better conserved, with the initial aligning programs finding little potential for variation.

The potential to homology model shikimate kinase on adenylate kinase is noted in chapter two. The low percentage sequence identity between shikimate kinase family members, may act to discourage such speculation because if aligning the sequences of family members is difficult, it may be harder to align to the sequence of a different enzyme family. However, identical residues are found spread throughout the alignment between shikimate kinase and adenylate kinase, and the percentage sequence identity is approximately four times higher than within the shikimate kinase family (figure 2.3).

Analysis of the Mix'n'Match consensus secondary structure predictions for shikimate kinase, shows that most positions are unambiguously predicted. The consensus predictions mostly use capital letters, when all or all bar one of the predictions for a position agree, with very few small letters seen. The different methods used to predict consensus secondary structures for a position, also show a high degree of similarity (figure 6.10). This uniformity in predicting secondary structure indicates that although there is only six percent sequence identity, there is a much stronger structural identity in this family.

Figure 6.10

Mix'n'Match alignments of the sequence of the shikimate pathway enzymes, e1-e5. These figures show the final output from Mix'n'Match, (normally held in a file called '`print.me2`'), with the addition of the line labelled 'SCR'. The sequences used are as listed in table 2.2. The meanings of the other legends found, above and below the sequence alignment are:

SCR = The Strongly Conserved Regions found by Mix'n'Match

GGBSM = Gascuel & Golmord consensus secondary structure prediction

CF = Chou & Fasman (as coded by PEPTIDESTRUCTURE) consensus secondary structure prediction

GOR = Garnier, Osguthorpe & Robson (as coded by PEPTIDESTRUCTURE) consensus secondary structure prediction

GOR2 = Garnier, Osguthorpe & Robson (as coded by PEPPLOT) consensus secondary structure prediction

GOR3 = Garnier, Osguthorpe & Robson (as coded by PEPPLOT with Decision Constants) consensus secondary structure prediction

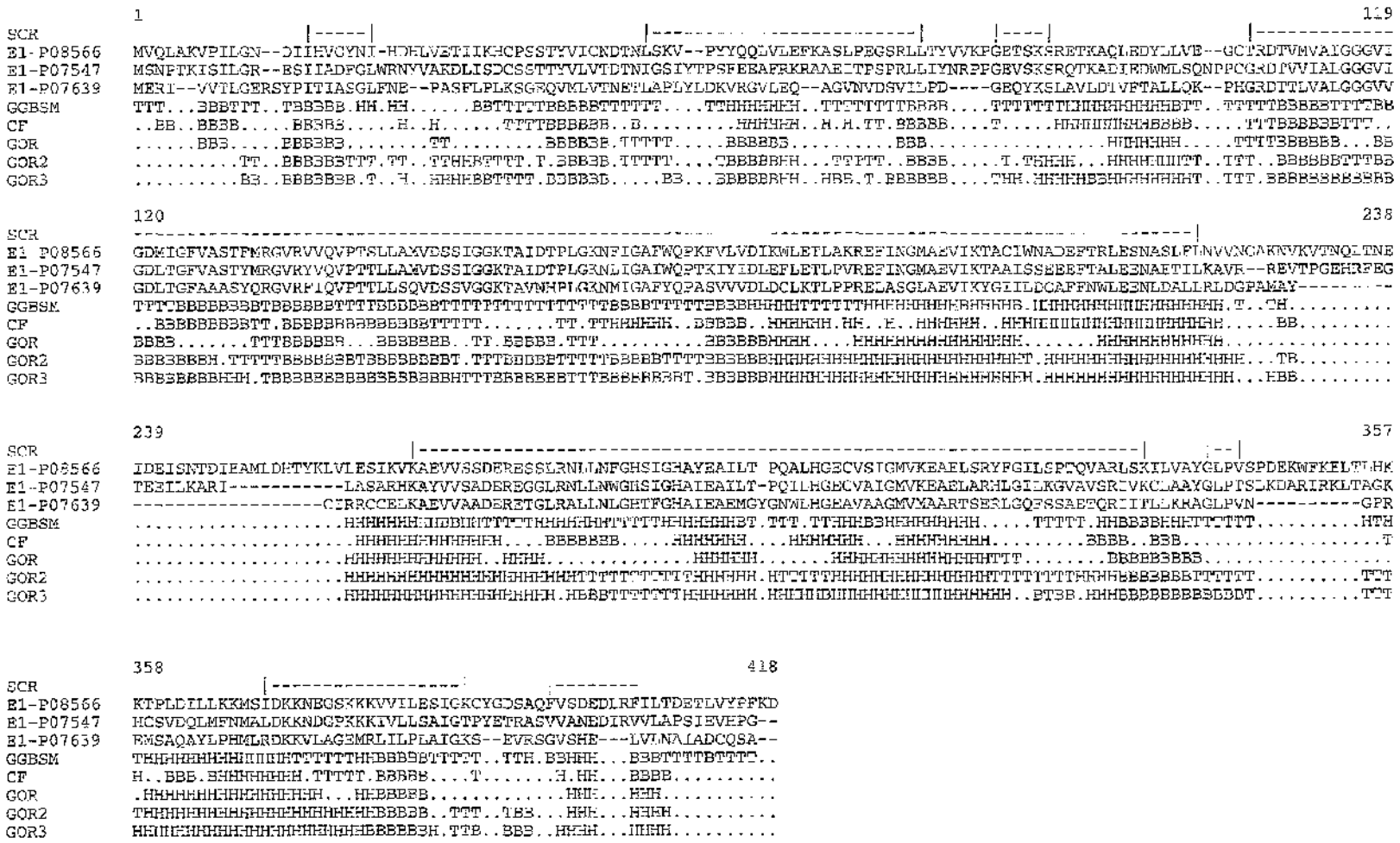
The secondary structure predictions are shown in a capital letter if all, or all bar one of the residues, have the same prediction. The characters used are:

b = β strand residue

h = α helical residue

t = turn residue

. = no prediction made



The Mix'n'Match alignment of shikimate pathway enzyme e1, 3-dehydroquinate synthase

The Mix'n'Match alignment of shikimate pathway enzyme e2, 3-dehydroquinase

The Mix'n'Match alignment of shikimate pathway enzyme e3, shikimate dehydrogenase

The Mix'n'Match alignment of shikimate pathway enzyme e4, shikimate kinase


1
119
SCR
E5_P07547
E5_P08565
E5_P07637
E5_P07638
E5_Pea
E5_P11043
E5_P10748
E5_P05466
E5_P12421
E5_P20691
GGBSM
CF
GOR
GOR2
GOR3
predicted
120
238
SCR
E5_P07547
E5_P08566
E5_P07637
E5_P07638
E5_Pea
E5_P11043
E5_P10748
E5_P05466
E5_P12421
E5_P20691
GGBSM
CF
GOR
GOR2
GOR3
predicted
239
357
SCR
E5_PC7547
E5_PC8566
E5_PC7637
E5_PC7638
E5_Pea
E5_P11043
E5_P10748
E5_P05466
E5_P12421
E5_P20691
GGBSM
CF
GOR
GOR2
GOR3
predicted

The Mix'n'Match alignment of shikimate pathway enzyme e5, EPSP synthase

A common core structure for type A purine nucleotide binding proteins, e.g., adenylate kinase and H-ras p21 protein, has been identified [50]. The predicted secondary structural units of shikimate kinase (shown in figure 6.10), show a good match with the order of adenylate kinase's secondary structural units (figure 6.11a). The secondary structure units, overlaid on the alignment of the proteins obtained using GAP during database searching (figure 6.12) shows an excellent fit, especially those purported to be in the common core. It is expected that further work could optimise the alignment to emphasise this similarity. For shikimate kinase, the agreement of secondary structure with adenylate kinase's, together with the existence of a putative ancestral core structure for nucleotide binding proteins, adds evidence to there being structural homology between these two proteins. Further, confirmation for the structural similarity between these proteins is found by Matsuo and Nishikawa [296] with a method based on sequence-structure compatibility.

This accumulation of evidence, supporting the hypothesis that these two proteins are structurally related, would appear to be justification for an attempt to homology model shikimate kinase, based on the structure of adenylate kinase.

A similar analysis of the consensus secondary structure predictions is carried out on the Mix'n'Match aligned EPSP synthase sequences. Only regions where there is consensus amongst the secondary structure predictions, are used. Stallings *et al.* [16] list the topology of the principal elements of secondary structure. The match, both in the order and number, between the predicted and actual secondary structures is poor. There are twelve predicted α-helices and eighteen predicted β-strands, compared to twenty three α-helices and twelve β-strands in the three dimensional structure (figure 6.11b). The six fold repeating secondary structure motif, found in the three dimensional structure, is not apparent from the predicted secondary structure. Although this is a poor result, it is perhaps expected given the low level of accuracy found in present methods of calculating secondary structure. It may also be seen as emphasising the strong match that is found between shikimate kinase and adenylate kinase.

(**a**) Comparison of the secondary structure of adenylate kinase to the predicted secondary structure of shikimate kinase

adenylate kinase	α1 β1 α2 [β2 α3 α4 α5] β3 α6 β4 [α7] α8 β5 [ ] α9
shikimate kinase	β1 α1 α2 α3 α4 β2 α5 β3 α6 α7 α8 β4 α9

Elements of the common core structure:

domain 1	β1 α2
domain 2	β3 α6 β4
domain 3	α8 β5
domain 4	α9
[...]	position of insertions in core structure, found in loops.

(**b**) Comparison of the secondary structure of EPSP synthase to the predicted secondary structure of EPSP synthase

Secondary structure of *Escherichia coli* e5

α α β α β α α α β α β α α α β α β α α α β α β α α α β α β
α β β α β α β α α β β β β β β α β α β β β α α β α β α β α

Predicted secondary structure of EPSP synthase

Figure 6.11

Comparing the predicted consensus secondary structure of e4 and e5 to secondary structures taken from three dimensional structures. A single composite secondary structure prediction is made from all the predictions in the Mix'n'Match alignment, using the rule that a secondary structure is only predicted where there is strong agreement between the different methods used. (**a**) The adenylate kinase sequence is from porcine muscle (database reference nrl_3d:3adk). The secondary structural elements, found in the four domains of the common core structural motif for type A purine nucleotide binding proteins is also shown, with the secondary structures that occur in between, enclosed in square brackets. (**b**) The EPSP synthase secondary structure is taken from the reference, cited in the text.

Figure 6.12

The alignment of the sequences of shikimate kinase and adenylate kinase, together with the predicted and actual secondary structure, respectively. The adenylate kinase sequence is from porcine muscle (database reference nrl_3d:3adk). The alignment is as described and shown previously (figure 2.3, chapter 2). The predicted secondary structure for shikimate kinase, is shown in the line above the sequence, labelled 'predicted', and the actual secondary structure for adenylate kinase is shown in the line below its sequence. Two structural types are shown, α-helical (indicated by 'h') and β-strand (indicated by 'b'). The structural types shown in bold, are those in one of the four domains of the common core structural motif for type A purine nucleotide binding proteins

Chapter 7 Conclusion

7.1 Conclusion

The objectives of this work were firstly, to investigate the possibility of homology modelling the shikimate enzymes e1 to e5; secondly, to investigate the possibility of *ab initio* modelling the shikimate enzymes e1 to e5; and thirdly, to determine weak links in the molecular modelling and sequence analysis process and develop methods if needed. The extent to which these objectives have been achieved shall be summarised.

To investigate the possibility of homology modelling the shikimate enzymes, e1 to e5, a thorough analysis was conducted using sequence databases and fast database searching algorithms. Many matches were found between shikimate enzymes and sequences in the databases, however, few of these showed any statistical significance when the sequences were realigned using the more sensitive, pairwise aligning program, GAP. To be of use in homology modelling, any matches found should show biological as well as statistical significance, and the structural details should be known for the database sequence. Only two matches were found that fitted this criterion, adenylate kinase and H-ras p21 protein, when aligned to shikimate kinase. The percentage sequence identity in the alignments between shikimate kinase and these two proteins is low. These alignments are not reliable enough on their own, to allow a homology modelling approach to be made. To attract greater credibility to the accuracy of alignments, more developmental work would be necessary in this area. Development of a model structure by homology modelling, is not presently possible for the other shikimate enzymes, e1, e2, e3, and e5, as no structural homologues were found during the database searches.

The second objective of the work, investigating the possibility of *ab initio* modelling the shikimate enzymes e1 to e5, involves as the initial step, aligning the sequences of interest. The results of database searches showed that only shikimate enzyme sequences

of the same family member were found to be similar to each other. For each of the enzymes, more than two sequences are available, which allows the use of multiple sequence aligning programs. This is beneficial, as multiple sequence aligning programs have been shown to produce more accurate alignments than other aligning methods. When this process was carried out, using two multiple sequence aligning programs, the alignments of the shikimate enzymes which were obtained, were found to be too variable to use. Altering the programs' parameters created different alignments, with no objective way to choose a single alignment or construct a composite alignment from the many produced for each shikimate enzyme.

Present multiple sequence aligning methods all exhibit this behaviour where the alignments produced are dependent on the scoring parameters used. This is most noticeable when aligning protein families with low sequence identity, which includes shikimate pathway enzymes. This crucial problem which appeared at the early stages of investigation, has been addressed and resolved by the development of the novel method, the third objective of this work.

This third objective, namely determining weak links in the molecular modelling and sequence analysis process and development of novel methods is achieved by the Mix'n'Match method [247]. This method provides a novel approach to multiple sequence aligning, based on finding alternating SCRs and LCRs in the protein sequences. The SCRs are used as 'anchors' in the alignment, and the alignment for each intervening LCR is chosen, independently of the other LCRs, from a selection of possible multiple alignments. As an aid in making this choice, consensus secondary structure predictions are shown alongside each different possible alignment for a LCR.

The Mix'n'Match method, which treats the sequences in two different ways, is based upon the observation from structurally known protein family sequences, that the structures alternate between conserved regions, corresponding to the framework of secondary structural units (e.g., α-helices and β-strands), and non conserved regions, corresponding to surface loops and coils or species specific secondary structural units.

This approach is in contrast to most multiple sequence alignment methods, which consider and align an entire sequence at a time. The Mix'n'Match method's two step approach, when aligning sequences, is a theoretically better approach to the problem of aligning low sequence identity protein families and has not been previously rigorously described.

Methods have been described for finding SCRs in protein sequence alignments. Mix'n'Match uses a novel method to define SCRs and also elevates the importance of aligning LCRs, the hardest to automatically define, yet biologically important regions of sequences, to a central role in the aligning procedure.

An advantage of the method used to define SCRs, compared to other multiple sequence aligning methods, in that the final alignment obtained is relatively independent of the choice of scoring parameters. This is of particular benefit when aligning low sequence identity protein families. The SCRs identified by Mix'n'Match, have also been shown to have a good correlation with the actual secondary structures found in the test enzyme families.

Of the results obtained using this method when aligning the shikimate enzymes, the most interesting result was obtained for the shikimate kinase sequences. Further information, obtained from the secondary structure predictions produced by the Mix'n'Match alignments, suggested a closer match with the actual secondary structure of adenylate kinase, than was found between their amino acid sequences. This homology was apparent particularly in adenylate kinase's putative type A purine nucleotide binding enzyme common core. This significant result is adequate evidence upon which to base a homology modelling attempt.

7.2 Future work

The Mix'n'Match method is a significant development in the area of multiple sequence aligning, however, it is not beyond the scope of improvement. One such improvement would be to increase the number of multiple sequence aligning methods used, from the two currently accepted. Also, recent secondary structure prediction algorithms, which claim to have more accurate predictions than the methods used in Mix'n'Match, could be incorporated. Due to the modular nature of the Mix'n'Match program, which was designed to be upgradeable, these improvements should not be difficult.

Additionally, optimising the FORTRAN code, either through reordering of the program or developing more efficient algorithms, could possibly result in significant speed increases. The time required for such an undertaking, however, may be better spent on completely rewriting Mix'n'Match in the computer languages, C or C++. The benefits of these languages compared to FORTRAN include easier structuring of a program, dynamic memory allocation and the availability of pointers. These changes would enable the number and size of sequences in an alignment to be increased and greater algorithmic freedom.

The Mix'n'Match program was used to increase the confidence in the homology between shikimate kinase and adenylate kinase. With these proteins identified as sharing functional, sequence and secondary structure homology, I predict that the tertiary structure of shikimate kinase will be closely related to the tertiary structure of adenylate kinase. An attempt could be made to homology model shikimate kinase based on the structure of adenylate kinase.

Appendix A Sequence data

The revised sequence for the Shikimate Dehydrogenase (e3) enzyme from the pentafunctional *Aspergillus nidulans* arom complex, as detailed by Dr. A. Hawkins, Newcastle University (personal communication, 29/10/91). The revised sequence is also shown aligned against the sequence for *Aspergillus nidulans* arom given in the database (OWL database accession number P07547). The alignment was carried out using the BESTFIT, local, global pairwise alignment method with the default settings, available in the GCG package [178]. Scoring ranges from one for identity to zero for complete dissimilarity. Identity is shown by '|', residues which have a score of greater than 0.5 with ':' and residues which score between 0.5 and 0.1 with '.'.

```
   1  PKKFAIFGSP ISQSRSPALH NTLFAQVGLP HNYTRLETTN AQDVQEFIRS
  51  PDFGGAFRNN SLKLDIMPLL DEVAAEAEII GAVNTIIPVS TGKNTPSRLV
 101  GRNTDWQGMI LSLRKAGVYG PKRKDQEQSA LVVGGGGTAR AAIYALENMG
 151  YSPIYIVGRT PSKLENMVST FPSSYNIRIV ESPSSFESVP HVAIGTIPAD
 201  QPIDPTMRET LCHMFERAQE ADAEAVKAIE HAPRILLEMA YKPQVTALMR
 251  LASDSGWKTI PGLEVLVGQG WYQVCFLASI ILIACELTER S
```

```
BESTFIT
Ratio:  1.127   Gaps:  4   Percent Similarity: 80.836   Percent Identity: 75.958
E3_New.Gcg x Old_Arom.Prot..

   1 PKKFAIFGSPISQSRSPALHNTLFAQVGLPHNYT...RLETTN.AQDVQE 46
     ||||||||||||| .|.| .| : . : ..  |   || |.. ...  .
1290 PKKFAIFGSPISQS.APQLSTTPYLPRSASPTTTPAWRLRTPKMCRSSSA 1338

  47 EIRSPDFGGAFRNNSLKLDIMPLLDEVAAEAEII GAVNTIIPVSTGKNTP 96
     :: |:. :..:|..| . .: . |.  : ..|::..    :|.. ::. .
1339 LLTSAAPSVTIRSSSTSCPFSTKLPRKPRSSELLTQ...SFPCRLARTLH 1385

  97 SRLVGRNTDWQGMILSLRKAGVYGPKRKDQEQSALVVGGGGTARAAIYAL 146
       .|||||||||||||||||||||||||||||||||||||||||||||||||
1386 HAYVGRNTDWQGMILSLRKAGVYGPKRKDQEQSALVVGGGGTARAAIYAL 1435

 147 HNMGYSPIYIVGRTPSKLENMVSTFPSSYNIRIVESPSSFESVPHVAIGT 196
     ||||||||||||||||||||||.|.||||||||||||||||||||||||||
1436 HNMGYSPIYIVGRTPSKLENMVSSFPSSYNIRIVESPSSFESVPHVAIGT 1485

 197 IPADQPIDPTMRETLCHMFERAQEADAEAVKAIEHAPRILLEMAYKPQVT 246
     ||||||||||||||||||||||||||||||||||||||||||||||||||||
1486 IPADQPIDPTMRETLCHMFERAQEADAEAVKAIEHAPRILLEMAYKPQVT 1535

 247 ALMRLASDSGWKTIPGLEVLVGQGWYQVCFLASIILIACELTERS 291
     |||||||||||||||||||||||||||||||||||||||||||||
1536 ALMRLASDSGWKTIPGLEVLVGQGWYQVCFLASIILIACELTERS 1580
```

The sequence for EPSP sythnase (e5) from pea, as detailed by Professor J. R. Coggins, Glasgow University (personal communication).

```
Pea.Pep  Length: 447  February 1, 1990  17:00  Check: 2445  ..

  1  KPSTAPEIVL EPISEISGTI TLPGSKSISN RILLLAALSE GTTVVENLLD
 51  SEDIHYMLEA LKTLGLRVED DKTTQRAVVE GSGGLFPIGR ESKDEVNLFL
101  GNAGTAMRPL TAALVAAGGN TRYILDGVPR MRERPIGDLV SGLKQLGADV
151  DCFLGTNCPP VRIIGKGGLP GGKVKLSGSI SSQYLTALLM AAPLALGDVE
201  IEIIDKLISV PYVEMTLKLM ERFGVSVEHS DNWDRFLVHG GQKYKSPGNA
251  FVEGDASSAS YFLAGAAVTG GTITVIGCGT SSLQGDVKFA EVLEKMGAKV
301  TWTENSVTVT GPPRDSSGRK VLQGIDVNMN KMPDVAMTLA VVALFANGPT
351  AIRDVASWRV KETERMIAIC TELRKLGATV EEGPDYCVIT PPEKLNVTSI
401  DTYDDHRMAM AFSLAACGDV PVTIKDPGCT RKTFPDYFQV LERFTKH
```

Appendix B Results for database searching using the BLAST program

The output obtained by searching the NRL_3D database of protein sequences with a known structure, with two shikimate kinase (e4) sequences, e4-P08329 and e4-P10880, using the BLAST program. Default settings are used for the BLAST program.

a) results for e4-P10880

Sequences producing High-scoring Segment Pairs:	High Score	Smallest Sum Probability P(N)	N
NRL:3ADK adenylate kinase (EC 2.7.4.3) pig	52	0.79	1
NRL:2HMGB hemagglutinin HA2 chain (bromelain digested wi...	43	0.94	2
NRL:2HMGD hemagglutinin HA2 chain (bromelain digested wi...	43	0.94	2
NRL:2HMGF hemagglutinin HA2 chain (bromelain digested wi...	43	0.94	2
NRL:3HMGB hemagglutinin HA2 chain (bromelain digested wi...	43	0.94	2
NRL:3HMGD hemagglutinin HA2 chain (bromelain digested wi...	43	0.94	2
NRL:3HMGF hemagglutinin HA2 chain (bromelain digested wi...	43	0.94	2
NRL:4HMGB hemagglutinin HA2 chain (bromelain digested wi...	43	0.94	2
NRL:4HMGD hemagglutinin HA2 chain (bromelain digested wi...	43	0.94	2
NRL:4HMGF hemagglutinin HA2 chain (bromelain digested wi...	43	0.94	2
NRL:5HMGB hemagglutinin HA2 chain mutant (D112G) (bromel...	43	0.94	2
NRL:5HMGD hemagglutinin HA2 chain mutant (D112G) (bromel...	43	0.94	2
NRL:5HMGF hemagglutinin HA2 chain mutant (D112G) (bromel...	43	0.94	2
NRL:1TLD trypsin (EC 3.4.21.4) (orthorhombic, pH 5.3) - ...	43	0.98	2
NRL:1NPP trypsin (EC 3.4.21.4) (isopropylphosphorylated)...	43	0.98	2
NRL:2PTCB trypsin (EC 3.4.21.4) beta (with basic protein...	43	0.98	2
NRL:2TGA trypsin (EC 3.4.21.4) precursor (2.4 M magnesiu...	43	0.98	2
NRL:2TGPZ trypsin (EC 3.4.21.4) precursor (with basic pr...	43	0.98	2
NRL:2TGT trypsin (EC 3.4.21.4) precursor (103 degrees K,...	43	0.98	2
NRL:3TPIZ trypsin (EC 3.4.21.4) precursor (with basic pr...	43	0.98	2
NRL:4PTP trypsin (EC 3.4.21.4) beta (diisopropylphosphor...	43	0.98	2
NRL:4TPIZ trypsin (EC 3.4.21.4) precursor (with mutant b...	43	0.98	2
NRL:1GBT trypsin (EC 3.4.21.4) beta (Ser-195 guanidinobe...	43	0.98	2
NRL:1PPEE trypsin (EC 3.4.21.4) (with winter squash tryp...	43	0.98	2
NRL:1PPHE Trypsin (EC 3.4.21.4) complex with noncovalent...	43	0.98	2
NRL:1TPO trypsin (EC 3.4.21.4) (orthorhombic, pH 5.0) - ...	43	0.98	2
NRL:1TPP trypsin (EC 3.4.21.4) (with p-amidino-phenyl-py...	43	0.98	2
NRL:1TGB trypsin (EC 3.4.21.4) precursor (with Ca, PEG) ...	43	0.98	2
NRL:1TGC trypsin (EC 3.4.21.4) precursor (0.50 methanol,...	43	0.98	2
NRL:1TGT trypsin (EC 3.4.21.4) precursor (173 degrees K,...	43	0.98	2
NRL:1TPAE trypsin (EC 3.4.21.4) (with basic proteinase i...	43	0.98	2
NRL:2PCN trypsin (EC 3.4.21.4) (orthorhombic, 2.4 M ammo...	43	0.98	2
NRL:3PTB trypsin (EC 3.4.21.4) beta (with benzamidine, p...	43	0.98	2
NRL:3PTN trypsin (EC 3.4.21.4) (trigonal, 2.4 M ammonium...	43	0.98	2
NRL:1TABE trypsin (EC 3.4.21.4) (with Bowman-Birk inhibi...	43	0.98	2
NRL:1TGSZ trypsin (EC 3.4.21.4) precursor (with pancreat...	43	0.98	2
NRL:1LAP2 leucyl aminopeptidase (EC 3.4.11.1) cytosolic,...	49	0.993	1
NRL:2TGD trypsin (EC 3.4.21.4) precursor (diisopropylpho...	42	0.996	2
NRL:1TGN trypsin (EC 3.4.21.4) precursor - bovine	42	0.996	2
NRL:421P H-ras p21 protein mutant with gly 12 replaced b...	40	0.998	2

```
>NRL:3ADK adenylate kinase (EC 2.7.4.3) - pig
  Length = 194

 Score = 52 (24.1 bits), Expect = 1.6, P = 0.79
 Identities = 11/29 (37%), Positives = 16/29 (55%)

Query:      5 IFMVGARGCGKTTVGRELARALGYEFVDT 33
              IF+VG  G GK T   ++ +  GY  + T
Sbjct:     11 IFVVGGPGSGKGTQCEKIVQKYGYTHLST 39
```

```
>NRL:2HMGB hemagglutinin HA2 chain (bromelain digested with mutant (G146D) HA1
  chains), chain B - influenza A virus
  Length = 175

 Score = 43 (19.9 bits), Expect = 2.8, Sum P(2) = 0.94
 Identities = 7/12 (58%), Positives = 9/12 (75%)

Query:     52 EGWPGFRRRESE 63
              +GW GFR + SR
Sbjct:     19 DGWYGFRHQNSR 30

 Score = 33 (15.3 bits), Expect = 2.8, Sum P(2) = 0.94
 Identities = 8/22 (36%), Positives = 11/22 (50%)

Query:    131 EMEAVLRREALYQDVAHYVVD 152
              ++E    E E   QD+  YV D
Sbjct:     65 QIEKEFSEVEGRIQDLEKYVED 86


>NRL:2HMGD hemagglutinin HA2 chain (bromelain digested with mutant (G146D) HA1
  chains), chain D - influenza A virus
  Length = 175

 Score = 43 (19.9 bits), Expect = 2.8, Sum P(2) = 0.94
 Identities = 7/12 (58%), Positives = 9/12 (75%)

Query:     52 EGWPGFRRRESE 63
              +GW GFR + SE
Sbjct:     19 DGWYGFRHQNSE 30

 Score = 33 (15.3 bits), Expect = 2.8, Sum P(2) = 0.94
 Identities = 8/22 (36%), Positives = 11/22 (50%)

Query:    131 EMEAVLRREALYQDVAHYVVD 152
              ++E    E E   QD+  YV D
Sbjct:     65 QIEKEFSEVEGRIQDLEKYVED 86


>NRL:2HMGF hemagglutinin HA2 chain (bromelain digested with mutant (G146D) HA1
  chains), chain F - influenza A virus
  Length = 175

 Score = 43 (19.9 bits), Expect = 2.8, Sum P(2) = 0.94
 Identities = 7/12 (58%), Positives = 9/12 (75%)

Query:     52 EGWPGFRRRESE 63
              +GW GFR + SE
Sbjct:     19 DGWYGFRHQNSE 30

 Score = 33 (15.3 bits), Expect = 2.8, Sum P(2) = 0.94
 Identities = 8/22 (36%), Positives = 11/22 (50%)

Query:    131 EMEAVLRREALYQDVAHYVVD 152
              ++E    E E   QD+  YV D
Sbjct:     65 QIEKEFSEVEGRIQDLEKYVED 86


>NRL:3HMGB hemagglutinin HA2 chain (bromelain digested with mutant (L226Q) HA1
  chains), chain B - influenza A virus
  Length = 175

 Score = 43 (19.9 bits), Expect = 2.8, Sum P(2) = 0.94
 Identities = 7/12 (58%), Positives = 9/12 (75%)

Query:     52 EGWPGFRRRESE 63
              +GW GFR + SE
Sbjct:     19 DGWYGFRHQNSE 30

 Score = 33 (15.3 bits), Expect = 2.8, Sum P(2) = 0.94
 Identities = 8/22 (36%), Positives = 11/22 (50%)

Query:    131 EMEAVLRREALYQDVAHYVVD 152
              ++E    E E   QD+  YV D
Sbjct:     65 QIEKEFSEVEGRIQDLEKYVED 86


>NRL:3HMGD hemagglutinin HA2 chain (bromelain digested with mutant (L226Q) HA1
  chains), chain D - influenza A virus
  Length = 175

 Score = 43 (19.9 bits), Expect = 2.8, Sum P(2) = 0.94
 Identities = 7/12 (58%), Positives = 9/12 (75%)

Query:     52 EGWPGFRRRESE 63
              +GW GFR + SE
Sbjct:     19 DGWYGFRHQNSE 30
```

```
 Score = 33 (15.3 bits), Expect = 2.8, Sum P(2) = 0.94
 Identities = 8/22 (36%), Positives = 11/22 (50%)

Query:   131 EMEAVLRREREALYQDVAHYVVD 152
             ++E     E E   QD+  YV D
Sbjct:    65 QIEKEFSEVEGRIQDLEKYVED 86


>NRL:3HMGF hemagglutinin HA2 chain (bromelain digested with mutant (L226Q) HA1
  chains), chain F - influenza A virus
  Length = 175

 Score = 43 (19.9 bits), Expect = 2.8, Sum P(2) = 0.94
 Identities = 7/12 (58%), Positives = 9/12 (75%)

Query:    52 EGWPGFRRRESE 63
             +GW GFR + SE
Sbjct:    19 DGWYGFRHQNSE 30

 Score = 33 (15.3 bits), Expect = 2.8, Sum P(2) = 0.94
 Identities = 8/22 (36%), Positives = 11/22 (50%)

Query:   131 EMEAVLRREREALYQDVAHYVVD 152
             ++E     E E   QD+  YV D
Sbjct:    65 QIEKEFSEVEGRIQDLEKYVED 86


>NRL:4HMOB hemagglutinin HA2 chain (bromelain digested with mutant (L226Q) HA1
  chains, sialic acid), chain B - influenza A virus
  Length = 175

 Score = 43 (19.9 bits), Expect = 2.8, Sum P(2) = 0.94
 Identities = 7/12 (58%), Positives = 9/12 (75%)

Query:    52 EGWPGFRRRESE 63
             +GW GFR + SE
Sbjct:    19 DGWYGFRHQNSE 30

 Score = 33 (15.3 bits), Expect = 2.8, Sum P(2) = 0.94
 Identities = 8/22 (36%), Positives = 11/22 (50%)

Query:   131 EMEAVLRREREALYQDVAHYVVD 152
             ++E     E E   QD+  YV D
Sbjct:    65 QIEKEFSEVEGRIQDLEKYVED 86


>NRL:4HMOD hemagglutinin HA2 chain (bromelain digested with mutant (L226Q) HA1
  chains, sialic acid), chain D - influenza A virus
  Length = 175

 Score = 43 (19.9 bits), Expect = 2.8, Sum P(2) = 0.94
 Identities = 7/12 (58%), Positives = 9/12 (75%)

Query:    52 EGWPGFRRRESE 63
             +GW GFR + SE
Sbjct:    19 DGWYGFRHQNSE 30

 Score = 33 (15.3 bits), Expect = 2.8, Sum P(2) = 0.94
 Identities = 8/22 (36%), Positives = 11/22 (50%)

Query:   131 EMEAVLRREREALYQDVAHYVVD 152
             ++E     E E   QD+  YV D
Sbjct:    65 QIEKEFSEVEGRIQDLEKYVED 86


>NRL:4HMOF hemagglutinin HA2 chain (bromelain digested with mutant (L226Q) HA1
  chains, sialic acid), chain F - influenza A virus
  Length = 175

 Score = 43 (19.9 bits), Expect = 2.8, Sum P(2) = 0.94
 Identities = 7/12 (58%), Positives = 9/12 (75%)

Query:    52 EGWPGFRRRESE 63
             +GW GFR + SE
Sbjct:    19 DGWYGFRHQNSE 30

 Score = 33 (15.3 bits), Expect = 2.8, Sum P(2) = 0.94
 Identities = 8/22 (36%), Positives = 11/22 (50%)

Query:   131 EMEAVLRREREALYQDVAHYVVD 152
             ++E     E E   QD+  YV D
Sbjct:    65 QIEKEFSEVEGRIQDLEKYVED 86
```

```
>NRL:5HMGB hemagglutinin HA2 chain mutant (D112G) (bromelain digested with
  sialic acid), chain B - influenza virus
  Length = 175

 Score = 43 (19.9 bits), Expect = 2.8, Sum P(2) = 0.94
 Identities = 7/12 (58%), Positives = 9/12 (75%)

Query:     52 EGWPGFRRRESE 63
              +GW GFR + SE
Sbjct:     19 DGWYGFRHQNSE 30

 Score = 33 (15.3 bits), Expect = 2.8, Sum P(2) = 0.94
 Identities = 8/22 (36%), Positives = 11/22 (50%)

Query:    131 EMEAVLREREALYQDVAHYVVD 152
              ++R     E E   QD+  YV D
Sbjct:     65 QIEKEFSEVEGRIQDLEKYVED 86


>NRL:5HMGD hemagglutinin HA2 chain mutant (D112G) (bromelain digested with
  sialic acid), chain D - influenza virus
  Length = 175

 Score = 43 (19.9 bits), Expect = 2.8, Sum P(2) = 0.94
 Identities = 7/12 (58%), Positives = 9/12 (75%)

Query:     52 EGWPGFRRRESE 63
              +GW GFR + SE
Sbjct:     19 DGWYGFRHQNSE 30

 Score = 33 (15.3 bits), Expect = 2.8, Sum P(2) = 0.94
 Identities = 8/22 (36%), Positives = 11/22 (50%)

Query:    131 EMEAVLREREALYQDVAHYVVD 152
              ++E     E E   QD+  YV D
Sbjct:     65 QIEKEFSEVEGRIQDLEKYVED 86


>NRL:5HMGF hemagglutinin HA2 chain mutant (D112G) (bromelain digested with
  sialic acid), chain F - influenza virus
  Length = 175

 Score = 43 (19.9 bits), Expect = 2.8, Sum P(2) = 0.94
 Identities = 7/12 (58%), Positives = 9/12 (75%)

Query:     52 EGWPGFRRRESE 63
              +GW GFR + SE
Sbjct:     19 DGWYGFRHQNSE 30

 Score = 33 (15.3 bits), Expect = 2.8, Sum P(2) = 0.94
 Identities = 8/22 (36%), Positives = 11/22 (50%)

Query:    131 EMEAVLREREALYQDVAHYVVD 152
              ++E     E E   QD+  YV D
Sbjct:     65 QIEKEFSEVEGRIQDLEKYVED 86


>NRL:1TLD trypsin (EC 3.4.21.4) (orthorhombic, pH 5.3) - bovine
  Length = 223

 Score = 43 (19.9 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 9/24 (37%), Positives = 13/24 (54%)

Query:      7 MVGARGCGKTTVGRELARALGYEF 30
              +VG   CG  TV  +++   GY F
Sbjct:      1 IVGGYTCGANTVPYQVSLNSGYHF 24

 Score = 33 (15.3 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 6/17 (35%), Positives = 12/17 (70%)

Query:     83 LLEQNRQFMRAHGTVVY 99
              ++E N QF- A  ++V+
Sbjct:     57 VVEGNEQFISASKSIVH 73


>NRL:1NTP trypsin (EC 3.4.21.4) (isopropylphosphorylated) - bovine
  Length = 223

 Score = 43 (19.9 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 9/24 (37%), Positives = 13/24 (54%)

Query:      7 MVGARGCGKTTVGRELARALGYEF 30
              -VG   CG  TV  +++   GY F
Sbjct:      1 IVGGYTCGANTVPYQVSLNSGYHF 24
```

```
 Score = 33 (15.3 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 6/17 (35%), Positives = 12/17 (70%)

Query:     83 LLEQNRQFMRAHGTVVY 99
              ++E N QF+ A  ++V+
Sbjct:     57 VVEGNEQFISASKSIVH 73


>NRL:2PTCE trypsin (EC 3.4.21.4) beta (with basic proteinase inhibitor), chain
  E - bovine
  Length = 223

 Score = 43 (19.9 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 9/24 (37%), Positives = 13/24 (54%)

Query:      7 MVGARCGKTTVGRELARALGYEF 30
              +VG   CG  TV  +++   GY F
Sbjct:      1 IVGGYTCGANTVPYQVSLNSGYHF 24

 Score = 33 (15.3 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 6/17 (35%), Positives = 12/17 (70%)

Query:     83 LLEQNRQFMRAHGTVVY 99
              ++E N QF+ A  ++V+
Sbjct:     57 VVEGNEQFISASKSIVH 73


>NRL:2TGA trypsin (EC 3.4.21.4) precursor (2.4 M magnesium sulfate) - bovine
  Length = 223

 Score = 43 (19.9 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 9/24 (37%), Positives = 13/24 (54%)

Query:      7 MVGARCGKTTVGRELARALGYEF 30
              +VG   CG  TV  +++   GY F
Sbjct:      1 IVGGYTCGANTVPYQVSLNSGYHF 24

 Score = 33 (15.3 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 6/17 (35%), Positives = 12/17 (70%)

Query:     83 LLEQNRQFMRAHGTVVY 99
              ++E N QF+ A  ++V+
Sbjct:     57 VVEGNEQFISASKSIVH 73


>NRL:2TGPZ trypsin (EC 3.4.21.4) precursor (with basic proteinase inhibitor),
  chain Z - bovine
  Length = 223

 Score = 43 (19.9 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 9/24 (37%), Positives = 13/24 (54%)

Query:      7 MVGARCGKTTVGRELARALGYEF 30
              +VG   CG  TV  +++   GY F
Sbjct:      1 IVGGYTCGANTVPYQVSLNSGYHF 24

 Score = 33 (15.3 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 6/17 (35%), Positives = 12/17 (70%)

Query:     83 LLEQNRQFMRAHGTVVY 99
              ++E N QF+ A  ++V+
Sbjct:     57 VVEGNEQFISASKSIVH 73


>NRL:2TGT trypsin (EC 3.4.21.4) precursor (103 degrees K, 0.70 methanol, 0.30
  water)  bovine
  Length = 223

 Score = 43 (19.9 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 9/24 (37%), Positives = 13/24 (54%)

Query:      7 MVGARCGKTTVGRELARALGYEF 30
              +VG   CG  TV  +++   GY F
Sbjct:      1 IVGGYTCGANTVPYQVSLNSGYHF 24

 Score = 33 (15.3 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 6/17 (35%), Positives = 12/17 (70%)

Query:     83 LLEQNRQFMRAHGTVVY 99
              ++E N QF+ A  ++V+
Sbjct:     57 VVEGNEQFISASKSIVH 73
```

```
>NRL:3TPIZ trypsin (EC 3.4.21.4) precursor (with basic proteinase inhibitor,
  Ile-Val), chain Z - bovine
  Length = 223

 Score = 43 (19.9 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 9/24 (37%), Positives = 13/24 (54%)

Query:      7 MVGARGCGKTTVGRELARALGYEF 30
              +VG   CG  TV  +++   GY F
Sbjct:      1 IVGGYTCGANTVPYQVSLNSGYHF 24

 Score = 33 (15.3 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 6/17 (35%), Positives = 12/17 (70%)

Query:     83 LLEQNRQFMRAHGTVVY 99
              ++E N QF+ A  ++V+
Sbjct:     57 VVEGNEQFISASKSIVH 73


>NRL:4PTP trypsin (EC 3.4.21.4) beta (diisopropylphosphorylated) - bovine
  Length = 223

 Score = 43 (19.9 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 9/24 (37%), Positives = 13/24 (54%)

Query:      7 MVGARGCGKTTVGRELARALGYEF 30
              +VG   CG  TV  +++   GY F
Sbjct:      1 IVGGYTCGANTVPYQVSLNSGYHF 24

 Score = 33 (15.3 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 6/17 (35%), Positives = 12/17 (70%)

Query:     83 LLEQNRQFMRAHGTVVY 99
              ++E N QF+ A  ++V+
Sbjct:     57 VVEGNEQFISASKSIVH 73


>NRL:4TPIZ trypsin (EC 3.4.21.4) precursor (with mutant basic proteinase
  inhibitor, Val-Val), chain Z - bovine
  Length = 223

 Score = 43 (19.9 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 9/24 (37%), Positives = 13/24 (54%)

Query:      7 MVGARGCGKTTVGRELARALGYEF 30
              +VG   CG  TV  +++   GY F
Sbjct:      1 IVGGYTCGANTVPYQVSLNSGYHF 24

 Score = 33 (15.3 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 6/17 (35%), Positives = 12/17 (70%)

Query:     83 LLEQNRQFMRAHGTVVY 99
              ++E N QF+ A  ++V+
Sbjct:     57 VVEGNEQFISASKSIVH 73


>NRL:2OLDE trypsin (EC 3.4.21.4) (with mutant streptomyces subtilisin
  inhibitor), chain E - bovine
  Length = 223

 Score = 43 (19.9 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 9/24 (37%), Positives = 13/24 (54%)

Query:      7 MVGARGCGKTTVGRELARALGYEF 30
              +VG   CG  TV  +++   GY F
Sbjct:      1 IVGGYTCGANTVPYQVSLNSGYHF 24

 Score = 33 (15.3 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 6/17 (35%), Positives = 12/17 (70%)

Query:     83 LLEQNRQFMRAHGTVVY 99
              ++E N QF+ A  ++V+
Sbjct:     57 VVEGNEQFISASKSIVH 73


>NRL:1GBT trypsin (EC 3.4.21.4) beta (Ser 195 guanidinobenzoylated, pH 3.5) -
  bovine
  Length = 223

 Score = 43 (19.9 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 9/24 (37%), Positives = 13/24 (54%)

Query:      7 MVGARGCGKTTVGRELARALGYEF 30
              +VG   CG  TV  +++   GY F
Sbjct:      1 IVGGYTCGANTVPYQVSLNSGYHF 24
```

```
 Score = 33 (15.3 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 6/17 (35%), Positives = 12/17 (70%)

Query:     83 LLEQNRQFMRAHGTVVY 99
              ++E N QF+ A  ++V+
Sbjct:     57 VVEGNEQFISASKSIVH 73


>NRL:1PPEE trypsin (EC 3.4.21.4) (with winter squash trypsin inhibitor), chain
  E - bovine
  Length = 223

 Score = 43 (19.9 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 9/24 (37%), Positives = 13/24 (54%)

Query:      7 MVGARGCGKTTVGRELARALGYEF 30
              +VG   CG  TV  +++   GY F
Sbjct:      1 IVGGYTCGANTVPYQVSLNSGYHF 24

 Score = 33 (15.3 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 6/17 (35%), Positives = 12/17 (70%)

Query:     83 LLEQNRQFMRAHGTVVY 99
              ++E N QF+ A  ++V+
Sbjct:     57 VVEGNEQFISASKSIVH 73


>NRL:1PPHZ Trypsin (EC 3.4.21.4) complex with noncovalently bound 3-tapap,
  chain E - bovine
  Length = 223

 Score = 43 (19.9 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 9/24 (37%), Positives = 13/24 (54%)

Query:      7 MVGARGCGKTTVGRELARALGYEF 30
              +VG   CG  TV  +++   GY F
Sbjct:      1 IVGGYTCGANTVPYQVSLNSGYHF 24

 Score = 33 (15.3 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 6/17 (35%), Positives = 12/17 (70%)

Query:     83 LLEQNRQFMRAHGTVVY 99
              ++E N QF+ A  ++V+
Sbjct:     57 VVEGNEQFISASKSIVH 73


>NRL:1TPO trypsin (EC 3.4.21.4) (orthorhombic, pH 5.0) - bovine
  Length = 223

 Score = 43 (19.9 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 9/24 (37%), Positives = 13/24 (54%)

Query:      7 MVGARGCGKTTVGRELARALGYEF 30
              +VG   CG  TV  +++   GY F
Sbjct:      1 IVGGYTCGANTVPYQVSLNSGYHF 24

 Score = 33 (15.3 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 6/17 (35%), Positives = 12/17 (70%)

Query:     83 LLEQNRQFMRAHGTVVY 99
              ++E N QF+ A  ++V+
Sbjct:     57 VVEGNEQFISASKSIVH 73


>NRL:1CPP trypsin (EC 3.4.21.4) (with p-amidino-phenyl-pyruvate) - bovine
  Length = 223

 Score = 43 (19.9 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 9/24 (37%), Positives = 13/24 (54%)

Query:      7 MVGARGCGKTTVGRELARALGYEF 30
              +VG   CG  TV  +++   GY F
Sbjct:      1 IVGGYTCGANTVPYQVSLNSGYHF 24

 Score = 33 (15.3 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 6/17 (35%), Positives = 12/17 (70%)

Query:     83 LLEQNRQFMRAHGTVVY 99
              ++E N QF+ A  ++V+
Sbjct:     57 VVEGNEQFISASKSIVH 73
```

```
>NRL:1TGB trypsin (EC 3.4.21.4) precursor (with Ca, PEG) - bovine
  Length = 223

 Score = 43 (19.9 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 9/24 (37%), Positives = 13/24 (54%)

Query:      7 MVGARCCGKTTVGRELARALGYEF 30
              +VG   CG  TV  +++   GY F
Sbjct:      1 IVGGYTCGANTVPYQVSLNSGYHF 24

 Score = 33 (15.3 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 6/17 (35%), Positives = 12/17 (70%)

Query:     83 LLEQNRQFMRAHGTVVY 99
              ++E N QF+ A  ++V+
Sbjct:     57 VVEGNEQFISASKSIVH 73


>NRL:1TGC trypsin (EC 3.4.21.4) precursor (0.50 methanol, 0.50 water) - bovine
  Length = 223

 Score = 43 (19.9 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 9/24 (37%), Positives = 13/24 (54%)

Query:      7 MVGARCCGKTTVGRELARALGYEF 30
              +VG   CG  TV  +++   GY F
Sbjct:      1 IVGGYTCGANTVPYQVSLNSGYHF 24

 Score = 33 (15.3 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 6/17 (35%), Positives = 12/17 (70%)

Query:     83 LLEQNRQFMRAHGTVVY 99
              ++E N QF+ A  ++V+
Sbjct:     57 VVEGNEQFISASKSIVH 73


>NRL:1TGT trypsin (EC 3.4.21.4) precursor (173 degrees K, 0.70 methanol, 0.30
  water) - bovine
  Length = 223

 Score = 43 (19.9 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 9/24 (37%), Positives = 13/24 (54%)

Query:      7 MVGARCCGKTTVGRELARALGYEF 30
              +VG   CG  TV  +++   GY F
Sbjct:      1 IVGGYTCGANTVPYQVSLNSGYHF 24

 Score = 33 (15.3 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 6/17 (35%), Positives = 12/17 (70%)

Query:     83 LLEQNRQFMRAHGTVVY 99
              ++E N QF+ A  ++V+
Sbjct:     57 VVEGNEQFISASKSIVH 73


>NRL:1TPAE trypsin (EC 3.4.21.4) (with basic proteinase inhibitor), chain E -
  bovine
  Length = 223

 Score = 43 (19.9 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 9/24 (37%), Positives = 13/24 (54%)

Query:      7 MVGARCCGKTTVGRELARALGYEF 30
              +VG   CG  TV  +++   GY F
Sbjct:      1 IVGGYTCGANTVPYQVSLNSGYHF 24

 Score = 33 (15.3 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 6/17 (35%), Positives = 12/17 (70%)

Query:     83 LLEQNRQFMRAHGTVVY 99
              ++E N QF+ A  ++V+
Sbjct:     57 VVEGNEQFISASKSIVH 73


>NRL:2PTN trypsin (EC 3.4.21.4) (orthorhombic, 2.4 M ammonium sulfate) - bovine
  Length = 223

 Score = 43 (19.9 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 9/24 (37%), Positives = 13/24 (54%)

Query:      7 MVGARCCGKTTVGRELARALGYEF 30
              +VG   CG  TV  +++   GY F
Sbjct:      1 IVGGYTCGANTVPYQVSLNSGYHF 24
```

```
 Score = 33 (15.3 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 6/17 (35%), Positives = 12/17 (70%)

Query:     83 LLEQNRQFMRAHGTVVY 99
              ++E N QF+ A  ++V+
Sbjct:     57 VVEGNEQFISASKSIVH 73


>NRL:3PTB trypsin (EC 3.4.21.4) beta (with benzamidine, pH 7) - bovine
  Length = 223

 Score = 43 (19.9 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 9/24 (37%), Positives = 13/24 (54%)

Query:      7 MVGARGCGKTTVGRELARALGYEF 30
              +VG   CG  TV  +++   GY F
Sbjct:      1 IVGGYTCGANTVPYQVSLNSGYHF 24

 Score = 33 (15.3 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 6/17 (35%), Positives = 12/17 (70%)

Query:     83 LLEQNRQFMRAHGTVVY 99
              ++E N QF+ A  ++V+
Sbjct:     57 VVEGNEQFISASKSIVH 73


>NRL:3PTN trypsin (EC 3.4.21.4) (trigonal, 2.4 M ammonium sulfate) - bovine
  Length = 223

 Score = 43 (19.9 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 9/24 (37%), Positives = 13/24 (54%)

Query:      7 MVGARGCGKTTVGRELARALGYEF 30
              +VG   CG  TV  +++   GY F
Sbjct:      1 IVGGYTCGANTVPYQVSLNSGYHF 24

 Score = 33 (15.3 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 6/17 (35%), Positives = 12/17 (70%)

Query:     83 LLEQNRQFMRAHGTVVY 99
              ++E N QF+ A  ++V+
Sbjct:     57 VVEGNEQFISASKSIVH 73


>NRL:1TABE trypsin (EC 3.4.21.4) (with Bowman-Birk inhibitor), chain E - bovine
  Length = 223

 Score = 43 (19.9 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 9/24 (37%), Positives = 13/24 (54%)

Query:      7 MVGARGCGKTTVGRELARALGYEF 30
              +VG   CG  TV  +++   GY F
Sbjct:      1 IVGGYTCGANTVPYQVSLNSGYHF 24

 Score = 33 (15.3 bits), Expect = 4.0, Sum P(2) = 0.98
 Identities = 6/17 (35%), Positives = 12/17 (70%)

Query:     83 LLEQNRQFMRAHGTVVY 99
              ++E N QF+ A  ++V+
Sbjct:     57 VVEGNEQFISASKSIVH 73


>NRL:1TGSZ trypsin (EC 3.4.21.4) precursor (with pancreatic secretory trypsin
  inhibitor), chain Z - bovine
  Length = 225

 Score = 43 (19.9 bits), Expect = 4.1, Sum P(2) = 0.98
 Identities = 9/24 (37%), Positives = 13/24 (54%)

Query:      7 MVGARGCGKTTVGRELARALGYEF 30
              +VG   CG  TV  +++   GY F
Sbjct:      3 IVGGYTCGANTVPYQVSLNSGYHF 26

 Score = 33 (15.3 bits), Expect = 4.1, Sum P(2) = 0.98
 Identities = 6/17 (35%), Positives = 12/17 (70%)

Query:     83 LLEQNRQFMRAHGTVVY 99
              ++E N QF+ A  ++V+
Sbjct:     59 VVEGNEQFISASKSIVH 75
```

```
>NRL:1LAP2 leucyl aminopeptidase (EC 3.4.11.1) cytosolic, fragment 2 - bovine
  Length = 470

 Score = 49 (22.7 bits), Expect = 5.0, P = 0.99
 Identities = 11/35 (31%), Positives = 17/35 (48%)

Query:   100 LFAPAEELALRLQASPQAHQRPTLTGRPIAEEMEA 134
             LFA  + LA RL  +P     PT     + E +++
Sbjct:   145 LFASGQNLARRLMETPANEMTPTKFAEIVEENLKS 179


>NRL:2TGD trypsin (EC 3.4.21.4) precursor (diisopropylphosphorylated) - bovine
  Length = 222

 Score = 42 (19.4 bits), Expect = 5.4, Sum P(2) = 1.0
 Identities = 9/23 (39%), Positives = 12/23 (52%)

Query:     8 VGARGCGKTTVGRELARALGYEF 30
             VG   CG  TV  +++   GY F
Sbjct:     1 VGGYTCGANTVPYQVSLNSGYHF 23

 Score = 33 (15.3 bits), Expect = 5.4, Sum P(2) = 1.0
 Identities = 6/17 (35%), Positives = 12/17 (70%)

Query:    83 LLEQNRQFMRAHGTVVY 99
             ++E N QF+ A  ++V+
Sbjct:    56 VVEGNEQFISASKSIVH 72


>NRL:1TGN trypsin (EC 3.4.21.4) precursor - bovine
  Length = 222

 Score = 42 (19.4 bits), Expect = 5.4, Sum P(2) = 1.0
 Identities = 9/23 (39%), Positives = 12/23 (52%)

Query:     8 VGARGCGKTTVGRELARALGYEF 30
             VG   CG  TV  +++   GY F
Sbjct:     1 VGGYTCGANTVPYQVSLNSGYHF 23

 Score = 33 (15.3 bits), Expect = 5.4, Sum P(2) = 1.0
 Identities = 6/17 (35%), Positives = 12/17 (70%)

Query:    83 LLEQNRQFMRAHGTVVY 99
             ++E N QF+ A  ++V+
Sbjct:    56 VVEGNEQFISASKSIVH 72


>NRL:421P H ras p21 protein mutant with gly 12 replaced by arg (g12r) Complex
  with guanosine-5'-[b,g-imido] triphosphate - human
  Length = 166

 Score =  40 (18.5 bits), Expect = 6.5, Sum P(2) = 1.0
 Identities = 8/18 (44%), Positives = 13/18 (72%)

Query:     5 IPMVGARGCGKTTVGREL 22
             + -VGARG GK+ +  +L
Sbjct:     6 LVVVGARGVGKSALTIQL 23

 Score = 33 (15.3 bits), Expect = 6.5, Sum P(2) = 1.0
 Identities = 5/14 (35%), Positives = 11/14 (78%)

Query:    20 RELARALGYEPVDT 33
             ++LAR+ G  -++T
Sbjct:   131 QDLARSYGIPYIET 144

Parameters:
  V=250     -ctxfactor=1.00
  B=100     E=10

  Query                          -----  As Used  -----     -----  Computed  ----
  Frame  MatID  Matrix name      Lambda     K        H      Lambda     K        H
   +0      0    BLOSUM62        0.321   0.133    0.382     same     same     same

  Query
  Frame  MatID  Length  Eff.Length    E     S W   T  X     E2  S2
   +0      0     173        173      10.  48 3   11 22    0.19 32

Statistics:
  Query          Expected        Observed          HSPs       HSPs
  Frame  MatID  High Score      High Score       Reportable  Reported
   +0      0    52 (24.1 bits)  52 (24.1 bits)      146        146

  Query          Neighborhd  Word       Excluded    Failed    Successful  Overlaps
  Frame  MatID    Words      Hits         Hits     Extensions  Extensions  Excluded
   +0      0      3103      321144       60004      260298       842          0
```

b) results for e4-P08329

```
                                                                  Smallest
                                                                    Sum
                                                          High   Probability
Sequences producing High-scoring Segment Pairs:           Score    P(N)       N

 NRL:1ZTA amino acid biosynthesis regulatory protein (GCN...    40  0.97       1
 NRL:2ZTAA amino acid biosynthesis regulatory protein (GC...    40  0.98       1
 NRL:2ZTAB amino acid biosynthesis regulatory protein (GC...    40  0.98       1


>NRL:1ZTA amino acid biosynthesis regulatory protein (GCN4 leucine zipper
  monomer) (NMR, 20 structures) - yeast (Saccharomyces cerevisiae) -
  Length = 35

 Score = 40 (18.3 bits), Expect = 3.4, P = 0.97
 Identities = 9/24 (37%), Positives = 15/24 (62%)

Query:   126 KPLSEEVQEVLEERDALYREVAHL 149
             K L ++V+E+L +    L  EVA +
Sbjct:     5 KQLEDKVEELLSKNYHLENEVARL 28


>NRL:2ZTAA amino acid biosynthesis regulatory protein (GCN4 leucine zipper
  dimer), chain A - yeast (Saccharomyces cerevisiae)
  Length = 31

 Score = 40 (18.3 bits), Expect = 3.8, P = 0.98
 Identities = 9/24 (37%), Positives = 15/24 (62%)

Query:   126 KPLSEEVQEVLEERDALYREVAHL 149
             K L ++V+E+L +    L  EVA +
Sbjct:     3 KQLEDKVEELLSKNYHLENEVARL 26


>NRL:2ZTAB amino acid biosynthesis regulatory protein (GCN4 leucine zipper
  dimer), chain B - yeast (Saccharomyces cerevisiae)
  Length = 31

 Score = 40 (18.3 bits), Expect = 3.8, P = 0.98
 Identities = 9/24 (37%), Positives = 15/24 (62%)

Query:   126 KPLSEEVQEVLEERDALYREVAHL 149
             K L ++V+E+L +    L  EVA +
Sbjct:     3 KQLEDKVEELLSKNYHLENEVARL 26


Parameters:
  V=250
  B=100

  -ctxfactor=1.00
  E=10

  Query                          -----  As Used  ------      ----  Computed  ----
  Frame  MatID  Matrix name      Lambda     K        H       Lambda     K        H
   +0      0    BLOSUM62         0.317   0.133    0.372     same     same     same

  Query
  Frame  MatID  Length  Eff.Length     E     S W   T  X     E2  S2
   +0      0     174        174      10.  48 3   11 22    0.21 32


Statistics:
  Query          Expected        Observed          HSPs       HSPs
  Frame  MatID  High Score      High Score      Reportable  Reported
   +0      0    52 (23.8 bits)  49 (22.4 bits)      16         16

  Query          Neighborhd  Word       Excluded    Failed      Successful  Overlaps
  Frame  MatID    Words      Hits         Hits     Extensions  Extensions  Excluded
   +0      0      3465      324276       62306      261115        765         0
```

Appendix C The scoring matrices used when running ALIEN and PILEUP

The scoring matrices used in running ALIEN [283] and PILEUP [178]. The matrices are: (i) log odds form of the mutation data matrix for 250 PAMs [216]; (ii) a log odds form of the mutation data matrix for 250 PAMs, modified so Y:R=0, W:F=2, W:Y=2 and Y:Y=12; (iii) a log odds form of the mutation data matrix for 250 PAMs, modified so there are no negative scores and "alters some scores to arguably more reasonable levels" (quote taken from the user documentation supplied with ALIEN) [283]; (iv) a log odds form of the mutation data matrix for 250 PAMs as rescaled and used by Gribskov and Burgess [291]; (v) a structure/genetics matrix [208]; and (vi) a structure/genetics matrix [209]. The default matrix for ALIEN is (iii) and for PILEUP is (iv).

```
i)
PAM 250 Matrix
C 12
S  0  2
T -2  1  3
P -3  1  0  6
A -2  1  1  1  2
G -3  1  0 -1  1  5
N -4  1  0 -1  0  0  2
D -5  0  0 -1  0  1  2  4
E -5  0  0 -1  0  0  1  3  4
Q -5 -1 -1  0  0 -1  1  2  2  4
H -3 -1 -1  0 -1 -2  2  1  1  3  6
R -4  0 -1  0 -2 -3  0 -1 -1  1  2  6
K -5  0  0 -1 -1 -2  1  0  0  1  0  3  5
M -5 -2 -1 -2 -1 -3 -2 -3 -2 -1 -2  0  0  6
I -2 -1  0 -2 -1 -3 -2 -2 -2 -2 -2 -2 -2  2  5
L -6 -3 -2 -3 -2 -4 -3 -4 -3 -2 -2 -3 -3  4  2  6
V -2 -1  0 -1  0 -1 -2 -2 -2 -2 -2 -2 -2  2  4  2  4
F -4 -3 -3 -5 -4 -5 -4 -6 -5 -5 -2 -4 -5  0  1  2  1  9
W  0 -3 -3 -5 -3  5 -2 -4 -4 -4  0 -4 -4  2  1 -1 -2  7 10
Y -8 -2 -5 -6 -6 -7 -4 -7 -7 -5 -3  2 -3 -4 -5 -2 -6  0  0 17
   C  S  T  P  A  G  N  D  E  Q  H  R  K  M  I  L  V  F  W  Y
```

ii)

Modified PAM 250 Matrix (Y:R=0 W:F=2 W:Y=2 F:Y=0 Y:Y=12)

```
C 12
S  0  2
T -2  1  3
P -3  1  0  6
A -2  1  1  1  2
G -3  1  0 -1  1  5
N -4  1  0 -1  0  0  2
D -5  0  0 -1  0  1  2  4
E -5  0  0 -1  0  0  1  3  4
Q -5 -1 -1  0  0 -1  1  2  2  4
H -3 -1 -1  0 -1 -2  2  1  1  3  6
R -4  0 -1  0 -2 -3  0 -1 -1  1  2  6
K -5  0  0 -1 -1 -2  1  0  0  1  0  3  5
M -5 -2 -1 -2 -1 -3 -2 -3 -2 -1 -2  0  0  6
I -2 -1  0 -2 -1 -3 -2 -2 -2 -2 -2 -2 -2  2  5
L -6 -3 -2 -3 -2 -4 -3 -4 -3 -2 -2 -3 -3  4  2  6
V -2 -1  0 -1  0 -1 -2 -2 -2 -2 -2 -2 -2  2  4  2  4
F -4 -3 -3 -5 -4 -5 -4 -6 -5 -5 -2 -4 -5  0  1  2 -1  9
W  0 -3 -3 -5 -3 -5 -2 -4 -4 -4  0 -4 -4 -2 -1 -1 -2  2 10
Y -8 -2 -5 -6 -6 -7 -4 -7 -7 -5 -3  0 -3 -4 -5 -2 -6  0  2 12
   C  S  T  P  A  G  N  D  E  Q  H  R  K  M  I  L  V  F  W  Y
```

iii)

No negative score PAM matrix

```
C 12
S  0  2
T  0  1  3
P  0  1  0  6
A  0  1  1  1  2
G  0  1  0  0  1  5
N  0  1  0  0  0  0  2
D  0  0  0  0  0  1  2  4
E  0  0  0  0  0  0  1  3  4
Q  0  0  0  0  0  0  1  2  2  4
H  0  0  0  0  0  0  2  1  1  3  6
R  0  0  0  0  0  0  0  0  0  1  2  6
K  0  0  0  0  0  0  1  0  0  1  0  3  5
M  0  0  0  0  0  0  0  0  0  0  0  0  0  6
I  0  0  0  0  0  0  0  0  0  0  0  0  0  2  5
L  0  0  0  0  0  0  0  0  0  0  0  0  0  4  2  6
V  0  0  0  0  0  0  0  0  0  0  0  0  0  2  4  2  4
F  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  2  0  9
W  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  7 10
Y  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2  2 12
   C  S  T  P  A  G  N  D  E  Q  H  R  K  M  I  L  V  F  W  Y
```

iv)

DAYHOFF MARK II for alien.

	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	W	Y
C	15																			
S	7	15																		
T	2	3	15																	
P	1	4	3	15																
A	3	4	4	5	15															
G	2	6	4	3	7	15														
N	-3	3	2	0	2	4	15													
D	-5	2	2	1	3	7	7	15												
E	-6	2	2	1	3	5	5	10	15											
Q	-6	-1	-1	3	2	2	4	7	7	15										
H	-1	-2	-1	2	-1	-2	5	4	4	7	15									
R	-3	1	1	3	-3	-3	1	0	0	4	5	15								
K	-6	2	2	1	0	-1	4	3	3	4	1	8	15							
M	-6	-3	0	-2	0	-3	-3	-4	-2	0	-3	2	2	15						
I	2	-1	2	-2	0	-3	-3	-2	-2	-3	-3	-3	-2	6	15					
L	-8	-4	-1	-3	-1	-5	4	5	-3	-1	-2	-4	-3	13	8	15				
V	2	-3	2	1	2	2	-3	-2	-2	-2	-3	-3	-2	6	11	8	15			
F	-1	-3	-3	-7	-5	-6	-5	-10	-7	-8	-6	-5	-7	5	7	12	2	15		
W	-12	14	-6	-8	-8	-10	-3	-11	11	-6	-1	14	1	-3	-5	5	-8	13	15	
Y	10	-6	-3	-3	-3	-7	-1	-5	-5	11	3	-6	-6	-1	1	3	-1	14	11	15

v)

BACON&ANDERSON Matrix

	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	W	Y
C	9																			
S	4	9																		
T	4	8	9																	
P	2	5	4	9																
A	6	5	5	3	9															
G	4	6	5	7	5	9														
N	3	7	7	4	4	4	9													
D	3	6	6	5	4	4	6	9												
E	5	5	5	3	5	3	6	6	9											
Q	4	6	6	3	5	3	7	6	7	9										
H	5	5	6	2	6	3	5	4	6	6	9									
R	4	6	6	3	5	3	6	5	6	6	8	9								
K	4	6	6	3	5	4	6	6	6	7	7	8	9							
M	7	3	4	0	6	3	2	5	4	2	6	4	4	9						
I	5	2	3	0	4	2	1	0	2	1	3	2	1	5	9					
L	7	2	3	1	6	3	2	1	4	3	5	3	3	7	7	9				
V	5	2	3	0	4	2	1	0	2	1	3	2	1	5	9	6	9			
F	6	3	5	1	6	3	3	2	5	4	6	4	4	7	7	7	6	9		
W	5	4	5	2	4	3	4	3	4	4	6	5	4	6	5	6	5	6	9	
Y	5	5	6	3	4	4	4	3	4	4	6	5	4	5	5	5	5	6	7	9

vi)

SG SCORING SCHEME FROM FENG(J.MOL.EVOL)1985

C	6																		
S	4	6																	
T	2	5	6																
P	2	4	4	6															
A	2	5	5	5	6														
G	3	5	2	3	5	6													
N	2	5	4	2	3	3	6												
D	1	3	2	2	4	4	5	6											
E	0	3	3	3	4	4	3	5	6										
Q	1	3	3	3	3	2	3	4	4	6									
H	2	3	3	3	2	1	4	3	2	4	6								
R	2	3	3	3	2	3	2	2	2	3	4	6							
K	0	3	4	2	3	2	4	3	4	4	3	5	6						
M	2	1	3	2	2	1	1	0	1	2	1	2	2	6					
I	2	2	3	2	2	2	2	1	1	1	1	2	2	4	6				
L	2	2	2	3	2	2	1	1	1	2	3	2	2	5	5	6			
V	2	2	3	3	5	4	2	3	4	2	1	2	3	4	5	5	6		
F	3	3	1	2	2	1	1	1	0	1	2	1	0	2	4	4	4	6	
W	3	2	1	2	2	3	0	0	1	1	1	2	1	3	2	4	3	3	6
Y	3	3	2	2	2	2	3	2	1	2	3	1	1	2	3	3	3	5	3	6
	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	W	Y

Appendix D User guide for the Mix'n'Match multiple sequence alignment programs

some familiarity with the VMS operating system is assumed.

Introduction to the Mix'n'Match programs

The Mix'n'Match suite of programs optimises protein multiple sequence alignment by a process of finding Strongly Conserved Regions (SCRs) that can be used to anchor an alignment, and then aligning the regions between these anchors. The anchor regions are found by analysis of initial alignments, previously generated using one or more automatic multiple alignment programs and a variety of scoring matrices and gap penalties. The suite consists of two programs: Mix'n'MatchA, which finds the SCRs, works out the different possible alignments for each Loosely Conserved Regions (LCRs), predicts the secondary structure or reads in the predictions if they have already been carried out and works out a consensus structure prediction for each of these possible alignments; and Mix'n'MatchB, which takes the alignments which the user has chosen for each LCR and the alignments for the SCRs and joins then to produce the final alignment. The University of Wisconsin Genetics Computer Group (GCG) suite of programs is also required. Two files are input to Mix'n'MatchA. The first is a file containing all the pre-generated alignments. The alignment formats that are supported are those of ALIEN [283] and PILEUP [178]. The second input file lists the sequences present in the multiple alignment. This sequence input file can be in either NBRF [259, 297] or GCG 'file of file names' [178] formats. Examples of the above mentioned formats are shown in Appendix D. The maximum number of alignments in the input alignment file is set at 100. The maximum number of protein sequences in an alignment is set at 30, with the maximum length of any sequence set at 2000, inclusive of any gaps that may have been introduced by the alignment process. If the input alignment file contains more alignments than allowed, or if the alignments have more or longer

sequences than the maximum values allowed, then this is detected by Mix'n'MatchA when reading in the alignments and an error message is written which explains the error and also how to alter the source code so that alignments of this size are allowed. The following table shows both the default and permissible values for the interactively-ntered program parameters.

Parameters	Default	Permissible range
Number of alignments used to calculate the SCRs	15	2-number of alignments input
Number of sequence used to calculate the SCRs	10	3-number of alignments input
Minimum width of an SCR (residues)	3	3-10
Width of the screen that program information is written to (characters)	80	40-132
Width of the final output (characters)	131	40-132

Four files are output, two of which are intended to be used by the next program, Mix'n'MatchB, and two for the user to help decide which alignments to use for each Loosely Conserved Region. The two files used by Mix'n'MatchB are called 'SCR.DATA' and 'LCR.DATA'. These files contain the alignments chosen as Strongly Conserved Regions and all the different alignments for all the Loosely Conserved Regions, respectively, together with the consensus secondary structure predictions for these alignments. The two files output for the users attention are called '<NAMEOFFILE>.PARAM_DATA' and 'PRINT.ME'. '<NAMEOFFILE>.PARAM_DATA' is a text file containing a list of the parameters used by each of the input alignments. 'PRINT.ME' is a text file which contains all the different possible alignments found for each of the different Loosely Conserved Regions, together with the consensus secondary structure predictions for these alignments. The user examines this latter file and chooses which alignment to use for each LCR, from the different possibilities listed. These values are used as input to Mix'n'MatchB together with the two files produced by Mix'n'MatchA for

Mix'n'MatchB. Output is a text file 'PRINT.ME2' which contains the final alignment, together with its consensus secondary structure predictions, in a form suitable for printing out. Additionally, Mix'n'MatchA will also output a 'debugging' file, named '<NAMEOFFILE>.DEBUG', that contains more information about the run, including the individual secondary structure predictions for each sequence, by each method, the positions of each of the SCRs and how the unique alignments for each LCR were found.

Instructions for use

1 Set up symbol names in the users 'login.com' file so that the programs can be run from any directory by typing the symbol names instead of the programs actual name. This only needs to be done once. If the programs are in the directory

```
[GBCXXX.PROGRAMS.ALIGNING]
```

and are called

```
MIX_N_MATCHA.EXE
MIX_N_MATCHB.EXE
```

then, to set up the symbols 'MNMA' and 'MNMB', the 'login.com' file in the users top directory should be edited to contain the lines:

```
$ MNMA :== [GBCXXX.PROGRAMS.ALIGNING]MIX_N_MATCHA.EXE
$ MNMB :== [GBCXXX.PROGRAMS.ALIGNING]MIX_N_MATCHB.EXE
```

2 Run the automatic multiple alignment programs to generate the initial alignments. A detailed guide to this is given in appendix E.

3 Concatenate all the initial alignments into one file. If the ALIEN and PILEUP alignments are in the files 'all.alien' and 'all.pileup', respectively (as produced by the example runs in Appendix E), this can be done using the following VMS commands:

3.1 Create the single file that will hold all the alignments.

```
$ create all.alignments <rtn>
cntrl-z
```

where 'cntrl-z' indicates pressing the control key and 'z' at the same time.

3.2 Append the initial alignments into this single file.

```
$ append all.alien all.alignments <rtn>
$ append all.pileup all.alignments <rtn>
```

3.3 Delete the initial alignments. This step is optional.

```
$ delete all.alien; <rtn>
$ delete all.pileup; <rtn>
```

4 Predict the secondary structures for the sequences using the UWGCG prediction programs. This step is optional because the program Mix'n'MatchA will carry out these predictions if this has not already been done. However, because running these prediction programs is a time consuming step and the slowest part of the Mix'n'MatchA program, it is recommended that the user run these prediction programs before running Mix'n'MatchA (e.g., while waiting for the automatic alignment programs to finish running). A detailed guide to running these prediction programs is given in Appendix E.

5 Initialise the UWGCG suite of programs before running the Mix'n'Match suite of programs by using the following VMS command:

```
$ GCG7 <rtn>
```

6 Run the Mix'n'MatchA multiple alignment program.

```
$ MNMA <rtn>
```

A detailed guide for using this program is presented below.

7 Print or examine the output file 'PRINT.ME' from Mix'n'MatchA.

7.1 The multiple alignment program generates a file named 'PRINT.ME' which contains all the different possible alignments found for each of the different

Loosely Conserved Regions found, together with the consensus secondary structure predictions for these alignments. In order to examine these alignments on screen, enter the following VMS command:

$ type PRINT.ME <rtn>

7.2 Alternatively, this file can be printed. The default number of characters in this file is set so that the format is suitable for printing on a line printer. To print these alignments on the system's default printer, enter the following VMS command:

$ print PRINT.ME <rtn>

To print to a printer other than the default, enter the following VMS command:

$ print/queue=*printername* PRINT.ME <rtn>

8 Run the Mix'n'MatchB multiple alignment program.

$ MNMB <rtn>

A detailed guide for using this program is presented below.

9 Print or examine the final alignment file, 'PRINT.ME2' from Mix'n'MatchB.

9.1 The file named 'PRINT.ME2' contains the final alignment, together with its predicted consensus secondary structure. In order to examine this alignments on screen, enter the following VMS command:

$ type PRINT.ME2 <rtn>

9.2 Alternatively, this file can be printed. The default number of characters in this file is set so that the format is suitable for printing on a line printer. To print this alignments on the system's default printer, enter the following VMS command:

$ print PRINT.ME2 <rtn>

To print to a printer other than the default, enter the following VMS command:

$ print/queue=*printername* PRINT.ME2 <rtn>

User guide for running the multiple alignment programs Mix'n'MatchA

First, an introductory explanation of the program is displayed on the screen.

Entering the alignment and sequence input files :

1 You will be prompted to input the name of the file containing all the initial alignments or to see the help files as follows:

Input the alignment file name or ? for more information.

1.1 Enter '? <rtn>' to go to the help section (see below). After displaying the help section you will then return to part 1 of this section.

1.2 Enter the name of the file containing all the alignments, followed by '<rtn>' to proceed to part 2 of this section.

1.3 If you enter a name of a file that does not exist, you will see displayed the following error message:

File name problem i.e. this file doesn't exist!

1.3.1 You will then be prompted again for the name of the file containing all the initial alignments or to see the help files with the following message:

Input the alignment file name or ? for more information.

1.3.2 The sequence of error messages and prompts for entry of data will continue until the name of a file that does exist or '?' is entered.

2 You will be prompted to input the name of the file containing the sequence file information or to see the help files as follows:

Input the sequence file name or ? for more information.

2.1 Enter '? <rtn>' to go to the help section (see below). After displaying the help section you will then return to part 2 of this section.

2.2 If the sequences are in NBRF format, enter the name of the file containing all the sequences, followed by '<rtn>' to proceed to the next section. If the

sequences are in UWGCG 'file of file names' format, you must add '@' as a prefix to the name of the file containing the list of sequence file names. Enter the name of the 'file of file names' file, with the prefix, followed by '<rtn>' to proceed to part 3 of this section.

2.3 If you enter a name of a file that does not exist, you will see displayed the following error message:

File name problem i.e. this file doesn't exist!

2.3.1 You will then be prompted again for the name of the file containing the sequence file information or to see the help files with the following message:

Input the sequence file name or ? for more information.

2.3.2 The sequence of error messages and prompts for entry of data will continue until the name of a file that does exist or '?' is entered.

2.4 The program then displays the following message:

Now checking that the files named in your GCG multiple sequence format file exist.

2.5 If all the files exist, the following message is displayed:

There were <number> sequences found in the GCG format file. If this is different from the number of sequences in your alignments, stop the program and check your GCG input file.

2.6 If a sequence file named in the UWGCG 'file of file names' format does not exist, the program will terminate after displaying the following error messages:

A sequence file <filename> named in the GCG format sequence file doesn't exist! Please check the file <fileoffilenames>

The program will now terminate. Please correct your GCG multiple sequences format file and start again.

2.6.1 You should then change your UWGCG 'file of file names' file to ensure that the names of the sequence files listed in it agree with the actual names of the sequence files.

Entering other program information:

1 You are then prompted to see if you have already run the UWGCG secondary structure prediction programs for the sequences with the following message:

Have you already run the GCG prediction programs? [no]:

(N.B. The value in square brackets is the default value and entering '<rtn>' will input this default value. The program will accept either upper or lower case letters and accepts abbreviations, e.g., 'y' for 'yes' and 'n' for 'no', as only the first letter of the input is checked.)

1.1 If you have not already run these programs, enter 'no <rtn>' or press '<rtn>' to accept this default value. You will then proceed to part 2 of this section.

1.2 If you have already run these programs, enter 'yes <rtn>' to proceed to part 2 of this section.

2 You are then prompted to see if you want the debugging information to be written with the following message:

Do you want debug data to be written? [no]:

2.1 If you do not wish the debugging information to be written, enter 'no <rtn>' or press '<rtn>'. You will then proceed to part 3 of this section.

2.2 If you wish the debugging information to be written, enter 'yes <rtn>' to proceed to part 3 of this section.

3 You are then prompted to see if you wish to change some of the program's variables with the following message:

Do you want to change any of the program variables? [no]:

3.1 If you do not wish to change any of the program variables, enter 'no <rtn>' or press '<rtn>'. You will then proceed to part 4 of this section.

3.2 If you do wish to change any of the program variables, enter 'yes <rtn>'. A menu detailing the variables that may be changed, along with their present values in square brackets, is then displayed. You are then prompted to enter the option number you require with the following menu:

The variables that may be altered are;
1) Screen width; [80] characters.
2) Maximum number of characters per line in output files; [131] characters.
3) Number of alignments used to calculate a SCR; [15] alignments.
4) Number of sequences used to calculate a SCR; [10] sequences.
5) Minimum width of a SCR; [3] residues.
6) Help - what do these variables do?
7) Return to running the program.
Enter a number now

3.3 Enter the number of the option you want, followed by '<rtn>' e.g., '7 <rtn>'

3.4 If the number you enter is not permissible, you will see displayed the following error message:

Number out of range - please re-enter a number

3.4.1 The sequence of error message, menu of available choices and prompt to enter a number will continue until an acceptable value is entered.

3.5 If you choose options numbers 1-5, the name of the variable you are changing and its present value is displayed, along with a prompt to enter a new value, as follows:

Changing the program variable
program variable
This is presently set at; *variable value*
Input your new value.

3.6 Enter the new variable value followed by '<rtn>'

3.7 A message is displayed showing the new value of the variable and a prompt to check that this is the correct value as follows:

Variable changed to *new variable value*
Is this correct? [yes]

3.8 If this value is wrong, enter 'no <rtn>'. You will then see redisplayed the name of the variable you are changing and its present, wrong value is displayed, along with a prompt to enter a new value, as shown in part 3.5 of this section

3.9 If this value is correct, enter 'yes <rtn>' or press '<rtn>'.

3.9.1 The program now checks to see that this value is permissible for each of the options 1 to 5. If the value is permissible, you will then return to the menu screen, as described in part 3.2 of this section. Any non-permissible values will cause the following error messages to be displayed for the options 1 to 5:

Option 1 If you alter the value of the screen width to be more than 132 or less than 40 characters, you are prompted to confirm this with the following message:

Are you sure that the screen is so wide (>132 chars) or so short (<40 chars)? [no]

a) If the screen is not so wide or narrow, enter 'no <rtn>' or press '<rtn>'. You will then be prompted to enter a new value with the messages shown in part 3.5 of this section.

b) If the screen is so wide or narrow, enter 'yes <rtn>'. You will then return to the menu screen, as described in part 3.2 of this section.

Option 2 If you alter the value of the maximum number of characters per line in output files to be more than 132 or less than 40, you are prompted to confirm this with the following message:

Are you sure that the width of the device that the final output will be to, is so wide (>132 chars) or so short (<40 chars)? [no]

a) If this value is wrong, enter 'no <rtn>' or press '<rtn>'. You will then be prompted to enter a new value with the messages shown in part 3.5 of this section.

b) If this value is correct, enter 'yes <rtn>'. You will then return to the menu screen, as described in part 3.2 of this section

Option 3 If you alter the value of the number of alignments used to calculate a SCR to be more than 100 (the maximum number of alignments allowed by the program), the following error message is displayed:

> You've chosen a number greater than the total number of alignments allowed by this program - please choose a lower number. If you do need this number then you will have to slightly alter the program. Please see the relevant section in the help.

a) If you alter the value of the number of alignments used to calculate a SCR to 1, the following error message is displayed:

> You've only chosen one alignment for the SCRs to be chosen from - this would make the entire alignment the SCR and is a mistake. Please choose again!.

b) If you alter the value of the number of alignments used to calculate a SCR to be less than 1, the following error message is displayed:

> Please try again with a sensible number.

c) After each of these error messages, you will then be prompted to enter a new value with the messages shown in part 3.5 of this section.

Option 4 If you alter the value of the number of sequences used to calculate a SCR to be more than 30 (the maximum number of sequences allowed by the program), the following error message is displayed:

> You've chosen a number greater than the total number of sequences allowed by this program - please choose a lower number. If you do need this number then you will have to slightly alter the program. Please see the relevant section in the help.

a) If you alter the value of the number of sequences used to calculate a SCR to be less than 3, the following error message is displayed:

> You've chosen less than 3 sequences for the SCR's to be worked out from. This would be a pairwise alignment and not a multiple alignment, and so is not allowed. Please choose a higher number.

b) After either of these error messages, you will then be prompted to enter a new value with the messages shown in part 3.5 of this section.

Option 5 If you alter the value of the minimum width of a SCR to be less than 3 or more than 10 residues, the following error message is displayed:

> You have chosen a width for the SCR's that is outside the permitted range of 3-10. You must choose another number. If you really do want to set a width outside this range, then you will have to alter the program.
> Change the values in the routine 'ALTVAR' after the jump label 958.

a) After this error message, you will then be prompted to enter a new value with the messages as shown in part 3.5 of this section.

3.10 If you choose option number 6, you will go to the help section (see below). After displaying the help section you will then return to part 3.2 of this section.

3.11 If you choose option number 7, you will return to running the program, proceeding to part 4 of this section.

(A worked example of changing the width of the screen, from its default value of 80 characters to a value of 132, from saying that you wish to change a value to returning to running the program, is given below. Words in italics are user-entered values.

Do you want to change any of the program variables? [no]:
yes <rtn>
The variables that may be altered are;
1) Screen width; [80] characters.
2) Maximum number of characters per line in output files; [131] characters.
3) Number of alignments used to calculate a SCR; [15] alignments.
4) Number of sequences used to calculate a SCR; [10] sequences.
5) Minimum width of a SCR; [3] residues.
6) Help - what do these variables do?
7) Return to running the program.
Enter a number now
1 <rtn>
Changing the program variable
Screen width
This is presently set at; 132
Input your new value.
132 <rtn>
Variable changed to 132
Is this correct? [yes]
yes <rtn>
The variables that may be altered are;
1) Screen width; [80] characters.
2) Maximum number of characters per line in output files; [131] characters.
3) Number of alignments used to calculate a SCR; [15] alignments.
4) Number of sequences used to calculate a SCR; [10] sequences.
5) Minimum width of a SCR; [3] residues.
6) Help - what do these variables do?
7) Return to running the program.
Enter a number now
7 <rtn>

4 You are then prompted to see if you want to change any of the values you have entered for the alignment and sequence input file section and for the other program information sections, with the following message:

Do you want to change any of the above answers? [no]

4.1 If you do not wish to change any of the above answers, enter 'no <rtn>' or press '<rtn>'. You will then proceed to the next section.

4.2 If you do wish to change any of the above answers, enter 'yes <rtn>'. You will then start again, entering information at part 1 of the 'Entering the alignment and sequence input files' section.

Explanatory messages

The program now displays messages indicating what it is doing in the following order:

1 If the sequences were entered in the NBRF format, the following message is displayed:

Reformatting the sequences from NBRF/PIR format to GCG format.

2 The program then displays the following messages:

Reading in the alignments

Finished reading the <number of alignments> alignments.

3 If a problem occurs when reading in the alignments, error messages will be displayed, as listed in the section below. Otherwise, the program continues to display progress reports as follows:

Predicting the secondary structure.

4 The program reads in the sequences so that their structure can be predicted. If a small letter is found in the sequences, this can lead to ambiguity, and the following message is displayed:

The non-capital letter <letter> has been found in the sequence <sequence name>

4.1 The following menu and prompt to enter a value is then displayed:

Enter

1) for more information/help.

2) for making this a cysteine 'C'.

3) for leaving it as the same residue type.

4) for leaving all small letters as the same residue type

4.2 Enter the number of the option you want, followed by '<rtn>' e.g., '4 <rtn>'

4.2.1 If the number you enter is not permissible, you will see displayed the following error message:

Error in input number, please reenter, range allowed is 1-4.

4.2.2 The sequence of error message, menu of available choices and prompt to enter a number will continue until an acceptable value is entered.

4.3 Enter '1 <rtn>' to go to the help section (see below). After displaying the help section you will then return to part 4.1 of this section.

4.4 Enter '2 <rtn>' to make the residue a cysteine.

4.5 Enter '3 <rtn>' to leave the residue as the same residue type.

4.6 Enter '4 <rtn>' to always leave any residues as the same type.

5 The following message is then displayed:

Converting sequences to GGBSM format and predicting their structure.

6 If the UWGCG structure prediction programs were not run before running Mix'n'Match, the following messages are displayed for each sequence in the alignment:

running PEPTIDESTRUCTURE.

running PEPPLOT.

Reading in these GCG predictions.

6.1 If the UWGCG structure prediction programs were run before running Mix'n'Match, the following message are displayed for each sequence in the alignment:

Reading in the GCG predictions.

7 The following messages are then displayed:

Calculating a consensus GGBSM 2ry structure prediction for each alignment.

Calculating a consensus CF 2ry structure prediction for each alignment.

Calculating a consensus GOR 2ry structure prediction for each alignment.

Calculating a consensus GOR2 2ry structure prediction for each alignment.

Calculating a consensus GOR3 2ry structure prediction for each alignment.

Calculating the Strongly Conserved Regions.

Calculating the Strongly Conserved Regions

1 If there are more sequences in the alignment than will be used to calculate the SCRs, the following messages will be displayed:

The number of sequences in the alignments is more than the number of sequences that are looked at in choosing the SCR's. So we will be picking, at random, which of the entered sequences will be looked at in choosing the SCR's.

<number> out of the <number> sequences will be used for SCR calculation

1.1 If there are not more sequences in the alignment than will be used to calculate the SCRs, the following message will be displayed:

All the sequences in the alignments are used for picking the SCR's.

2 The names of the sequences used to calculate the SCR's are then displayed.

3 If there are less alignments in the alignment input file than the number of alignments that will be used to calculate the SCRs, the following message is displayed:

> The number of alignments to be looked at, has been set to more than the total number of alignments that were entered into this program. So, we are resetting the number of alignments to be looked at, to be the same as the number of alignments that were used as input to this program.

4 You are then prompted to see how the SCRs should be calculated with the following menu:

> Choose how the SCR's are to be calculated. Enter number now ....
> 1) Automatic. Both the alignments to look at & the areas of identity are automatically done.
> 2) Semi- automatic. User chooses the alignments, areas of identity automatically done.
> 3) Manual. User chooses the Strongly Conserved Regions.
> 4) Change the number of alignments that are looked at in choosing the Strongly Conserved Regions.
> 5) Change the minimum width that a Strongly Conserved Region can be.
> 6) Help.

4.1 Enter the number of the option you want, followed by '<rtn>' e.g., '1 <rtn>'

4.2 If the number you enter is not permissible, you will see displayed the following error message:

> This number is out of range - please re-enter a number

4.2.1 The sequence of error message, menu of available choices and prompt to enter a number will continue until an acceptable value is entered.

5 Enter '1 <rtn>' if you want the program to automatically choose both which alignments to look at in choosing the SCRs, and the SCRs. You will then proceed to part 11 of this section.

6 Enter '2 <rtn>' if you want to choose which alignments to look at in choosing the SCRs and the program to automatically calculate the SCRs. The following messages and prompt will then be displayed:

User choosing alignments to calculate the Strongly Conserved Regions.
You have to enter <number> alignments.
NB - you cannot choose the same alignment twice.
Alignments are numbered in the same order that they are found in, in the alignment file used as input. A list of the parameters for each alignment can be found in the file '<nameofile>.param_data'.
This file can be printed just now on the default printer if you enter print (at Glasgow, this can also be printed on the Biochem. Dept line printer by typing BPRINT and the Chemistry Dept. line printer by typing CPRINT) - if you do not want this file to be printed, just press return.

6.1 If you want to print the file containing the parameter data on the system's default printer, type 'print <rtn>'. You will then proceed to part 6.2 of this section.

6.1.1 If you want to print the file containing the parameter data on the Biochem. Dept line printer, type 'bprint <rtn>'. You will then proceed to part 6.2 of this section.

6.1.2 If you want to print the file containing the parameter data on the Chemistry Dept. line printer, type 'cprint <rtn>'. You will then proceed to part 6.2 of this section.

6.1.3 If you do not want this file to be printed, enter '<rtn>'. You will then proceed to part 6.2 of this section.

6.2 The following message and prompt will then be displayed:

Choosing <number> alignments: Chosen <number>
Enter the number of the alignment you want to user for the SCRs

6.2.1 If any alignments to be chosen in calculating SCRs have been entered,, the following message will be displayed:

The alignments already chosen to calculate the SCRs are numbers <numbers>

6.3 Enter the number of the alignment that you wish to use to calculate the SCRs, followed by '<rtn>'. If you have entered all the alignments that you need to choose to calculate the SCRs, you will then proceed to part 11 of this section, else you will proceed to part 6.2 of this section.

6.4 If the alignment number entered is not permissible, the following error message will be displayed:

That number is not in range - please re-enter a number.

6.4.1 The sequence of error message and prompt to enter data will continue until an acceptable value is entered.

6.5 If that alignment has already been entered, the following error message will be displayed:

This alignment has already been chosen - please choose a different one.

6.5.1 The sequence of error message and prompt to enter data will continue until an acceptable value is entered.

7 Enter '3 <rtn>' if you want to manually pick the SCRs. The following message and prompt will then be displayed:

Choosing your own Strongly Conserved Regions.
Input the number of the alignment (numbering the alignments as they are found in the input file) that you want the SCRs to be taken from.

7.1 Enter the number of the alignment that you wish to use to calculate the SCRs, followed by '<rtn>'.

7.2 If the alignment number entered is not permissible, the following error message will be displayed:

This number is out of range, please re-enter

7.2.1 The sequence of error message and prompt to enter data will continue until an acceptable value is entered.

7.3 You will then be prompted with the following message:

Starting position of SCR number <number> (or -1 to stop).

7.4 Enter '-1 <rtn>', if you wish to stop entering SCRs. You will then proceed to part 13 of this section.

7.5 Enter the starting position of the SCR, followed by '<rtn>'.

7.6 If this would mean the length between SCRs is 2 or less residues, this is too close to the end of the last SCR, and the following error message will be displayed:

This is too close to the end of the last SCR. Re-enter number.

7.6.1 The sequence of error message and prompt to enter data will continue until an acceptable value is entered.

7.7 If this starting position is at the end of the alignment, the following error message will be displayed:

This is past or at the end of the alignment. Re-enter number.

7.7.1 The sequence of error message and prompt to enter data will continue until an acceptable value is entered.

7.8 You will then be prompted with the following message:

Finishing position of SCR number <number>

7.9 Enter the finishing position of the SCR, followed by '<rtn>'.

7.10 If the position entered is before the start of the SCR or after the end of the alignment, the following error message will be displayed:

This number is out of range, please re-enter

7.10.1 The sequence of error message and prompt to enter data will continue until an acceptable value is entered.

7.11 The residues in this SCR for one of the sequences in the alignment is then displayed, and the user is prompted to check that this is correct, as follows:

This is the SCR for sequence <sequence name>
<residues in SCR>
Type yes if correct, no if wrong. [yes]

7.12 If this is correct, enter 'yes <rtn>' or press '<rtn>'. You will then return to part 7.3 of this section.

7.13 If this is wrong, enter 'no <rtn>'. You will then return to part 7.3 of this section.

8 Enter '4 <rtn>' if you want to change the number of alignments that are looked at in choosing the SCRs. This will automatically start option 3 of the menu displayed in part 3.2 of the 'Entering other program information' section with the messages and prompt of part 3.5 being displayed. Parts 3.5 to 3.9.1 in that section will be run through, except that in part 3.9.1, after your new value is checked, you proceed to part 8.1 of this section, and not as described, to the menu screen of part 3.2 of that section.

8.1 If the new value entered is not permissible, the following message is displayed:

You chose a number of alignments to be looked at, that is greater than the total number of alignments that were entered into this program. So, we are resetting the number of alignments to be looked at, to be the same as the number of alignments that were used as input to this program.

8.2 You will then return to the menu displayed in part 4 of this section.

9 Enter '5 <rtn>' if you want to change the minimum length that an SCR can be. This will automatically start option 5 of the menu displayed in part 3.2 of the 'Entering other program information' section with the messages and prompt of part 3.5 being displayed. Parts 3.5 to 3.9.1 in that section will be run through,, except that in part 3.9.1, after your new value is checked, you proceed to part 4 of this section, and not to the menu screen of part 3.2 of that section.

10 Enter '6 <rtn>' to go to the help section (see below). After displaying the help section you will then return to the menu displayed in part 4 of this section.

11 The program will then display the following message:

There were <number> alignments chosen
The alignments used to calculate the SCRs are numbers <numbers>

11.1 The program then calculates the SCRs. Very rarely, however, a problem may arise. This problem is, if there are 2 or more identical SCRs, the program will not be able to decide where about the SCR actually is. Before continuing to run, proceeding to part 12 of this section, the following messages will be displayed:

There were <number> places where <sequence> was found in the test alignment.
This is really confusing - which hit should be chosen for the Strongly Conserved Regions. Rule of thumb used is that the last place found is the SCR. Frankly your computer recommends that you start again, but this time, you should choose your own SCRs.

12 If more SCRs were found in the alignments than allowed by the program, the program will stop after displaying the following error messages:

There are more Strongly Conserved Regions in the alignments than are allowed in this program.
This is not a major problem - the limit on this number was only placed for programming reasons (to allow other variables to be initialised) - however, you will have to change this number before re-running.
The number allowed is stored in the variable MAXNUMSCR.

To change this, edit the program (type 'edit MIX_N_MATCHA.FOR'). The variable will be given a value in a line that starts with the word 'parameter'. Increase the variable to a number greater than the number you are using as input. Exit from editing the program. Then recompile, relink and you are then ready to run the program again.
Deleting the work files generated by this program.

13 The program will then check to see if any SCRs have been found. If no SCRs have been found, the following error message will be displayed, before returning to part 4 of this section:

There is a problem - no Strongly Conserved Regions have been found. You should try again with a different option or if you are using the automatic method to work out which alignments the SCRs should be chosen from, you can reduce the number of alignments used to calculate the SCR. This should improve the chances of finding a SCR.

14 The following messages are then displayed:

There were <number> Strongly Conserved Regions
There were <number> Loosely Conserved Regions

Explanatory messages

The program now displays messages indicating what it is doing.

1 The following message is displayed:

Finding the Loosely Conserved Regions

1.1 This program finds the SCRs in each alignment, and then finds the Loosely Conserved Regions in between. Very rarely, however, a problem may arise. If a SCR is found in 2 or more places in the same alignment, the program will not be able to decide where about the SCR actually is. Before continuing to run, the following messages will be displayed:

There were <number> places where <sequence> was found in the test alignment.
This is really confusing - which hit should be chosen for the Strongly Conserved Regions. Rule of thumb used is that the last place found is the

SCR. Frankly your computer recommends that you start again, but this time, you should choose your own SCRs.

2 The following messages are then displayed:

There were <number> different alignments found in these LCR's.
Finding the unique Loosely Conserved Regions

2.1 For each LCR, the following messages are then displayed:

For LCR number <number>, there were <number> possible alignments.
And of these, <number> were unique.

3 The following messages are then displayed:

Writing out the unique Loosely Conserved Regions (lcr.data) for use by the program MMB
Writing out the Strongly Conserved Regions (scr.data) for use by the program MMB

4 The following message is then displayed:

Do you want to try and get different SCR regions [no]:

4.1 If you do not want to try and get different SCRs, enter 'no <rtn>' or press '<rtn>'.

4.2 If you do want to try and get different SCRs, enter 'yes <rtn>'. You will then proceed to part 1 of the 'Calculating the Strongly Conserved Regions' section.

5 The following messages are then displayed:

Deleting the work files generated by this program.
For each Loosely Conserved Region (the region in between the 'anchor' SCR's) there will be a choice of differing alignments blocks. The unique blocks found for each LCR are in the file 'PRINT.ME'. Print this file and choose which one of the alignments blocks is 'best' for each of these regions.
Then run the program MMB, which will ask you which blocks you have picked for each LCR and then produce a final alignment (in the file 'PRINT.ME2')

6 The program then ends.

The help section

This section lists the help that is available with this program. If, when available, the user chose the help option, the following menu will first be displayed:

1) A quick look at the theoretical basis of this program.
2) What input files are needed by this program?
3) Miscellaneous.
4) Changing the program variables.
5) How the program calculates Strongly Conserved Regions.
6) Return to running the program.
There is no other help at the minute

1 Enter the number of the option, followed by '<rtn>' e.g., '2 <rtn>'.

2 If the number you enter is not permissible, you will see displayed the following error message:

That number is out of range - please re-enter a number

2.1 The sequence of error message, menu of available choices and prompt to enter a number will continue until an acceptable value is entered.

3 If you chose option 1, the following message will be displayed:

This program takes a large number of differing alignments and uses them to work out Strongly Conserved 'anchor' Regions. It then lists the different aligned blocks that were found in between these SCR's, together with their predicted secondary structure. The user can then pick whichever of these aligned blocks is best, (eg. using knowledge from empirical studies or the knowledge that gaps should only occur in turn regions), giving a final alignment reasonably independant of the starting parameters/programs used and containing more expert information than a purely 'mathematical' approach can give.

3.1 The following prompt will then be displayed:

Press RETURN when finished reading.

3.2 Press '<rtn>' to return to the menu displayed at the start of this section.

4 If you chose option 2, the following message will be displayed:

This program uses 2 files as input. The first of these is the alignment file. Different multiple alignments can be generated using different programs and using different gap parameters/scoring matrices. All these differing alignments should be placed in a single file, and it is the name of this single file that should be entered (at the minute, this program can read in alignments made by ALIEN and PILEUP - more formats will be added whenever possible).

The second file is the sequence file. This should contain all of the sequences that were used in making up the alignments. This sequence file can be in 2 formats, NBRF or UWGCG. If the sequences are in NBRF format, all the sequences should be in this single file. If the sequences are in GCG format, the GCG 'file of file names' should be used. This format requires that in a single file there should be a list of the file names of the GCG sequences. To show that you are using this type of file, and not a sequence file, you should precede the file name with a '@'. (NB - If the GCG format is used, all the individual sequence names are presumed to end in '.GCG' and this program will crash if they do not!)

4.1 The following prompt will then be displayed:

Press RETURN when finished reading.

4.2 Press '<rtn>' to return to the menu displayed at the start of this section.

5 If you chose option 3, the following message will be displayed:

GCG must be initialised before this program is run - otherwise this program crashes. The GCG package is needed to predict the secondary structures by the methods of Chou & Fasman (CF) and Garnier, Osguthorpe & Robson (GOR), carried out by the GCG programs PEPTIDESTRUCTURE and PEPPLOT. PEPTIDESTRUCTURE gives a CF and GOR prediction and PEPPLOT gives a GOR prediction without decision constants and a GOR prediction with decision constants (output is called cf, gor, gor2, gor3 respectively). To run these programs, sequences need to be in GCG format and so if sequences were input as NBRF format, they are reformatted with the GCG program FROMPIR. This is done by this program automatically

Residues in a sequence are indicated in a short-hand manner by using capital letters of the alphabet eg. A=alanine. If a small letter is found in

the sequence data, it must be changed to the correct capital letter. The problem is that although 'a' may indicate alanine, some sequence formats use small letters to indicate a cysteine residue, with the next 'a' found, bonded to the first 'a' residue, forming a cystine residue. You are asked to decide whether small letters are being used to indicate cysteines or if they mean the same as capital letters.

5.1 The following prompt will then be displayed:

Press RETURN when finished reading.

5.2 Press '<rtn>' to return to the menu displayed at the start of this section.

6 If you chose option 4, the following message will be displayed:

Screen width holds the number of characters that can be displayed horizontally across the screen (eg. on a vt100 terminal, either 80 or 132 characters can be displayed). The files that can be printed out are the final results files (PRINT.ME) and the debugging data files (xxxx.sec_dat) and (xxxx.run_dat) if these were asked to be printed. The number of characters per line has a default value of132, the same width as a line printer, so that these files can be printed for hard copy. The other variables are used in calculating the Strongly Conserved Regions - there is a seperate help section on these variables which you should read before modifying them.

6.1 The following prompt will then be displayed:

Press RETURN when finished reading.

6.2 Press '<rtn>' to return to the menu displayed at the start of this section.

7 If you chose option 5, the following message will be displayed:

Calculating the Strongly Conserved Regions (SCR). The program defines a SCR as being any region which is the same in all the alignments looked at. NB - not all of the alignments are looked at because some of the alignments ALIEN & PILEUP produce can be very bad. The program uses a default value of 15 alignments to look at, however, the user can change this number if they wish - either at the start (in the change the program variables section) or in the calculate SCR section (with option number 4)

Once the number to look at is set, which alignments are looked at can be chosen by the user or are generated automatically.

The method to automatically choose alignments, uses the number of identities in an alignment to pick out the 'good' from 'bad'; the alignments with the highest number of id's are picked, then those with the next highest etc.

With the number of alignments to be looked at set, and with which specific alignments these are set, the program then calculates the SCR's.

7.1 The user is then prompted to display another screen of information, with the following message:

Press RETURN for more information.

7.2 Press '<rtn>' to display more information.

However, instead of these semi-automatic or automatic methods, an option exists for the user to completely define the SCR's themselves. To do this, you first enter the number of the alignment that the SCR's will be chosen from (eg. if the alignment was the 5th alignment in your input alignment file, you would enter '5'). You then enter the starting & finishing positions of the SCR's (the numbering should be taken from the alignment you have chosen.).

Calculating identical regions; after the alignments to be looked at are chosen, regions that are identical in all of these alignments are defined as SCR's. The minimum length that a region can be is initially set at 3 residues. This was chosen after careful consideration, but the user can change this if they wish (although the range of values is set between 3 and 10) Note that, the more sequences there are in an alignment, the less likelihood there is of finding regions that are identical (very similar, yes, identical, no). To try and reduce this effect, we limit the number of sequences that we look at. If there are more than this set number of sequences in an alignment, we pick sequences to calculate the SCR's from at random.

7.3 The following prompt will then be displayed:

Press RETURN when finished reading.

7.4 Press '<rtn>' to return to the menu displayed at the start of this section.

8 If you chose option 6, the user will return to running the program from where they asked for help.

Error messages displayed if there is a problem with the alignment input file.

If no alignments were found in the input alignment file, the program will stop after displaying the following error messages:

> No alignments have been found!
> This program will now terminate due to an error in the alignment file.

If an alignment with a format unsupported by this program is found in the input alignment file, the program will stop after displaying the following error messages:

> An unknown alignment type has been found in the alignment file. Only files generated by ALIEN or PILEUP may be used. Please check your alignment file and start again.
> The first 6 characters of this unknown alignment type are <xxxxxx>
> This program will now terminate due to an error in the alignment file.

If more alignments were found in the input alignment file than permitted by the program, the program will stop after displaying the following error messages:

> There are more alignments in the alignment input file than are allowed in this program.
> This is not a major problem - the limit on this number was only placed for programming reasons (to allow other variables to be initialised) - however, you will have to change this number before re-running.
> The number allowed is stored in the variable MAXALISNUM.
> To change this, edit the program (type 'edit MIX_N_MATCHA.FOR'). The variable will be given a value in a line that starts with the word 'parameter'. Increase the variable to a number greater than the number you are using as input. Exit from editing the program. Then recompile, relink and you are then ready to run the program again.
> This program will now terminate due to an error in the alignment file.

If more sequences are found in the alignments than allowed by the program, the program will stop after displaying the following error messages:

> There are more sequences in the alignments than are allowed in this program.

This is not a major problem - the limit on this number was only placed for programming reasons (to allow other variables to be initialised) - however, you will have to change this number before re-running.
The number allowed is stored in the variable MAXSEQNUM.
To change this, edit the program (type 'edit MIX_N_MATCHA.FOR'). The variable will be given a value in a line that starts with the word 'parameter'. Increase the variable to a number greater than the number you are using as input. Exit from editing the program. Then recompile, relink and you are then ready to run the program again.
This error took place in alignment number <number>
This program will now terminate due to an error in the alignment file.

If longer sequences are found in the alignments than allowed by the program, the program will stop after displaying the following error messages:

The lengths of the sequences in the alignments file are longer than are allowed in this program.
This is not a major problem - the limit on this number was only placed for programming reasons (to allow other variables to be initialised) - however, you will have to change this number before re-running.
The number allowed is stored in the variable MAXSEQLEN.
To change this, edit the program (type 'edit MIX_N_MATCHA.FOR'). The variable will be given a value in a line that starts with the word 'parameter'. Increase the variable to a number greater than the number you are using as input. Exit from editing the program. Then recompile, relink and you are then ready to run the program again.
This error took place in alignment number <number>
This program will now terminate due to an error in the alignment file.

If there are less than 3 sequences found in the alignments, the program will stop after displaying the following error messages:

There are not enough sequences for an alignment! This program needs at least 3 sequences in the alignments to work.
This error took place in alignment number <number>
This program will now terminate due to an error in the alignment file.

If the names of the sequences in the alignments do not agree with the names of the sequences as given in the sequence input file, the program will stop after displaying the following error messages:

> A sequence file named in the alignments could not be found. The sequences must have different names in the alignment and sequence files. Please change the sequence names so that the names found in the alignments and the sequence file agree. This problem can occur because the initial alignment program may have changed the sequence name. Or, if your sequence input file was in NBRF format, then reformatting the sequences to GCG format (done by this program with the GCG command FROMPIR), may have changed the sequence names. Or, if your sequence input file was in GCG format, ensure that the files all have the suffix '.GCG'. We have to assume that all sequence files have this suffix because that is what FROMPIR appends to sequence names and otherwise we would never be able to find them
> This program will now terminate due to an error in the alignment file.

If the names of the sequences differ between the different alignments in the alignment input file, the program will stop after displaying the following error messages:

> A sequence name has changed between the first alignment and this one.
> The sequence named <sequence name> could not be found.
> This error took place in alignment number <number>
> This program will now terminate due to an error in the alignment file.

User guide for running the multiple alignment programs Mix'n'MatchB

1 If the program cannot find the input file 'SCR.DATA', it will stop after displaying the following error message:

> The file 'SCR.DATA' produced by Mix'n'MatchA cannot be found. This file is needed for Mix'n'MatchB to run, so Mix'n'MatchB will now stop.

2 If the program cannot find the input file 'LCR.DATA', it will stop after displaying the following error message:

> The file 'LCR.DATA' produced by Mix'n'MatchA cannot be found. This file is needed for Mix'n'MatchB to run, so Mix'n'MatchB will now stop.

3 An explanatory message about this program is then displayed.

4 You will be prompted to enter the alignment number you have chosen for each of the LCRs with the following message:

What alignment number for LCR <number>

5 Enter the chosen alignment number followed by '<rtn>' e.g., '7 <rtn>'.

6 If this value is not permissible the following error message is displayed:

This number is out of range - please re-enter another number.

7 The sequence of error message and prompt for entering data will continue until a permissible value is entered.

8 If this alignment cannot be found, the following error message is displayed:

There has been an in/out error when reading from 'LCR.DATA'.
Obviously, an incorrect alignment number was given.

9 The sequence of error message and prompt for entering data will continue until a permissible value is entered.

10 After entering the alignment numbers for all the LCRs, the following message is displayed:

Do you want the data files (LCR.DATA, SCR.DATA and PRINT.ME) produced by MMA deleted [yes] - or not deleted in case you want to run this programme again [no]? The default answer is no.

11 If you want these files deleted, enter 'yes <rtn>'.

12 If you do not want these files deleted, enter 'no <rtn>' or press '<rtn>'.

13 The program will then end, after displaying the final message:

The final alignment is stored in the file 'PRINT.ME2'. Print this file to see this alignment.

Appendix E Examples of file formats used by Mix'n'Match

The format used for the NBRF sequence file

All lines must be fully left-justified. The first line of each entry is the protein/DNA identifier code, preceded by four characters of the form '>xx;'. Typical NBRF prefixes are '>P1;', '>F1;' and '>DL;'. For example, for the second sequence, the identifier code is 'HAHO'. The second line of each entry is a title line saying what the entry is and its source. This line is compulsory but its content is ignored. The sequence starts on line 3 of each entry and may spread over several lines; it must however end with the asterisk character ('*').

Sample output for the NBRF sequence file

A set of 3 haemoglobin sequences are used in this example and <SOF> and <EOF> are used to indicate the start and end of the NBRF format sequence file, respectively.

```
<SOF>
>P1;GGLMS
line containing a description of the protein sequence
PIVDTGSVAPLSAAEKTKIRSAWAPVYSTYETSGVDILVKFFTSTPAAQEFFPKFKGLTTAD
QLKKSADVRWHAERIINAVNDAVASMDDTEKMSMKLRDLSGKHAKSFQVDPQYFKVLAAVIA
DTVAAGDAGFEKLMSMICILLRSAV*
>P1;HAHO
line containing a description of the protein sequence
VLSAADKTNVKAAWSKVGGHAGEYGAEALERMFLGFPTTKTYFPHFDLSH
GSAQVKAHGKKVGDALTLAVGHLDDLPGALSNLSDLHAHKLRVDPVNFKLLSHCLLSTL
AVHLPNDFTPAVHASLDKFLSSVSTVLTSKYR*
>P1;MYWHP
line containing a description of the protein sequence
VLSEGEWQLVLHVWAKVEADVAGHGQDILIRLFKSHPETLEKFDRFKHLKTEA
EMKASEDLKKHGVTVLTALGAILKKKGHHEAELKPLAQSHATKHKIPIKYLEFISEAIIHVL
HSRHPGDFGADAQGAMNKALELFRKDIAAKYKELGYQG*
<EOF>
```

The format used for the UWGCG 'file of file names' file

All lines are fully left-justified. Each line should contain the name of a UWGCG format sequence file (see below), that you want to be multiply aligned. NB - for the Mix'n'Match program, the sequence files should all have the extension '.GCG', otherwise the program will not be able to find them and will stop with an error message.

Sample output for the UWGCG 'file of file names' file

A set of 3 haemoglobin sequence file names are used in this example and <SOF> and <EOF> are used to indicate the start and end of the 'file of file names' file, respectively.

```
<SOF>
GGLMS.GCG
HAHO.GCG
MYWHP.GCG
<EOF>
```

The format used for UWGCG sequence files

The start of the file contains information about the sequence, and UWGCG specific information. This continues until the two characters '..' are found. The sequence starts after this and may spread over several lines and may contain blank lines. Each line of residues starts with the number of the first residue in that line. The number of residues in each line and whether or not spaces occurs along the residues, are variable and can be set by the user.

Sample output for the UWGCG format sequence files

The set of 3 haemoglobin sequences named in the UWGCG 'file of file names' file shown above, are listed in this example. <SOF> and <EOF> are used to indicate the start and end of the sequence files, respectively.

Example output for the file GGLMS.GCG

```
<SOF>
p1;GGLMS          -

Gglms.Gcg  Length: 149  February 16, 1993  14:20  Type: P  Check: 7918  ..

       1  PIVDTGSVAP LSAAEKTKIR SAWAPVYSTY ETSGVDILVK FFTSTPAAQE

      51  FFPKFKGLTT ADQLKKSADV RWHAERIINA VNDAVASMDD TEKMSMKLRD

     101  LSGKHAKSFQ VDPQYFKVLA AVIADTVAAG DAGFEKLMSM ICILLRSAY
<EOF>
```

Example output for the file HAHO.GCG

```
<SOF>
p1;HAHO            -

Haho.Gcg  Length: 141  February 16, 1993  14:20  Type: P  Check: 83  ..

       1  VLSAADKTNV KAAWSKVGGH AGEYGAEALE RMFLGFPTTK TYFPHFDLSH

      51  GSAQVKAHGK KVGDALTLAV GHLDDLPGAL SNLSDLHAHK LRVDPVNFKL

     101  LSHCLLSTLA VHLPNDFTPA VHASLDKFLS SVSTVLTSKY R
<EOF>
```

Example output for the file MYWHP.GCG

```
<SOF>
p1;MYWHP           -

Mywhp.Gcg  Length: 153  February 16, 1993  14:20  Type: P  Check: 3184  ..

       1  VLSEGEWQLV LHVWAKVEAD VAGHGQDILI RLFKSHPETL EKFDRFKHLK

      51  TEAEMKASED LKKHGVTVLT ALGAILKKKG HHEAELKPLA QSHATKHKIP

     101  IKYLEFISEA IIHVLHSRHP GDFGADAQGA MNKALELFRK DIAAKYKELG

     151  YQG
<EOF>
```

The format used by the ALIEN multiple alignment program

All lines are fully left-justified. The first six lines are messages identifing the alignment type and where the start of the multiple alignment is. Then follows blocks of aligned sequences, with two lines between, the first of which identifies identical and similar residues in the aligned sequences. The end of the aligned sequences is indicated by the 'End of Multiple Alignment' message. The format for the blocks of sequences is sequence name from position 1 to 6 (6 characters per name) and the aligned sequences from positions 15 to 76 (62 characters per line). Then follows a list of the parameters used by the program.

Sample output from the ALIEN multiple alignment program

A set of 3 haemoglobin sequences are used in this example and <SOF> and <EOF> are used to indicate the start and end of the ALIEN output file, respectively. The sequence file input to this program is that given in the example of the NBRF sequence file format.

```
<SOF>
ALIEN: Multiple Sequence alignment program
by
Alan J Bleasby

Start of Multiple Alignment

GGLMS           PIVDTGSVAPLSAAEKTKIRSAWAPVYSTYETSGVDILVKFFTSTPAAQEFFPKFKGLTTAD
HAHO            V----------LSAADKTNVKAAWSKVGGHAGEYGAEALERMFLGFPTTKTYFPHF-DLSH--
MYWHP           ----------VLSEGEWQLVLHVWAKVEADVAGHGQDILIRLFKSHPETLEKFDRFKHLKTEA
                          #~  ~   ~   #  #       # ~ # ~ ~   # ~   #  #  #

GGLMS           QLKKSADVRWHAERIINAVNDAVASMDDTEKMSMKLRDLSGKHAKSFQVDPQYFKVLAAVIA
HAHO            ---GSAQVKAHGKKVGDALTLAVGHLDDLPGALSNLSDLHAHKLRVDPVNFKLLSHCLLSTL
MYWHP           EMKASEDLKKHGVTVLTALGAILKKKGHHEAELKPLAQSHATKHKIPIKYLEFISEAIIHVL
                    ~  ~~ #~  ~  ~    ~   ~~       # ~                ~

GGLMS           DTVAAGDAG-------------FEKLMSMICILLRSAY
HAHO            AVHLPNDFTPAVHASLDKFLSSVSTVLTSKYR------
MYWHP           HSRHPGDFGADAQGAMNKALELFRKDIAAKYKELGYQG
                       ~                 ~~

End of Multiple Alignment


Key:
     Identity:     #
     Conservative: ~


Parameters:

Ktup 1
Wilbur Gap 5
Cutoff   1.00
Diagonal 10
Fixed Gap 10
Floating Gap 10

Sequences were: Proteins
Absolute Identities Were Scored
Sequence Input file: haem.seqs
Matrix used: No penalty
<EOF>
```

The format used by the PILEUP multiple alignment program

The first two lines are a message identifing the alignment type, followed by a blank line. The next 7 lines are a list of the parameters used by the program, followed by a blank line. The next lines identitfy which sequences are used in this alignment, with the end of the sequence names being identified by a blank line followed by the two characters '//' at positions 1 and 2 of a new line, and another blank line. Then follows blocks of aligned sequences, with a line above the sequences showing the position numbers of the aligned sequences, and a blank line below the sequences. In the blocks of sequences, the sequence names are right-justified, from position 1 to *x* (where *x* is the length of the sequence name that is the longest) and the aligned sequences start 3 characters further to the right. The length of the aligned sequences and whether or not spaces occurs along the aligned sequences, are variable and can be set by the user.

Sample output from the PILEUP multiple alignment program

A set of 3 haemoglobin sequences are used in this example and <SOF> and <EOF> are used to indicate the start and end of the PILEUP output file, respectively. The sequence file input to this program is that given in the example of the UWGCG 'file of file names' format.

```
<SOF>
PileUp of: @Haem.Names

 Symbol comparison table: GenRunData:PileUpPep.Cmp  CompCheck: 1254

                  GapWeight: 3.0
            GapLengthWeight: 1.5

 Haem.Msf  MSF: 165  Type: P  February 11, 1992  13:52  Check: 1211  ..

 Name: Haho             Len:   165  Check: 6265  Weight:  1.00
 Name: Mywhp            Len:   165  Check:   78  Weight:  1.00
 Name: Gglms            Len:   165  Check:  475  Weight:  1.00

//

       1                                                    50
 Haho  .........V LSAADKTNVK AAWSKVGGHA GEYGAEALER MFLGFPTTKT
Mywhp  .........V LSEGEWQLVL HVWAKVEADV AGHGQDILIR LFKSHPETLE
Gglms  PIVDTGSVAP LSAAEKTKIR SAWAPVYSTY ETSGVDILVK FFTSTPAAQE

       51                                                   100
 Haho  YFPHF.DLSH .....GSAQV KAHGKKVGDA LTLAVGHLDD ...LPGALSN
Mywhp  KFDRFKHLKT EAEMKASEDL KKHGVTVLTA LGAILKKKGH ...HEAELKP
Gglms  FTPKFKGLTT ADQLKKSADV RWHAERIINA VNDAVASMDD TEKMSMKLRD

       101                                                  150
 Haho  LSDLHAHKLR VDPVNFKLLS HCLLSTLAVH LPNDFTPAVH ASLDKFLSSV
Mywhp  LAQSHATKHK IPIKYLEFIS EAIIHVLHSR HPGDFGADAQ GAMNKALELF
Gglms  LSGKHAKSFQ VDPQYFKVLA AVIADTVAAG .DAGFEKLMS MICILLRSAY

       151          165
 Haho  STVLTSKYR. .....
Mywhp  RKDIAAKYKE LGYQG
Gglms  .......... .....

<EOF>
```

Appendix F User guide for running the automatic multiple alignment programs ALIEN and PILEUP

Introduction to ALIEN

ALIEN first calculates an optimal pairwise alignment of every sequence in the data set using the Wilbur/Lipman algorithm. After cluster analysis of the results from the pairwise alignment step, a Myers/Miller algorithm is used to multiply align the sequences in the data set. The parameters for both the pairwise and multiple alignment steps are set by command line qualifiers (see below). For a fuller guide to the ALIEN program, see the program documentation on SEQNET at Daresbury, Cheshire, U.K..

The parameters used by ALIEN

The list of qualifiers is shown below.

Pairwise alignment parameters	Type	Default value	Permissable range
K-tuple	Number	1	1-2
gap penalty	Number	3	1-500
Score	Text	Absolute	Absolute or Percentage
Cutoff	Number	1.0	0.1-20.0
Diagonal	Number	10	1-50
Multiple alignment parameters			
Fixed gap penalty	Number	10	1-100
Floating gap penalty	Number	10	1-100
Other parameters			
Infile file	Text	name of sequence input file	Compulsory
Matrix	Text	DEF	DEF, PAM, or user-file

How to run ALIEN

An example of command line operation would be:

```
$ alien -infile=myseqs.pro -ktup=2 -float=6 -wgap=4
```

This would tell ALIEN that your sequence data set is in the file 'myseqs.pro', you want a K-tuple of 2, a floating gap penalty of 6 and a Wilbur/Lipman gap penalty of 4. The order of qualifiers in the command line is irrelevant and all qualifiers may be abbreviated unless such abbreviation results in ambiguity with another qualifier.

To run many alignments at one time, and to automatically place all the alignments in one file, you create a command file, containing all the appropriate commands. An example command file, shown below, will create a file that will contain all the alignments, run the alignment program with various parameters, place all these alignments into the single file and then deleter. Create a file containing the following UNIX commands:

```
>all.alien
alien -in=any.seq -wg=8 -dia=20 -fl=10 -fi=10 -mat=pam
cat haem.ali > all.alien
alien -in=any.seq -wg=8 -dia=20 -fl=15 -fi=15 -mat=pam
cat haem.ali > all.alien
alien -in=any.seq -wg=8 -dia=20 -fl=20 -fi=20 -mat=pam
cat haem.ali > all.alien
alien -in=any.seq -wg=3 -dia=10 -fl=10 -fi=10 -mat=pam
cat haem.ali > all.alien
rm any.wil
rm any.ali
^d
```

Then run this file in the background using the following UNIX commands

```
nohup sh filename &
```

Introduction to PILEUP

PILEUP first calculates an optimal pairwise alignment of every sequence in the data set using the Needleman and Wunsch algorithm. After cluster analysis of the results from the pairwise alignment step, the Needleman and Wunsch algorithm is used to successively align pairs of sequences to produce the multiple alignment. The parameters are set by command line qualifiers (see below). For a fuller guide to the ALIEN program, see the UWGCG program documentation.

The parameters used by PILEUP

The list of qualifiers is shown below.

Alignment parameter name	Default value
Gapweight	3.0
Lenweight	0.1
Data	PileUpPep.Cmp

How to run PILEUP

An example of command line operation would be:

```
$pileup/infile=@seqs.names/out=any.msf/gap=2.0/len=1.5/default
```

This would tell PILEUP that your sequence data set is in the file 'seq.names', the resulting alignment will be placed in the file 'seqs.msf', you want a gap penalty of 2.0, a len of 1.5 and '/default' ensures that the default values are used for the other parameters. The order of qualifiers in the command line is irrelevant, as long as '/default' is last, and qualifiers may be abbreviated (see the UWGCG program documentation).

To run more than one alignment at a time, placing all the alignments in one file (i.e., suitable for input to Mix'n'Match), you create a 'command' file which contains all the appropriate commands and then run this file. An example command file, shown below, will create a file that will contain all the alignments, run the alignment program with various parameters, place these alignments into the single file and then delete any

unnecessary files. To make this command file, create a file containing the following VMS commands:

```
$ create all.pileup
$ gcg7
$pileup/infile=@seqs.names/out=seqs.msf/gap=4.0/len=0.1/default
$append seqs.msf all.pileup
$pileup/infile=@seqs.names/out=seqs.msf/gap=3.0/len=1.5/default
$append seqs.msf all.pileup
$pileup/infile=@seqs.names/out=seqs.msf/gap=2.0/len=1.0/default
$append seqs.msf all.pileup
$pileup/infile=@seqs.names/out=seqs.msf/gap=2.0/len=0.5/default
$append seqs.msf all.pileup
$delete seqs.msf;*
```

If the above example command file was called 'run.pileup', then to run this file interactively, use the following VMS commands

```
$ @run.pileup <rtn>
```

To run this command file in the background, you would type the following VMS commands:

```
$ submit/queue=queue_name run.pileup <rtn>
```

Introduction to the UWGCG secondary structure prediction programs

There are two UWGCG programs for predicting secondary structure. The program PEPTIDESTRUCTURE predicts the secondary structure using the methods of Chou and Fasman (CF) and Garnier, Osguthorpe and Robson (GOR). The program PEPPLOT predicts the structure using the same two methods, although only the results from the GOR analysis are in a readily useable form. PEPPLOT calculates a GOR prediction both with and without Decision Constants. For a fuller guide to these programs, see the UWGCG program documentation.

How to run the UWGCG secondary structure prediction programs

Examples of running these programs would be:

```
$pepplot/noplot/infile=seq1.gcg/gar=seq1.gar/default
$peptidestructure/infile=seq1.gcg/default
```

This would tell PEPPLOT that you do not want the predictions plotted, your sequence data is in the file 'seq1.gcg', you want the GOR predictions to be output to the file 'seq1.gar', and '/default' ensures that the default values are used for the other parameters and would tell PEPTIDESTRUCTURE that your sequence data is in the file 'seq1.gcg', and '/default' ensures that the default values are used for the other parameters (e.g., the CF and GOR predictions will be output to the file 'seq1.p2s'). The order of qualifiers in the command line is irrelevant and qualifiers may be abbreviated (see the UWGCG program documentation).

To run these programs for all your sequences, at one time, you create a 'command' file which contains the appropriate commands for each sequence and then run this file. An example command file, shown below, will run the programs PEPPLOT and PEPTIDESTRUCTURE for three sequences, named 'seq1.gcg', 'seq2.gcg' and 'seq3.gcg'. To make this command file, create a file containing the following VMS commands:

```
$pepplot/noplot/gar=seq1.gar/infile=seq1.gcg/default
$peptidestructure/infile=seq1.gcg/default
$pepplot/noplot/gar=seq2.gar/infile=seq2.gcg/default
$peptidestructure/infile=seq2.gcg/default
$pepplot/noplot/gar=seq3.gar/infile=seq3.gcg/default
$peptidestructure/infile=seq3.gcg/default
```

If the above example command file was called 'run.predictions', then to run this file interactively, use the following VMS commands

```
$ @run.predictions <rtn>
```

To run this command file in the background, you would type the following VMS commands:

```
$ submit/queue=queue_name run.predictions <rtn>
```

Appendix G The FORTRAN source code used to program the Cohen *et al.* turn prediction algorithm [146]

```
      PROGRAM TURNGEN
c
c a simple program to reproduce the algorithm in Cohen et al,
c Biochemistry, 1983, v22, p4894-4904
c the sequence to predict is hard wired into the code.
c
      IMPLICIT NONE
      INTEGER MAX_SEQ_LEN,X
      PARAMETER (MAX_SEQ_LEN=600)
      CHARACTER*(MAX_SEQ_LEN) ARR_STRING,TEMP_STR,T_ONE,TMP_STR

c open output file
      OPEN (4,FILE='V3.DATA')
 10   FORMAT (TR1,A)
c initialise
      do x=1,Max_seq_len
          temp_str(x:x) = "-"
          t_one(x:x) = "-"
      end do
      ARR_STRING='MALSSTSTTNSLLPNRSLVQNQPLLPSPLKNAFFSNNSTKTV
     &  RFVQPISAVHSSDSNKIPIVSDKPSKSSPPAACATTAPAPAVTKTEWAVDSWKSKKAL
     &  QLPEYPNQEELRSVLKTIDEFPPIVFAGEARSLEERLGEAAMGRAFLLQGGDCAES
     &  FKEFNANNIRDSFRILLQMGAVLMFGGQMPVIKVGRMAGQFAKP
     &  RSDSPEEKDGVKLPSYRGDNVNGDAFDVKSRTPDPQRLIRAYCQSAATLNLLRAFA
     &  TGGYAAMQRTNQWNLDFTFHSEQGDRYRELASRVDEALGFMTAA
     &  GLTMDHPIMKTTEFWISHECLLLPYEQSLTRRDSTSGLYYDCSAHFLWVGERTRQLDGAHV
     &  EFLRGIANPLGIKVSDKMDPSALVKLIEILNPQNKAGRI
     &  TIITRMGAENMRVKLPHLIRAVRRAGQIVIWVSDPMHGNTIKAPCGLRTRPFDSIRAEVRAFFDVH
     &  DQEGSHPGGVHLEMTGQNVTECIGGSRTVTFDDL
     &  SSRSLSLNV'
c convert the residues to capitals
      CALL CONV_STRING (ARR_STRING)
c find regions of t1
      CALL T_ONEG (ARR_STRING,TEMP_STR,T_ONE,TMP_STR)
      DO X=1,600,100
          WRITE (4,10) ARR_STRING(X:X+99)
          WRITE (4,10) TEMP_STR(X:X+99)
          WRITE (4,10) TMP_STR(X:X+99)
          WRITE (4,10) T_ONE(X:X+99)
          END DO
      CLOSE (4)
      END

      SUBROUTINE T_ONEG (ARR_STRING,TEMP_STR,T_ONE,TMP_STR)
c
c take a sequence in the string arr_string, search it for occurence of t1 residues
c (in tone) (cohen et al. biochem. 22 4894-4904)
c and places 't' in the string temp_str where these are found. also places a y
c where t2 residue (y) is found.
c takes residues in temp_str, search it for occurence of t1 patterns ( found in
c a/b/c_t_one ) and place these patterns in the string t_one.
c
      CHARACTER*(*) ARR_STRING,TEMP_STR,T_ONE,TMP_STR
      CHARACTER*1 TONE(11)
      CHARACTER*4 A_T_ONE(4)
      CHARACTER*5 B_T_ONE(3)
      CHARACTER*7 C_T_ONE(10)
      CHARACTER*10 T,G
      INTEGER X,Y,LOOP,POS
      DATA A_T_ONE /'YTTT','TYTT','TTYT','TTTY'/
```

```
      DATA B_T_ONE /'T-TTT','TT-TT','TTT-T'/
      DATA C_T_ONE /'T--TTTT','T-T-TTT','T-TT-TT','T-TTT-T','TT--TTT','TT-T-
TT','TT-TT-T','TTT--TT','TTT-T-T','TTTT--T'/
      DATA TONE /'D','E','G','H','K','N','P','Q','R','S','T'/
c put 't' in temp_str where t1 is in arr_string
 11   FORMAT (TR1,A)
 12   FORMAT (TR1,I)
      DO LOOP=1,11
        X=1
 110    POS=INDEX(ARR_STRING(X:),TONE(LOOP))
        IF (POS.EQ.0)GOTO 120
        Y=X+POS-1
        TEMP_STR(Y:Y)='T'
        X=X+POS
        GOTO 110
 120    CONTINUE
      END DO
c put 'y' in temp_str where t2 is in arr_seq (t2=t1+y)
      X=1
 130  POS=INDEX(ARR_STRING(X:),'Y')
      IF (POS.EQ.0)GOTO 140
      Y=X+POS-1
      TEMP_STR(Y:Y)='Y'
      X=X+POS
      GOTO 130
 140  CONTINUE
      TMP_STR=TEMP_STR
c find long sections first of all, then remove the parts that generate them
c from temp_str so that they don't interfere with the other patterns.
      X=1
      T='TTTTTTTTTT'
      G='----------'
      DO LOOP=10,4,-1
 203      POS=INDEX(TEMP_STR(X:),T(1:LOOP))
          IF (POS.EQ.0)GOTO 205
          Y=POS+X-1
          T_ONE(Y:Y+LOOP-1)=T(1:LOOP)
          X=X+POS
          TEMP_STR(Y:Y+LOOP-1)=G(1:LOOP)
          GOTO 203
 205      CONTINUE
          X=1
      END DO
c find the patterns that are given in cohen's paper.
      X=1
      DO LOOP=1,4
 210      POS=INDEX(TEMP_STR(X:),A_T_ONE(LOOP))
          IF (POS.EQ.0)GOTO 220
          Y=POS+X-1
          T_ONE(Y:Y+3)='TTTT'
          X=X+POS
          GOTO 210
 220      CONTINUE
          X=1
      END DO
      X=1
      DO LOOP=1,3
 230      POS=INDEX(TEMP_STR(X:),B_T_ONE(LOOP))
          IF (POS.EQ.0)GOTO 240
          Y=POS+X-1
          T_ONE(Y:Y-4)='TTTTT'
          X=X+POS
          GOTO 230
 240      CONTINUE
          X=1
      END DO
      X=1
      DO LOOP=1,10
 250      POS=INDEX(TEMP_STR(X:),C_T_ONE(LOOP))
          IF (POS.EQ.0)GOTO 260
```

```
            Y=POS+X-1
            T_ONE(Y:Y+6)='TTTTTTT'
            X=X+POS
            GOTO 250
260         CONTINUE
            X=1
          END DO
          END

          SUBROUTINE CONV_STRING  (STRING)
c
c convert characters in the string into uppercase.
c searches for lowercase letters in the string & if found, replaces these with
c uppercase letters. this will be fast if there are few lowercase letters but slow
c if the majority of the string is lowercase.
c
          CHARACTER*(*) STRING
          INTEGER X,POS
          CHARACTER*1 CH
          X=97
          DO WHILE (X.LT.123)
            CH=CHAR(X)
310         POS=INDEX(STRING,CH)
            IF (POS.EQ.0)GOTO 320
            STRING(POS:POS)=CHAR(X-32)
            GOTO 310
320         X=X+1
          END DO
          END
```

Appendix H The FORTRAN source code for the Mix'n'MatchA program

```
      PROGRAM mixnmatcha
      IMPLICIT none
c****************************************
c
c the program for multiple alignment.
c (c) Lachlan H. Bell
c       changed 11/5/93
c
c The mix'n'match suite of programs optimise protein multiple alignment
c by a process of finding Strongly Conserved Regions (SCRs) that can be used
c to anchor an alignment and then aligning the regions between these anchors.
c The anchor regions are found by analysis of initial alignments, previously
c generated using one or more automatic multiple alignment programs and a
c variety of scoring matrices and gap penalties. The suite of programs
c consists of two programs: Mix'n'MatchA (this beaut), which finds the SCRs,
c works out the different possible alignments for each of the Loosely
c Conserved Regions (LCRs), predicts the secondary structure or reads in
c the predictions if they have already been carried out and works out a
c consensus structure prediction for each of these possible alignments;
c and Mix'n'MatchB which takes the alignments which the user has chosen for
c each LCR and the alignments for the SCRs and joins them to produce the final
c alignment. The UWGCG suite of programs is also required. Two files are used
c as iput to MNMA. The first is a file containing all the pre-generated
c alignments. The alignment formats supported are those of ALIEN and PILEUP.
C The second input file lists the sequences present in the multiple alignments.
c This sequence input file can be in either NBRF or UWGCG 'file of file names'
c format
c
c
c****************************************
c
c If you change the max no. of sequences, length of sequences or
c no. of 2ry structure predictions methods in this program, you must
c change them in program MMB also.
c

c maximum number of sequences allowed in an alignment.
      INTEGER maxseqnum
      PARAMETER (maxseqnum=30)

c maximum length of sequences allowed in an alignment.
      INTEGER maxseqlen
      PARAMETER (maxseqlen=2000)

c maximum number of alignments allowed in the alignment input file
c this value is set so that other arrays that store values for each alignment
c do not run out of array space. These arrays are; idscores, alitord, idmaxs.
      INTEGER maxalisnum
      PARAMETER (maxalisnum=100)

c maximum number of SCR that can be found
c this value is set so that other arrays that store values for each scr
c do not run out of array space. This array is scrpos
      INTEGER maxnumscr
      PARAMETER (maxnumscr=100)

c the number of different secondary structure predictions carried out
      INTEGER no_preds
      PARAMETER (no_preds=5)

c the length of the input sequence names and the length of the
c secondary structure prediction methods names
```

```
c
c please note . If you change the length of these strings, you must
c change the length of the equivalent strings in MMB also.
c You will also have to change the FORMAT statements in the routine
c WRARRAY in the COMMON.FOR file.
        INTEGER namlen
        PARAMETER (namlen=11)

c the arrays that hold the names of the sequences in each individual
c alignment - used for checking that the sequences are in the same order
c in each alignment
        CHARACTER*(namlen) ordera(maxseqnum+no_preds)
        CHARACTER*(namlen) orderb(maxseqnum)

c holds the name of the file that each secondary structure prediction
c methods' will be output to. prednam must be the same length as ordera for
c the second program's sake
        CHARACTER*(namlen) prednam(no_preds)


c the arrays that hold the alignments and secondary structure predictions
c for the sequences in an alignment
        CHARACTER*(maxseqlen) arr_a(maxseqnum+no_preds)
        CHARACTER*(maxseqlen) arr_b(maxseqnum+no_preds)
        CHARACTER*(maxseqlen) arr_c(maxseqnum), seq_a, seq_c

c unify the names of the output files for the LCR and SCR data.
        CHARACTER*(*) lcrtmpa, lcrtmpb, scrout, lcrout
        PARAMETER (lcrtmpa='lcr.tmpa', lcrtmpb='lcr.tmpb',
     & lcrout='lcr.data', scrout='scr.data')

c certain messages that are passed to individual routines
        CHARACTER*(*) messa, messd
        PARAMETER(
     &messa=' This is not a major problem - the limit on this number
     & was only placed for programming reasons (to allow other
     & variables to be initialised) - however, you will have to change
     & this number before re-running.')
        PARAMETER (
     &messd=' To change this, edit the program
     &( type ''edit MMA.FOR''). The variable will be given a value in a
     & line that starts with the word ''parameter''. Increase the
     & variable to a number greater than the number you are using as
     & input. Exit from editing the program. Then recompile, relink and
     & you are then ready to run the program again.')

c a string containing asterices, used to seperate some of the screen info.
        CHARACTER*132 asterix
c These strings contain the names of the input files
        CHARACTER*20 outfile, seqname, alisname
c alisgap is the gap char used by alien,
c nopredchar is the char used when neither sheet, helix or turn is pred 2ry.
        CHARACTER*1 alisgap, nopredchar
c string used for char input in the main routine
        CHARACTER*3 strinp
c this string is needed for testing the SCR's. It must be at least as wide
c as the maximum length of the scr width. This is set in the routine
c altvar. any attempt to modify scrwidth greater than 10 is stopped.
        CHARACTER*10 hitstring

c the minimum width of a scr
        INTEGER scrwidth

c number of alignments found in the alignment file
        INTEGER totalifnd

c number of sequences found in the alignment
        INTEGER totseqfnd

c number of lcr's found
```

```
      INTEGER totlcrfnd

c number of scr's found
      INTEGER totscrfnd

c the length of the longest alignment found - operations on the arrays only
c have to be done up to this length.
      INTEGER maxlenfnd

c some variables used in the main program
      INTEGER pos, x, len_a, row

c some of the program variables - width of screen, output device,
c number of alis and seqs that will be used to calculate the SCRs from.
      INTEGER screenwidth, outwidth, numscrali, numscrseq

c array that will hold the identity scores found for each alignment
      INTEGER idscores(maxalisnum)

c array that holds the alignment numbers that have to be used
c to calculate the SCR's from
      INTEGER alitord(maxalisnum)

c array that will hold the highest to lowest identity scores, listed in idscores
      INTEGER idmaxs(maxalisnum)

c array that holds the sequences, to be used to calculate the SCR's from
      INTEGER seqstord(maxseqnum)

c this array will hold the sequence data in numerical form, as needed to
c calculate the GGBSM 2ry structure prediction.
      INTEGER seq_b(maxseqlen)

c this will hold the start (1,x) and end (2,x) positions of every SCR found
      INTEGER scrpos(2,maxnumscr)

c specify unit numbers here to help in avoiding reading/writing to the
c same channel.
      INTEGER unit_a
      INTEGER unit_b
      INTEGER unit_c
      INTEGER unit_d
      INTEGER alifil
      INTEGER debugI
      INTEGER screenwr

      LOGICAL crash, gcg, pred, debugL, extra
c
      data asterix/'*******************************************
     &*************************************************************
     &************************'/
      data seqstord /maxseqnum*0/
      data idmaxs /maxalisnum*-1/
      data prednam(1) /'GGBSM.DATA '/, prednam(2) /'CF.DATA     '/
      data prednam(3) /'GOR.DATA   '/, prednam(4) /'GOR2.DATA   '/
      data prednam(5) /'GOR3.DATA  '/
c
      COMMON /a1/ maxlenfnd
      COMMON /scrnwrite/ screenwr, screenwidth
      COMMON /a3/ outwidth
      COMMON /a4/ totseqfnd
      COMMON /a5/ gcg
      COMMON /a6/ crash
      COMMON /a7/ numscrseq
      COMMON /a8/ totalifnd
      COMMON /a11/ alifil
      COMMON /a12/ unit_a
      COMMON /a13/ unit_b
      COMMON /a14/ unit_c
      COMMON /a15/ unit_d
```

```
      COMMON /b1/ pred
      COMMON /b2/ totscrfnd
      COMMON /b3/ totlcrfnd
      COMMON /c1/ debugL, debugI
      COMMON /d1/ numscrali
      COMMON /d3/ scrwidth
      COMMON /d4/ alisgap
      COMMON /d5/ nopredchar
 10   FORMAT (a)
c
      CALL desc_prog
c
c input the names of the files used in program
c
      CALL input (alisname, seqname, debugL,
     & maxalisnum, maxseqnum)
      IF (crash) GOTO 9998
      pos=index(alisname,'.')
      outFILE=alisname(1:pos)//'zzzz'
      IF (debugL)
     & OPEN (debugI,FILE=alisname(1:pos)//'debug', STATUS='new')
c
c reformat sequences to be in GCG format, ready for
c peptidestructure/pepplot. do now because when you read in the first
c alignment we carry out checks on the sequences to make sure the
c necessary sequence files are present.
c
      IF (.NOT.gcg)THEN
        CALL wrwords ('Reformatting the sequences from NBRF/PIR
     & format to GCG format.', screenwr, screenwidth)
         CALL reform (seqname)
      ENDIF
c
c now read in the alignments. the formats readable are those from
c alien and pileup.
c
      CALL wrwords (asterix(1:screenwidth),screenwr, screenwidth)
      CALL wrwords ('Reading in the alignments.',
     &screenwr, screenwidth)
      CALL doalis (alisname, ordera, orderb, idscores, maxalisnum,
     &maxseqnum, asterix, arr_a, seq_a, outfile,
     &messa, messd)
      IF (crash) GOTO 9997
c
c predict the secondary structure for the sequences
c
      CALL wrwords (asterix(1:screenwidth),screenwr, screenwidth)
      CALL wrwords ('Predicting the secondary structure.',
     &screenwr, screenwidth)
      CALL second (ordera, seq_b, seqname, seq_c(1:maxlenfnd),
     & prednam, arr_a, seq_a(1:maxlenfnd))
c
c if reform was called above, extra files were created - so delete them.
c
      IF (.NOT.gcg)THEN
        DO row=1,totseqfnd
          pos=index(ordera(row),' ')-1
          OPEN (1,FILE=ordera(row)(1:pos)//'.gcg',STATUS='old')
          CLOSE (1,STATUS='delete')
        END DO
      ENDIF
      IF (crash) GOTO 9996
c
c calculate the consensus secondary structure predictions for each
c alignment, by each of the different prediction methods.
c
      DO x=1,no_preds
        CALL calc_cons (arr_b, arr_c, prednam(x), ordera, arr_a,
     &seq_a, outfile)
      END DO
```

```
c
c now calculate the scr's. The alignment from which the scr's are taken
c is held in arr_a
c
 7786   CONTINUE
        CALL wrwords (asterix(1:screenwidth),screenwr, screenwidth)
        CALL wrwords ('Calculating the Strongly Conserved Regions.',
     & screenwr, screenwidth)
        CALL wrwords (asterix(1:screenwidth),screenwr, screenwidth)
        CALL scr (alisname, ordera, arr_b, seq_c, idscores,
     &idmaxs, seqstord, len_a, arr_a, seq_a, asterix, maxalisnum,
     & maxseqnum, outfile, alitord, scrpos, maxnumscr,
     & messa, messd, hitstring)
        IF (crash) GOTO 9999
c
c now find out the lcr's
c
        CALL wrwords (asterix(1:screenwidth),screenwr, screenwidth)
        CALL wrwords ('Finding the Loosely Conserved Regions',
     & screenwr, screenwidth)
        CALL fndlcr (arr_b, len_a, seqstord, lcrtmpa, arr_a, seq_a,
     & outfile, scrpos)
c
        CALL wrwords (asterix(1:screenwidth),screenwr, screenwidth)
        CALL wrwords ('Finding the unique LCR''s.', screenwr, screenwidth)
        CALL uniq (arr_b, arr_c, seq_c, lcrtmpa, lcrtmpb)
c
c write the data for the lcr's
c
        DO x=1, no_preds
          ordera(totseqfnd+x)=prednam(x)
        END DO
        IF (numscrseq.NE.totseqfnd)THEN
          extra=.TRUE.
          DO x=1,totseqfnd
            seqstord(x)=x
          END DO
        ELSE
          extra=.FALSE.
        ENDIF
        CALL wrwords (asterix(1:screenwidth), screenwr, screenwidth)
        CALL wrwords ('Writing out the unique Loosely Conserved Regions
     & (lcr.data) for use by the program MMB.', screenwr, screenwidth)
        CALL wrlcr1 (arr_b, seq_c, ordera, seqstord, lcrout, lcrtmpb,
     & prednam, no_preds, outfile, extra, arr_a)
c
c now write out the scr's that have been found
c
        CALL wrwords (asterix(1:screenwidth),screenwr, screenwidth)
        CALL wrwords ('Writing out the Strongly Conserved Regions
     & (scr.data) for use by the program MMB.', screenwr, screenwidth)
        CALL wr_scr (len_a, seqstord, scrout, ordera, prednam,
     &no_preds, arr_a, seq_a, outfile, alitord, scrpos, extra)
c
c can repeat the finding out of SCR's and LCR's if you wish
c
        CALL wrwords (asterix(1:screenwidth), screenwr, screenwidth)
        CALL wrwords ('Do you want to try and get different SCR
     & regions [no]:', screenwr, screenwidth)
        READ (*,10) strinp
        IF (strinp(1:1).EQ.'y'.or.strinp(1:1).EQ.'Y') GOTO 7786
c
 9999   CONTINUE
        CALL wrwords (asterix(1:screenwidth),screenwr, screenwidth)
        CALL wrwords ('Deleting the work files generated by this
     & program.', screenwr, screenwidth)
        DO x=1, no_preds
          OPEN (1,FILE=prednam(x),STATUS='old',form='unformatted')
          CLOSE (1,STATUS='delete')
        END DO
```

```
c If there is an error in the scr routine, we jump to 9999. However, we
c do not want the next message printed if there is a crash, so we jump
c over the next bit
        IF (crash) GOTO 9996
c
c Write a wee explanation of what this program has done, and what needs to
c be done next
c
        CALL wrwords (asterix(1:screenwidth), screenwr, screenwidth)
        CALL wrwords ('For each Loosely Conserved Region (the region
     &in between  the  'anchor'' SCR''s) there will be a choice of
     & differing alignments blocks. The unique blocks found for each
     & LCR are in the file ''PRINT.MB''. Print this file and choose
     & which one of the alignments blocks is ''best'' for each
     & of these regions.', screenwr, screenwidth)
        CALL wrwords ('Then run the program MMB, which will ask
     & you which blocks you have picked for each LCR and then produce
     & a final alignment ( in the file ''PRINT.ME2'').',
     & screenwr, screenwidth)
c
 9996   CONTINUE
        OPEN (UNIT=alifil,FILE=outfile,STATUS='old')
        CLOSE (alifil,STATUS='delete')
c
 9997   CONTINUE
        IF (debugL) CLOSE (debugI)
c
 9998   CONTINUE
        STOP
        END

        BLOCK DATA init
        INTEGER screenwidth, outwidth, numscrali, numscrseq
        INTEGER unit_a
        INTEGER unit_b
        INTEGER alifil
        INTEGER unit_c
        INTEGER unit_d
        INTEGER debugI
        INTEGER screenwr
        INTEGER totalifnd
        INTEGER maxlenfnd
        INTEGER scrwidth
        CHARACTER*1 alisgap, nopredchar
        data alisgap /'-'/, nopredchar /'.'/
        data maxlenfnd/0/, screenwidth/80/, outwidth/131/
        data alifil/1/, debugI/15/
        data unit_a/2/, unit_b/3/, unit_c/4/, unit_d/7/
        data screenwr/6/
        data numscrali/15/, numscrseq/10/
        data totalifnd /0/
        data scrwidth /3/
        LOGICAL crash, gcg, pred, debugL
        data gcg /.FALSE./
        data pred /.FALSE./
        data debugL /.FALSE./
        data crash /.FALSE./
        COMMON /a1/ maxlenfnd
        COMMON /scrnwrite/ screenwr, screenwidth
        COMMON /a3/ outwidth
        COMMON /a4/ totseqfnd
        COMMON /a5/ gcg
        COMMON /a6/ crash
        COMMON /a7/ numscrseq
        COMMON /a8/ totalifnd
        COMMON /a11/ alifil
        COMMON /a12/ unit_a
        COMMON /a13/ unit_b
        COMMON /a14/ unit_c
        COMMON /a15/ unit_d
```

```
      COMMON /b1/ pred
      COMMON /b2/ totscrfnd
      COMMON /b3/ totlcrfnd
      COMMON /c1/ debugL, debugI
      COMMON /d1/ numscrali
      COMMON /d3/ scrwidth
      COMMON /d4/ alisgap
      COMMON /d5/ nopredchar
      END

      SUBROUTINE reform (seqname)
      IMPLICIT none
c
c takes an NBRF format sequence file, called seqname and reformats it to GCG
c format by spawning a GCG command.
c
      CHARACTER*(*) seqname
      INTEGER pos, c
      CHARACTER*80 cmd1
      pos=index(seqname,' ')-1
      c=pos+19
      cmd1='frompir/in='//seqname(1:pos)//'/default'
      CALL gcgcmds (c,cmd1,'gcg_spawn')
      END

      SUBROUTINE check_gcg (ordera)
      IMPLICIT none
c
c checks that the sequence files named in the array ordera exist.
c the files are all assumed to have the suffix '.gcg'
c
      INTEGER row, totseqfnd, pos, screenwr, screenwidth
      LOGICAL oldfil, crash
      CHARACTER*(*) ordera(*)
      COMMON /scrnwrite/ screenwr, screenwidth
      COMMON /a4/ totseqfnd
      COMMON /a6/ crash
      DO row=1,totseqfnd
        CALL mkcapb (ordera(row))
        pos=index (ordera(row),' ')-1
        inquire (FILE=ordera(row)(1:pos)//'.gcg',exist=oldfil)
        IF (.NOT.oldfil)THEN
          CALL wrwords ('A sequence file named in the
     & alignments could not be found.
     & The sequences must have different names in the alignment and
     & sequence files.', screenwr, screenwidth)
          crash=.TRUE.
        ENDIF
      END DO
      IF (crash) CALL wrwords ('Please change the sequence names
     & so that the names found in the alignments and the
     & sequence file agree. This problem can occur because the
     & initial alignment program may have changed the sequence
     & name. NB ALIGN truncates sequence names if they are more than
     & 6 characters long. Or, if your sequence input file was in NBRF
     & format, the reformatting of it to GCG format
     & (done by this program with the GCG command FROMPIR),
     & may have changed the sequence names (of course if you have not
     & initialised the GCG package before running this program, then
     & FROMPIR will not work and therefore the sequences will not be
     & found). Or, if your sequence input file was in GCG format, ensure
     & that the files all have the suffix ''.GCG''. We have to assume
     & that all sequence files have this suffix because that is what
     & FROMPIR appends to sequence names, and otherwise we would never
     & be able to find them ', screenwr, screenwidth)
      END

      SUBROUTINE desc_prog
      INTEGER screenwr, screenwidth
c
```

```
      COMMON /scrnwrite/ screenwr, screenwidth
c prints out a short desciption of the program
      CALL wrwords ('The Mix''n''Match suite of programs optimise
     & protein multiple alignment by a process of finding
     & Strongly Conserved Regions (SCRs) that can be used to anchor
     & an alignment, and then aligning the Loosely Conserved Regions
     & (LCRs) between these anchors. The anchor regions
     & are found by analysis of initial alignments, previously
     & generated
     & using one or more automatic multiple alignment programs and a
     & variety of scoring matrices and gap penalties.',
     & screenwr, screenwidth)
      CALL wrwords ('Mix''n''MatchA
     & finds the SCRs, works out the different possible alignments for
     & each of the Loosely Conserved Regions (LCRs), predicts the
     & secondary structure - or reads in the predictions if they have
     & already been carried out - and works out a consensus structure
     & prediction for each of these possible alignments, and
     & Mix''n''MatchB produces the final alignment.',
     & screenwr, screenwidth)
      CALL wrwords ('Two files are used as
     & input to MNMA. The first is a file containing all the
     & pre-generated alignments. The alignment formats supported
     & are those of ALIEN and PILEUP. The second input file lists
     & the sequences
     & present in the multiple alignments. This sequence input file
     & can be in either NBRF or UWGCG ''file of file names'' formats.',
     & screenwr, screenwidth)
      CALL wrwords ('The maximum number of
     & alignments in the input alignment file is set at 100. The
     & maximum number of protein sequences in an alignment is set at
     & 30, with the maximum length of any sequence set at 2000,
     & inclusive of any gaps
     & that may have been introduced by the alignment process.',
     & screenwr, screenwidth)
 30   FORMAT (a)
      WRITE (screenwr, 30) '             (c) Lachlan Bell 1992 '
      END

      SUBROUTINE title
 10   FORMAT (a)
        WRITE (6, 10)' l            a        cccc    h     h   l       a       nn
n'
        WRITE (6, 10)' l          a a      c    c   h     h   l     a a      nnn
n'
        WRITE (6, 10)' l         a   a     c        hhhhhhh   l     a   a     n n
n'
        WRITE (6, 10)' l        aaaaaaa    c    c   hhhhhhh   l    aaaaaaa    n
n  n'
        WRITE (6, 10)' llll     a       a   c    c   h     h   llllll   a       a
n    nnn'
        WRITE (6, 10)' llllll   a       a    cccc   h     h   llllll   a       a
n     nn '
        WRITE (6, 10)' '
        WRITE (6, 10)'         bbbbb             eeeeeeeee        ll          ll'
        WRITE (6, 10)'         bb   bb           eeeeeeee         ll          ll'
        WRITE (6, 10)'         bb   bb           cc               ll          ll'
        WRITE (6, 10)'         bbbbb             eeeeee           ll          ll'
        WRITE (6, 10)'         bb   bbb          eeeeee           ll          ll'
        WRITE (6, 10)'         bb      bb        ee               ll          ll'
        WRITE (6, 10)'         bb      bb        ee               ll          ll'
        WRITE (6, 10)'         bb     bb         eeeeeeee         llllll
llllll  ..'
        WRITE (6, 10)'         bbbbbbb           eeeeeeeee        lllllll
lllllll ..'
        WRITE (6, 10)'                 (c) lachlan h. bell 17/12/91'
      END

      SUBROUTINE mkcapb (string)
      IMPLICIT none
```

```
c
c this routine makes all the letter characters in string capitals.
c checks every position of the string and checks it isn't a small letter
c this routine will be faster for small strings,
c mkcapc faster for large strings.
c
        CHARACTER*(*) string
        INTEGER x, y
 310    do x=1,len(string)
          y=ichar(string(x:x))
          IF (y.GT.96.AND.y.LT.123)THEN
            string(x:x)=char(y-32)
          ENDIF
        END DO
        END

        SUBROUTINE mkcapc (string, name)
        IMPLICIT none
c
c this routine takes a sequence string and capitalises all the
c residue letters.
c looks for occurence in the string of small letter, and makes them capitals
c or cysteine residues (C)
c
        CHARACTER*(*) string, name
        CHARACTER*(*) str1, str2, str3, str4, str5, str6
        INTEGER x, pos, numinp
        INTEGER screenwidth, screenwr
        LOGICAL miss
        COMMON /scrnwrite/ screenwr, screenwidth
        PARAMETER (str1=' the non-capital letter ',
     &str2=' has been found in the sequence, ',
     &str3=' 2) for making this a cysteine ''C''.',
     &str4=' 3) for leaving it as the same residue type.',
     &str5=' 4) for leaving all small letters as the same residue type',
     &str6=' 1) for more information/help.')
 301    FORMAT (a,a1,2a)
 305    FORMAT (a,/,a,/,a,/,a,/,a)
 302    FORMAT (a)
 304    FORMAT (i)
        miss=.FALSE.
        DO x=97,122
 310            CONTINUE
          pos=index(string,char(x))
          IF (pos.EQ.0)GOTO 320
c small letter found
          IF (miss) GOTO 318
 315            CONTINUE
          WRITE (6,301) str1, char(x), str2, name
          WRITE (6,305) ' enter', str6, str3, str4, str5
 317            CONTINUE
          READ (*,304) numinp
          IF (numinp.LT.1.or.numinp.GT.4)THEN
            CALL wrwords ('Error in input number,
     & please reenter, range allowed is 1-4.', screenwr, screenwidth)
            GOTO 317
          ENDIF
          IF (numinp.EQ.1)THEN
            CALL help
            GOTO 315
          ENDIF
          IF (numinp.EQ.2)THEN
            string(pos:pos)='C'
            GOTO 310
          ENDIF
          IF (numinp.EQ.4) miss=.TRUE.
 318            CONTINUE
          string(pos:pos)=char(x-32)
          GOTO 310
 320            CONTINUE
```

```
        END DO
        END

        SUBROUTINE rem_space (string)
        IMPLICIT none
 10     FORMAT (a)
c this routine accepts a string and removes all spaces and numbers
c from it
c maxpos is the limit up to which to test  but can only really be used when
c there is an end-of-sequence marker (like * in NBRF sequence data)- this
c stops you testing the entire length of the string which may be set at a
c far greater number than the length of interest.
c it finds the position of the first space, counts along until
c the next alpha or '*' or '.' (the gap in pileup format) char is found
c and removes everything in between.
c this is repeated until no next char or until maxpos is reached.
c if no alpha character is found, then no removal occurs.
c
        CHARACTER*(*) string, test*1
        INTEGER pos, max_pos, x, y
        max_pos=index(string,'*')
        IF (max_pos.EQ.0) max_pos=len(string)
 4010   CONTINUE
        pos=index(string(:max_pos),' ')
        IF (pos.EQ.0) GOTO 4030
        x=1
c set y to position after first found space
        y=pos+x
 4020   CONTINUE
        IF (y.GT.max_pos) GOTO 4030
        test=string(y:y)
c doesn't really remove all but alphabetics - leaves in
c [\]^_`{|}~
c also the delete character
c
        IF ((lge(test,'A')).or.test.EQ.'*'.or.test.EQ.'.')THEN
c IF at position y there is a alpha char, remove the space in between
            string(pos:)=string(y:)
            max_pos=max_pos-x
            GOTO 4010
        ELSE
c IF at position y there's not a alpha char, make y point 1 further along string
            x=x+1
            y=pos+x
            GOTO 4020
        ENDIF
 4030   CONTINUE
        END
c alternative way of testing
c       tmp=ichar (string(y:y))
c       IF (tmp.LT.65.AND.tmp.NE.42)THEN

        SUBROUTINE reorder (no_chosen, numarr)
        IMPLICIT none
c
c this routine reorders numbers, held in array numarr, so that they are
c smallest to largest. 2 methods thought of.
c there are no_chosen numbers in the array to reorder.
c 1) go along array until you find a number out of sequence.
c 2) go from start of array and find out where this out-of-order number fits in
c 3) move all the intervening numbers along
c 4) put out-of-order number in place
c
c a) move along array and swop any number found in wrong order.
c b) go from start again and repeat 1 until all array done
c
c when arr_a/13,15,45,9,44/,arr_b /13,9,45,47,11/
c --> m1 takes 90 & 96 steps in debugger. m2 takes 69 & 69 steps.
c when= 13,15,17,19,21,28,30,32,37,43,44,45,51,52,53,9,11,23,29,31,33,46
c --> m1<350, m2>3550
```

```
c method one is chosen
        INTEGER x, a, b, c, no_chosen, numarr(*), temp
        DO x=1,no_chosen-1
            c=x+1
            IF (numarr(x).GT.numarr(c))THEN
          temp=numarr(c)
          IF (x.EQ.1)THEN
              a=1
              GOTO 7210
          ENDIF
          DO a=1,x
              IF (temp.LT.numarr(a))GOTO 7210
          END DO
 7210           CONTINUE
          DO b=c,a,-1
              numarr(b)=numarr(b-1)
              IF (b-1.EQ.a)GOTO 7215
          END DO
 7215           CONTINUE
          numarr(a)=temp
            endif
        END DO
        END
c
c       x=1
c 10    a=x+1
c       IF (numarr(x).LT.numarr(a))GOTO 20
c       temp=numarr(x)
c       numarr(x)=numarr(a)
c       numarr(a)=temp
c       x=1
c       GOTO 10
c 20    x=x+1
c       IF (x.LT.no_chosen)GOTO 10
c

        SUBROUTINE doalis (alisname, ordera, orderb, idscores,
     & maxalisnum, maxseqnum, asterix, arr_a, seq_a, outfile,
     & messa, messd)
        IMPLICIT none
c
c read in the alignments and output them in machine-readable form.
c
        CHARACTER*(*) alisname, ordera, orderb, arr_a(*), seq_a
        CHARACTER*(*) asterix, outfile
        CHARACTER*(*) errora, errord, errore
        CHARACTER*(*) messa, messc, messd, messk
        CHARACTER string*6, blank*1
        INTEGER unit_a, unit_b, alifil, totalifnd, maxalisnum, maxseqnum
        INTEGER idscores(*), maxlenfnd
        INTEGER totseqfnd, pos, type, screenwidth
        INTEGER maxseqlen, screenwr
        LOGICAL crash
        PARAMETER (
     &messc=' The number allowed is stored in the variable maxseqnum.',
     &messk=' The length allowed is stored in the variable maxseqlen.',
     &errora='There are not enough sequences for an alignment! This
     & program needs at least 3 sequences in the alignments to work.',
     &errord='There are more sequences in the alignments
     &than are allowed in this program.',
     &errore='The length of the sequences in the alignments file are
     & longer than are allowed in this program.')
        COMMON /a1/ maxlenfnd
        COMMON /scrnwrite/ screenwr, screenwidth
        COMMON /a4/ totseqfnd
        COMMON /a6/ crash
        COMMON /a8/ totalifnd
        COMMON /a11/ alifil
        COMMON /a12/ unit_a
        COMMON /a13/ unit_b
```

```
 10     FORMAT (a)
 20     FORMAT (2(a1,/),a1)
 27     FORMAT (' Finished reading the ',i4,' alignments.')
c
        maxseqlen=len(arr_a(1))
        pos=index(alisname,'.')
        OPEN (unit_a,FILE=alisname,STATUS='old')
        REWIND (unit_a)
        OPEN (unit_b,FILE=alisname(1:pos)//'param_data',STATUS='new')
        OPEN (alifil,FILE=outfile,form='unformatted',STATUS='new')
 90     CONTINUE
        CALL alis_type (type, string, unit_a)
 91     CONTINUE
c file reading error
        IF (type.EQ.0)THEN
          IF (totalifnd.EQ.0)THEN
            CALL wrwords
     &(asterix(1:screenwidth),screenwr, screenwidth)
            CALL wrwords ('No alignments have been found!',
     &screenwr, screenwidth)
            crash=.TRUE.
            GOTO 999
          ENDIF
          WRITE (6,27) totalifnd
          GOTO 999
c
c alignment of unknown type in the alignment file
c
        ELSE IF (type.EQ.-1)THEN
          CALL wrwords
     &(asterix(1:screenwidth),screenwr, screenwidth)
          CALL wrwords ('An unknown alignment type has been found in
     &the alignment file. Only files generated by ALIEN or PILEUP
     & may be used. Please check your alignment file and start again.',
     &screenwr, screenwidth)
          CALL wrwords ('The first 6 characters of this unknown
     & alignment type are ', screenwr, screenwidth)
          CALL wrwords (string, screenwr, screenwidth)
          crash=.TRUE.
          GOTO 999
        ENDIF
 100    CONTINUE
        totalifnd=totalifnd+1
        IF (totalifnd.GT.maxalisnum)THEN
          CALL wrwords ('There are more alignments in the alignment
     &input file than are allowed in this program', screenwr,
     & screenwidth)
          CALL wrwords (messa, screenwr, screenwidth)
          CALL wrwords ('The number allowed is stored in the
     & variable maxalisnum.', screenwr, screenwidth)
          CALL wrwords (messd, screenwr, screenwidth)
          crash=.TRUE.
          GOTO 999
        ENDIF
        READ (unit_a,20) blank,blank,blank
c
        IF (type.EQ.1)THEN
          CALL alien_rd (ordera, orderb, idscores, maxseqnum,
     & arr_a, seq_a, errord, errore, messa, messc, messd,
     & messk, errora, asterix, maxseqlen)
          IF (crash) GOTO 999
          GOTO 90
c
        ELSE IF (type.EQ.2)THEN
          CALL pileup_rd (type, string, ordera, orderb, idscores,
     & maxseqnum, arr_a, seq_a, errord, errore, messa, messc,
     & messd, messk, errora, asterix, maxseqlen)
          IF (crash) GOTO 999
c
c Pileup_rd calls alis_type itself.
```

```
          GOTO 91
        ENDIF
 999    CONTINUE
        CLOSE (unit_a)
        IF (crash)THEN
          CALL wrwords ('This program will now terminate due
     & to an error in the alignment file.', screenwr, screenwidth)
          CLOSE (unit_b,STATUS='delete')
          CLOSE (alifil,STATUS='delete')

        ELSE
          CLOSE (unit_b,STATUS='keep')
          CLOSE (alifil,STATUS='keep')
        ENDIF
        END

        SUBROUTINE alis_type (type, string, unit_a)
        IMPLICIT none
c
c reads first 6 characters of the alignment file.
c the name of the method is stored here.
c type = 1 if ALIEN
c type = 2 if pileup
c type = 0 if file error
c type =-1 if unknown alignment method found
c string is passed back, and can be used by the calling routines to work out
c what kind of error this implies.
c
        CHARACTER*(*) string
        INTEGER type, ifail, unit_a
 888    FORMAT (a)
        READ (unit_a,888,iostat=ifail) string
        IF (ifail.NE.0)THEN
          type=0
        ELSE
          IF (string.EQ.'ALIEN:')THEN
            type=1
          ELSE IF (string.EQ.'PileUp')THEN
            type=2
          ELSE
            type=-1
          ENDIF
        ENDIF
        END

        SUBROUTINE ali_wr (seq_len, aligarr, unit)
        IMPLICIT none
c simple routine to write out the alignments - length first then the array
        INTEGER seq_len, row, totseqfnd, unit
        CHARACTER*(*) aligarr(*)
        COMMON /a4/ totseqfnd
        WRITE (unit) seq_len
        DO row=1,totseqfnd
          WRITE (unit) aligarr(row)(1:seq_len)
        END DO
        END


        SUBROUTINE alien_rd (ordera, orderb, idscores, maxseqnum,
     & arr_a, seq_a, errord, errore, messa, messc, messd,
     & messk, errora, asterix, maxseqlen)
        IMPLICIT none
c
c this routine reads in an alignment from the already open alignment
c file on unit unit_a, calculates the overall number of gaps,
c the overall number of identities and reads in the parameters
c (gap penalties/matrix) which were used to generate the alignment and
c outputs these parameters to the parameter data file, already opened
c on unit_b. the seq. alignment is output to file on unit alifil.
c the FORMAT for alien alignments is sequence name from 1:6
```

```
c(6 chars per line) alignments from 15:76 (62 CHARACTERs per line)
c 63 CHARACTERs are read in so that the last CHARACTER is always a
c blank, which enables the length of the alignment to be calculated.
c
        CHARACTER*(*) ordera(*), orderb(*), arr_a(*), seq_a
        CHARACTER*(*) errora, errord, errore
        CHARACTER*(*) messa, messc, messd, messk
        CHARACTER*(*) asterix
        CHARACTER*80 alis_input
        CHARACTER*60 paras(8)
        CHARACTER*1 alisgap, blank

c
        INTEGER block_no, posa, posb
        INTEGER row, totseqfnd, length, totgapfnd
        INTEGER totalifnd, idscores(*), maxlenfnd, maxseqnum
        INTEGER unit_a, unit_b, alifil, maxseqlen, screenwr, screenwidth
        LOGICAL crash
        COMMON /a1/ maxlenfnd
        COMMON /scrnwrite/ screenwr, screenwidth
        COMMON /a4/ totseqfnd
        COMMON /a6/ crash
        COMMON /a8/ totalifnd
        COMMON /a11/ alifil
        COMMON /a12/ unit_a
        COMMON /a13/ unit_b
        COMMON /c4/ alisgap
 5001   FORMAT (a)
 5002   FORMAT (a1,/,a1)
 5003   FORMAT (' This error took place in alignment number ',i)
 5004   FORMAT (a40,5(/,a40) )
 5005   FORMAT (/,' ALIEN parameters read from alignment number  ',i)
 5006   FORMAT (a60,/,a60 )
 5007   FORMAT (a,2(/,a))
 5008   FORMAT (3(tr1,a40),/,3(tr1,a40),/,2(tr1,a60))
 5009   FORMAT (' Total number of gaps is',i5,
     &'  Total number of id''s is',i5,' Alignment length is',i5)
 5011   FORMAT (a,8(/,a))
        READ (unit_a,5002) blank,blank
        block_no=1
c read in the first block of aligned sequecnes.
        IF (totalifnd.EQ.1)THEN
c read in the sequence names in the order they will be used
c in every alignment.
          totseqfnd=0
 5010           CONTINUE
          READ (unit_a,5001) alis_input
c check if all the sequences are read in - this would be '  '
          IF (alis_input(1:2).EQ.'  ') GOTO 5015
          totseqfnd=totseqfnd+1

c now check to see if the array is about to be bust, and halt if it is
          IF (totseqfnd.GT.maxseqnum)THEN
            CALL wrwords (errord,screenwr, screenwidth)
            CALL wrwords (messa,screenwr, screenwidth)
            CALL wrwords (messc,screenwr, screenwidth)
            CALL wrwords (messd,screenwr, screenwidth)
            crash=.TRUE.
            GOTO 5050
          ENDIF
c
c put the aligned sequences into array
          arr_a(totseqfnd)(1:63)=alis_input(15:77)
c
c put the names of the sequences into array
          ordera(totseqfnd)=alis_input(1:10)
          GOTO 5010
 5015           CONTINUE
c
c check if enough sequences
```

```
          IF (totseqfnd.LT.3)THEN
            CALL wrwords (errora, screenwr, screenwidth)
            crash=.TRUE.
            GOTO 5050
          ENDIF
c
c have already checked that gcg files exist but only if originally gcg format
c and only using the names as given in the sequence files, but only a very
c surface check. what has to be checked is if the names of the sequences in
c the alignments (ordera) corresponds to the names of actual sequence files.
c
          CALL check_gcg (ordera)
          IF (crash) GOTO 5055
        ELSE
c
c read in the sequence names in the order they are used in this alignment.
          DO row=1,totseqfnd
            READ (unit_a,5001)alis_input
            arr_a(row)(1:63)=alis_input(15:77)
            orderb(row)=alis_input(1:10)
          END DO
          READ (unit_a,5001) blank
        ENDIF
c
c check if there are any more blocks of sequences to be read in.
 5035   CONTINUE
        READ (unit_a,5001) blank
c
c either read in a new block of aligned sequecnes or find the end.
        READ (unit_a,5001) alis_input
        IF (alis_input(1:3).EQ.'End') GOTO 5037
c
c read in the other blocks of sequences.
        block_no=block_no+1
        posa=(((block_no-1)*62)+1)
        posb=posa+62

c now check to see if the array is about to be bust, and halt if it is
        IF (posb.GT.maxseqlen)THEN
          CALL wrwords (errore,screenwr, screenwidth)
          CALL wrwords (messa,screenwr, screenwidth)
          CALL wrwords (messk,screenwr, screenwidth)
          CALL wrwords (messd,screenwr, screenwidth)
          crash=.TRUE.
          GOTO 5050
        ENDIF
        arr_a(1)(posa:posb)=alis_input(15:77)
c
c the first sequence of a new block has already been read in.
        DO 5030 row=2,totseqfnd
            READ (unit_a,5001)alis_input
            arr_a(row)(posa:posb)=alis_input(15:77)
 5030   END DO
        READ (unit_a,5001) blank
        GOTO 5035
 5037   CONTINUE
        length=index(arr_a(1),' ')-1
        IF (length.GT.maxlenfnd) maxlenfnd=length
c
c in alien, id's and sims are marked - but we only count the identities as
c similarities are matrix dependant
        CALL ident (arr_a, idscores(totalifnd), length, totseqfnd)
c
c some end gaps are represented by '?', change these to the gap symbol '-'
c
        DO row=1,totseqfnd
          CALL changechar (arr_a(row)(1:maxlenfnd), '?', alisgap)
        END DO
c
c calculate the number of gaps
```

```
c
        CALL char_cnt (arr_a, totgapfnd, length, totseqfnd, alisgap)
c
c reading in the alien parameters for the alignment
c
        READ (unit_a,5011)
     & blank,blank,blank,blank,blank,blank,blank,blank,blank
        READ (unit_a,5004)
     &paras(1),paras(2),paras(3),paras(4),paras(5),paras(6)
        READ (unit_a,5007) blank,blank,blank
        READ (unit_a,5006) paras(7),paras(8)
c
c writing the parameter data
c
        WRITE (unit_b,5005) totalifnd
        WRITE (unit_b,5008) paras(1), paras(2), paras(3), paras(4),
     & paras(5), paras(6), paras(7), paras(8)
        WRITE (unit_b,5009) totgapfnd, idscores(totalifnd), length
c
c make sure that the sequences in the present alignment are in the same
c order as in the first alignment
c
        IF (totalifnd.NE.1) CALL order_seqs (ordera, orderb,
     &arr_a, seq_a(1:length), length, asterix)
        CALL ali_wr (length, arr_a, alifil)
c
c jump to here if crash and want to write where error occured
 5050   CONTINUE
        IF (crash) WRITE (6,5003) totalifnd
c
c jump to here if crash and do not want to write where error occured
 5055   CONTINUE
        END

        SUBROUTINE pileup_rd (type, string, ordera, orderb, idscores,
     & maxseqnum, arr_a, seq_a, errord, errore, messa, messc,
     &messd, messk, errora, asterix, maxseqlen)
        IMPLICIT none
c
c The format for pileup alignments is
c sequence names are found from 8:x
c alignments occur from (length of longest seq name +3):
c When alien & pileup alignments are in the alignment file, the longest
c file name is 6 long, as alien truncates the sequence name. However, if
c only pileup alignments are to be used by this program, the length of the
c sequence name, must be taken into consideration. THis is why
c reseqnames is used to get the length of the sequence name
c
        CHARACTER*(*) ordera(*), orderb(*)
        CHARACTER*(*) arr_a(*), seq_a, string
        CHARACTER*(*) errora, errord, errore
        CHARACTER*(*) messa, messc, messd, messk
        CHARACTER*(*) pileupgap
        CHARACTER*(*) asterix
        CHARACTER*20 paras(2), alis_input*200
        CHARACTER*1 alisgap, blank
        INTEGER unit_a, unit_b
        INTEGER maxlenfnd, alifil, totgapfnd, maxseqlen
        INTEGER idscores(*), maxseqnum, screenwr, screenwidth
        INTEGER dummy, dummyb
        INTEGER length, len_b
        INTEGER pos, pos3, row
        INTEGER totseqfnd, type, totalifnd
        INTEGER maxnamlen
        LOGICAL crash
        LOGICAL called
        PARAMETER (pileupgap='.')
        DATA maxnamlen /0/
        DATA called /.FALSE./
        COMMON /a1/ maxlenfnd
```

```
      COMMON /scrnwrite/ screenwr, screenwidth
      COMMON /a4/ totseqfnd
      COMMON /a6/ crash
      COMMON /a8/ totalifnd
      COMMON /a11/ alifil
      COMMON /a12/ unit_a
      COMMON /a13/ unit_b
      COMMON /d4/ alisgap
 4001 FORMAT (a)
 4002 FORMAT (a1,2(/,a1))
 4003 FORMAT (/,' PileUp parameters read from alignment number ',i)
 4004 FORMAT (tr19,a14,/,tr13,a20)
 4005 FORMAT (' This error took place in alignment number ',i)
 4707 FORMAT (' Total number of gaps is',i5,
     &'  Total number of id''s is',i5,' Alignment length is',i5)
 11   FORMAT (tr1,2a20)
      READ (unit_a,4004) paras(1),paras(2)
      READ (unit_a,4002) blank,blank,blank
c
c Now read in the sequences used in the aligments.
      IF (totalifnd.EQ.1)THEN
        CALL rdseqnames (ordera,totseqfnd,unit_a,
     & maxnamlen, maxseqnum, errord, messa, messc, messd, crash)
        IF (crash) GOTO 4050
        called=.TRUE.
        IF (totseqfnd.LT.3)THEN
          CALL wrwords (errora, screenwr, screenwidth)
          crash=.TRUE.
          GOTO 4050
        ENDIF
        CALL check_gcg (ordera)
        IF (crash) GOTO 4055
      ELSE
c If we have already called this routine, then we know the total no. of seqs,
c the maxseqlen, and whether or not a crash will occur
c If we haven't, then you need to now these things. However, because we are
c calling in this routine when (totalifnd<>1), then already some ALIEN
c alignments must have been read in, so we know the total no. of seqs
c and whether or not a crash will occur.
        IF (called) THEN
          CALL rdseqnames (orderb, dummy, unit_a,
     & dummyb, maxseqnum, errord, messa, messc, messd, crash)
        ELSE
          CALL rdseqnames (orderb, dummy, unit_a,
     & maxnamlen, maxseqnum, errord, messa, messc, messd, crash)
          called=.TRUE.
        ENDIF
      ENDIF
      READ (unit_a,4002) blank,blank,blank
      length=1
 4030 CONTINUE
c
c read in the first sequence and determine length from it
      READ (unit_a,4001) alis_input
c
c remove the sequence name and any spaces from the input.
      alis_input(1:)=alis_input(maxnamlen:)
      CALL rem_space (alis_input)
      pos=index(alis_input,' ')-1
      len_b=length+pos-1
      IF (len_b.GT.maxseqlen)THEN
        CALL wrwords (errore,screenwr, screenwidth)
        CALL wrwords (messa,screenwr, screenwidth)
        CALL wrwords (messk,screenwr, screenwidth)
        CALL wrwords (messd,screenwr, screenwidth)
        crash=.TRUE.
        GOTO 4050
      ENDIF
      arr_a(1)(length:len_b)=alis_input(:pos)
c
```

```
c read the other sequences in
        DO row=2,totseqfnd
          READ (unit_a,4001) alis_input
c
c remove the sequence name and any spaces from the input.
          alis_input(1:)=alis_input(maxnamlen:)
          CALL rem_space (alis_input)
          arr_a(row)(length:len_b)=alis_input(:pos)
        END DO
        length=len_b+1
        READ (unit_a,4001) alis_input
        CALL alis_type (type, string, unit_a)
c
c the variable type determines what happens next. If type 0, 1, 2 found
c then we are at the start of the next alignment or there is a file error.
c either way we don't jump back to 4030, but finish this routine & pass
c the type to the calling routine to deal with this type.
c However, if type =-1(unknown type), this is either unknown type or the
c blank space in between sequence blocks in the present pileup alignment,
c ie. we should continue reading this into the present array.
c We presume that if it's a genuine unknown type, then there won't be blank
c spaces for the first 6 characters and so this is used as a marker for it
c being still the same alignment.
c
        IF (type.EQ.-1.AND.string.EQ.'      ') GOTO 4030
        length=length-1
        IF (length.GT.maxlenfnd) maxlenfnd=length
        CALL idcnt (arr_a, idscores(totalifnd), length, totseqfnd)
        DO row=1,totseqfnd
            CALL changechar (arr_a(row)(1:length), pileupgap, alisgap)
        END DO
        CALL char_cnt ( arr_a, totgapfnd, length, totseqfnd, alisgap)
        WRITE (unit_b,4003) totalifnd
        WRITE (unit_b,11) paras(1), paras(2)
        WRITE (unit_b,4707) totgapfnd, idscores(totalifnd), length
c
c make sure that the sequences in the present alignment are in the same
c order as in the first alignment.
c
        IF (totalifnd.NE.1) CALL order_seqs (ordera, orderb,
     &arr_a, seq_a(1:length), length, asterix)
        CALL ali_wr (length, arr_a, alifil)
c
c if crash, jump here if you want where crash occured output
 4050   CONTINUE
        IF (crash) WRITE (6,4005) totalifnd
c
c if crash, jump here if you do not want where crash occured output
 4055   CONTINUE
        END

        SUBROUTINE changechar (string, char1, char2)
        IMPLICIT none
c
c this routine takes finds every occurence of string  'char1' in
c the string string it and replaces it with string  'char2'.
c the lines that have been commented are those that would enable the 2 strings
c to be greater than 1 character long &
c checks that the 2 strings are same length
c
        CHARACTER*(*) string, char1, char2
        INTEGER pos
c       x=len(char2)-1
c       IF (len(char1)-1.NE.x)THEN
c               WRITE (6,FMT='a')' the 2 strings are different lengths!'
c               crash=.TRUE.
c       endif
 1835   CONTINUE
        pos=index(string(1:),char1)
        IF (pos.EQ.0)GOTO 1836
```

```
        string(pos:pos)=char2
c       string(pos:pos+x)=char2
        GOTO 1835
 1836   CONTINUE
        END

        SUBROUTINE char_cnt (array, totchar, searchcol, searchrow, char)
c
c this will count the number of CHARACTERs char, at each position in the
c passed array , the total number of CHARACTERs being stored in totchar.
c searchcol & searchrow tell you how much of the array to search through.
c
        CHARACTER*(*) array(*), char
        INTEGER totchar, searchcol, searchrow, colm, row
        totchar=0
        DO 4070 colm=1,searchcol
          DO 4060 row=1,searchrow
              IF (array(row)(colm:colm).EQ.char) totchar=totchar+1
 4060           END DO
 4070   END DO
        END

        SUBROUTINE idcnt (array, totids, searchcol, searchrow)
        IMPLICIT none
c
c calculate the total number of identical CHARACTERs (in each column) in
c array array, total stored in totids
c searchcol & searchrow tell you how much of the array to search through.
c
        CHARACTER*(*) array(*)
        INTEGER totids, searchcol, searchrow, row, colm
        totids=0
        DO colm=1,searchcol
          DO row=1,searchrow-1
            IF (array(row)(colm:colm).NE.array(row+1)(colm:colm))
     &GOTO 5500
          END DO
          totids=totids+1
 5500   CONTINUE
        END DO
        END

        SUBROUTINE order_seqs (ordera, orderb, array, seq_a,
     & length, asterix)
        IMPLICIT none
c
c this routine reorders array to be in the same order as ordera.
c a list of sequence names are in ordera & orderb.
c
        CHARACTER*(*) ordera(*), orderb(*), array(*), seq_a
        CHARACTER*(*) asterix
        INTEGER totseqfnd, y, length, len_b, row, screenwidth, screenwr
        LOGICAL crash
        COMMON /scrnwrite/ screenwr, screenwidth
        COMMON /a4/ totseqfnd
        COMMON /a6/ crash
c
c make all the names in orderb capitals firstly - this has already been done to
c the names in ordera. this ensures that the same name will be found
c (if present)
        DO row=1,totseqfnd
            CALL mkcapb (orderb(row))
        END DO
c
        len_b=len(ordera(1))
        DO row=1,totseqfnd
c find the value of row and y where the sequences name are the same in
c the 2 arrays
          DO y=row,totseqfnd
            IF (ordera(row).EQ.orderb(y))GOTO 4620
```

```
          END DO
c if the sequence names have changed (a name in orderb cannot be found
c in ordera) - this is major error and must stop program.
          CALL wrwords
     &(asterix(1:screenwidth),screenwr, screenwidth)
          CALL wrwords (' A sequence name has changed between the
     & first alignment and this one. The sequence named ',
     & screenwr, screenwidth)
          CALL wrwords (ordera(row), screenwr, screenwidth)
          CALL wrwords (' could not be found.',screenwr,screenwidth)
          crash=.TRUE.
          GOTO 4630
 4620           IF (row.NE.y)THEN
c swop the sequences
            seq_a(1:length)=array(y)(1:length)
            array(y)(1:length)=array(row)(1:length)
            array(row)(1:length)=seq_a(1:length)
c
c swop the names of the sequences
            seq_a(1:len_b)=orderb(y)
            orderb(y)=orderb(row)
            orderb(row)=seq_a(1:len_b)
          ENDIF
        END DO
 4630   CONTINUE
        END

        SUBROUTINE input (alisname, seqname, debugL,
     & maxalisnum, maxseqnum)
        IMPLICIT none
c
c the routine where the names of the input files are fetched and
c some variables may be altered.
c
        CHARACTER*(*) alisname, seqname
        CHARACTER strinp*3, tmp_nam*20
        LOGICAL gcg, debugL, pred
        LOGICAL oldfil, crash
        INTEGER maxalisnum, maxseqnum, numscrali, numscrseq
        INTEGER ifail, cntr
        INTEGER screenwr, screenwidth
c
c these variables are needed for altvar & say if we are only calling a part
c of altvar, wwhich part - set both for false
c
        LOGICAL bit
        INTEGER dummy
        data bit/.FALSE./, dummy /0/
c
        COMMON /scrnwrite/ screenwr, screenwidth
        COMMON /a5/ gcg
        COMMON /a6/ crash
        COMMON /a7/ numscrseq
        COMMON /b1/ pred
        COMMON /d1/ numscrali
c
 10     FORMAT (a)
 11     FORMAT (a,i4,a)
 12     FORMAT (3a)
 37     CONTINUE
        CALL wrwords ('Input the alignment file name or ?
     &for more information.',screenwr, screenwidth)
        READ (*,10) alisname
        IF (alisname(1:1).EQ.'?')THEN
          CALL help
          GOTO 37
        ENDIF
        inquire (FILE=alisname,exist=oldfil)
        IF (.NOT.oldfil)THEN
          CALL wrwords ('File name problem ie. this file
```

```
     & doesn''t exist!', screenwr, screenwidth)
          GOTO 37
        ENDIF
c
 35     CONTINUE
        CALL wrwords ('Input the sequence file name or ?
     & for more information.',screenwr, screenwidth)
        READ (*,10) seqname
        IF (seqname(1:1).EQ.'?')THEN
          CALL help
          GOTO 35
        ENDIF
        IF (seqname(1:1).EQ.'@')THEN
          inquire (FILE=seqname(2:),exist=oldfil)
        ELSE
          inquire (FILE=seqname,exist=oldfil)
        ENDIF
        IF (.NOT.oldfil)THEN
          CALL wrwords ('File name problem ie. this file
     & doesn''t exist!', screenwr, screenwidth)
          GOTO 35
        ENDIF
c
c if the sequence file is in GCG format (This is a file containing the names
c of the sequence files used), check that the sequence files named in
c the input file exist
c
        IF (seqname(1:1).EQ.'@')THEN
          gcg=.TRUE.
          cntr=0
          OPEN (1,FILE=seqname(2:),STATUS='old')
          REWIND (1)
c
c a check is made of the number of files in the gcg input file so that the user
c can check if this number agrees with the number present in the alignment.
c
          CALL wrwords ('Now checking that the files named in your
     & GCG multiple sequence format file exist.', screenwr, screenwidth)
 39             CONTINUE
          READ (1,10,iostat=ifail) tmp_nam
c
c the 2 below conditions used to be ORed together so that either failure would
c lead to the program stopping. Now seperated so that there can be a blank
c line in the GCG multiple sequence name format file, seperating 2 real sequence
c names, and they still will get read.
c
          IF (ifail.NE.0) GOTO 41
          IF (tmp_nam(1:2).EQ.'  ') GOTO 39
          inquire (FILE=tmp_nam,exist=oldfil)
          IF (.NOT.oldfil)THEN
            CALL wrwords (' A sequence file '
     &//tmp_nam(1:index(tmp_nam,' '))//
     &', named in your GCG format sequence file doesn''t exist! Please
     & check the file '''//seqname(2:index(seqname,' '))//''' .',
     & screenwr, screenwidth)
            crash=.TRUE.
            GOTO 39
          ENDIF
          cntr=cntr+1
          GOTO 39
 41             CONTINUE
          CLOSE (1)
          IF (crash)THEN
            CALL wrwords ('The program will now terminate.
     & Please correct your GCG multiple sequence format file and start
     & again.', screenwr, screenwidth)
            GOTO 43
          ENDIF
          WRITE (6,11)' There were ',cntr,' sequences found
     & in the GCG format file.'
```

```
          CALL wrwords ('If this is different from the number
     & of sequences in your alignments, stop the program and check
     & your GCG input file.' ,screenwr, screenwidth)
c
c write a blank line so that everything is nice and legible.
c
          WRITE (6,FMT='(a1)')' '
        ENDIF
        CALL wrwords ('Have you already run the gcg prediction
     & programs? [no]:',screenwr, screenwidth)
        READ (*,10) strinp
        IF (strinp(1:1).EQ.'y'.or.strinp(1:1).EQ.'Y') pred=.TRUE.
        CALL wrwords ('Do you want debug data to be written? [no]: ',
     &screenwr, screenwidth)
        READ (*,10) strinp
        IF (strinp(1:1).EQ.'y'.or.strinp(1:1).EQ.'Y') debugL=.TRUE.
        CALL wrwords ('Do you want to change any of the program
     & variables? [no]: ',screenwr, screenwidth)
        READ (*,10) strinp
        IF (strinp(1:1).EQ.'y'.or.strinp(1:1).EQ.'Y')
     &CALL altvar (maxalisnum, maxseqnum, dummy, bit)
c
c place the re-initialiseing statements here because if they were above,
c say just below the format statements, then because they are initialised
c before getting passed to this program, you would in effect be initialising
c them twice!
c

        CALL wrwords ('Do you want to change any of the above answers?
     & [no]: ',screenwr, screenwidth)
        READ (*,10) strinp
        IF (strinp(1:1).EQ.'y'.or.strinp(1:1).EQ.'Y')THEN
          debugL=.FALSE.
          pred=.FALSE.
          gcg=.FALSE.
          GOTO 37
        ENDIF
c
 43     CONTINUE
        END

        SUBROUTINE altvar (maxalisnum, maxseqnum, number, bit)
        IMPLICIT none
c
c allows some of the system variables to be altered.
c if bit is true, then we are only altering some of the variables,
c which one is set in number, else we can alter any of the variables
c
        INTEGER screenwidth, outwidth, numscrali
        INTEGER numscrseq, numinp, scrwidth
        INTEGER maxalisnum, maxseqnum, screenwr
        INTEGER number
        LOGICAL bit
        CHARACTER*(*) a, b, d, e, g, message
        CHARACTER*3 string
c
        COMMON /d1/ numscrali
        COMMON /a7/ numscrseq
        COMMON /scrnwrite/ screenwr, screenwidth
        COMMON /a3/ outwidth
        COMMON /d3/ scrwidth
        PARAMETER (a='screen width; ',
     &g='maximum number of characters per line in output files; ',
     &b='number of alignments used to calculate a SCR; ',
     &d='number of sequences used to calculate a SCR; ',
     &e='minimum width of a SCR; ',
     &message=' If you do need this number then you will have to
     & slightly alter the program. Please see the relevant section in
     & the help.')
 11     FORMAT (a)
```

```
 12     FORMAT (i)
 13     FORMAT (3a,i4,a)
c
        IF (bit) THEN
          numinp=number
          GOTO 8888
        ENDIF
 8787   CONTINUE
        IF (bit) GOTO 989
        CALL wrwords ('The variables that may be altered are; ',
     & screenwr, screenwidth)
        WRITE (6,13)' 1) ',a,'[',screenwidth,'] characters.'
        WRITE (6,13)' 2) ',g,'[',outwidth,'] characters.'
        WRITE (6,13)' 3) ',b,'[',numscrali,'] alignments.'
        WRITE (6,13)' 4) ',d,'[',numscrseq,'] sequences.'
        WRITE (6,13)' 5) ',e,'[',scrwidth,'] residues.'
        CALL wrwords ('6) help - what do these variables do?',
     & screenwr, screenwidth)
        CALL wrwords ('7) return to running the program.',
     & screenwr, screenwidth)
        CALL wrwords ('Enter a number now', screenwr, screenwidth)
        READ (*,12) numinp
        IF (numinp.LT.1.or.numinp.GT.7)THEN
          CALL wrwords (' Number out of range - please re-enter
     & a number', screenwr, screenwidth)
          WRITE (screenwr,FMT='(a1)')' '
          GOTO 8787
        ENDIF
 8888   CONTINUE
        GOTO (901,923,934,956,958,978,989) numinp
c
 901    CONTINUE
        CALL changevar (a, screenwidth)
        IF (screenwidth.GT.132.OR.screenwidth.LT.40)THEN
          CALL wrwords ('Are you sure that the screen is so wide
     & (>132 chars) or so narrow (<40 chars)? [no]:', screenwr,
     & screenwidth)
          READ (*,11) string
          IF (string(1:1).EQ.'y'.or.string(1:1).EQ.'Y') GOTO 8787
          GOTO 901
        ENDIF
        GOTO 8787
c
 923    CONTINUE
        CALL changevar (g, outwidth)
        IF (outwidth.GT.132.OR.outwidth.LT.40)THEN
          CALL wrwords ('Are you sure that the width of the device
     & that the final output will be to, is so wide (>132 chars)
     & or so narrow (<40 chars)? [no]:', screenwr, screenwidth)
          READ (*,11) string
          IF (string(1:1).EQ.'y'.or.string(1:1).EQ.'Y') GOTO 8787
          GOTO 923
        ENDIF
        GOTO 8787
c
 934    CONTINUE
        CALL changevar (b, numscrali)
        IF (numscrali.GT.maxalisnum)THEN
          CALL wrwords ('You''ve chosen a number greater than
     & the total number of alignments allowed by this program - please
     & chose a lower number.', screenwr, screenwidth)
          CALL wrwords (message,screenwr, screenwidth)
          GOTO 934
        ENDIF
        IF (numscrali.EQ.1)THEN
          CALL wrwords ('You''ve only chosen one alignment for
     & the SCR''s to be chosen from - this would make the entire
     & alignment the SCR and is a mistake. Please choose again!',
     & screenwr, screenwidth)
          GOTO 934
```

```
        ENDIF
        IF (numscrali.LT.1)THEN
          CALL wrwords ('Whoops! That number is a silly number.
     & Please try again with a sensible number. Your computer thanks
     & you.', screenwr, screenwidth)
          GOTO 934
        ENDIF
        GOTO 8787
c
 956    CONTINUE
        CALL changevar (d, numscrseq)
        IF (numscrseq.GT.maxseqnum)THEN
          CALL wrwords (' You''ve chosen a number greater than
     & the total number of sequences allowed by this program - please
     & chose a lower number.', screenwr, screenwidth)
          CALL wrwords (message,screenwr, screenwidth)
          GOTO 956
        ENDIF
        IF (numscrseq.LT.3)THEN
          CALL wrwords (' You''ve chosen less than 3 sequences
     & for the SCR''s to be worked out from. This would be a pairwise
     & alignment and not a multiple alignment, and so is not allowed.
     & Please chose a higher number.', screenwr, screenwidth)
          GOTO 956
        ENDIF
        GOTO 8787
c
 958    CONTINUE
        CALL changevar (e, scrwidth)
C
c ensure minimum width of scr is between 3 and 10
c for hackers - if you change the minimum width the scr can be, to be above 10,
c then you must also change the length of the string HITSTRING,
c defined in the main program to be 10.
C
        IF (scrwidth.LT.3.OR.scrwidth.GT.10) THEN
          CALL wrwords ('You have chosen a width for the SCR''s
     & that is outside the permitted range of 3-10. You must choose
     & another number. If you really do want to set a width outside
     & this range, then you will have to change the program. Change
     & the values in the routine ''ALTVAR'' after the jump label 958.',
     & screenwr, screenwidth)
          GOTO 958
        ENDIF
        GOTO 8787
c
 978    CONTINUE
        CALL help
        GOTO 8787
c
 989    CONTINUE
        END

        SUBROUTINE changevar (string, int)
        IMPLICIT none
c
c routine for changing program  variables. the variable is in int and a
c short desription of what the variable is held in string.
c
        INTEGER int, temp
        CHARACTER*(*) string
        CHARACTER*3 inp
 10     FORMAT (a1,a)
 11     FORMAT (a)
 12     FORMAT (i)
 14     FORMAT (a,i)
 100    CONTINUE
        WRITE (6,11)' Changing the program variable;'
        WRITE (6,10)' ', string
        WRITE (6,14)' This is presently set at ;',int
```

```
        WRITE (6,11)' Input your new value (you then confirm it).'
        READ (*,12) temp
        WRITE (6,14)' Variable changed to ',temp
        WRITE (6,11)' Is this correct? [yes]:'
        READ (*,11) inp
        IF (inp(1:1).EQ.'n'.or.inp(1:1).EQ.'N') GOTO 100
        int=temp
        END

        SUBROUTINE uniq (arr_b, arr_c, seq_c, lcrtmpa, lcrtmpb)
        IMPLICIT none
c
c in the file lcrtmpa, we have all the loosely conserved regions, alignment
c by alignment. we now need to find just the unique lcr's.
c for each lcr, all the variations found are written to  the file unit_c
c this is passed to routine uniq_lcrs where only the unique variations are kept,
c stored in the file lcrtmpb. this is repeated for each of the lcr's.
c
        CHARACTER*(*) arr_b(*), arr_c(*), seq_c
        CHARACTER*(*) lcrtmpa, lcrtmpb
        INTEGER numseqseq, unit_a, unit_b, totlcrfnd
        INTEGER lcrnum, diff_a, startpos, endpos, alig_no, ifail
        INTEGER unit_c, lcrfnd, x
c
        COMMON /a7/ numseqseq
        COMMON /a12/ unit_a
        COMMON /a13/ unit_b
        COMMON /a14/ unit_c
        COMMON /b3/ totlcrfnd
c
 24     FORMAT (/,' For LCR number ',i3,', there were ',i4,
     &' possible alignments.')
c
        lcrnum=0
        OPEN (unit_a,FILE=lcrtmpa,STATUS='old',form='unformatted')
        OPEN (unit_b,FILE=lcrtmpb,STATUS='new',form='unformatted')
 8310   CONTINUE
        REWIND (unit_a)
        diff_a=0
        lcrnum=lcrnum+1
        IF (lcrnum.GT.totlcrfnd) GOTO 8327
c
c read in the differing alignments found for lcr number lcrnum
c and write them to unit_c
c
        OPEN (unit_c,STATUS='scratch',form='unformatted')
 8320   CONTINUE
        CALL rd_lcr (lcrfnd, startpos, endpos, alig_no, unit_a,
     & numseqseq, arr_b, ifail, 1)
        IF (ifail.NE.0) GOTO 8325
        IF (lcrfnd.EQ.lcrnum)THEN
          CALL wr_lcr (lcrnum, startpos, endpos, alig_no, unit_c,
     & numseqseq, arr_b, 1)
          diff_a=diff_a+1
        ENDIF
        GOTO 8320
c
c all the differing regions for lcr number lcrnum are in unit_c
c find the unique alignments for this lcr and write them to lcrtmpb (unit_b)
c
 8325   CONTINUE
        WRITE (6,24) lcrnum, diff_a
        CALL uniq_lcrs (arr_b, arr_c, seq_c)
        GOTO 8310
 8327   CONTINUE
        CLOSE (unit_a,STATUS='delete')
        CLOSE (unit_b)
        END

        SUBROUTINE uniq_lcrs (arr_b, arr_c, seq_c)
```

```
      IMPLICIT none
c
c a number of different region have been found for lcr number x (at least 1)
c these lcr's have been stored in the scratch file unit_c
c example: test data;
c       1* 2  3* 4  5* 6* 7
c the numbers refer to the alignments these lcr's come from
c (the * after the number means that these are unique regions.)
c 2 & 4 are identical to 1 and 7 is identical to 3
c
c read in first region (#1) from pointera (unit_c), output to unique (unit_b),
c and test first region against the rest.
c any other unique regions are written to pointerb (unit_d),
c at end, unit_d contains; 3* 5* 6* 7.
c then, we close the file the lcr's were read from, pointera (unit_c)
c change the pointers (files _c & _d) around and start again
c
c read in first region (#3) from pointera (unit_d), output to unique (unit_b),
c and test first region against the rest.
c any other unique regions are written to pointerb (unit_c),
c at end, unit_c contains; 5* 6*.
c
c repeat again; unit_d contains; 6*
c repeat again. read in first region (#6) and write to unique
c when you try to read in another lcr, theres no more to read and the routine
c finishes, have transferred 1*, 3*, 5* and 6* to unique file (unit_b)
      CHARACTER*(*) arr_b(*), arr_c(*), seq_c
      INTEGER numscrseq, unit_b, unit_c, unit_d
      INTEGER cntr, x
      INTEGER lcrfnd
      INTEGER startpos, endpos
      INTEGER alig_no, ifail
      INTEGER len_a, len_b
      INTEGER y, hits, dummy, pointera, pointerb, store
      LOGICAL finish
      INTEGER temp, tmparr(200)
      INTEGER debugI
      LOGICAL debugL
      COMMON /a7/ numscrseq
      COMMON /a13/ unit_b
      COMMON /a14/ unit_c
      COMMON /a15/ unit_d
      COMMON /c1/ debugL, debugI
 34   FORMAT (' Alignment number ',i4,
     &' is unique, starting at pos. ',i4,' ends at ',i4)
 35   FORMAT (tr1,a1,i2)
 37   FORMAT (' And of these, ',i4,' were unique.')
 77   FORMAT (tr1,'lcr''s knocked out= ',17(i4,','))
c
c read in alignment from (pointera) & write to (pointerb).
      cntr=0
      pointera=unit_c
      pointerb=unit_d
c
c read in the first lcr from the file containing all the lcr's and write it
c to the unique output ( the first has to be unique)
 2010 CONTINUE
      temp=0
      REWIND (pointera)
      OPEN (pointerb,STATUS='scratch',form='unformatted')
      CALL rd_lcr (lcrfnd, startpos, endpos, alig_no, pointera,
     & numscrseq, arr_b, ifail, 1)
        CALL wr_lcr
     &(lcrfnd, startpos, endpos, alig_no, unit_b, numscrseq, arr_b, 1)
      cntr=cntr+1
      IF (debugL) WRITE (debugI,34) alig_no, startpos, endpos
c test the lcr thats just been read in against the rest to eliminate any
c identical ones.
      finish=.TRUE.
      len_a=endpos-startpos+1
```

```
        seq_c=arr_b(1)(1:len_a)
 2035   CONTINUE
c
c read in the next region to test against.
c if it is different, we will output it.
c If no more to read in, then we've tested all the LCR's for uniqueness
c and can exist.
        CALL rd_lcr (lcrfnd, startpos, endpos, alig_no, pointera,
     & numscrseq, arr_c, ifail, 1)
        IF (ifail.NE.0)THEN
          IF (debugL)THEN
            IF (temp.EQ.1)THEN
               WRITE (debugI,77) tmparr(1)
            ELSE IF (temp.GT.1)THEN
               WRITE (debugI,77) (tmparr(y),y=1,temp)
            ENDIF
          ENDIF
          IF (finish) GOTO 2075
          GOTO 2040
        ENDIF
        len_b=endpos-startpos+1
        IF (len_a.NE.len_b) GOTO 2037
        CALL ident (arr_b, arr_c, 1, len_a, dummy, 1, seq_c(1:len_a),
     & numscrseq, hits)
        IF (hits.NE.0)THEN
c
c store the lcr from alignment number alig_no that is the same as the
c testing lcr so that it can be written with the debug data.
          IF (debugL)THEN
            temp=temp+1
            tmparr(temp)=alig_no
          ENDIF
          GOTO 2035
        ENDIF
c
c a different region !!
 2037   CONTINUE
        finish=.FALSE.
        CALL wr_lcr
     &(lcrfnd, startpos, endpos, alig_no, pointerb, numscrseq, arr_c, 1)
        GOTO 2035
 2040   CONTINUE
c
c finished testing the lcr. read the lcr's from unit_d back to unit_c
c so that the next lcr can be tested for uniqueness.
        CLOSE (pointera)
        store=pointera
        pointera=pointerb
        pointerb=store
        GOTO 2010
 2075   CONTINUE
        WRITE (6,37) cntr
        IF (debugL) WRITE (debugI,37) cntr
        CLOSE (pointera)
        CLOSE (pointerb)
        END

        SUBROUTINE fndlcr (arr_b, len_a, seqstord, lcrtmpa, arr_a,
     & seq_a, outfile, scrpos)
        IMPLICIT none
c
c this routine finds the lcr's and outputs them to the file lcrtmpa.
c diff_a is the overall total of different lcr's found in all the alignments.
c temp & tmparr record for each alignment what lcr's (lcrfnd) were found.
        CHARACTER*(*) arr_a(*), arr_b(*), seq_a, lcrtmpa, outfile
        INTEGER alifil, unit_a, totalifnd, numscrseq, len_a, totseqfnd
        INTEGER seqstord(*), totlcrfnd
        INTEGER alig_no, diff_a, len_b, start_a
        INTEGER lcrfnd, screenwidth, totscrfnd
        INTEGER lcr_start_pos, lcr_end_pos, hits
```

```
      INTEGER scr_cntr, scr_width, len_bb
      INTEGER scrpos(2,*), x, end_a
      INTEGER scr_start, screenwr
      LOGICAL debugL
      INTEGER debugI
      LOGICAL prev_scr
      COMMON /scrnwrite/ screenwr, screenwidth
      COMMON /a4/ totseqfnd
      COMMON /a7/ numscrseq
      COMMON /a8/ totalifnd
      COMMON /a11/ alifil
      COMMON /a12/ unit_a
      COMMON /b2/ totscrfnd
      COMMON /b3/ totlcrfnd
      COMMON /c1/ debugL, debugI
c
 18   FORMAT (17(i4,','))
 19   FORMAT (tr1,i)
 21   FORMAT (a,i2,a)
 22   FORMAT (a,i4,a)
 23   FORMAT (tr1,a80)
      diff_a=0
      OPEN (alifil,FILE=outfile,STATUS='old',form='unformatted')
      OPEN (unit_a,FILE=lcrtmpa,STATUS='new',form='unformatted')
      REWIND (alifil)
c read in an alignment from outfile into arr_b, arr_a holds the alignment where
c the scr positions are known. the number of seqs read in is numscrseq
      DO alig_no=1,totalifnd
        CALL rd_ali2
     &(arr_b, seqstord, totseqfnd, len_b, seq_a, alifil)
c initialise
        prev_scr=.TRUE.
        lcr_start_pos=1
        lcrfnd=0
        scr_cntr=0
c find the position of the scr.
 8210         CONTINUE
        scr_cntr=scr_cntr+1
        IF (scr_cntr.GT.totscrfnd) GOTO 8212
        start_a=scrpos(1,scr_cntr)
        END_a=scrpos(2,scr_cntr)
c see if this scr is found in the test alignment
        IF (start_a.NE.1) lcrfnd=lcrfnd+1
        scr_width=end_a-start_a+1
        len_bb=len_b-scr_width+1
        seq_a=arr_a(1)(start_a:end_a)
        CALL ident (arr_a, arr_b, start_a, end_a, scr_start,
     & len_bb, seq_a(1:scr_width), numscrseq, hits)
c no scr found in arr_b
        IF (hits.EQ.0)THEN
          prev_scr=.FALSE.
          GOTO 8210
        ENDIF
c scr found but there wasn't a previous scr
        IF (.NOT.prev_scr)THEN
          prev_scr=.TRUE.
          lcr_start_pos=scr_start+scr_width
          GOTO 8210
        ENDIF
c the special case where the scr starts at the beginning of the alignment
        IF (scr_start.EQ.1)THEN
          lcr_start_pos=scr_start+scr_width
          GOTO 8210
        ENDIF
c scr found.
        IF (hits.GT.1)THEN
            WRITE (6,21)' There were ',hits,' places where ( '
            WRITE (6,23) arr_a(1)(start_a:end_a)
            WRITE (6,23)') was found in the test alignment.'
            CALL wrwords ('This is very confusing - which hit
```

```
     & should be
     & chosen for the strongly conserved region?. Rule of thumb used is,
     & the last place found is the SCR. Frankly, your computer
     & recommends that you start again, but this time, you should
     & choose your own SCR''s.', screenwr, screenwidth)
          ENDIF
          lcr_end_pos=scr_start-1
          CALL wr_lcr (lcrfnd, lcr_start_pos, lcr_end_pos,
     & alig_no, unit_a, numscrseq, arr_b, lcr_start_pos)
          lcr_start_pos=scr_start+scr_width
          diff_a=diff_a+1
          GOTO 8210
 8212           CONTINUE
c no more scr's but if the previous scr was found in the test ali,
c and there is sequence after it, then it's a lcr.
c write it & then read a new ali. lcr_start_pos has already been set to the
c start of the lcr, end of lcr is the end of sequence,= len_b.
          IF (prev_scr.AND.lcr_start_pos.LE.len_b)THEN
            lcrfnd=lcrfnd+1
            diff_a=diff_a+1
            CALL wr_lcr (lcrfnd, lcr_start_pos, len_b,
     & alig_no, unit_a, numscrseq, arr_b, lcr_start_pos)
          ENDIF
        END DO
        CLOSE (alifil)
        CLOSE (unit_a)
 25     FORMAT (' There were ',i4,
     &' different alignments found in these LCR''s.')
        WRITE (6,25) diff_a
c       IF (debugL)  WRITE (debugI,25) diff_a
        END


        SUBROUTINE ident (array_one, array_two, start_a, end_a,
     & start_b, len_bb,test_str, numscrseq, hits)
        IMPLICIT none
c
c this routine finds a region in array_two that is identical to array_one (
c start_a:end_a)
c test_str is array_one(1)(start_a:end_a) & this is checked against the other
c array to see where/if it exists.
c only then, are further row by row comparisons made. the number of rows to
c check is in numscrseq. If a hit was found, then a further check is made
c to see if there is anywhere else that the test str is found in array_two
c If there is, we again check if the other rows of the 2 arrays are in
c agreement. This checks for any ambiguity in the site of the SCR found.
c len_bb is the length of the second array minus the
c length of the test string itself, which we need to ensure that we don't
c check beyond this.
c if an identical region is found, hits returns the number of times this was
c found (in a window start_a+_30) &
c if no identical region is found, hits contains 0
c
        CHARACTER*(*) array_one(*), array_two(*), test_str
        INTEGER start_a, end_a, start_b, end_b, len_bb, first_a, last_a
        INTEGER windowa, windowb, row, hits, numscrseq, width, store
        width=end_a-start_a
        hits=0
c
c the special case where testing 2 lcr's against each other.
c we know where about test_str must be in array two.
c
        IF (len_bb.EQ.1)THEN
          start_b=1
          IF ( test_str.EQ.array_two(1)(start_a:end_a) )GOTO 6430
          GOTO  6440
        ENDIF
c
c we don't know where about (if at all) test_str is in array_two.
c so generate a starting position & work along the array from it.
```

```
c
        windowa=start_a-30
        windowb=start_a+30
        IF (windowa.LT.1) windowa=1
        IF (windowb.GT.len_bb) windowb=len_bb
c
 6425   CONTINUE
c
c Have previously found a hit at windowb-1, we now look at position windowb
c to see if generates a hit also
c
        IF (windowa.EQ.windowb)THEN
          start_b=windowa
          IF ( test_str.EQ.array_two(1)(start_b:start_b+width) )
     &GOTO 6430
        ELSE
          DO start_b=windowa,windowb
            IF ( test_str.EQ.array_two(1)(start_b:start_b+width) )
     &GOTO 6430
          END DO
        ENDIF
        GOTO 6440
c
c check to see that the other rows of the arrays are the same.
c
 6430   CONTINUE
        END_b=start_b+width
        DO row=2,numscrseq
            IF (array_one(row)(start_a:end_a).NE.
     &array_two(row)(start_b:end_b)) GOTO 6435
        END DO
c
c identity found.
c
        hits=hits+1
c
c store where this hit was found - we now search again for a hit, which puts
c other values in start_b. The line before exiting this routine restores
c where about the hit was found into start_b
c
        store=start_b
 6435   CONTINUE
c
c see if there are any more hits to be found.
c
        windowa=start_b+1
c
c this line for when 2 lcr's tested.
c
        IF (windowa.GT.len_bb) GOTO 6440
c
c check to see if the test_str is found anywhere else in the window in array2
c
        IF (windowa.LE.windowb) GOTO 6425
c
c checked the entire window. exit
c
 6440   CONTINUE
        IF (hits.GT.0) start_b=store
        END

        SUBROUTINE wr_scr (len_a, seqstord, scrout,  ordera, prednam,
     &no_preds, arr_a, seq_a, outfile, alitord, scrpos, extra)
        IMPLICIT none
        CHARACTER*(*) arr_a(*), prednam(*)
        CHARACTER*(*) seq_a, scrout, ordera(*), outfile
        INTEGER len_a, totseqfnd, seqstord, alitord(*), alifil, unit_b
        INTEGER totscrfnd, totlcrfnd, totalifnd, no_preds, debugI
        INTEGER row, alig_rd, x, y, scrpos(2,*), seqlen, pos
        INTEGER cntr, outwidth, screenwidth
```

```
      INTEGER screenwr
      LOGICAL debugL, extra
      COMMON /scrnwrite/ screenwr, screenwidth
      COMMON /a3/ outwidth
      COMMON /a4/ totseqfnd
      COMMON /a8/ totalifnd
      COMMON /a11/ alifil
      COMMON /a13/ unit_b
      COMMON /b2/ totscrfnd
      COMMON /b3/ totlcrfnd
      COMMON /c1/ debugL, debugI
c
 31   FORMAT (' Starting Position ',i,' End Position ',i)
      alig_rd=1
      IF (debugL) WRITE (debugI,FMT='(a,i)')
     &' SCR positions are taken from alignment number ', alitord(1)
c only numscrseq sequences are in arr_a, there should be totseqfnd.
c read the relevant alignment into arr_a.
      IF (extra)THEN
            OPEN (alifil,FILE=outfile,STATUS='old',form='unformatted')
            REWIND (alifil)
            CALL rd_ali (arr_a, seqstord, alitord(1), alig_rd,
     & totseqfnd, len_a, seq_a, alifil)
            CLOSE (alifil)
      ENDIF
c read in the consensus secondary structure predictions for this alignment.
c this is output along with the strongly conserved regions from this alignment
      DO x=1,no_preds
        OPEN (7,FILE=prednam(x),STATUS='old',form='unformatted')
        REWIND (7)
        cntr=0
 8320         CONTINUE
        cntr=cntr+1
        READ (7) seqlen
        READ (7) arr_a(totseqfnd+x)(1:seqlen)
        IF (cntr.NE.alitord(1) )GOTO 8320
        CLOSE (7)
      END DO
c
c open output files and write some data we need
c
      OPEN (unit_b,FILE=scrout,STATUS='new',form='unformatted')
      WRITE (unit_b) totseqfnd
      WRITE (unit_b) totalifnd
      WRITE (unit_b) totscrfnd
      WRITE (unit_b) totlcrfnd
      WRITE (unit_b) outwidth
      WRITE (unit_b) screenwidth
      DO x=1,totseqfnd
        WRITE (unit_b) ordera(x)
      END DO
      DO x=1, no_preds
        WRITE (unit_b) prednam(x)
      END DO
c write out the scr's
      cntr=0
 33   CONTINUE
      cntr=cntr+1
      IF (debugL)THEN
        write
     &(debugI,FMT='(a,i)')' Positions for SCR number ',cntr
        WRITE (debugI,31) scrpos(1,cntr), scrpos(2,cntr)
      ENDIF
      WRITE (unit_b) scrpos(1,cntr)
      WRITE (unit_b) scrpos(2,cntr)
      DO row=1,totseqfnd+no_preds
        WRITE (unit_b) arr_a(row)(scrpos(1,cntr):scrpos(2,cntr))
      END DO
      IF (cntr.NE.totscrfnd) GOTO 33
      CLOSE (unit_b)
```

```
      END

      SUBROUTINE wrlcrl (arr_b, seq_c, ordera, seqstord, lcrout,
     & lcrtmpb, prednam, no_preds, outfile, extra, arr_a)
      IMPLICIT none
c
c this routine will output the unique lcr's that have have by fndlcr & uniq
c it also outputs the consensus secondary structure predictions along with
c these lcr's.
c
      CHARACTER*(*) arr_a, arr_b(*), seq_c, ordera(*), lcrout, lcrtmpb
      CHARACTER*(*) prednam(*), outfile
      CHARACTER*1 alisgap
      CHARACTER*2 dummychar
      INTEGER alifil, unit_a, unit_b, unit_c, numscrseq, seqstord(*)
      INTEGER oldnum
      INTEGER no_preds, x
      INTEGER alig_rd, lcrnum, length, store, cntr
      INTEGER totids, totgap, outwidth
      INTEGER start, end
      INTEGER alig_no, ifail, dummy
      INTEGER width, row, pos, totseqfnd
      LOGICAL extra
      COMMON /a3/ outwidth
      COMMON /a4/ totseqfnd
      COMMON /a7/ numscrseq
      COMMON /a11/ alifil
      COMMON /a12/ unit_a
      COMMON /a13/ unit_b
      COMMON /a14/ unit_c
      COMMON /d4/ alisgap
 13   FORMAT (/,' These are the different alignments found for
     & loosely conserved region ',i4)
 14   FORMAT (/,' LCR number:',i4,' from alignment number:------>',i4)
 15   FORMAT (' Names of seqs &',/,' Struct. Pred. Methods.')
 16   FORMAT (' Total number of gaps in LCR : ',i,/,
     &' Total number of identities in LCR : ', i)
c
c the number of sequences stored in the lcr is numscrseq whereas the
c number of sequences in the alignment is totseqfnd. if these are different, we
c need to read all the sequences back in from the raw alignment data else we can
c read the lcr in from lcrout.
c
      IF (extra)THEN
            alig_rd=1
            OPEN (alifil,FILE=outfile,STATUS='old',form='unformatted')
            REWIND (alifil)
      ENDIF
      OPEN (unit_a,FILE=lcrtmpb,STATUS='old',form='unformatted')
      REWIND (unit_a)
      OPEN (unit_b,FILE='print.me',STATUS='new',form='formatted')
      OPEN (unit_c,FILE=lcrout,STATUS='new',form='unformatted')
      DO x=1,no_preds
           OPEN (x+6, FILE=prednam(x),STATUS='old',form='unformatted')
           REWIND (x+6)
      END DO
c
c counter that tells us which secondary struc pred is getting read in
c
      oldnum=0
c
c counter that tells us which lcr is getting read in
c
      store=1
      WRITE (unit_b,13) store
c
c read in the lcr. if i/o error, we must have read in all the lcr's.
c
 8340 CONTINUE
      CALL rd_lcr
```

```
     &(lcrnum, start, end, alig_no, unit_a, numscrseq, arr_b, ifail, 1)
        IF (ifail.NE.0) GOTO 8383
        width=end-start+1
c
c if lcrnum<>store, then we've read in all the diff alignments for lcr #store,
c so we are reading the aligs for the next lcr - init all the files.
c
        IF (lcrnum.NE.store)THEN
          store=lcrnum
          WRITE (unit_b,13) store
          IF (extra)THEN
            REWIND (alifil)
            alig_rd=1
          ENDIF
          DO x=1, no_preds
            REWIND (x+6)
          END DO
          oldnum=0
        ENDIF
c
c if there are more sequences to be read in for the lcr, read them in
c
        IF (extra)THEN
          CALL rd_ali (arr_b, seqstord, alig_no, alig_rd,
     & totseqfnd, dummy, seq_c, alifil)
          DO x=1,totseqfnd
            arr_b(x)(1:width)=arr_b(x)(start:end)
          END DO
        ENDIF
c
c get the extra data (id's and gap numbers, the consensus secondary structure
c predictions ) that the user might like to have with the lcr to help in
c the choice.
c
        DO x=1,no_preds
          cntr=oldnum
 8350           CONTINUE
          READ (x+6) length
          READ (x+6) seq_c(1:length)
          cntr=cntr+1
          IF (cntr.NE.alig_no) GOTO 8350
          arr_b(totseqfnd+x)(1:width)=seq_c(start:end)
        END DO
        oldnum=alig_no
        CALL char_cnt (arr_b, totgap, width, totseqfnd, alisgap)
        CALL idcnt (arr_b, totids, width, totseqfnd)
c
c write out the lcr results to the output file & to the print file
c
        CALL wr_lcr (lcrnum, start, end, alig_no, unit_c,
     & totseqfnd+no_preds, arr_b, 1)
        WRITE (unit_b,14) lcrnum, alig_no
        WRITE (unit_b,16) totgap, totids
        WRITE (unit_b,15)
        CALL wrarray (width, totseqfnd+no_preds, seq_c, ordera, arr_b,
     & dummychar, dummychar, unit_b, outwidth)
c
c get the next alignment for the lcr # and repeat
c
        GOTO 8340
c
 8383   CONTINUE
        IF (extra) CLOSE (alifil)
        CLOSE (unit_a, STATUS='delete')
        CLOSE (unit_b)
        CLOSE (unit_c)
        DO x=1, no_preds
          CLOSE (x+6)
        END DO
        END
```

```
      SUBROUTINE rd_ali (arr_x, seqstord, alig_to_rd, alig_rd,
     & totseqfnd, length, string, unit)
      IMPLICIT none
c this routine reads in alignment data on the already opened unit number unit.
c the alignment about to be read is alig_rd and the alignment to be read and
c kept is alig_to_rd.
      CHARACTER*(*) arr_x(*), string
      INTEGER seqstord(*)
      INTEGER alig_to_rd, alig_rd
      INTEGER length, totseqfnd, unit, x
c
 5310 IF (alig_to_rd.NE.alig_rd)THEN
          READ (unit) length
          do x=1,totseqfnd
        READ (unit) string(1:length)
          END DO
          alig_rd=alig_rd+1
          GOTO 5310
      ELSE
          CALL rd_ali2 (arr_x, seqstord, totseqfnd, length, string,
     & unit)
          alig_rd=alig_rd+1
      ENDIF
      END

      SUBROUTINE rd_ali2 (arr_x, seqstord, totseqfnd, length, string,
     & unit)
      IMPLICIT none
      INTEGER totseqfnd, length, seqstord(*), cntr, unit, x
      CHARACTER*(*) string, arr_x(*)
      cntr=1
      READ (unit) length
      DO x=1,totseqfnd
        READ (unit) string(1:length)
        IF (x.EQ.seqstord(cntr))THEN
            arr_x(cntr)(1:length)=string(1:length)
            cntr=cntr+1
        ENDIF
      END DO
      END

      SUBROUTINE wr_lcr (lcrnum, start, end, alig_no, unit, numrows,
     & arr_x, pos2)
      IMPLICIT none
      INTEGER lcrnum
      INTEGER start, end
      INTEGER alig_no, unit, x, numrows, pos2, width
      CHARACTER*(*) arr_x(*)
      WRITE (unit) lcrnum
      WRITE (unit) start
      WRITE (unit) end
      WRITE (unit) alig_no
      width=end-start+pos2
      DO x=1,numrows
        WRITE (unit) arr_x(x)(pos2:width)
      END DO
      END

      SUBROUTINE rdseqa (ordera, seq_a, arr_a, seqname)
      IMPLICIT none
c routine to read in the sequences into an array. the formats supported are
c gcg (1 seq per file) and nbrf (all sequence in 1 file) formats
      CHARACTER*(*) ordera(*)
      CHARACTER*(*) arr_a(*), seq_a, seqname
      CHARACTER*15 name
      INTEGER totseqfnd, maxlenfnd, row, pos, unit, x
      INTEGER screenwr, screenwidth
      INTEGER length, error
      LOGICAL crash, gcg
```

```
      PARAMETER (UNIT=1)
      COMMON /a1/ maxlenfnd
      COMMON /scrnwrite/ screenwr, screenwidth
      COMMON /a4/ totseqfnd
      COMMON /a5/ gcg
      COMMON /a6/ crash
 3000 FORMAT (a)
c
      IF (gcg)THEN
        DO row=1,totseqfnd
          name=
     &ordera(row)(1:index(ordera(row),' ')-1)//'.gcg'
          OPEN (unit,FILE=name,STATUS='old')
          REWIND (unit)
          CALL rd_gcg (seq_a, error, unit, length)
          IF (error.NE.0)THEN
             CALL wrwords ('Something is funny with
     & the sequence; '//ordera(row) , screenwr, screenwidth)
             CALL wrwords ('This program will be
     & terminating.' , screenwr, screenwidth)
             crash=.TRUE.
          ENDIF
c make a blank after the last residue in the sequence so that the length
c of the sequence can be calculated
          seq_a(length:length)=' '
          CALL mkcapc (seq_a, name)
          arr_a(row)(1:maxlenfnd)=seq_a(1:maxlenfnd)
          CLOSE (unit)
        END DO
      ELSE
c if the input sequence file was nbrf format, the individual seqs will
c be read in one at a time from the multiple file.
        OPEN (unit,FILE=seqname,STATUS='old')
        REWIND (unit)
        error=0
        DO x=1,totseqfnd
          CALL rd_pir (seq_a, error, name, unit, length)
          IF (error.EQ.1) THEN
             CALL wrwords ('The End-of-File
     & character has been found too soon in the NBRF format sequence
     & file. There may be not enough sequences in the sequence file.
     & Please check this file and start again.', screenwr, screenwidth)
             CALL wrwords ('Program will be
     & terminating.', screenwr, screenwidth)
             crash=.TRUE.
             GOTO 3040
          ENDIF
          IF (error.EQ.2)   THEN
             CALL wrwords ('There is no
     & End-of-Sequence character ie., an asterix, at the end of the
     & sequence '//name//'. Please check the sequence file and start
     & again.', screenwr, screenwidth)
             CALL wrwords ('Program will be
     & terminating.', screenwr, screenwidth)
             crash=.TRUE.
             GOTO 3040
          ENDIF
c make a blank after the last residue in the sequence so that the length
c of the sequence can be calculated
          seq_a(length:length)=' '
          CALL mkcapb (name)
          CALL mkcapc (seq_a, name)
          DO row=1,totseqfnd
             IF (name.EQ.ordera(row)) GOTO 3030
          END DO
C Do not beleive it is possible for this to happen.
c Any mistakes in sequence names between the sequence file and the
c names in the alignment should be picked up earlier by the
c routine ORDER_SEQS, called when reading in the alignments
          CALL wrwords ('Error - the sequence name from
```

```
     & the sequence file and that in the alignment do not agree.',
     & screenwr, screenwidth)
           crash=.TRUE.
           GOTO 3040
3030                    CONTINUE
           arr_a(row)(1:maxlenfnd)=seq_a(1:maxlenfnd)
         END DO
3040          CONTINUE
         CLOSE (unit)
       ENDIF
       END

       SUBROUTINE rd_gcg (seq_a, ifail, unit, length)
       IMPLICIT none
c the length of input, is set so big - normal gcg files are only 80 chars
c across - just to be on the safe side.
       CHARACTER*140 input
       CHARACTER*(*) seq_a
       INTEGER pos, length, tmp, ifail, unit
 401   FORMAT (a)
c read past the general info - ie get to start of seq data
 410   CONTINUE
       READ (unit,401,iostat=ifail) input
       IF (ifail.NE.0) GOTO 440
       IF (index(input,'..').EQ.0) GOTO 410
c read in seq data
       length=1
 420   CONTINUE
       READ (unit,401,iostat=ifail) input
       IF (ifail.NE.0) GOTO 430
       CALL rem_space (input)
       pos=index(input,' ')-1
c if a blank string
       IF (pos.EQ.0) GOTO 420
c place the sequence data into the string that's passed back.
       tmp=length+pos-1
       seq_a(length:tmp)=input(1:pos)
       length=tmp+1
       GOTO 420
 430   CONTINUE
c reset error counter to no-error position
       ifail=0
 440   CONTINUE
       END

       SUBROUTINE rd_pir (seq_a, error, name, unit, length)
       IMPLICIT none
c a sequence file in nbrf/pir FORMAT has been opened on unit. this routine
c reads the file and the sequence is read into var_str
c name contains the sequence name (important if the multiple sequence file
c FORMAT is used)
c error =1 if end of file before time eg. not enough sequences in the file
c or this is a file in the wrong format
c error=2 if no end_of_sequence character is found. This is '*'.
c if no end of sequence or the presence of '>' in string (a new name!)
c indicates read past where end of sequence should be.
       INTEGER x, ifail, length, pos, unit, tmp, error
       CHARACTER*(*) seq_a, name
       CHARACTER*255 var_str
       CHARACTER blank*1
 401   format(a)
c read in the name
 410   CONTINUE
       READ (unit,401,iostat=ifail) name
       IF (ifail.NE.0) THEN
         error=1
         GOTO 435
       ENDIF
       IF (name(1:1).NE.'>') GOTO 410
c remove any special CHARACTERs from name ('>p1;')
```

```
c - ALIEN & FROMPIR do this, so to recognise sequence names as used by
C alig prog and found in the sequence file, we must do this also.
        name(1:)=name(5:)
c  get to start of seq data
        READ (unit,401) blank
c read in sequence data
        length=1
 420    CONTINUE
        READ (unit,401,iostat=ifail) var_str
c if a sequence does not have end_of_seq ('*') char,
c we'll read past eof or get new name. !error!
        IF (ifail.NE.0.or.index(var_str,'>').GT.0)THEN
          error=2
          GOTO 435
        ENDIF
c remove spaces from the sequence
        CALL rem_space (var_str)
        pos=index(var_str,' ')-1
        tmp=length+pos-1
        seq_a(length:tmp)=var_str(1:pos)
        IF (index(var_str,'*').GT.0) GOTO 430
        length=tmp+1
        GOTO 420
 430    CONTINUE
c remove the asterix at the end of the sequence.
        length=index(seq_a,'*')
 435    CONTINUE
        END


        SUBROUTINE rdseqnames (array, cntr, unit, seqlen, maxseqnum,
     &errord, messa, messc, messd, crash)
        IMPLICIT none
c this routine reads the sequence names used in a alignment
c by the multiple alignment program pileup (gcg7)
c the dummy values.
        INTEGER cntr, unit, seqlen, maxseqnum
        CHARACTER*(*) errord, messa, messc, messd
        CHARACTER*(*) array(*)
        LOGICAL crash
c
        INTEGER pos
        CHARACTER*30 name_input
c
        INTEGER screenwr, screenwidth
        COMMON /scrnwrite/ screenwr, screenwidth
 1271   FORMAT (a)
        cntr=0
 1272   READ (unit,1271) name_input
        IF (name_input(1:2).EQ.'  ') GOTO 1276
c if still seq names to read in.
        cntr=cntr+1
c check to see if there is an allowable no. of seqs in the alignments
        IF (cntr.GT.maxseqnum)THEN
          CALL wrwords (errord,screenwr, screenwidth)
          CALL wrwords (messa,screenwr, screenwidth)
          CALL wrwords (messc,screenwr, screenwidth)
          CALL wrwords (messd,screenwr, screenwidth)
          crash=.TRUE.
          GOTO 1276
        ENDIF
        pos=index( name_input(8:),' ')-1
c work out the longest name.
        IF (pos.GT.seqlen) seqlen=pos
        array(cntr)=name_input(8:7+pos)
        GOTO 1272
 1276   CONTINUE
c now set seqlen to point at the start of the alignments,
c this is 3 places after the end of the longest sequence's name
        seqlen=seqlen+3
```

```
      END

      SUBROUTINE scr (alisname, ordera, arr_b, seq_c, idscores,
     & idmaxs, seqstord, len_a, arr_a, seq_a, asterix, maxalisnum,
     &maxseqnum, outfile, alitord, scrpos, maxnumscr,
     & messa, messd, hitstring)
      IMPLICIT none
c
c routine to calculate the strongly conserved regions.
c 1) choose which sequences in the alignments will be used in SCR calculations
c 2) pick the alignments to look at in SCR calculations
c       a) Automatically
c       b) Manually
c 3) Choose the SCR's knowing the sequences & alignments to use
c       a) Automatically
c       b) Manually - if this is the case, we do not do #2 above, the user
c          picks the alignments
c
      CHARACTER*(*) alisname, ordera(*), arr_a(*)
      CHARACTER*(*) arr_b(*), seq_a, seq_c
      CHARACTER*(*) outfile, asterix
      CHARACTER*(*) hitstring
      CHARACTER*(*) messa, messe, messd
      INTEGER totalifnd, totseqfnd, idscores(*), idmaxs(*)
      INTEGER totidsfnd
      INTEGER seqstord(*), x, numimp, numscrali
      INTEGER maxlenfnd, maxnumscr
      INTEGER len_a
      INTEGER alifil
      INTEGER scrpos(2,*)
      INTEGER debugI
      INTEGER maxalisnum, maxseqnum
      INTEGER screenwidth, alitord(*), numscrseq, screenwr
      INTEGER dummy
      LOGICAL bit
      LOGICAL debugL, crash
      PARAMETER(
     &messe=' The number allowed is stored in the variable maxnumscr.')
c
      COMMON /a1/ maxlenfnd
      COMMON /scrnwrite/ screenwr, screenwidth
      COMMON /a4/ totseqfnd
      COMMON /a6/ crash
      COMMON /a7/ numscrseq
      COMMON /a8/ totalifnd
      COMMON /a11/ alifil
      COMMON /c1/ debugL, debugI
      COMMON /d1/ numscrali
 10   FORMAT (a)
 11   FORMAT (i)
 5307 FORMAT (7(tr1,a8,','))
c
c we do not use all the sequences in an alignment to generate scr's.
c if there are > numscrseq seqs in the alignment, select which seqs to use
c the seqs in use are in seqstord
c
      IF (numscrseq.GE.totseqfnd)THEN
        numscrseq=totseqfnd
        DO x=1,numscrseq
          seqstord(x)=x
        END DO
      ELSE
        CALL wrwords ('The number of sequences in the
     & alignments is more than the number of sequences that are
     & looked at in choosing the SCRS''s. So we will be picking,
     & at random, which of the entered sequences will be looked
     & at in choosing the SCR''s.', screenwr, screenwidth)
        CALL rand (numscrseq, totseqfnd, seqstord)
        CALL reorder (numscrseq, seqstord)
      ENDIF
```

```
c
c write out which sequences have been picked for chosing the scr's
c
 23    FORMAT (i,' out of the ',i,' sequences will be used for SCR
     & calculation')
        IF (totseqfnd.LE.numscrseq)THEN
              CALL wrwords ('All the sequences in the
     & alignments are used for picking the SCR''s.', screenwr,
     & screenwidth)
          IF (debugL) CALL wrwords ('All the sequences in the
     & alignments are used for picking the SCR''s.', debugI,
     & screenwidth)
        ELSE
          WRITE (screenwr,23) numscrseq, totseqfnd
          IF (debugL) WRITE (debugI,23) numscrseq, totseqfnd
        ENDIF
        CALL wrwords ('These sequences are;', screenwr, screenwidth)
        WRITE (screenwr,5307) ( ordera(seqstord(x)),x=1,numscrseq )
        IF (debugL) THEN
            CALL wrwords ('These sequences are;', debugI,
     & screenwidth)
            WRITE (debugI,5307)
     &( ordera(seqstord(x)),x=1,numscrseq )
        ENDIF
c
c we do not use all the alignments to generate scr's, only numscrali number.
c make sure numscrali is not greater than the number of actual alignments.
c
        IF (numscrali.GT.totalifnd) THEN
          CALL wrwords ('The number of alignments to be
     & looked at, has been set to more than the total number of
     & alignments that were entered into this program. So, we are
     & resetting the number of alignments to be looked at, to be
     & the same as the number of alignments that were used as input
     & to this program.', screenwr, screenwidth)
          numscrali=totalifnd
        ENDIF
c
c let the user choose the method of scr calculation.
c print out what the options are
c
 4241   CONTINUE
        WRITE (screenwr,10)' '
        CALL wrwords ('Choose how the SCR''s are to be calculated.
     & Enter number now ....', screenwr, screenwidth)
        CALL wrwords (' 1) Automatic. Both the alignments to look at &
     & the areas of identity are automatially done.', screenwr,
     & screenwidth)
        CALL wrwords (' 2) Semi- automatic. User chooses the alignments,
     & areas of identity automatically done.', screenwr, screenwidth)
        CALL wrwords (' 3) Manual. User chooses the Strongly
     & Conserved Regions.', screenwr, screenwidth)
        CALL wrwords (' 4) Change the number of alignments that are
     & looked at in choosing the Strongly Conserved Regions.',
     & screenwr, screenwidth)
        CALL wrwords (' 5) Change the minimum width that a
     & Strongly Conserved Region can be.',
     & screenwr, screenwidth)
        CALL wrwords (' 6) Help.', screenwr, screenwidth)
        READ (*,11) numinp
        IF (numinp.LT.1.or.numinp.GT.6)THEN
          CALL wrwords ('This number is out of range, please
     & re-enter a number.', screenwr, screenwidth)
          GOTO 4241
        ENDIF
        GOTO (4230,4240,4250,4255, 4257, 4260)numinp
c
c automatically pick the alignments to use. This is done on basis of
c identity scores, the alignments with the highest # of identities being
c picked first then next highest, etc.
```

```
c Work out the highest to lowest identity scores - we only need to do this once
c we put the different ( there are totidsfnd diff) identity scores ( held in
c idscores) into the array idmaxs.
c
 4230    CONTINUE
         CALL wrwords (asterix(1:screenwidth),screenwr, screenwidth)
         IF (idmaxs(1).EQ.-1) CALL id_gen (idscores, idmaxs, totidsfnd)
         CALL auto (totidsfnd, idscores, idmaxs, alitord)
         GOTO 4245
c
c user picks the alignments to use.
c
 4240    CONTINUE
         CALL wrwords (asterix(1:screenwidth),screenwr, screenwidth)
         CALL scr_alis_inp (alisname, alitord)
         GOTO 4245
c
c automatically pick the scrs from already chosen alignments and sequences
c find the scr's in these alignments & mark with '*' in seq_c
c
 4245    CONTINUE
           CALL scr_fnd (ordera, seqstord, arr_b, seq_c, len_a, alitord,
     &arr_a, seq_a, outfile, hitstring)
c
c and convert '*' to numbers - also check for a bug in finding the scr's!
c
           CALL convert (numscrseq, arr_a, seq_c(1:len_a), scrpos,
     & messa, messe, messd, maxnumscr)
           IF (crash) GOTO 4275
         GOTO 4270
c
c user chooses the scr's themselves.
c
 4250    CONTINUE
         CALL wrwords (asterix(1:screenwidth),screenwr, screenwidth)
         CALL scrinp (seqstord, ordera, outfile, arr_a, seq_a,
     & scrpos, alitord)
         GOTO 4270
c
c change the number of alignments used in automatically getting the scr's
c
 4255    CONTINUE
         bit=.TRUE.
         CALL altvar (maxalisnum, dummy, 3, bit)
c
c now check that the number of alignments chosen is not greater than the
c actual number of alignments
c
         IF (numscrali.GT.totalifnd) THEN
           CALL wrwords ('You choose a number of alignments to be
     & looked at, that is greater than the total number of alignments
     & that were entered into this program. So, we are resetting the
     & number of alignments to be looked at, to be the same as the
     & number of alignments that were used as input to this program.',
     & screenwr, screenwidth)
           numscrali=totalifnd
         ENDIF
         GOTO 4241
c
c change the minimum width of an scr
c
 4257    CONTINUE
         bit=.TRUE.
         CALL altvar (dummy, dummy, 5, bit)
         GOTO 4241
c
c the help routines
c
 4260    CONTINUE
         CALL help
```

```
        GOTO 4241
c
c check if any scr's, have been found
c
 4270   CONTINUE
        IF (scrpos(1,1).EQ.0)THEN
          CALL wrwords ( 'There is a problem - no Strongly
     & Conserved Regions have
     & been found. You should try again with a different option or if
     & you are using the automatic method to work out which alignments
     & the SCR''s should be chosen from, you can reduce the number of
     & alignments used to calculate the SCR. This should improve the
     & chances of finding a SCR.', screenwr, screenwidth)
          GOTO 4241
        ENDIF
c
c count the number of SCR's (and therfore LCRs) that have been found
c
        CALL scrlercnt (len_a, scrpos)
 4275   CONTINUE
        END

        SUBROUTINE id_gen (idscores, idmaxs, totidsfnd)
        IMPLICIT none
c
c there are totalifnd number of scores in idscores.
c the top score is put in idmaxs(1), next highest in idmaxs(2)...etc., with
c totidsfnd is the number of different identity scores.
c initialise the array idmaxs to -1, this is done at start, & this routine
c is only called if idmaxs(1)=-1, ie., it has never been called before!
c
        INTEGER totalifnd, idscores(*), idmaxs(*)
        INTEGER totidsfnd, x, cntr, debugI
        LOGICAL debugL
        COMMON /a8/ totalifnd
        COMMON /c1/ debugL, debugI
 5305   FORMAT (' The highest id score was ',i)
 5306   FORMAT (' The next highest id score was ',i)
c
c set a top limit for the number of diff scores.
        totidsfnd=totalifnd
        cntr=1
 5310   CONTINUE
c
c get the highest score.
        DO x=1,totalifnd
          IF (idscores(x).GT.idmaxs(cntr)) idmaxs(cntr)=idscores(x)
        END DO
c
c write out what this score is and what alignments have it.
        IF (debugL)THEN
          IF (cntr.EQ.1) THEN
            WRITE (debugI,5305) idmaxs(1)
            CALL wrdebugali (idmaxs(1), totalifnd, idscores)
          ELSE
            WRITE (debugI,5306) idmaxs(cntr)
            CALL wrdebugali (idmaxs(cntr), totalifnd, idscores)
          ENDIF
        ENDIF
c
c remove this score from the data, so that the next highest score can be counted
c (unless you have finished reading in all the identity scores.)
        IF (idmaxs(cntr).EQ.0)THEN
          totidsfnd=cntr
c if only 1 id score found and it was 0, then don't need to re-negate.
          IF (totidsfnd.EQ.1) GOTO 5320
          GOTO 5315
        ENDIF
c -1 signifies that all the identities have been read in.
        IF (idmaxs(cntr).EQ.-1)THEN
```

```
          totidsfnd=cntr-1
          GOTO 5315
        ENDIF
c
c negate the scores so that the next highest can be picked up.
        DO x=1,totalifnd
          IF (idscores(x).EQ.idmaxs(cntr)) idscores(x)=-idscores(x)
        END DO
        cntr=cntr+1
        GOTO 5310
c
c restore the data ( it's all been negated) to original form
 5315   CONTINUE
        DO x=1,totalifnd
          IF (idscores(x).LT.0) idscores(x)=-idscores(x)
        END DO
 5320   CONTINUE
 5307   FORMAT (' There are ',i4,' different identity scores.')
        IF (debugL) WRITE (debugI,5307) totidsfnd
        END

        SUBROUTINE wrdebugali (score, tosearch, idscores)
        IMPLICIT none
c routine to work out which alignments have the same number of identities
c as the number 'score'. The numbers of these alignments is written to
c the debug file.
        INTEGER score, tosearch, idscores(*)
c
        INTEGER debugI, screenwidth, screenwr
        LOGICAL debugL
        COMMON /scrnwrite/ screenwr, screenwidth
        COMMON /cl/ debugL, debugI
c
        INTEGER x
c
 30     FORMAT (tr1,i4)
        CALL wrwords ('The alignments with this score were numbers:',
     & debugI, screenwidth)
        DO x=1,tosearch
          IF (idscores(x).EQ.score) WRITE (debugI,30) x
        END DO
        END



        SUBROUTINE rand (numbera, numberb, rand_nos)
c
c picks numbera number of random numbers out of a range of numberb number.
c the numbers are stored in the array rand_nos.
c
c This routine was supplied  by Ian Walker, Computer Service Dept.,
c Glasgow University - suitable for vax.
c modified, lhb, 11/95 to cope with openvms 6.1 fortran
c This uses ran(seed) intrinsic function to generate random numbers.
c Unfortunately, because seed can't be changed during run, this is only
c pseudo-random behaviour
        INTEGER numberb, numbera, rand_nos(*), x, x2, y
        integer*4 seed
        real*4 temp
        seed=11111
        temp=real(numberb)
        DO x=1,numbera
 15         y=INT(1.0+(temp*RAN(seed)))
c check this number has not been picked before - pick again if it has
           IF (x.GT.1)THEN
          DO x2=1,x-1
             IF ( rand_nos(x2).EQ.y )GOTO 15
          END DO
            ENDIF
            rand_nos(x)=y
```

```
        END DO
        END

        SUBROUTINE auto (totidsfnd, idscores, idmaxs, alitord)
        IMPLICIT none
c
c In idscores are a list of identities found for each alignment.
c In idmaxs are the individual numbers found, in order of highest to lowest.
c the total of different numbers found is stored in totidsfnd
c this routine finds the alignments with the highest number of id's, then
c those with the next highest etc. and stores them in the array alitord
c until the number of alignments used to calculate the SCRs have been found
        INTEGER totidsfnd, totalifnd, idscores(*),idmaxs(*), numscrali
        INTEGER alitord(*),x, y, cntr
        COMMON /a8/ totalifnd
        COMMON /d1/ numscrali
        cntr=0
 5310   CONTINUE
        DO y=1,totidsfnd
          DO x=1,totalifnd
            IF (idmaxs(y).EQ.idscores(x))THEN
              cntr=cntr+1
              alitord(cntr)=x
              IF (cntr.EQ.numscrali) GOTO 5330
            ENDIF
          END DO
        END DO
 5330   CONTINUE
        END

        SUBROUTINE scrlcrcnt (length, scrpos)
        IMPLICIT none
c
c routine to work out the numbers of LCRs and SCRs found. the actual positions
c of the scrs in an alignment are stored in scrpos
        INTEGER totscrfnd, totlcrfnd, scrpos(2,*), debugI, length
        LOGICAL debugL
        COMMON /b2/ totscrfnd
        COMMON /b3/ totlcrfnd
        COMMON /c1/ debugL, debugI
        totscrfnd=0
c
c find the total # of SCR's, recorded in scrpos (pos+1 is a 0)
c
 222    CONTINUE
        IF (scrpos(1,totscrfnd+1).EQ.0)GOTO 223
        totscrfnd=totscrfnd+1
        GOTO 222
c
c determine LCR's empirically. There will be 1 more LCR than SCR, unless
c there is an SCR at the start or finish (reduce lcr number by 1 on either
c condition)
 223    CONTINUE
        totlcrfnd=totscrfnd+1
c
c there is no lcr at start of sequence
        IF (scrpos(1,1).EQ.1) totlcrfnd=totlcrfnd-1
c
c there is no lcr at end of sequence
        IF (scrpos(2,totscrfnd).EQ.length) totlcrfnd=totlcrfnd-1
c
c write out the numbers found
 224    format(' There were ',i4,a,' Conserved Regions.')
        WRITE (6,224) totscrfnd,' Strongly'
        WRITE (6,224) totlcrfnd,' Loosely'
        IF (debugL)THEN
          WRITE (debugI,224) totscrfnd,' Strongly'
          WRITE (debugI,224) totlcrfnd,' Loosely'
        ENDIF
        END
```

```
      SUBROUTINE scr_fnd (ordera, seqstord, arr_b, seq_c, len_a,
     &alitord, arr_a, seq_a, outfile, hitstring )
      IMPLICIT none
c
      CHARACTER*(*) ordera(*), arr_a(*), arr_b(*), seq_c, seq_a
      CHARACTER*(*) outfile, hitstring
      INTEGER alitord(*), seqstord(*)
      INTEGER numscrseq, totseqfnd, numscrali, len_a
      INTEGER alig_rd, x, len_b, cntr, temp
      INTEGER alifil, debugI, maxlenfnd
      INTEGER scrwidth
      INTEGER screenwr, pos, screenwidth
      LOGICAL debugL
c
      COMMON /a1/ maxlenfnd
      COMMON /scrnwrite/ screenwr, screenwidth
      COMMON /a4/ totseqfnd
      COMMON /a7/ numscrseq
      COMMON /a11/ alifil
      COMMON /c1/ debugL, debugI
      COMMON /d1/ numscrali
      COMMON /d3/ scrwidth
 5301 FORMAT (a)
 5302 FORMAT (tr1,17(i4,','))
 5303 FORMAT (' There were ',i4,' alignments chosen.')
c
c the alignment numbers to use have previously been calculated & stored
c in alitord. renumber so that the alignment numbers are in numerical order,
c smallest to largest & write them to screen.
c
      CALL reorder (numscrali, alitord)
      WRITE (6,5303) numscrali
      CALL wrwords ('The alignments used to calculate the SCR''s
     & are numbers', screenwr, screenwidth)
      WRITE (6,5302) (alitord(x),x=1,numscrali)
      IF (debugL)THEN
        WRITE (debugI,5303) numscrali
        CALL wrwords ('The alignments used to calculate the
     & SCR''s are numbers', debugI, screenwidth)
        WRITE (debugI,5302) (alitord(x),x=1,numscrali)
      ENDIF
c
c the routine next marks in the hit string the asterices.
c we make up a string to make this simpler
c
      DO x=1,scrwidth
        hitstring(x:x)='*'
      END DO
c
c pull in the 1st alignment
c
      OPEN (alifil,FILE=outfile,form='unformatted',STATUS='old')
      REWIND (alifil)
      alig_rd=1
      cntr=1
      CALL rd_ali (arr_a, seqstord, alitord(1), alig_rd, totseqfnd,
     & len_a, seq_a, alifil)
c
c initialise seq_c (the results string ) remove any symbols that indicate hits
c
      DO x=1,len_a
        seq_c(x:x)=' '
      END DO
c
c pull in the next alignment - unless all the alignments have been read in.
c
 5325 CONTINUE
      cntr=cntr+1
      IF (cntr.GT.numscrali) GOTO 5389
```

```
        CALL rd_ali (arr_b, seqstord, alitord(cntr), alig_rd, totseqfnd, len_b,
     & seq_a, alifil)
c
c test the alignments -this is done pairwise, the first against the next.
c regions, >= SCRWIDTH long, identical in both the alignments are marked in the
c 'hit' string seq_a with an '*'. This is why we remove asterices from
c this string before calling the testing routine
c seq_a is based on arr_a , the first alignment read in (alitord(1)).
c
 5330   CONTINUE
        pos=INDEX(seq_a(1:len_a),'*')
        IF (pos.EQ.0) GOTO 5332
        seq_a(pos:pos)=' '
        GOTO 5330
 5332   CONTINUE
        CALL test (hitstring(1:scrwidth), len_a, len_b, arr_b, arr_a,
     & seq_a, ordera)
c
c Mark the regions that are conserved between all the alignments looked at.
c Place the results found for a pairwise comparison, stored in the hit string
c into the results string.
c First time round, the hits are placed in results string.
c Later times, by a  logical AND of seq_a and seq_c.
c
        IF (cntr.EQ.2) THEN
c
c consequence of wiping out the * in seq_a here is that the second time we
c call test, the bit before that to wipe out * is unneeded, however,
c the rest of the time it will be. The code is elegant & readable this way
c and only 2 lines extra are carried out so there.
c
 5428     CONTINUE
          pos=INDEX(seq_a(1:len_a),'*')
          IF (pos.EQ.0) GOTO 5430
          seq_c(pos:pos)='*'
          seq_a(pos:pos)=' '
          goto 5428
 5430     CONTINUE
        ELSE
          DO x=1,len_a
            IF (seq_c(x:x).EQ.'*'.AND.seq_a(x:x).NE.'*')
     &seq_c(x:x)=' '
          END DO
        ENDIF
        GOTO 5325
 5389   CONTINUE
        CLOSE (alifil)
        END

        SUBROUTINE test (hitstring, len_a, len_b, arr_b, arr_a,
     & seq_a, ordera)
        IMPLICIT none
c
c
        CHARACTER*(*) hitstring
        CHARACTER*(*) arr_a(*), arr_b(*), seq_a, ordera(*)
        INTEGER start_a, end_a
        INTEGER scrwidth
        INTEGER len_a, len_b, numscrseq, len_ab, len_bb
        INTEGER hits, screenwr, screenwidth, x
c
        COMMON /scrnwrite/ screenwr, screenwidth
        COMMON /a7/ numscrseq
        COMMON /d3/ scrwidth
 21     FORMAT (/,tr1,a11,i2,a16)
 20     FORMAT (tr1,a)
c
c initialise the string position counters
c if len_x is the length of array_x, len_xb is the length of that array
c that has to be searched.
```

```
c
 6405   CONTINUE
        len_ab=len_a-scrwidth+1
        len_bb=len_b-scrwidth+1
        start_a=0
        end_a=scrwidth-1
c
 6410   CONTINUE
        start_a=start_a+1
        IF (start_a.GT.len_ab) GOTO 6415
        end_a=end_a+1
        CALL ident (arr_a, arr_b, start_a, end_a, x, len_bb,
     & arr_a(1)(start_a:end_a), numscrseq, hits)
        IF (hits.EQ.0)THEN
          GOTO 6410
        ELSE IF (hits.EQ.1)THEN
          seq_a(start_a:end_a)=hitstring
          GOTO 6410
        ELSE
          WRITE (6,21)'There were ',hits,' places where ( '
          DO x=1, numscrseq
            CALL wrwords (arr_a(x)(start_a:end_a),
     & screenwr, screenwidth)
          END DO
          WRITE (6,20)' was found in the test alignment.'
          CALL wrwords ('This is very confusing - which hit
     & should be chosen for the strongly conserved region?.
     & Rule of thumb used is,
     & the last place found is the SCR. Frankly, your computer
     & recommends that you start again, but this time, you should
     & choose your own SCR''s.', screenwr, screenwidth)
          seq_a(start_a:end_a)=hitstring
          GOTO 6410
        ENDIF
 6415   CONTINUE
        END

        SUBROUTINE scr_alis_inp (alisname, alitord)
        IMPLICIT none
c
c the routine that allows the user to input which alignments will be used
c to calculate the scrs
c the routine for automatically working this out is called auto
c
        CHARACTER*(*) alisname
        CHARACTER charinp*6, cmd1*80, param_file*30
        INTEGER alitord(*), totalifnd, pos, x, numinp, numscrali, cntr
        INTEGER screenwr, screenwidth
        LOGICAL used
c
        COMMON /scrnwrite/ screenwr, screenwidth
        COMMON /a8/ totalifnd
        COMMON /d1/ numscrali
 5410   FORMAT (a)
 5411   FORMAT (a,i4,a)
 26     FORMAT (2(a,i4))
 27     FORMAT (i)
 28     FORMAT (tr1,17(i4,','))
 29     FORMAT (tr1,a)
c
        cntr=0
        used=.FALSE.
        pos=index(alisname,'.')
        param_FILE=alisname(1:pos)//'param_data'
        pos=pos+10
c
c print info about this routine, about choosing your own alignments and see
c if the user wants the parameter data file printed
c
        CALL wrwords ('User choosing alignments to calculate
```

```
     & Strongly Conserved Regions.', screenwr, screenwidth)
        WRITE (6,5411)' You have to enter ',numscrali,
     &' alignments'
        CALL wrwords ('NB-you cannot choose the same alignment twice.',
     & screenwr, screenwidth)
        CALL wrwords ('Alignments are numbered in the same order
     & that they are found in, in the alignment file used for input.
     & A list of the parameters for each
     & alignment can be found in the file ', screenwr, screenwidth)
        WRITE (6,29) param_file(1:pos)
        CALL wrwords ('This file can be printed just now on the
     & default printer if you enter print ( at Glasgow, this can
     & also be printed on the Biochem. Dept. line-printer by typing
     & BPRINT and the Chemistry Dept. line-printer by typing CPRINT) -
     & if you do not want this file printed, just press return.',
     & screenwr, screenwidth)
        READ (*,5410) charinp
        IF (charinp(1:1).EQ.'p'.or.charinp(1:1).EQ.'P')THEN
            x=16+pos
            cmd1='print '//param_file(1:pos)
            CALL gcgcmds (x, cmd1, 'print_spawn')
            GOTO 5430
        ENDIF
        IF (charinp(1:1).EQ.'b'.or.charinp(1:1).EQ.'B')THEN
            x=30+pos
            cmd1='print/queue=biochem '//param_file(1:pos)
            CALL gcgcmds (x, cmd1, 'print_spawn')
            GOTO 5430
        ENDIF
        IF (charinp(1:1).EQ.'c'.or.charinp(1:1).EQ.'C')THEN
            x=31+pos
            cmd1='print/queue=chemistry '//param_file(1:pos)
            CALL gcgcmds (x, cmd1, 'print_spawn')
            GOTO 5430
        ENDIF
C
c input the alignment numbers.
C
 5430   CONTINUE
        WRITE (6,26)' Choosing ',numscrali,' alignments:    Chosen ;', cntr
        CALL wrwords ('Enter the number of the alignment you want to use
     & for the SCR''s.', screenwr, screenwidth)
        IF (cntr.EQ.0) GOTO 5435
        CALL wrwords ('The alignments already chosen to calculate
     & the SCR''s are numbers', screenwr, screenwidth)
        IF (cntr.EQ.1)THEN
          WRITE (6,27) alitord(1)
        ELSE
          WRITE (6,28) (alitord(x),x=1,cntr)
        ENDIF
 5435   CONTINUE
        READ (*,27) numinp
        IF (numinp.LT.1.or.numinp.GT.totalifnd)THEN
          CALL wrwords ('That number is not in range -
     & please re-enter a number.', screenwr, screenwidth)
          GOTO 5430
        ENDIF
C
c check to see that this number is not already in use.
C
        IF (cntr.EQ.0)THEN
          GOTO 5440
        ELSE IF (cntr.EQ.1)THEN
          IF (alitord(1).EQ.numinp) used=.TRUE.
        ELSE
          DO x=1,cntr
            IF (alitord(x).EQ.numinp) used=.TRUE.
          END DO
        ENDIF
C
```

```
c the number has already been used!
c
        IF (used) THEN
          used=.FALSE.
          CALL wrwords ('This alignment has
     & already been chosen - please choose a different one.',
     & screenwr, screenwidth)
          GOTO 5430
        ENDIF
c
c  Number hasn't previously been used, so use this alignment
c
 5440   CONTINUE
        cntr=cntr+1
        alitord(cntr)=numinp
        IF (cntr.NE.numscrali) GOTO 5430
        END


        SUBROUTINE convert (numscrseq, arr_a, seq_c, scrpos,
     & messa, messe, messd, maxnumscr)
        IMPLICIT none
c
c this routine takes a string in which regions of interest in an array are
c marked by '*' along it's length, and it finds out the positions of these
c regions of interest, start & end.
c
        CHARACTER*(*) arr_a, seq_c, messa, messe, messd
        INTEGER start, width, end, cntr, numscrseq, scrpos(2,*)
        INTEGER maxnumscr
        INTEGER screenwr, screenwidth
        LOGICAL finish, bug, crash
c
        COMMON /scrnwrite/ screenwr, screenwidth
        COMMON /a6/ crash
c
        finish=.FALSE.
        end=0
        cntr=0
 111    CONTINUE
        IF (finish) GOTO 113
        start=index(seq_c(end+1:),'*')
        IF (start.EQ.0) GOTO 113
        start=start+end
        width=index(seq_c(start:),' ')-2
        IF (width.EQ.-2)THEN
          finish=.TRUE.
          width=len(seq_c)-start
        ENDIF
        end=start+width
        CALL bugremove (arr_a, start, end, bug)
c
c if a false SCR, ignore it, else count it
c
        IF (bug) GOTO 111
        cntr=cntr+1
c
c check to make sure that were not going to knacker the SCRPOS array
c This should never happen, as the array is set to very high levels.
c
        IF (cntr.EQ.maxnumscr)THEN
          CALL wrwords ('There are more Strongly Conserved
     & Regions in the alignments than are allowed in this program.',
     & screenwr, screenwidth)
          CALL wrwords (messa,screenwr, screenwidth)
          CALL wrwords (messe,screenwr, screenwidth)
          CALL wrwords (messd,screenwr, screenwidth)
          crash=.TRUE.
          GOTO 114
        ENDIF
```

```
        scrpos(1,cntr)=start
        scrpos(2,cntr)=end
        GOTO 111
 113    CONTINUE
c
c set the next position to zero for the other parts to realise where the end is
c
        scrpos(1,cntr+1)=0
        scrpos(2,cntr+1)=0
 114    CONTINUE
        END

        SUBROUTINE bugremove (arr_a, start, end, bug)
        IMPLICIT none
c
c the bug is
c          ****   these are not scr's!            ******** these are scr's!
c seq1  pa-----p                                  pa-----p
c seq2  pacdefgp                                  pacdefgp
c
c seq1  p-----ap                                  pa-----p
c seq2  pacdefgp                                  pacdefgp
c *=strongly conserved residue
c
c Not all regions that are ifentical in all the alignments looked at, are
c true SCRs (see above), but the false ones are easily weeded out.
c the cure is to check if an scr has been found which has gaps along its
c entire length in any of the sequences. this is of course a false scr and
c shouldn't be counted.
c
        CHARACTER*(*) arr_a(*)
        CHARACTER*1 alisgap
        INTEGER row, col, start, end, numscrseq
        LOGICAL bug
        COMMON /a7/ numscrseq
        COMMON /d4/ alisgap

        bug=.FALSE.
        DO row=1,numscrseq
          DO col=start,end
            IF (arr_a(row)(col:col).NE.alisgap)GOTO 777
          END DO
          bug=.TRUE.
          GOTO 779
 777      CONTINUE
        END DO
 779    CONTINUE
        END


        SUBROUTINE scrinp (seqstord, ordera, outfile, arr_a, seq_a,
     &scrpos, alitord)
        IMPLICIT none
c
c choosing the SCR's manually.
c 1) set the alignment the scrs are to be picked from.
c 2) pick the scr's
c
        CHARACTER*(*) seq_a, arr_a(*), ordera(*), outfile
        CHARACTER*(3) charinp
        INTEGER seqstord(*), totalifnd, totseqfnd, scrpos(2,*), unit
        INTEGER ali_no, start, end, seqlen, scrnum, lastpos, x
        INTEGER screenwr, screenwidth, outwidth, alitord(*)
        INTEGER alig_rd
        PARAMETER (UNIT=1)
c
        COMMON /scrnwrite/ screenwr, screenwidth
        COMMON /a3/ outwidth
        COMMON /a8/ totalifnd
        COMMON /a4/ totseqfnd
```

```
c
 21     FORMAT (a)
 22     FORMAT (a,i2,a)
 23     FORMAT (i)
 25     FORMAT (a,i)
 27     FORMAT (2a)
c
        CALL wrwords ('Choosing your own Strongly Conserved Region''s.',
     & screenwr, screenwidth)
c
c get the alignment that the scr are taken from & read into array arr_a
c
 24     CONTINUE
        CALL wrwords ('Input the number of the alignment (numbering
     & the alignments as they are found in the input alignment file)
     & that you want the SCRs to be taken from.',
     & screenwr, screenwidth)
        READ (*,23) ali_no
        IF (ali_no.LT.1.or.ali_no.GT.totalifnd)THEN
          CALL wrwords ('This number is out of range,
     & please re-enter.', screenwr, screenwidth)
          GOTO 24
        ENDIF
        alitord(1)=ali_no
        alig_rd=1
        OPEN (unit,FILE=outfile,STATUS='old',form='unformatted')
        REWIND (unit)
        CALL rd_ali (arr_a, seqstord, ali_no, alig_rd, totseqfnd, seqlen,
     & seq_a, unit)
        CLOSE (unit)
c
c now read in the scr's.
c
        scrnum=0
        lastpos=-2
 36     CONTINUE
        WRITE (6,22)' Starting position of SCR number ',scrnum+1,
     &' (or -1 to stop.)'
        READ (*,23) start
        IF (start.LT.0)GOTO 32
c
c we ensure that there must be at least 2 residues in the lcr
c
        IF (start.LT.lastpos+3)THEN
          CALL wrwords ('This is too close to the end of the last SCR.
     & Re-enter number.', screenwr, screenwidth)
          GOTO 36
        ENDIF
        IF (start.GE.seqlen)THEN
          CALL wrwords ('This is past or at the end of the alignment.
     & Re-enter number.', screenwr, screenwidth)
          GOTO 36
        ENDIF
 28     CONTINUE
        WRITE (6,25)' Finishing position of SCR number ',scrnum+1
        READ (*,23) end
        IF (end.GT.seqlen.or.end.LE.start)THEN
          CALL wrwords ('This number is out of range,
     & please re-enter.', screenwr, screenwidth)
          GOTO 28
        ENDIF
        WRITE (6,27)' This is the SCR for sequence: ',ordera(seqstord(1))
c
c we use wrarray and not wrwords, because wrwords copes with printing a
c line bigger than the screenwidth by breaking the line up at a convenient
c space. wrarray does this where there is no spaces in the line ie seq data
c
        CALL wrarray (end-start+1, 1, seq_a, ordera, arr_a,
     &ordera(seqstord(1)), arr_a(seqstord(1))(start:end), 5, outwidth)
        CALL wrwords ('Type yes if correct , no if wrong. [yes]:',
```

```
     & screenwr, screenwidth)
        READ (*,21) charinp
        IF (charinp(1:1).EQ.'n'.or.charinp(1:1).EQ.'N') GOTO 36
        scrnum=scrnum+1
        scrpos(1,scrnum)=start
        scrpos(2,scrnum)=end
        lastpos=end
        GOTO 35
 32     CONTINUE
c
c zero the position 1 after the scr's we we can work out how many scr there are
c
        scrpos(1,scrnum+1)=0
        scrpos(2,scrnum+1)=0
        END

        SUBROUTINE second (ordera, seq_b, seqname, seq_c, prednam, arr_a,
     & seq_a)
        IMPLICIT none
c
c this routine predicts the secondary structure of the sequences - there
c are totseqfnd sequences and their names are held in ordera.
c the length of the predicted sequence is output, then the actual prediction
c (the CHARACTERs used to represent each state are unified so that
c H=alpha helix, B=beta-strand, T=turn/random coil and nopredchar=no prediction)
c methods used are GGBSM ,PEPTIDESTRUCTURE , PEPPLOT.
c these give predictions by the methods of CGBSM, CF and GOR.
c the output names are held in prednam
c
        CHARACTER*(*) ordera(*), arr_a(*), prednam(*)
        CHARACTER*(*) seq_a, seqname, seq_c
        CHARACTER*(*) aa_type*20
        INTEGER totseqfnd, seq_b(*)
        INTEGER x, pos, row, length, y, loop
        INTEGER*2 aa_no(20)
        INTEGER screenwr, screenwidth
        LOGICAL crash, pred
        data aa_type /'ACDEFGHIKLMNPQRSTVWY'/
        data aa_no /1,2,10,10,8,6,3,4,12,5,5,11,7,11,12,9,9,4,8,8/
c
        COMMON /scrnwrite/ screenwr, screenwidth
        COMMON /a6/ crash
        COMMON /a4/ totseqfnd
        COMMON /b1/ pred
 3000   FORMAT (a)
 3001   FORMAT (a,i2)
c
c get the sequences into array (arr_a), in the order specified in ordera.
c
        CALL rdseqa (ordera, seq_a, arr_a, seqname)
        IF (crash) GOTO 3070
c
c run ggbsm
c
        OPEN (1,FILE=prednam(1),STATUS='new',form='unformatted')
        CALL wrwords ('Converting sequences to CGBSM format and
     & predicting their structure.', screenwr, screenwidth)
c
c firstly convert the amino-acid data into the numerical data that ggbsm uses
c each residue (these are stored in aa_type) is converted to the appropriate
c number in aa_no
c
        DO row=1,totseqfnd
          length=index(arr_a(row),' ')-1
c
c it an unknown residue is located eg., B, then zero is placed at that
c position in seq_b, and so this doesn't count when calculating the
c predictions of alpha/beta/strand
c
          DO x=1,length
```

```
            seq_b(x)=aa_no(index(aa_type,arr_a(row)(x:x)))
          END DO
          CALL ggbsm (seq_b, seq_a, length)
          WRITE (1) length
          WRITE (1) seq_a(1:length)
        END DO
        CLOSE (1)
        OPEN (3,FILE=prednam(2),STATUS='new',form='unformatted')
        OPEN (4,FILE=prednam(3),STATUS='new',form='unformatted')
        OPEN (7,FILE=prednam(4),STATUS='new',form='unformatted')
        OPEN (8,FILE=prednam(5),STATUS='new',form='unformatted')
        DO row=1,totseqfnd
          pos=index(ordera(row),' ')-1
          IF (.NOT.pred)THEN
c
c the gcg prediction programs have not been run before and so are now called
c
            CALL gcgprd (ordera(row)(1:pos))
            CALL wrwords
     &('Reading in these GCG predictions.',screenwr, screenwidth)
          ELSE
c
c the gcg prediction programs have been run before.
c
            CALL wrwords
     &('Reading in the GCG predictions.',screenwr, screenwidth)
          ENDIF
c
c read in these predictions and output them to seperate files.
c
          length=index(arr_a(row),' ')-1
          CALL p2s_rd (ordera(row)(1:pos), seq_a, seq_c)
          WRITE (3) length
          WRITE (3) seq_a(1:length)
          WRITE (4) length
          WRITE (4) seq_c(1:length)
          CALL gar_rd (ordera(row)(1:pos), seq_a, seq_c)
          WRITE (7) length
          WRITE (7) seq_a(1:length)
          WRITE (8) length
          WRITE (8) seq_c(1:length)
        END DO
        CLOSE (3)
        CLOSE (4)
        CLOSE (7)
        CLOSE (8)
 3070   CONTINUE
        END

        SUBROUTINE gcgprd (seqname)
        IMPLICIT none
c runs the gcg secondary structure prediction programs.
        CHARACTER*(*) seqname
        CHARACTER*80 cmd1, in_name*20, out_name*20, subprc*9
        INTEGER c,pos,pos2
        data subprc/'gcg_spawn'/
 1401   FORMAT (a)
c run the prediction programs
        pos=len(seqname)
        pos2=pos+4
        in_name=seqname//'.gcg'
        out_name=seqname//'.gar'
        WRITE (6,1401)' running peptidestructure.'
        c=37+pos
        cmd1(1:c)='peptidestructure/infile=''//in_name(1:pos2)//''/default'
        CALL gcgcmds (c,cmd1,subprc)
        WRITE (6,1401) ' running pepplot.'
        c=c+pos+7
        cmd1(1:c)='pepplot/noplot/infile1=''//in name(1:pos2)//''/gar=''//
     &out_name(1:pos2)//''/default'
```

```
        CALL gcgcmds (c,cmd1,subprc)
        END

        SUBROUTINE gcgcmds (c, cmds, subprc)
c takes a command in cmds of length, c, and carries it out.
c subprc will be 'gcg_spawn' if cmds is a gcg command or the actual
c command if its a vms command, d is the length of the command word.
        CHARACTER*(*) subprc, cmds
        INTEGER c, d, istat, ipid
        d=len (subprc)
        istat=0
        ipid=0
        istat=lib$spawn(cmds(1:c),,'sys$output',,subprc(1:d),ipid,,,,,)
        END

        SUBROUTINE gar_rd (seqname, str_a, str_b)
        IMPLICIT none
c read/write pepplot data
c nb - the first & last 8 residues are not predicted by the pepplot program
c and so these predictions are replaced with nopredchar
        CHARACTER*(*) seqname, str_a, str_b
        CHARACTER*80 input, blank*6
        CHARACTER*1 nopredchar
        INTEGER depstn, length, x
        LOGICAL pred
        COMMON /b1/ pred
        COMMON /d5/ nopredchar
 1401   FORMAT (a)
        OPEN (1,FILE=seqname//'.gar',STATUS='old')
        REWIND (1)
c find where the predictions are for the 'gor with decision constants' are.
        CALL defind (depstn)
c read past the header information
 1410   CONTINUE
        READ (1,1401) blank
        IF (blank(4:6).NE.'Pos') GOTO 1410
c insert the blank CHARACTERs for the first 8 residues that pepplot skips
        DO x=1,8
          str_a(x:x)=nopredchar
          str_b(x:x)=nopredchar
        END DO
        length=9
 1430   read(1,1401,end=1440) input
c
c read in the GOR with no Decision Constant prediction
        str_a(length:length)=input (13:13)
c
c read in the GOR with Decision Constant prediction
        str_b(length:length)=input (depstn:depstn)
        length=length+1
        GOTO 1430
c the last 8
 1440   CONTINUE
        DO x=length, length+7
          str_a(x:x)=nopredchar
          str_b(x:x)=nopredchar
        END DO
1470    CONTINUE
c
c We do not check the first or last 8 predictions because we already
c know what they are - nopredchar
c
        CALL changechar (str_a(9:length), 'A', 'H')
        CALL changechar (str_b(9:length), 'A', 'H')
        CALL changechar (str_a(9:length), 'C', 'L')
        CALL changechar (str_b(9:length), 'C', 'L')
        CALL changechar (str_a(9:length), '?', nopredchar)
        CALL changechar (str_b(9:length), '?', nopredchar)
        IF (pred)THEN
          CLOSE (1,STATUS='keep')
```

```
        ELSE
          CLOSE (1,STATUS='delete')
        ENDIF
        END

        SUBROUTINE dcfind (dcpstn)
        IMPLICIT none
c calculate where the dc info is & what FORMAT it is in.
c this routine reads a gcg produced 'pepplot' file already open on unit 1.
c firstly it works out what FORMAT statement is needed to read in the
c (predicted) percentage of alpha and beta in the protein, then it reads in
c the %s and then works out at what postion in the pepplot file,
c is the prediction with that decision constant. this is stored in dcpstn
c and passed back.
c
        CHARACTER*80 input, form*25
        INTEGER dcpstn, da, db, dcint(2), x
        REAL dcreal(2)
 1601   FORMAT (a)
c
c read the file until get to line with the % predictions in it
 1603   READ (1,1601) input
        IF (index (input,'alpha =').EQ.0)GOTO 1603
c
c work out the lengths of the alpha & beta predictions & so set the format.
        da=index(input(1:),'%')
        db=index(input(da+1:),'%')+da
c the % sign for the percentage of alpha residues can be at 63 if 0<%<9.9,
c or at 64 if 10.0<%<99.9. Similar idea for beta.
c NB both alpha and beta max % is 99.9, and not 100% because it is the number
c of sequences predicted/number of sequences (remember first and last 8 are
c not predicted).
        IF (da.EQ.64)THEN
          form(1:11)='(tr59,f4.1,'
          IF (db.EQ.79)THEN
            form(12:)='tr11,f4.1)'
          ELSE
            form(12:)='tr11,f3.1)'
          ENDIF
        ELSE
          form(1:11)='(tr59,f3.1,'
          IF (db.EQ.77)THEN
            form(12:)='tr11,f3.1)'
          ELSE
            form(12:)='tr11,f4.1)'
          ENDIF
        ENDIF
c
c read in the decision constants.
c
        READ (UNIT=input,FMT=form) dcreal(1),dcreal(2)
c
c work out where the pred with that dc is in file.
c
        DO x=1,2
          IF (dcreal(x).LT.20.0)THEN
            dcint(x)=1
          ELSE IF ((dcreal(x).GE.20.0).AND.(dcreal(x).LE.50.0))THEN
            dcint(x)=2
          ELSE
            dcint(x)=3
          ENDIF
        END DO
        dcpstn=( dcint(1)*21-1)+( ( dcint(2)-1 )*7 )
        END

        SUBROUTINE p2s_rd (seqname, str_1, str_2)
        IMPLICIT none
c
c this routine reads in the peptidestructure predictions of the file seqname,
```

```
c and places the predictions in str_1 & str_2
c
        CHARACTER*(*) seqname, str_1, str_2
        CHARACTER*80 input, blank*3
        INTEGER length, ifail
        LOGICAL pred
        COMMON /b1/ pred
 1401   FORMAT (a)
c
c read peptidestructure data
        OPEN (1,FILE=seqname//'.p2s',STATUS='old')
        REWIND (1)
 1405   CONTINUE
        READ (1,1401) blank
        IF (blank.NE.'Pos') GOTO 1405
        READ (1,1401) blank
        length=1
 1410   READ (1,1401,iostat=ifail) input
        IF (ifail.NE.0) GOTO 1420
c read in the CF prediction
        str_1(length:length)=input(47:47)
c read in the GOR prediction
        str_2(length:length)=input(55:55)
        length=length+1
        GOTO 1410
 1420   CONTINUE
        IF (pred)THEN
          CLOSE (1,STATUS='keep')
        ELSE
          CLOSE (1,STATUS='delete')
        ENDIF
        END

        SUBROUTINE ggbsm (input, output, length)
        IMPLICIT none
        INTEGER pos, input(*), first(3), last(3), length, x
        CHARACTER*(*) output
        real pis(-6:11,3), its(0:12,3), ns(3), c_sum, ecs(3)
        data first /-3,-6,-3/, last /6,11,3/
        data ns(1)/1.0/
        data ns(2)/1.2311/
        data ns(3)/1.5451/
c data for turn residues
        data (its(x,1),x=0,12) /0.0, 0.018074, 0.073735, 0.071903,
     & -0.03926,
     & -0.018393, 0.294359, 0.445666, 0.012034, 0.16152, 0.17015,
     & 0.187471, 0.109817/
        data (pis(x,1),x=-3,6) /0.154212 , 0.434159, 0.706102, 1.0,
     & 0.915659, 0.559442, 0.336078, 0.109662, 0.098532, 0.098014/
c data for helical residues
        data (its(x,2),x=0,12) /0.0, 0.147418, 0.024205, 0.114178,
     & 0.037925,
     & 0.111157, -0.081108, -0.144076, 0.063689, -0.031711, 0.050114,
     &0.01434, 0.093082/
        data (pis(x,2),x=-6,11) /0.169079, 0.195383, 0.326344, 0.430292,
     & 0.610814, 0.711503, 1.0, 0.996892, 0.845693, 0.74947, 0.589796,
     &0.538197, 0.458475, 0.374935, 0.332311, 0.291095, 0.221007,
     & 0.159564/
c data for beta strand residues
        data (its(x,3),x=0,12) /0.0, -0.005853, 0.096215, 0.016752,
     & 0.214617,
     & 0.098072, 0.01415, -0.078651, 0.124579, 0.079342, -0.062711,
     &-0.001842, -0.014743/
        data (pis(x,3),x=-3,3) /0.16148, 0.498551, 0.864631, 1.0,
     & 0.850783,  0.49398, 0.169746/
        COMMON /data/first, last, pis, its, ns
        external c_sum
        DO pos=1,length
          ecs(1)=exp(c_sum(pos,1,input,length) )
          ecs(2)=exp(c_sum(pos,2,input,length) )
```

```
          ecs(3)=exp(c_sum(pos,3,input,length) )
            IF (( (ecs(1).GE.ecs(2)).AND.(ecs(1).GE.ecs(3)) ).OR.
     &(pos.LT.4).OR.(pos.GT.(length-3)))THEN
          output(pos:pos)='T'
            ELSE IF (ecs(2).GE.ecs(3))THEN
          output(pos:pos)='H'
            ELSE
          output(pos:pos)='B'
            ENDIF
        END DO
        END

        FUNCTION c_sum (pos, type, input, length)
        real pis(-6:11,3), its(0:12,3), ns(3), total
        INTEGER cntr, length, input(*)
        INTEGER first(3), last(3), first_p, windowb
        INTEGER type, pos
        COMMON /data/first, last, pis, its, ns
        first_p=first(type)+pos
        windowb=last(type)+pos
        IF (first_p.LT.1)first_p=1
        IF (windowb.GT.length)windowb=length
        total=0.0
        DO cntr=first_p,windowb
            total=total+( pis(cntr-pos,type)*its(input(cntr),type) )
        END DO
        c_sum=total*ns(type)
        END

        SUBROUTINE calc_cons (arr_b, arr_c, prednam,
     &ordera, arr_a, seq_a, outfile)
        IMPLICIT none
c
c
c
        CHARACTER*(*) arr_a(*), arr_b(*), arr_c(*), ordera(*)
        CHARACTER*(*) seq_a, prednam
        CHARACTER*(*) outfile
        CHARACTER*1 alisgap, dummy
        INTEGER totseqfnd, x, y, length, len_b
        INTEGER row, pos, ifail, unit_a
        INTEGER outwidth, alifil, temp, debugI, maxlenfnd
        INTEGER screenwr, screenwidth
        INTEGER seqlen
        LOGICAL debugL
        COMMON /a1/ maxlenfnd
        COMMON /scrnwrite/ screenwr, screenwidth
        COMMON /a3/ outwidth
        COMMON /a4/ totseqfnd
        COMMON /a11/ alifil
        COMMON /a12/ unit_a
        COMMON /c1/ debugL, debugI
        COMMON /d4/ alisgap

 10     FORMAT (tr1,a)
 11     FORMAT (i)
 17     FORMAT (a,i4)
c
c initialise some variables used for writing debug data.
        IF (debugL) THEN
          seqlen=0
          temp=0
        ENDIF
        pos=index(prednam,'.')-1
        CALL wrwords ('Calculating a consensus '//prednam(1:pos)//
     &' 2ry structure prediction for each alignment.',screenwr,
     & screenwidth)
c
c read in the 2ry structure predictions into array arr_b
        OPEN (unit_a,FILE=prednam,form='unformatted',STATUS='old')
```

```
        REWIND (unit_a)
        DO row=1,totseqfnd
          READ (unit_a) length
          READ (unit_a) arr_b(row)(1:length)
c
c then we will be writing this data, so make sure its tidy with blanks
          IF (debugL)THEN
            DO x=length+1, maxlenfnd
               arr_b(row)(x:x)=' '
            END DO
            IF (length.GT.seqlen) seqlen=length
          ENDIF
        END DO
        CLOSE (unit_a,STATUS='delete')
c
c read in the alignment into arr_a and
c make arr_c the same as arr_b(2ry struc).
c also change prednam name (xx.data) to the new output name (xx.data2)
        pos=index(prednam,' ')-1
        prednam=prednam(1:pos)//'2'
        OPEN (unit_a,FILE=prednam,STATUS='new',form='unformatted')
        OPEN (alifil,FILE=outfile,STATUS='old',form='unformatted')
        REWIND (alifil)
 5510   CONTINUE
        READ (alifil,iostat=ifail) length
        IF (ifail.NE.0) GOTO 5550
        DO row=1,totseqfnd
          READ (alifil) arr_a(row)(1:length)
          arr_c(row)(1:length)=arr_b(row)(1:length)
        END DO
        IF (debugL) temp=temp+1
c
c place the gaps which are found in arr_a (the sequence alignment)
c into arr_c (the 2ry structure predictions) the 2ry prediction gap CHARACTER
c is the same as the alignment gap CHARACTER, alisgap.
        len_b=length-1
        DO row=1,totseqfnd
          x=1
 5520           CONTINUE
          pos=index(arr_a(row)(x:length),alisgap)
          IF (pos.EQ.0) GOTO 5530
c
c y holds the position in the string of the gap
          y=x+pos-1
          x=y+1
c
c if a gap is the last CHARACTER, x will be one greater than length
c (which will be y)
          IF (x.GT.length) GOTO 5525
          arr_c(row)(x:length)=arr_c(row)(y:len_b)
          arr_c(row)(y:y)=alisgap
          GOTO 5520
 5525           CONTINUE
          arr_c(row)(y:y)=alisgap
 5530           CONTINUE
        END DO
c
c now we have the predictions in the array with all the gaps in the right places
c then we can calculate the consensus sequence and write it out.
        CALL cons (arr_c, length, seq_a)
        WRITE (unit_a) length
        WRITE (unit_a) seq_a (1:length)
c
c If we want debug data written, write it.
c This info is for alignment no 4, we first write out the actual alignment (do
c this only once -during GGBSM run).
        IF (debugL.AND.temp.EQ.4) THEN
c put the consensus sequence into the array of predictions
          arr_c(totseqfnd+1)(1:length)=seq_a(1:length)
          ordera(totseqfnd+1)='          '
```

```
c write out the original 2ry predictions
          IF (prednam(1:4).EQ.'GGBS') THEN
            write (debugI, 17)' The secondary
     & structure predictions for alignment number ',4
            CALL wrarray
     &(seqlen, totseqfnd, seq_a, ordera, arr_a, dummy, dummy,
     & debugI, outwidth)
          ENDIF
c now write out the actual preds by this method
          pos=INDEX(prednam,'.')-1
          CALL wrwords ('The individual and
     & consensus structure predictions for this alignment by the method
     & of '//prednam(1:pos), debugI, screenwidth)
          CALL wrarray (length, totseqfnd+1,
     & seq_a, ordera, arr_c, dummy, dummy, debugI, outwidth)
        ENDIF
        GOTO 5510
 5550   CONTINUE
        CLOSE (alifil)
        CLOSE (unit_a)
        END

        SUBROUTINE cons (arr_c, length, seq_a)
        IMPLICIT none
c now calc consensus secondary sequence. rules involved are:--->
c if less than 3 predictions are involved, no prediction is made (nopredchar).
c if all preds (except 1) are in agreement, then capital letter is used for
c that pred, else, the one in the majority is chosen.
c if there is more than one state with the highest #, nopredchar is used
c the possible states are;
c 'h'= alpha helix
c 'b'=beta strand
c 't'=turn or random coil ( ie. not the 2 above)
c
        CHARACTER*(*) arr_c(*), seq_a
        CHARACTER*1 alisgap, nopredchar, temp3
        INTEGER totseqfnd, length, colm, row, tmp, alpha, beta, turn
        INTEGER votes, max_no
        COMMON /a4/ totseqfnd
        COMMON /d4/ alisgap
        COMMON /d5/ nopredchar
 6101   FORMAT (a)
        DO 6160 colm=1,length
          alpha=0
          beta=0
          turn=0
c work out the votes for each prediction.
          DO 6140 row=1,totseqfnd
            temp3=arr_c(row)(colm:colm)
            IF (temp3.EQ.nopredchar.or.temp3.EQ.alisgap)
     & GOTO 6140
            if(temp3.EQ.'t'.or.temp3.EQ.'T')THEN
              turn=turn+1
            ELSE IF(temp3.EQ.'h'.or.temp3.EQ.'H')THEN
              alpha=alpha+1
            ELSE IF(temp3.EQ.'b'.or.temp3.EQ.'B')THEN
              beta=beta+1
            ENDIF
 6140         END DO
c work out which vote is the determining one
          votes=alpha+beta+turn
          max_no=max(alpha,beta,turn)
c less than 3 predictions
          IF (votes.LT.3)THEN
            seq_a(colm:colm)=nopredchar
c there's a majority vote for one prediction
          ELSE IF (max_no.GE.votes-1)THEN
            if(alpha.GE.votes-1)THEN
              seq_a(colm:colm)='H'
            ELSE IF(beta.GE.votes-1)THEN
```

```
                seq_a(colm:colm)='B'
              ELSE IF(turn.GE.votes-1)THEN
                seq_a(colm:colm)='T'
              ENDIF
c no clear majority
            ELSE
              tmp=0
              IF (alpha.EQ.max_no)THEN
                seq_a(colm:colm)='h'
                tmp=tmp+1
              ENDIF
              IF (beta.EQ.max_no)THEN
                seq_a(colm:colm)='b'
                tmp=tmp+1
              ENDIF
              IF (turn.EQ.max_no)THEN
                seq_a(colm:colm)='t'
                tmp=tmp+1
              ENDIF
c equal votes for at least 2 prediction states
              IF (tmp.NE.1) seq_a(colm:colm)=nopredchar
            ENDIF
 6150             CONTINUE
 6160     END DO
          END
c the way that ggbsm calculates a consensus.
c         IF ((alpha.GE.beta).AND.(alpha.GE.turn) )THEN
c                 seq_a(colm:colm)='h'
c         ELSE IF (beta.GE.turn)THEN
c                 seq_a(colm:colm)='b'
c         else
c                 seq_a(colm:colm)='t'
c         endif
```

Appendix I The FORTRAN source code for the Mix'n'MatchB program

```
c
c       (c) Lachlan H. Bell
c       changed 11/5/93
c
c The mix'n'match suite of programs optimise protein multiple alignment
c by a process of finding Strongly Conserved Regions (SCRs) that can be used
c to anchor an alignment and then aligning the regions between these anchors.
c The anchor regions are found by analysis of initial alignments, previously
c generated using one or more automatic multiple alignment programs and a
c variety of scoring matrices and gap penalties. The suite of programs
c consists of two programs: Mix'n'MatchA, which finds the SCRs,
c works out the different possible alignments for each of the Loosely
c Conserved Regions (LCRs), predicts the secondary structure or reads in
c the predictions if they have already been carried out and works out a
c consensus structure prediction for each of these possible alignments;
c and Mix'n'MatchB, (this beaut) which takes the alignments which the user has
c chosen for each LCR and the alignments for the SCRs and joins them to
c produce the final alignment. The UWGCG suite of programs is also required.
c Two files are used as iput to MNMA. The first is a file containing all the
c pre-generated alignments. The alignment formats supported are those of ALIEN
c and PILEUP. The second input file lists the sequences present in the
c multiple alignments.
c This sequence input file can be in either NBRF or UWGCG 'file of file names'
c format
c
c
        SUBROUTINE help
        IMPLICIT none
        CHARACTER*(1) dummy
        INTEGER numinp, screenwr, screenwidth
        COMMON /scrnwrite/ screenwr, screenwidth
 30     FORMAT (i)
 40     FORMAT (a)
c
 300    CONTINUE
        CALL wrwords
     &(' 1) A quick look at the theoretical basis of this program.',
     & screenwr, screenwidth)
        CALL wrwords
     &(' 2) What input files are needed by this program?',screenwr,
     &screenwidth)
        CALL wrwords
     &(' 3) Miscellaneous.',screenwr, screenwidth)
        CALL wrwords
     &(' 4) Changing the program variables.',screenwr,
     &screenwidth)
        CALL wrwords
     &(' 5) How the program calculates Strongly Conserved
     &Regions.',screenwr, screenwidth)
        CALL wrwords
     &(' 6) Return to running the program.', screenwr, screenwidth)
        CALL wrwords ('   There is no other help at the minute',
     & screenwr, screenwidth)
        READ (*,30) numinp
        IF (numinp.LT.1.or.numinp.GT.6)THEN
          CALL wrwords ('That number is out of range - please
     & re-enter a number', screenwr, screenwidth)
          GOTO 300
        ENDIF
c
        IF (numinp.EQ.1)THEN
          CALL wrwords (
```

```
     &'   This program takes a large number of differing alignments
     & and
     & uses them to work out Strongly Conserved ''anchor'' Regions.
     & It then lists the different aligned blocks
     & that were found in between these SCR''s, together with their
     & predicted secondary structure. The user can then pick whichever
     & of these aligned blocks is best, (eg. using knowledge from
     & empirical
     & studies or the knowledge that gaps should only occur in turn
     & regions), giving a final alignment reasonably independant of
     & the starting parameters/programs used and containing more
     & expert information than a purely ''mathematical'' approach
     & can give.', screenwr, screenwidth)
C
        ELSE IF (numinp.EQ.2)THEN
          CALL wrwords ('     This program uses 2 files as input.
     & The first
     & of these is the alignment file. Different multiple alignments can
     & be generated using different programs and using
     & different gap parameters/scoring matrices.
     & All these differing alignments should be placed in
     & a single file, and it is the name of this single file that
     & should be entered (at the minute, this program
     & can read in alignments made by ALIEN and PILEUP - more formats will
     & be added whenever possible).',
     & screenwr, screenwidth)
          CALL wrwords ('
     &   The second file is the sequence file.
     & This should contain all of the sequences that were used in making
     & up the alignments. This sequence file can be in 2 formats,
     & NBRF or UWGCG. IF the sequences are in NBRF format, all the
     & sequences should be in this single file.
     & If the sequences are in
     & GCG format, the GCG ''file of file names'' should be used.
     & This format requires that in a single file there should be a
     & list of the file names of
     & the GCG sequences. To show that you are using this type of
     & file, and not a sequence file, you should precede the file
     & name with a ''@''.',
     & screenwr, screenwidth)
          CALL wrwords ('(NB if the GCG format is used, all the
     & individual sequence names are presumed to end in ''.GCG'' and
     & this program will crash if they do not!)', screenwr, screenwidth)
C
        ELSE IF (numinp.EQ.3)THEN
          CALL wrwords('   GCG must be initialised before this program
     & is run - otherwise this program crashes. The GCG package is
     & needed to predict the secondary structures by the methods of
     & Chou & Fasman (CF) and Garnier, Osguthorpe & Robson (GOR),
     & carried out by the GCG programs PEPTIDESTRUCTURE and PEPPLOT.
     & PEPTIDESTRUCTURE gives a CF and GOR prediction and PEPPLOT gives
     & a GOR prediction without decision constants and a GOR prediction
     & with decision constants (output is called cf, gor, gor2, gor3
     & respectively). To run these programs, sequences need to be in
     & GCG format and so if sequences were input as NBRF format, they
     & are reformatted with the GCG program FROMPIR. This is done by
     & this program automatically', screenwr, screenwidth)
          CALL wrwords ('   Residues in a sequence are indicated in
     & a short hand manner by using capital letters of the alphabet
     & eg. A=alanine. If a small letter is found in the
     & sequence data, it must be changed to the correct capital letter.
     & The problem is that although ''a'' may indicate alanine, some
     & sequence formats use small letters to indicate a cysteine
     & residue, with the next ''a'' found, bonded to
     & the first ''a'' residue, forming a cystine residue.
     & You are asked to decide whether small letters are being used to
     & indicate cysteines or if they mean the same as capital letters.',
     & screenwr, screenwidth)
C
        ELSE IF (numinp.EQ.4)THEN
```

```
      CALL wrwords (
     &'   Screen width holds the number of characters that can be
     & displayed horizontally across the screen (eg. on a vt100
     & terminal, either 80 or 132 characters can be displayed).
     & The files that can be printed out are the final results file
     & (PRINT.ME) and the debugging data files (xxxx.debug)
     & if these were asked to be printed. The number
     & of characters per line has a default value of 132, the same
     & width as a line printer, so that these files can be printed for
     & hard copy. The other variables are used in calculating the
     & Strongly Conserved Regions - there is a seperate help section
     & on these variables which you should read before modifying them.',
     & screenwr, screenwidth)
C
      ELSE IF (numinp.EQ.5)THEN
        CALL wrwords ('   Calculating the Strongly Conserved
     &Regions (SCR). The
     & program defines a SCR as being any region
     & which is the same in all the alignments looked at. NB -
     & not all of the alignments are looked at because some of the
     & alignments ALIEN & PILEUP produce can be very bad.
     & The program uses a default value of 15 alignments to look at,
     & however, the user can change this number if they wish -
     & either at the start (in the change the program variables section)
     & or in the calculate SCR section (with option number 4)',
     & screenwr, screenwidth)
        CALL wrwords ('   Once the number to look at is set,
     & which alignments are looked at can be choosen by the user or are
     & generated automatically.', screenwr, screenwidth)
        CALL wrwords (
     &'   The method to automatically choose alignments, uses the
     & number of identities in an alignment to pick out the ''good''
     & from ''bad''; the alignments with the highest number
     & of id''s are picked, then those with the next highest etc..'
     & , screenwr, screenwidth)
        CALL wrwords ('   With the number of alignments to be
     & looked at set, and with which specific alignments these are set, the
     & program then calculates the SCR''s.',
     & screenwr, screenwidth)
        CALL wrwords ('Press RETURN for more information.',
     & screenwr, screenwidth)
        READ (*,40) dummy
        CALL wrwords (
     &'   However, instead of these semi-automatic or
     & automatic methods, an option exists for
     & the user to completely define the SCR''s themselves.  To
     & do this, you first enter the number of the alignment that the
     & SCR''s will be  choosen from (eg. if the alignment was the
     & 5th alignment in your input alignment file, you would enter
     & ''5''). You then enter the starting & finishing positions
     & of the SCR''s (the numbering should be taken from the alignment
     & you have choosen.).' , screenwr, screenwidth)
        CALL wrwords ('   Calculating identical regions;
     & after the alignments to be looked at are choosen, regions
     & that are identical in all of these
     & alignments are defined as SCR''s. The minimum length that
     & a region can be is initially set at 3 residues. This was choosen
     & after careful consideration, but the user can change this if
     & they wish (although the range of values is set between 3 and 10).
     & Note that, the more sequences
     & there are in an alignment, the less likelihood there is
     & of finding regions that are identical (very similar, yes,
     & identical, no). To try and reduce this effect, we limit
     & the number of sequences that we look at. If there are more
     & than this set number of sequences in an alignment, we pick
     & sequences to calculate the SCR''s from at random.',
     & screenwr, screenwidth)
C
      ELSE IF (numinp.EQ.6)THEN
        GOTO 310
```

```
        ENDIF
c get them to enter return when finished reading and then they go back to the
c menu
        CALL wrwords ('Press RETURN when finished reading.',
     & screenwr, screenwidth)
        READ (*,40) dummy
        GOTO 300
c make a blank line so that everything is nice and legible
 310    CONTINUE
        WRITE (6,40) ' '
        END

        SUBROUTINE rd_lcr (lcrnum, start, end, alig_no, unit, numrows,
     &arr_x, ifail, offset)
        IMPLICIT none
c
c
        INTEGER lcrnum, start, end, alig_no, unit
        INTEGER numrows, ifail, offset
        INTEGER width, x
        CHARACTER*(*) arr_x(*)
        READ (unit,iostat=ifail) lcrnum
        IF (ifail.NE.0) GOTO 8420
        READ (unit,iostat=ifail) start
        IF (ifail.NE.0) GOTO 8420
        READ (unit,iostat=ifail) end
        IF (ifail.NE.0) GOTO 8420
        READ (unit,iostat=ifail) alig_no
        width=end-start+offset
        DO x=1,numrows
          READ (unit,iostat=ifail) arr_x(x)(offset:width)
          IF (ifail.NE.0) GOTO 8420
        END DO
 8420   CONTINUE
        END

        SUBROUTINE wrwords (string, unit, screenwidth)
        IMPLICIT none
C
c this routine will write a string to unit of width, screenwidth
c string should be just the length of the message or else lots of blanks may
c get  written. the width of the screen is set in screenwidth.
c the string does need to have spaces in it separating words,
c ie. this routine cannot be used for sequence output.
C
        INTEGER screenwidth, strlen, pos, z, tmpwid, unit
        CHARACTER*(*) string
 401    FORMAT (tr1,a)
        strlen=len (string)
        pos=1
 415    CONTINUE
        IF (strlen.LE.screenwidth)THEN
          WRITE (unit,401) string(pos:)
        ELSE
c
c find the end of a word and write up to that word
          tmpwid=pos+screenwidth-1
 417      continue
          IF (string(tmpwid:tmpwid).NE.' ')THEN
            tmpwid=tmpwid-1
            GOTO 417
          ENDIF
          WRITE (unit,401) string(pos:tmpwid)
          strlen=strlen-(tmpwid-pos+1)
          pos=tmpwid+1
          GOTO 415
        ENDIF
        END
c the old routine was this - the above will be faster
C         tmpwid=screenwidth-1
```

```
C 417            CONTINUE
C          z=pos+tmpwid
C          IF (string(z:z).NE.' ')THEN
C            tmpwid=tmpwid-1
C            GOTO 417
C          ENDIF
C          WRITE (unit,401) string(pos:z)
C          strlen=strlen-tmpwid
C          pos=z+1
C          GOTO 415
C        ENDIF
C to right this entire routine as a do routine:
c        pos=1
c        strlen=len(string)
c        DO WHILE (strlen.GT.screenwidth)
c          tmp=pos+screenwidth-1
c          DO WHILE (string(tmp:tmp.NE.' ')
c            tmp=tmp-1
c          END DO
c          WRITE (unit,401) string(pos:tmp)
c          strlen=strlen-(tmp-pos+1)
c          pos=tmp+1
c        END DO
c        WRITE (unit,401) string(pos:)
c        END


         SUBROUTINE wrarray (length, rows, tmpstr, ordera, array, name,
     &seq, unit, outwidth)
         IMPLICIT none
c
c this routine can output sequences (either many or a single ) in the format
c of, numbering on 1 line and on the lines underneath, the sequences.
c rows is the number of rows in the array or 1 if just a single sequence
c to be outted
c tmpstr is a string, at least the length of outwidth in which the data
c to be output is loaded - the data is names, space, sequence
c ordera contains the names of the sequences
c name is the name of the sequence if only a single sequence to be output
c array contains the sequences
c seq is this single sequence
c
c length is the length of the sequence strings to be outputted
c unit is the channel on which to output the data
c
         CHARACTER*(*) array(*), ordera(*), tmpstr, name, seq
         CHARACTER*200 blank
         INTEGER length, tmp_length, rows, cntr1, cntr2, x, unit
         INTEGER outwidth, tmp_width, row, temp2, jmp
         INTEGER namlen
         data blank/'
     &
     &
     &
     &                                                              '/
c
c if x is the length of ordera (in which are stored the names of the CHARACTERs)
c tmp_width=outwith-(x+1) - uses length of ordera+1 for the space in
c between the 2 strings.
c FORMAT (trx+2, ) - 1 for non-printing 1st space & 1 for space in between the 2
c strings
c temp2=(cntr2-cntr1)+x+3
c temp2 uses x+3 because; 1 space before the string, 1 space between the name
c and the sequence(s) & 1 for the maths of subtracting 1 number from another.
c
 6301    FORMAT (a)
c
c these FORMAT statements set up for a length of ordera of 11.
c
 30      FORMAT (tr13,i1)
```

```
 31     FORMAT (tr13,i2)
 32     FORMAT (tr13,i3)
 33     FORMAT (tr13,i4)
 34     FORMAT (tr13,i5)
 35     FORMAT (tr13,i1,a,i2)
 36     FORMAT (tr13,i1,a,i3)
 37     FORMAT (tr13,i2,a,i2)
 38     FORMAT (tr13,i2,a,i3)
 39     FORMAT (tr13,i3,a,i3)
 40     FORMAT (tr13,i3,a,i4)
 41     FORMAT (tr13,i4,a,i4)
 42     FORMAT (tr13,i4,a,i5)
 43     FORMAT (tr13,i5,a,i5)
c
c ensure that we know the value of the length of the input names
c so far, name should be the same length as ordera or smaller, but you
c never know if this may change in the future
c
        x=LEN(name)
        namlen=LEN(ordera(1))
        IF (x.GT.namlen) namlen=x
c
        cntr1=1
        cntr2=0
        tmp_width=outwidth-(namlen+1)
        tmp_length=length
 50     CONTINUE
        IF ( tmp_length.LT.tmp_width)THEN
          cntr2=cntr2+tmp_length
        ELSE
          cntr2=cntr2+tmp_width
        ENDIF
c if the numbers would be too close together, just write the first number
        IF (cntr2-cntr1.LT.16)THEN
          IF (cntr1.LT.10)THEN
            WRITE (unit,30) cntr1
          ELSE IF (cntr1.LT.100)THEN
            WRITE (unit,31) cntr1
          ELSE IF (cntr1.LT.1000)THEN
            WRITE (unit,32) cntr1
          ELSE IF (cntr1.LT.10000)THEN
            WRITE (unit,33) cntr1
          ELSE
            WRITE (unit,34) cntr1
          ENDIF
        ELSE
c
c writing both the numbers - the length of the region in between is variable
c
 10             CONTINUE
          IF (cntr1.LT.10)THEN
            IF (cntr2.LT.100)THEN
              WRITE (unit,35) cntr1,blank(1:cntr2-cntr1-2),cntr2
            ELSE
              WRITE (unit,36) cntr1,blank(1:cntr2-cntr1-3),cntr2
            ENDIF
          ELSE IF (cntr1.LT.100)THEN
            IF (cntr2.LT.100)THEN
              WRITE (unit,37) cntr1,blank(1:cntr2-cntr1-3),cntr2
            ELSE
              WRITE (unit,38) cntr1,blank(1:cntr2-cntr1-4),cntr2
            ENDIF
          ELSE IF (cntr1.LT.1000)THEN
            IF (cntr2.LT.1000)THEN
              WRITE (unit,39) cntr1,blank(1:cntr2-cntr1-5),cntr2
            ELSE
              WRITE (unit,40) cntr1,blank(1:cntr2-cntr1-6),cntr2
            ENDIF
          ELSE IF (cntr1.LT.10000)THEN
            IF (cntr2.LT.10000)THEN
```

```
            WRITE (unit,41) cntr1,blank(1:cntr2-cntr1-7),cntr2
          ELSE
            WRITE (unit,42) cntr1,blank(1:cntr2-cntr1-8),cntr2
          ENDIF
        ELSE
          WRITE (unit,43) cntr1, blank(1:cntr2-cntr1-9),cntr2
        ENDIF
      ENDIF
c
c write the information
c
      temp2=(cntr2-cntr1)+(namlen+3)
      IF (rows.EQ.1)THEN
        tmpstr=' '//name//' '//seq(cntr1:cntr2)
        WRITE (unit,6301) tmpstr(1:temp2)
      ELSE
        DO row=1,rows
          tmpstr=' '//ordera(row)//' '//array(row)(cntr1:cntr2)
          WRITE (unit,6301) tmpstr(1:temp2)
        END DO
      ENDIF
      WRITE (unit,fmt='(/)')
      cntr1=cntr1+tmp_width
      tmp_length=tmp_length-tmp_width
      IF (tmp_length.GT.0) GOTO 50
      END
```

Appendix J The FORTRAN source code for routines common to Mix'n'MatchA and Mix'n'MatchB

```
c       some subroutines comon to both mma and mmb
c       (c) Lachlan M. Bell
c       changed 11/5/93
c
        SUBROUTINE help
        IMPLICIT none
        CHARACTER*(1) dummy
        INTEGER numinp, screenwr, screenwidth
        COMMON /scrnwrite/ screenwr, screenwidth
 30     FORMAT (i)
 40     FORMAT (a)
c
 300    CONTINUE
        CALL wrwords
     &(' 1) A quick look at the theoretical basis of this program.',
     & screenwr, screenwidth)
        CALL wrwords
     &(' 2) What input files are needed by this program?',screenwr,
     &screenwidth)
        CALL wrwords
     &(' 3) Miscellaneous.',screenwr, screenwidth)
        CALL wrwords
     &(' 4) Changing the program variables.',screenwr,
     &screenwidth)
        CALL wrwords
     &(' 5) How the program calculates Strongly Conserved
     &Regions.',screenwr, screenwidth)
        CALL wrwords
     &(' 6) Return to running the program.', screenwr, screenwidth)
        CALL wrwords ('   There is no other help at the minute',
     & screenwr, screenwidth)
        READ (*,30) numinp
        IF (numinp.LT.1.or.numinp.GT.6)THEN
          CALL wrwords ('That number is out of range - please
     & re-enter a number', screenwr, screenwidth)
          GOTO 300
        ENDIF
c
        IF (numinp.EQ.1)THEN
          CALL wrwords (
     &'   This program takes a large number of differing alignments
     & and
     & uses them to work out Strongly Conserved ''anchor'' Regions.
     & It then lists the different aligned blocks
     & that were found in between these SCR''s, together with their
     & predicted secondary structure. The user can then pick whichever
     & of these aligned blocks is best, (eg. using knowledge from
     & empirical
     & studies or the knowledge that gaps should only occur in turn
     & regions), giving a final alignment reasonably independant of
     & the starting parameters/programs used and containing more
     & expert information than a purely ''mathematical'' approach
     & can give.', screenwr, screenwidth)
c
        ELSE IF (numinp.EQ.2)THEN
          CALL wrwords ('     This program uses 2 files as input.
     & The first
     & of these is the alignment file. Different multiple alignments can
     & be generated using different programs and using
     & different gap parameters/scoring matrices.
```

```
     & All these differing alignments should be placed in
     & a single file, and it is the name of this single file that
     & should be entered (at the minute, this program
     & can read in alignments made by ALIEN and PILEUP - more formats will
     & be added whenever possible).',
     & screenwr, screenwidth)
          CALL wrwords ('
     &   The second file is the sequence file.
     & This should contain all of the sequences that were used in making
     & up the alignments. This sequence file can be in 2 formats,
     & NBRF or UWGCG. IF the sequences are in NBRF format, all the
     & sequences should be in this single file.
     & If the sequences are in
     & GCG format, the GCG ''file of file names'' should be used.
     & This format requires that in a single file there should be a
     & list of the file names of
     & the GCG sequences. To show that you are using this type of
     & file, and not a sequence file, you should precede the file
     & name with a ''@''.',
     & screenwr, screenwidth)
          CALL wrwords (''(NB if the GCG format is used, all the
     & individual sequence names are presumed to end in ''.GCG'' and
     & this program will crash if they do not!)', screenwr, screenwidth)
C
        ELSE IF (numinp.EQ.3)THEN
          CALL wrwords('   GCG must be initialised before this program
     & is run - otherwise this program crashes. The GCG package is
     & needed to predict the secondary structures by the methods of
     & Chou & Fasman (CF) and Garnier, Osguthorpe & Robson (GOR),
     & carried out by the GCG programs PEPTIDESTRUCTURE and PEPPLOT.
     & PEPTIDESTRUCTURE gives a CF and GOR prediction and PEPPLOT gives
     & a GOR prediction without decision constants and a GOR prediction
     & with decision constants (output is called cf, gor, gor2, gor3
     & respectively). To run these programs, sequences need to be in
     & GCG format and so if sequences were input as NBRF format, they
     & are reformatted with the GCG program FROMPIR. This is done by
     & this program automatically', screenwr, screenwidth)
          CALL wrwords ('   Residues in a sequence are indicated in
     & a short-hand manner by using capital letters of the alphabet
     & eg. A=alanine. If a small letter is found in the
     & sequence data, it must be changed to the correct capital letter.
     & The problem is that although ''a'' may indicate alanine, some
     & sequence formats use small letters to indicate a cysteine
     & residue, with the next ''a'' found, bonded to
     & the first ''a'' residue, forming a cystine residue.
     & You are asked to decide whether small letters are being used to
     & indicate cysteines or if they mean the same as capital letters.',
     & screenwr, screenwidth)
C
        ELSE IF (numinp.EQ.4)THEN
          CALL wrwords (
     &'   Screen width holds the number of characters that can be
     & displayed horizontally across the screen (eg. on a vt100
     & terminal, either 80 or 132 characters can be displayed).
     & The files that can be printed out are the final results file
     & (PRINT.ME) and the debugging data files (xxxx.debug)
     & if these were asked to be printed. The number
     & of characters per line has a default value of 132, the same
     & width as a line printer, so that these files can be printed for
     & hard copy. The other variables are used in calculating the
     & Strongly Conserved Regions - there is a seperate help section
     & on these variables which you should read before modifying them.',
     & screenwr, screenwidth)
C
        ELSE IF (numinp.EQ.5)THEN
          CALL wrwords ('   Calculating the Strongly Conserved
     &Regions (SCR). The
     & program defines a SCR as being any region
     & which is the same in all the alignments looked at. NB -
     & not all of the alignments are looked at because some of the
```

```
     & alignments ALIEN & PILEUP produce can be very bad.
     & The program uses a default value of 15 alignments to look at,
     & however, the user can change this number if they wish -
     & either at the start (in the change the program variables section)
     & or in the calculate SCR section (with option number 4)',
     & screenwr, screenwidth)
          CALL  wrwords ('   Once the number to look at is set,
     & which alignments are looked at can be choosen by the user or are
     & generated automatically.', screenwr, screenwidth)
          CALL wrwords (
     &'   The method to automatically choose alignments, uses the
     & number of identities in an alignment to pick out the ''good''
     & from ''bad''; the alignments with the highest number
     & of id''s are picked, then those with the next highest etc..'
     & , screenwr, screenwidth)
          CALL wrwords ('   With the number of alignments to be
     & looked at set, and with which specific alignments these are set, the
     & program then calculates the SCR''s.',
     & screenwr, screenwidth)
          CALL wrwords ('Press RETURN for more information.',
     & screenwr, screenwidth)
          READ (*,40) dummy
          CALL wrwords (
     &'   However, instead of these semi-automatic or
     & automatic methods, an option exists for
     & the user to completely define the SCR''s themselves.  To
     & do this, you first enter the number of the alignment that the
     & SCR''s will be  choosen from (eg. if the alignment was the
     & 5th alignment in your input alignment file, you would enter
     & ''5''). You then enter the starting & finishing positions
     & of the SCR''s (the numbering should be taken from the alignment
     & you have choosen.).' , screenwr, screenwidth)
          CALL wrwords ('   Calculating identical regions;
     & after the alignments to be looked at are choosen, regions
     & that are identical in all of these
     & alignments are defined as SCR''s. The minimum length that
     & a region can be is initially set at 3 residues. This was choosen
     & after careful consideration, but the user can change this if
     & they wish (although the range of values is set between 3 and 10).
     & Note that, the more sequences
     & there are in an alignment, the less likelihood there is
     & of finding regions that are identical (very similar, yes,
     & identical, no). To try and reduce this effect, we limit
     & the number of sequences that we look at. If there are more
     & than this set number of sequences in an alignment, we pick
     & sequences to calculate the SCR''s from at random.',
     & screenwr, screenwidth)
c
          ELSE IF (numinp.EQ.6)THEN
            GOTO 310
          ENDIF
c get them to enter return when finished reading and then they go back to the
c menu
          CALL wrwords ('Press RETURN when finished reading.',
     & screenwr, screenwidth)
          READ (*,40) dummy
          GOTO 300
c make a blank line so that everything is nice and legible
 310      CONTINUE
          WRITE (6,40) ' '
          END

          SUBROUTINE rd_lcr (lcrnum, start, end, alig_no, unit, numrows,
     &arr_x, ifail, offset)
          IMPLICIT none
c
c
          INTEGER lcrnum, start, end, alig_no, unit
          INTEGER numrows, ifail, offset
          INTEGER width, x
```

```
      CHARACTER*(*) arr_x(*)
      READ (unit,iostat=ifail) lcrmun
      IF (ifail.NE.0) GOTO 8420
      READ (unit,iostat=ifail) start
      IF (ifail.NE.0) GOTO 8420
      READ (unit,iostat=ifail) end
      IF (ifail.NE.0) GOTO 8420
      READ (unit,iostat=ifail) alig_no
      width=end-start+offset
      DO x=1,numrows
        READ (unit,iostat=ifail) arr_x(x)(offset:width)
        IF (ifail.NE.0) GOTO 8420
      END DO
 8420 CONTINUE
      END

      SUBROUTINE wrwords (string, unit, screenwidth)
      IMPLICIT none
C
c this routine will write a string to unit of width, screenwidth
c string should be just the length of the message or else lots of blanks may
c get  written. the width of the screen is set in screenwidth.
c the string does need to have spaces in it separating words,
c ie. this routine cannot be used for sequence output.
C
      INTEGER screenwidth, strlen, pos, z, tmpwid, unit
      CHARACTER*(*) string
 401  FORMAT (tr1,a)
      strlen=len (string)
      pos=1
 415  CONTINUE
      IF (strlen.LE.screenwidth)THEN
        WRITE (unit,401) string(pos:)
      ELSE
C
c find the end of a word and write up to that word
        tmpwid=pos+screenwidth-1
 417    continue
        IF (string(tmpwid:tmpwid).NE.' ')THEN
          tmpwid=tmpwid-1
          GOTO 417
        ENDIF
        WRITE (unit,401) string(pos:tmpwid)
        strlen=strlen-(tmpwid-pos+1)
        pos=tmpwid+1
        GOTO 415
      ENDIF
      END
c the old routine was this - the above will be faster
C       tmpwid=screenwidth-1
C 417           CONTINUE
C       z=pos+tmpwid
C       IF (string(z:z).NE.' ')THEN
C         tmpwid=tmpwid-1
C         GOTO 417
C       ENDIF
C       WRITE (unit,401) string(pos:z)
C       strlen=strlen-tmpwid
C       pos=z+1
C       GOTO 415
C     ENDIF
C to right this entire routine as a do routine:
c     pos=1
c     strlen=len(string)
c     DO WHILE (strlen.GT.screenwidth)
c       tmp=pos+screenwidth-1
c       DO WHILE (string(tmp:tmp.NE.' ')
c         tmp=tmp-1
c       END DO
c       WRITE (unit,401) string(pos:tmp)
```

```
c          strlen=strlen-(tmp-pos+1)
c          pos=tmp+1
c        END DO
c        WRITE (unit,401) string(pos:)
c        END


         SUBROUTINE wrarray (length, rows, tmpstr, ordera, array, name,
     &seq, unit, outwidth)
         IMPLICIT none
c
c this routine can output sequences (either many or a single ) in the format
c of, numbering on 1 line and on the lines underneath, the sequences.
c rows is the number of rows in the array or 1 if just a single sequence
c to be outted
c tmpstr is a string, at least the length of outwidth in which the data
c to be output is loaded - the data is names, space, sequence
c ordera contains the names of the sequences
c name is the name of the sequence if only a single sequence to be output
c array contains the sequences
c seq is this single sequence
c
c length is the length of the sequence strings to be outputted
c unit is the channel on which to output the data
c
         CHARACTER*(*) array(*), ordera(*), tmpstr, name, seq
         CHARACTER*200 blank
         INTEGER length, tmp_length, rows, cntr1, cntr2, x, unit
         INTEGER outwidth, tmp_width, row, temp2, jmp
         INTEGER namlen
         data blank/'
     &
     &
     &
     &                                                          '/
c
c if x is the length of ordera (in which are stored the names of the CHARACTERs)
c tmp_width=outwith-(x+1) - uses length of ordera+1 for the space in
c between the 2 strings.
c FORMAT (trx+2, ) - 1 for non-printing 1st space & 1 for space in between the 2
c strings
c temp2=(cntr2-cntr1)+x+3
c temp2 uses x+3 because; 1 space before the string, 1 space between the name
c and the sequence(s) & 1 for the maths of subtracting 1 number from another.
c
 6301    FORMAT (a)
c
c these FORMAT statements set up for a length of ordera of 11.
c
 30      FORMAT (tr13,i1)
 31      FORMAT (tr13,i2)
 32      FORMAT (tr13,i3)
 33      FORMAT (tr13,i4)
 34      FORMAT (tr13,i5)
 35      FORMAT (tr13,i1,a,i2)
 36      FORMAT (tr13,i1,a,i3)
 37      FORMAT (tr13,i2,a,i2)
 38      FORMAT (tr13,i2,a,i3)
 39      FORMAT (tr13,i3,a,i3)
 40      FORMAT (tr13,i3,a,i4)
 41      FORMAT (tr13,i4,a,i4)
 42      FORMAT (tr13,i4,a,i5)
 43      FORMAT (tr13,i5,a,i5)
c
c ensure that we know the value of the length of the input names
c so far, name should be the same length as ordera or smaller, but you
c never know if this may change in the future
c
         x=LEN(name)
         namlen=LEN(ordera(1))
```

```
        IF (x.GT.namlen) namlen=x
c
        cntr1=1
        cntr2=0
        tmp_width=outwidth-(namlen+1)
        tmp_length=length
 50     CONTINUE
        IF ( tmp_length.LT.tmp_width)THEN
          cntr2=cntr2+tmp_length
        ELSE
          cntr2=cntr2+tmp_width
        ENDIF
c if the numbers would be too close together, just write the first number
        IF (cntr2-cntr1.LT.16)THEN
          IF (cntr1.LT.10)THEN
            WRITE (unit,30) cntr1
          ELSE IF (cntr1.LT.100)THEN
            WRITE (unit,31) cntr1
          ELSE IF (cntr1.LT.1000)THEN
            WRITE (unit,32) cntr1
          ELSE IF (cntr1.LT.10000)THEN
            WRITE (unit,33) cntr1
          ELSE
            WRITE (unit,34) cntr1
          ENDIF
        ELSE
c
c writing both the numbers - the length of the region in between is variable
c
 10             CONTINUE
          IF (cntr1.LT.10)THEN
            IF (cntr2.LT.100)THEN
               WRITE (unit,35) cntr1,blank(1:cntr2-cntr1-2),cntr2
            ELSE
               WRITE (unit,36) cntr1,blank(1:cntr2-cntr1-3),cntr2
            ENDIF
          ELSE IF (cntr1.LT.100)THEN
            IF (cntr2.LT.100)THEN
               WRITE (unit,37) cntr1,blank(1:cntr2-cntr1-3),cntr2
            ELSE
               WRITE (unit,38) cntr1,blank(1:cntr2-cntr1-4),cntr2
            ENDIF
          ELSE IF (cntr1.LT.1000)THEN
            IF (cntr2.LT.1000)THEN
               WRITE (unit,39) cntr1,blank(1:cntr2-cntr1-5),cntr2
            ELSE
               WRITE (unit,40) cntr1,blank(1:cntr2-cntr1-6),cntr2
            ENDIF
          ELSE IF (cntr1.LT.10000)THEN
            IF (cntr2.LT.10000)THEN
               WRITE (unit,41) cntr1,blank(1:cntr2-cntr1-7),cntr2
            ELSE
               WRITE (unit,42) cntr1,blank(1:cntr2-cntr1-8),cntr2
            ENDIF
          ELSE
            WRITE (unit,43) cntr1, blank(1:cntr2-cntr1-9),cntr2
          ENDIF
        ENDIF
c
c write the information
c
        temp2=(cntr2-cntr1)+(namlen+3)
        IF (rows.EQ.1)THEN
          tmpstr=' '//name//' '//seq(cntr1:cntr2)
          WRITE (unit,6301) tmpstr(1:temp2)
        ELSE
          DO row=1,rows
            tmpstr=' '//ordera(row)//' '//array(row)(cntr1:cntr2)
            WRITE (unit,6301) tmpstr(1:temp2)
          END DO
```

```
ENDIF
WRITE (unit,fmt='(/)')
cntr1=cntr1+tmp_width
tmp_length=tmp_length-tmp_width
IF (tmp_length.GT.0) GOTO 50
END
```

Appendix K The selection of alignments for each Loosely Conserved Region in eleven sequences of the serine protease family

A Mix'n'Match alignment using sixty five inital alignments of eleven serine protease sequences, as listed in table 6.2, produces the output file 'print.me' from the Mix'n'MatchA program. The sixty five alignments are generated using the gap penalties as listed in table 3.2 and the scoring matrices listed in table 3.3 and shown in appendix C. The alignment programs used are ALIEN [283] and PILEUP [178]. The contents of the file 'print.me', showing the different possible alignments found for the nine Loosely Conserved Regions are shown below. The user choses a single alignment for each LCR and inputs it into the Mix'n'MatchB program which produces the final alignment.

```
These are the different alignments found for loosely conserved region    1

LCR number:   1 from alignment number;---->   1
Total number of gaps in LCR :           36
Total number of identities in LCR :              0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       QVSLNSGYHF-----C
E2EC58      MAHLDIVTEKGLRVIC
HPHB        QAKMVSHHNLT----T
KFBO        QVLL-HGEIAAF---C
PLHU        QVSLRTRFGM- HFC
TBBO        QVMLFRKSPQEL--LC
N1CHG       QVSLQDKTGF---HFC
N1SGT       MVRLSMG--------C
N3EST       QISLQYRSGSSWAHTC
EXBO        QALLVNEENEGF---C
N2PKA       QVAIYHYSSFQ----C
GGBSM.DATA2 bbbb.ttttTt.Btt
CF.DATA2    bbbbbbttt....BBb
GOR.DATA2   bbb.tTTTTTTT.TTT
GOR2.DATA2  hb.ttTTttttt..tt
GOR3.DATA2  bbbbttttthtb.Bbt


LCR number:   1 from alignment number;---->   8
Total number of gaps in LCR :           36
Total number of identities in LCR :              0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       ----QVSLNSGYHF-C
E2EC58      MAHLDIVTEKGLRVIC
HPHB        QAKMVSHH----NLTT
KFBO        QVLL-HGEIAAF---C
PLHU        QVSLRTRFGM---HFC
TBBO        QVMLFRKSPQEL--LC
N1CHG       QVSLQDKTGF---HFC
N1SGT       MVRLSMG--------C
N3EST       QISLQYRSGSSWAHTC
EXBO        QALLVNEENEGF---C
N2PKA       ----QVAIYHYSSFQC
GGBSM.DATA2 bbbbbttttttthttt
CF.DATA2    bbbbbb.b.tt..BBb
GOR.DATA2   ...httttttTTTTTT
GOR2.DATA2  hbbttttttttthttt
GOR3.DATA2  bbbbbttttthtbbtt
```

```
LCR number:   1 from alignment number:-  >  17
Total number of gaps in LCR :          36
Total number of identities in LCR :             0
Names of seqs &
Struct. Pred. Methods.
              1
N1NTP         QVSLN----SGYHF-C
EZEC58        MAHLDIVTEKGLRVIC
HPHB          QAKM   VSHHNLTT
KFBO          QVLL--HGEI--AAFC
PLHU          QVSLRTRFGMHF---C
TBBO          QVMLFRKSPQ--ELLC
N1CHG         QVSLQDKTGFHF---C
N1SGT         MVRLSMG--------C
N3EST         QISLQYRSGSSWAHTC
EXBO          QALLV-NEEN--EGFC
N2PKA         QVAIY----HYSSFQC
GGBSM.DATA2   bbbbtTttt.ttttt
CF.DATA2      bbbbbbbbh.bbtbbb
GOR.DATA2     bbb.bTTTbTTTTTT
GOR2.DATA2    hb.ttTTT.cTT.ttt
GOR3.DATA2    bbbbtttt.h.b.btt


LCR number:   1 from alignment number:---->  19
Total number of gaps in LCR :          36
Total number of identities in LCR :             0
Names of seqs &
Struct. Pred. Methods.
              1
N1NTP         QVSL--NSGYHF---C
EZEC58        MAHLDIVTEKGLRVIC
HPHB          QAKMV----SHHNLTT
KFBO          QVLL-H---GEIAAFC
PLHU          QVSLRTRFG---MHFC
TBBO          QVMLFRK--SPQELLC
N1CHG         QVSLQDKTG---FHFC
N1SGT         MVRL--SMG------C
N3EST         QISLQYRSGSSWAHTC
EXBO          QALLVN---EENEGFC
N2PKA         QVAI--YHYSSFQ--C
GGBSM.DATA2   bbbbttttTtttt.tt
CF.DATA2      bbbbbB....t.bbbb
GOR.DATA2     bbb.bTTTTTTTTTT
GOR2.DATA2    hb.ttTtTTTtTtttt
GOR3.DATA2    bbbbbttTtt..bbtt


LCR number:   1 from alignment number:---->  21
Total number of gaps in LCR :          36
Total number of identities in LCR :             0
Names of seqs &
Struct. Pred. Methods.
              1
N1NTP         QVSL--NSGYHF---C
EZEC58        MAHLDIVTEKGLRVIC
HPHB          QAKMV----SHHNLTT
KFBO          QVLL-H---GEIAAFC
PLHU          QVSLRTRFG---MHFC
TBBO          QVMLFRK--SPQELLC
N1CHG         QVSLQDKTG---FHFC
N1SGT         MVRLSMG--------C
N3EST         QISLQYRSGSSWAHTC
EXBO          QALLVN---EENEGFC
N2PKA         QVAI--YHYSSFQ--C
GGBSM.DATA2   bbbb.tttTtttt.tt
CF.DATA2      bbbbbbttbb.t.bbbb
GOR.DATA2     bbb.tTtTTTTTTTTT
GOR2.DATA2    hb.ttTtTTTtTtttt
GOR3.DATA2    bbbb.tTTtt..bbtt


LCR number:   1 from alignment number:---->  25
Total number of gaps in LCR :          36
Total number of identities in LCR :             0
Names of seqs &
Struct. Pred. Methods.
              1
N1NTP         ----QVSLNSGYHF-C
EZEC58        MAHLDIVTEKGLRVIC
HPHB          QAKMV----SHHNLTT
KFBO          QVLL-E---GEIAAFC
PLHU          QVSLRTRFG---MHFC
TBBO          QVMLFRK--SPQELLC
N1CHG         QVSLQDKTG---FHFC
N1SGT         MVRLSMG--------C
N3EST         QISLQYRSGSSWAHTC
EXBO          QALLVN---EENEGFC
N2PKA         ----QVAIYHYSSFQC
GGBSM.DATA2   bbbbbtt.Ttttttt
CF.DATA2      bbbbbb.bb.tbbbbb
GOR.DATA2     ...httttTTTTTTT
GOR2.DATA2    hbhttttttTtT.ttt
GOR3.DATA2    bbbbbttttttc.bbtt
```

```
LCR number:    1 from alignment number:---->  26
Total number of gaps in LCR :          58
Total number of identities in LCR :             0
Names of seqs &
Struct. Pred. Methods.
           1                  18
N1NTP      --QVSL--NSGYHF---C
EZEC58     MAHLDIVTEKGLRVI--C
HPHB       --QAKMV----SHHNLTT
KFBO       --QVLL-H---GEIAAFC
PLHU       --QVSLRTRFG---MHFC
TBBO       --QVMLFRK--SPQELLC
N1CHG      --QVSLQDKTG---FHFC
N1SGT      MVRLSM----G------C
N3EST      --QISLQYRSGSSWAHTC
EXBO      --QALLVN---EENEGFC
N2PKA      --QVAI--YHYSSFQ--C
GGBSM.DATA2  ..bbbb.tTTTt.ttttt
CF.DATA2     ..bbbbbB.b.tt.bbbb
GOR.DATA2    ..bbb.tTtTTTTTTTTT
GOR2.DATA2   ..hb.ttBtTTTttttttt
GOR3.DATA2   ..bbbbbtTTtt..bbtt


LCR number:    1 from alignment number:---->  27
Total number of gaps in LCR :          36
Total number of identities in LCR :             0
Names of seqs &
Struct. Pred. Methods.
           1
N1NTP      ----QVSLNSGYHF-C
EZEC58     MAHLDIVTEKGLRVIC
HPHB       QAKMV----SHHNLTT
KFBO       QVLL-H---GEIAAFC
PLHU       QVSLRTRFG---MHFC
TBBO       QVMLFRK--SPQELLC
N1CHG      QVSLQDKTG---FHFC
N1SGT      MVRL--SMG------C
N3EST      QISLQYRSGSSWAHTC
EXBO       QALLVN---EENEGFC
N2PKA      ----QVAIYHYSSFQC
GGBSM.DATA2  bbbbbt.tTttttttt
CF.DATA2     hbbbhBbb..tbbbbb
GOR.DATA2    ...h.ttttTTTTTTT
GOR2.DATA2   hbhttttttBtT.ttt
GOR3.DATA2   bbbbbttttttt.bbtt


LCR number:    1 from alignment number:---->  28
Total number of gaps in LCR :          36
Total number of identities in LCR :             0
Names of seqs &
Struct. Pred. Methods.
           1
N1NTP      QVSLN------SGYHF-C
EZEC58     MAHLDIVTEKGLRVIC
HPHB       QAKMVSHHN----LTT
KFBO       QVLLHGEI----AAFC
PLHU       QVSLRTRFG---MHFC
TBBO       QVMLFRKSPQ---ELLC
N1CHG      QVSLQDKTG---FHFC
N1SGT      MVRL---------SMGC
N3EST      QISLQYRSGSSWAHTC
EXBO       QALLVNEE-N--EGFC
N2PKA      QVAIY----HYSSFQC
GGBSM.DATA2  bbbbttttTTTTbttte
CF.DATA2     bbbbbbh.ttTBbbbb
GOR.DATA2   bbb.tTTTTTTTTTC
GOR2.DATA2   hb.ttTttTTTT.ttt
GOR3.DATA2   bbbbttttt.TBbbtt


LCR number:    1 from alignment number:---->  35
Total number of gaps in LCR :          36
Total number of identities in LCR :             0
Names of seqs &
Struct. Pred. Methods.
           1
N1NTP      ----QVSLNSGYHF-C
EZEC58     MAHLDIVTEKGLRVIC
HPHB       QAKMV----SHHNLTT
KFBO       QVLLHGEI----AAFC
PLHU       QVSLRTRFG---MHFC
TBBO       QVMLFRKSPQ--ELLC
N1CHG      QVSLQDKTG---FHFC
N1SGT      MVRLS--------MGC
N3EST      QISLQYRSGSSWAHTC
EXBO       QALLVNEE-N--EGFC
N2PKA      ----QVAIYHYSSFQC
GGBSM.DATA2  bbbbbttTTttttttt
CF.DATA2     bbbbbB.b.tt.bbbb
GOR.DATA2    ...httttttTTTTTT
GOR2.DATA2   hbhttttttTTT.ttt
GOR3.DATA2   bbbbbttttt.tbbbtt
```

```
LCR number:   1 from alignment number:---->  37
Total number of gaps in LCR :           36
Total number of identities in LCR :              0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       QVSLNSGYHF-----C
EZEC58      MAILDIVTEKGLRVIC
HPHB        QAKMVSHHNLTT----
KFBO        QVLL-H-GEIAA--FC
PLHJ        QVSLRTRFGM---HFC
TBBO        QVMLFRKSPQEL--LC
N1CHG       QVSLQDKTGF---HFC
N1SGT       MVRLSMG--------C
N3EST       QISLQYRSGSSWAHTC
EXBO        QALLVN-EENEG--FC
N2PKA       QVAIYHYSSFQ----C
GGBSM.DATA2 bbbb.ttttTt.Btt
CF.DATA2    bbbbbbtttb.b.Bbb
GOR.DATA2   bbb.tTTTTTTT.TTT
GOR2.DATA2  hb.ttTTTttt..tt
GOR3.DATA2  bbbbttttthhb.B.t


LCR number:   1 from alignment number:---->  44
Total number of gaps in LCR :           36
Total number of identities in LCR :              0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       ----QVSLNSGYHF-C
EZEC58      MAILDIVTEKGLRVIC
HPHB        QAKMVSHHNLTT----
KFBO        QVLL-H-GEIAA--FC
PLHJ        QVSLRTRFGM---HFC
TBBO        QVMLFRKSPQEL- LC
N1CHG       QVSLQDKTGF---HFC
N1SGT       MVRLSMG--------C
N3EST       QISLQYRSGSSWAHTC
EXBO        QALLVN-EENEG--FC
N2PKA       ----QVAIYHYSSFQC
GGBSM.DATA2 bbbbbttttttttB.tt
CF.DATA2    bbbbbb.btbttbBBbb
GOR.DATA2   ...httttttTTTTTTT
GOR2.DATA2  hbhttttttttttBttt
GOR3.DATA2  bbbbbttttt..bBBtt


LCR number:   1 from alignment number:---->  46
Total number of gaps in LCR :           36
Total number of identities in LCR :              0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       QVSLNSGYHF-----C
EZEC58      MAILDIVTEKGLRVIC
HPHB        QAKMVSHHNLTT----
KFBO        QVLL-HGEIAAF---C
PLHJ        QVSLRTRFGM---HFC
TBBO        QVMLFRKSPQEL--LC
N1CHG       QVSLQDKTGF---HFC
N1SGT       MVRLSMG--------C
N3EST       QISLQYRSGSSWAHTC
EXBO        QALLVNEENEGF---C
N2PKA       QVAI---YHYSSFQ-C
GGBSM.DATA2 bbbbttttttttTbtt
CF.DATA2    bbbbbbtb..tbBBBb
GOR.DATA2   bbb.tTTtTTTT.TTT
GOR2.DATA2  hb.ttTTttttttBttt
GOR3.DATA2  bbbbtttttht.BBbt


LCR number:   1 from alignment number:---->  53
Total number of gaps in LCR :           36
Total number of identities in LCR :              0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       QVSLNSGYHF-----C
EZEC58      MAILDIVTEKGLRVIC
HPHB        QAKMVSHENLT----T
KFBO        QVLL-HGEIAAF---C
PLHJ        QVSLRTRFGM---HFC
TBBO        QVMLFRKSPQEL--LC
N1CHG       QVSLQDKTGF---HFC
N1SGT       MVRLSMG--------C
N3EST       QISLQYRSGSSWAHTC
EXBO        QALLVNEENEGF   C
N2PKA       QVAI---YHYS-SFQC
GGBSM.DATA2 bbbbttttttttEbtt
CF.DATA2    bbbbbbtb..t.BBBb
GOR.DATA2   bbb.tTTtTTTT.TTT
GOR2.DATA2  hb.ttTTttttttBttt
GOR3.DATA2  bbbbttttthtbBB.t
```

```
LCR number:   1 from alignment number:---->  55
Total number of gaps in LCR :            36
Total number of identities in LCR :               0
Names of seqs &
Struct. Pred. Methods.
              1
N1NTP         QVSL--NSG---YHSC
EZEC58        MARLDEVTERGLRVIC
HPHB          QAKMVSHHNLT----T
KFBC          QVLL-H-GEIAA--FC
PLHU          QVSLRCRFG---MHFC
TBBC          QVMLFRKSPQEL--LC
N1CHG         QVSLQDKTG---FHFC
N1SGT         MVRLSMG---------C
N3EST         QISLQYRSGSSWAHTC
EXBC          QALLVN-EENEG--FC
N2PKA         QVAIYHYSS---FQ-C
GGBSM.DATA2   bbbbbttttTTttbtt
CF.DATA2      bbbbbbttt...BBbb
GOR.DATA2     bbb.tTTTTTTTTTT
GOR2.DATA2    hb.ttTTTttt..tt
GOR3.DATA2    bbbbttttthhbBBbt


LCR number:   1 from alignment number:---->  56
Total number of gaps in LCR :            36
Total number of identities in LCR :               0
Names of seqs &
Struct. Pred. Methods.
              1
N1NTP         QVSLNSGYHF-----C
EZEC58        MARLDEVTERGLRVIC
HPHB          QAKMVSHHNLT----T
KFBC          QVLL-H-GEIA--AFC
PLHU          QVSLRTRFG---MHFC
TBBC          QVMLFRKSPQE--LLC
N1CHG         QVSLQDKTG---FHFC
N1SGT         MVRLSMG--------C
N3EST         QISLQYRSGSSWAHTC
EXBC          QALLVN-EENE--GFC
N2PKA         QVAIYHYSSFQ----C
GGBSM.DATA2   hbbb.ttttTT..btt
CF.DATA2      bbbbbbttttt..Bbbb
GOR.DATA2     bbb.tTTTTTT.TTTT
GOR2.DATA2    hb.ttTTttt.B.tt
GOR3.DATA2    bbbbttttthh.Bb.t


These are the different alignments found for loosely conserved region    2

LCR number:   2 from alignment number:---->   1
Total number of gaps in LCR :           135
Total number of identities in LCR :               0
Names of seqs &
Struct. Pred. Methods.
              1                                                    56
N1NTP         Y-----K-----SGIQVRLGEDNINVVEGNEQFISASKSIVHPSY----DSNTLNN
EZEC58        ----------KGREITVILGAHDVRKAESTQQKIKVEKQIIHESYNSAPNL----H
HPHB          LFLNHSENA-TAKDIAFTLTLYVGKK-----QLVEIEKVVLHPNYS--------QV
KFBC          I---KPGVKITV-----VAGEHNTEKPEPTEQKRNVIRAIPYHSYMASIN--KYSH
PLHU          L------EKSPRPSSYKVILGAHQEVNLEPHVQEIEVSRLFLEPTRK----------
TBBC          LLYPPWDKNFTVDDLLVRIGKHSRTRYERKVEKISMLDKIYIHPRYNWKE--NLDR
N1CHG               GVTTSDVVVAGEKIQKLKIAKVFKNSKYNSLTIN---N-----------
N1SGT         VSGSGNNTSITATGGVVDLQSGAAVKVRST-------KVLQAPGYNGT------GK
N3EST         V-----DRELT---FRVVVGEHNLNQNNGTEQYVGVQKIVVHPYWN--TDDVAAGY
EXBC          L---HQAKRFTV-----RVGDRNTEQEEGNEMAHEVEMTVKHSRFVK--E--TYDF
N2PKA         K-----N-----DNYEVWLGRHNLFENENTAQFFGVTADFPHPGFNLSADGKDYSH
GGBSM.DATA2   hTTTtTtt..btthbbBbtt..thbttttttthhbhb.bbtTttt.tth.HHHhh
CF.DATA2      bBBTTTt.BBbbttBBBBBb..hhhhhhhhhhhhhhhh...httttTT...ThHtt
GOR.DATA2     h....Tthbbbbt.bbbbbhHHHHHhHhHHhhhhhhhhbbbbtttT.TTT.ThHH
GOR2.DATA2    hB.TTTt.tbbbt.bBbb.ttttt.tthhhhhhhhhhhhbbbbtTttTTTTtTT.hh
GOR3.DATA2    bB.T.ttbbBbb.bbBBBbb.ttthhhhhhhhhhhhhhhbbbbbttttT.ttT...b


LCR number:   2 from alignment number:---->   8
Total number of gaps in LCR :           135
Total number of identities in LCR :               0
Names of seqs &
Struct. Pred. Methods.
              1                                                    56
N1NTP         ----------YKSGIQVRLGEDNINVVEGNEQFISASKSIVHPSY----DSNTLNN
EZEC58        ----------KGREITVILGAHDVRKAESTQQKIKVEKQIIHESYNSAPNL----H
HPHB          LFLNHSENA-TAKDIAFTLTLYVGKK-----QLVEIEKVVLHPNYS--------QV
KFBC          I---KPGVKITV-----VAGEHNTEKPEPTEQKRNVIRAIPYHSYM  ASINKYSH
PLHU          L-----EKSPRPSSYKVILGAHQEVNLEPHVQEIEVSRLFLEPTRK----------
TBBC          LLYPPWDKNFTVDDLLVRIGKHSRTRYERKVEKISMLDKIYIHPRY--NWKENLDR
N1CHG          . . . . GVTTSDVVVAGEKIQKLKIAKVFKNSKYNSLTIN---N----------
N1SGT         VSGSGNNTSITATGGVVDLQSGAAVKVRST-------KVLQAPGYNGT------GK
N3EST         V-----DRELT---FRVVVGEHNLNQNNGTEQYVGVQKIVVHPYWN--TDDVAAGY
EXBC          L---HQAKRFTV-----RVGDRNTEQEEGNEMAHEVEMTVKHSRFV--K--ETYDF
N2PKA         ----------KNDNYEVWLGRHNLFENENTAQFFGVTADFPEPGFNLSADGKDYSH
GGBSM.DATA2   hTTTtTtt...btthbbBbtt...thbttttttthhbhb.bbtTttTT.hhUUUhh
CF.DATA2      bBBTTTt.B2bbttBBBBBb..hhhhhhhhhhhhhhhh...htttt..tthhhHtt
GOR.DATA2     h....T.hbb..t.bbbbbhHHHHHhHhHHhhhhhhhhbbbbttt..TTTTThHH
GOR2.DATA2    hB.TTTt.tbtbt.bBbb.ttttt.tthhhhhHhhhhhhhbbbtTttTTTTtT.hh
GOR3.DATA2    bB.T.ttbbBbb.bbBBBbb.ttthhhhhhhhhhhhhhhbbbbbttttT.ttt...b
```

```
LCR number:   2 from alignment number:---->  17
Total number of gaps in LCR :          113
Total number of identities in LCR :            0
Names of seqs &
Struct. Pred. Methods.
                1                                                    54
N1NTP           YK--------SGIQVRLGEDNINVVEGNEQFISASKSIVHPSYDSNT----LNN
EZEC58          KG---------REITVILGAHDVRKAESTQQKIKVEKQIIHESYNSAPNL----E
H2HB            LFLN------HSENATAKDIAPTLTLYVGKKQLVEIEKVVLHPNYSQV--------
KFBO            IK---------PGVKITVVAGEHNTEKPEPTEQKRNVIRAIPYHSYNASINKYSE
PLHU            LE--------K--SPRPSSYKVILGAHQEVNLEPHVQEIEVSRLFL---EPTRK
TBBO            LLYPPWDKNFTVDDLLVRIGKHSRTRYERKVEKISKLDKIYIHPRYNWKENLDR
N1CHG           GV--------T--TS------DVVVAGEKIQKLKIAKVFKNSKYNS---LTINN
N1SGT           VSGSGNNTSITATGGVVDL-------QSGAAVKVRSTKVLQAPGYNGT-----GK
N3EST           VD--------RELTFRVVVGEHNLNQNNGTEQYVGVQKIVVHPYWNTDDVAAGY
EXBO            LH--------QAKRFTVTVGDRNTEQEEGNEMAHEV--EMTVKHSRFVKETYDF
N2PKA           KN--------DNVEVWLGRHNLFENENTAQFFGVTADFPHPGFNLSADGKDYSH
GGBSM.DATA2     htTT......thhhbbbbttttt..ttttttthhhhh.bb.ttttttttHHHhhhh
CF.DATA2        btBT.....Bttbbbbbbb.bbbb.hhhhhhhhhhhhhhh.ttttt.BE.htt
GOR.DATA2       tt........t..bbbbbbBBBBB.hHHHHhhhhhhhhbbbbttttttTTTThHH
GOR2.DATA2      tt.T....TBth.bbbbbttttttthhhhHhhhhhhhhbbbbttttttTttttthh
GOR3.DATA2      bb.T....BB.bbbbbbbbbbtttbhhhhhhhhhhhhbbbbbttttttt..b...b


LCR number:   2 from alignment number:---->  19
Total number of gaps in LCR :          135
Total number of identities in LCR :            0
Names of seqs &
Struct. Pred. Methods.
                1                                                    56
N1NTP           --YKSG---IQ----VRLG-EDNINVVEGNEQFISASKSIVHPSY-   DSNTLNN
EZEC58          K-----GREIT----VILGA-HDVRKAESTQQKIKVEKQIIHESYN-SAPNL---H
H2HB            LFLNHSE-NATAKD---IAPTLTLYV--GKKQLVEIEKVVLHPNY--------SQV
KFBO            I----KPGVKITV      VAGEHNTEKPEPTEQKRNVIRAIPYHSYNASINKY-SH-
PLHU            L-EKSP---RPSSYKVILGAHQEVNLEPHV-QEIEVSRLFLEPTRK----------
TBBO            LLYPPWDKNFTVDDLLVRIGKHSRTRYERKVEKISMLDKIYIHPR--YNWKENLDR
N1CHG           ·····G   VTTSD VVVAGEK-------I-QKLKIAKVFKNSKYN--S--LTINN
N1SGT           VSGSGNNTSITATGGVV-------DLQSGAAVKVRSTKVLQAPGYN-------GTGK
N3EST           V-DR-E---LT--FRVVVG-EHNLNQNNGTEQYVGVQKIVVHPYWN--TDDVAAGY
EXBO            L---HQAKRFTV------RVGDRNTEQEEGNEMAHEVEMTVK-HSR---FVKETYDF
N2PKA           KNLNYE---VW-------LG-RHNLFENENTAQFFGVTADFPHPGFNLSADGKDYSH
GGBSM.DATA2     hThtttTThhbbtbbBBb.ttttttttTttttthhh.hhhbbtTttt...hhhhhhh
CF.DATA2        b.ttttTBBbbbtbBBBBBH.hhhhhhhhhhhhhhhhhhh.tttt...h.thhhtt
GOR.DATA2       ht±TTttbbbbb..BBBb.H.B.,b.HHHHHhhhhhhhbbbbttT..TTTTThHh
GOR2.DATA2      h.±TTttt.bbb..BBBb.htttttttthHHHhhhhhhhhbbbbtTTT.TTTtttt.h
GOR3.DATA2      .Bttt....bbbbbBBBbbhttttthhhhhhhhhh.hhhbbbbbttt.T.tt.tbbb


LCR number:   2 from alignment number:---->  21
Total number of gaps in LCR :          113
Total number of identities in LCR :            0
Names of seqs &
Struct. Pred. Methods.
                1                                                    54
N1NTP           ---YKSGIQVRLGEDNINVVEGNEQFISASKSIVHPSY---------DSNTLNN
EZEC58          ---KGREITVILGAHDVRKAESTQQKIKVEKQII--------HESYNSAPNL-E
H2HB            LF-------LNHSENATAKDIAPTLTLYVGKKQLVEIEKVVLHPNYSQV-----
KFBO            I---KPGVKITV------VAGEHNTEKPEPTEQKRNVIRAIPYHSYNASINKYSH
PLHU            L-EKSP--RPSSYKVILGAHQEVNLEPHVQEIEVSRL----------FLEPTRK
TBBO            LLYPPWDKNFTVDDLLVRIGKHSRTRYERKVEKISMLDKTYIHPRYNWKENLDR
N1CHG           --------GVTTSDVVVAGEKIQKLKI-AKVFKNSKY----------NSLTINN
N1SGT           VSGSGNNFTSITATGGVVDLQSGAAVKVRSTKVLQAPGYN----------GTGK
N3EST           V-DRELTFRVVVGEHNLNQNNGTEQYVCVQKIVVHPYWN---------TDDVAAGY
EXBO            L---HQAKRFTV------RVGDRNTEQEEGNEMAHEVEMTVK-HSRFVK-ETYDF
N2PKA           ---KNLNYEVWLGRHNLFENENTAQFFGVTADFPHFGFNLS-----SADGKDYSH
GGBSM.DATA2     hTTttt..bbb.tthhb..ttttttbbb.thhb..ttt...Bttthtthhhhhhh
CF.DATA2        bB.tTttBBBBB.tbbbb.hh.hhhhhhhhhhhhhtthh...Bttttthhthtte
GOR.DATA2       h.tTttbbbbbb...h.hHhhHhh.hhhhhhhh.httH...b.TTTttTThHH
GOR2.DATA2      hBtTttt.bbbbt...hhhh,.hhh.hhhhhhhh..tt...B.tTTttttthh
GOR3.DATA2      bB.ttttbbbbbbbb.bbhhttt.bbbhhhh..h..tt..bBBbTt....hh.b


LCR number:   2 from alignment number:---->  23
Total number of gaps in LCR :          113
Total number of identities in LCR :            0
Names of seqs &
Struct. Pred. Methods.
                1                                                    54
N1NTP           YKSG--------IQVRLGEDNINVVEGNEQFISASKSIVHPSY----DSNTLNN
EZEC58          K-----GREITVI---LGAHDVRKAESTQQKIKVEKQIIHESYNSAPNL----H
HPHB            LFLN-HSENATAKDI---APTLTLYVGKKQLVEIEKVVLHPNYS--------QV
KFBO            I---KPGVKITV------VAGEHNTEKPEPTEQKRNVIRAIPYHSYNASINKYSH
PLHU            L-EKS--------P--RPSSYKVILGAHQEVNLEPHVQEIEVSRL---PLEPTRK
TBBO            LLYPPWDKNFTVDDLLVRIGKHSRTRYERKVEKISMLDKIYIHPRYNWKENLDR
N1CHG           ----------------GVTTSDVVVAGEKIQKLKI-AKVFKNSKY---NSLTINN
N1SGT           VSGSGNNFTSIT------ATGGVVDLQSGAAVKVRSTKVLQA-----PGYNGTGK
N3EST           VDRE---------LTFRVVVGEHNLNQNNGTEQYVGVQKIVVHPYWNTDDVAAGY
EXBO            L---HQAKRFTV------RVGDRNTEQEEGNEMAHEVEMTVK-HSRFVK-ETYDF
N2PKA           KNDN--------YEVWLGRHNLFENENTAQFFGVTADFPHPGFNLSADGKDYSH
GGBSM.DATA2     htttTtTTh.BbHHbbbbttttbbttttttthhhhh.bb.tttttt.thhhhhhh
CF.DATA2        bttTTTTbBBBBbBBBDDbtbbbbbhhhhhhhhhhhhhhh.ttttttthht.tt
GOR.DATA2       .ttT.TT.bBBBhh...bbBBBBbhhHHHhhhhhhh.bbbttttTTTttTTtHb
GOR2.DATA2      .tTTTTt.BBBhhb.h..tt..tthhhhHhhhhhhh.bbbttttTTTttttthh
GOR3.DATA2      .bttttt.h.BBBbbBbbbbbbb..bhhhhhhhhhhhhbbbbbttttTtbttt..bb
```

```
LCR number:   2 from alignment number:---->  25
Total number of gaps in LCR :          113
Total number of identities in LCR :            0
Names of seqs &
Struct. Pred. Methods.
                1                                                    54
N1NTP           YKSG--------IQVRLGEDNINVVEGNEQFISASKSIVHPSY---DSNTLNN
EZEC58          KGRE----------ITVILGAHDVRKAESTQQKIKVEKQIIHESYN-SAPNL---H
HPHB            LFLNESE-NATAKDI---APTLTLYVGKKQLVEIEKVVLHPNYS--------QV
KFBO            I---KPGVKITV-----VAGEHNTEKPEPTEQKRNVIRAIPYHSYNASINKYSH
PLHU            L-----EKSP----RPSSYKVILGAHQEVNLEPHVQEIEVSRL---FLEPTRK
TBBO            LLYPPWDKNFTVDDLLVRIGKHSRTRYERKVEKISMLDKIYIHPRYNWKENLDR
N1CHG           ----------------GVTTSDVVVAGEKIQKLKI-AKVFKNSKY---NSLTINN
N1SGT           VSGSGNNTSITATGGVVDLQS-------GAAVKVRSTKVLQAPGYNGT----GK
N3EST           V-----DRELTF---RVVVGEHNLNQNNGTEQYVGVQKIVVHPYWNTDDVAAGY
EXBO            L---HQAKRFTV-----RVGDRNTEQEEGNEMAHEVEMTVK-HSRFVK-ETYDF
N2PKA           KNDN--------YEVWLGRHNLFENENTAQFFGVTADFPHPGFNLSADGKDYSH
CGBSM.DATA2     hTt.tTtthhb.hhbbbbttttt...ttttthhhhh..b.bttttttt.hhhhhh
CF.DATA2        bttTTTt.BbBBbbBBBBbbbbbb.hhhhhhhhhhhhhhh..ttTTtt.h.htt
GOR.DATA2       .tTT.T..bbbbbbbbbbbbbBB..hHHHHhhhhhhh.bbbttTTTTttThHh
GOR2.DATA2      .tTTTTtttbbBbbbb...ttttttthhhHhhhhhhh.bbbtttTTTttttthh
GOR3.DATA2      .btt.ttbbbBBbbBbbbbbbbttbhhhhhhhhhhhhhbbbbbttttTttt....bb


LCR number:   2 from alignment number:   -->  26
Total number of gaps in LCR :          135
Total number of identities in LCR :            0
Names of seqs &
Struct. Pred. Methods.
                1                                                    56
N1NTP           . YK- . . . ---SGIQVRLGEDNINVVEGNEQFISASKSIVHPSY----DSNTLNN
EZEC58          KG----------REITVILGAHDVRKAESTQQKIKVEKQIIHESYN-SA--PNL-H
HPHB            LFLN-HSENATARDI-----APTLTLYVGKKQLVEIEKVVLHPNY--SQV------
KFBO            I---KPGVKITV-----VAGEHNTEKPEPTEQKRNVIRAIPYHSYNASINKY-SH-
PLHU            LEKSP----RPSSYKVIL-GAHQEVNLEPHV-QEIEVSRLFLEPTRK----------
TBBO            LLYPPWDKNFTVDDLLVRIGKHSRTRYERKVEKISMLDKIYIHPR---YNWKENLDR
N1CHG           ----G----VTTSD-VVVAGEK-------i-QKLKIAKVFKNSKYNS--S--TINN
N1SGT           VSGSGNNTSITATGGVV-------DLQSGAAVKVRSTKVLQAPGYN---------GTGK
N3EST           VDR-E----LT--FRVVVG-EHNLNQNNGTEQYVGVQKIVVHPYWNTDDV--AAGY
EXBO            L---HQAKRFTV-----RVGDRNTEQEEGNEMAHEVEMTVK-HSR---FVKETYDF
N2PKA           KNDN--------YEVW--LGRHNLFENENTAQFFGVTADFPHPGFNLSADGKDYSH
GGBSM.DATA2     hthtttTT.hbtthbDBB...tt.ttttttttthhh.hhhbbtTtttt.hhhhhhth
CF.DATA2        bthTTTTbBbbbtbBBBBBHhhhhhhhhhhhhhhhhhhhhhhh.ttTt.thhtt.htt
GOR.DATA2       htTT.TTbbbbbtbbhBb..HH..b.HhHHHHhhhhhhhbbbttT.TttTTTHHH
GOR2.DATA2      htTTtTT..bbbt.BbBhh.ttttttthhHIIIhhhhhhhhbbbtTtTTTttTt..h
GOR3.DATA2      .btT.tt.bbbb.bBBBBbh.tttbhhhhhhhhhh.hhhhbbbbttttB..htt...b


LCR number:   2 from alignment number:---->  27
Total number of gaps in LCR :          135
Total number of identities in LCR :            0
Names of seqs &
Struct. Pred. Methods.
                1                                                    56
N1NTP           ----------YKSGIQVRLGEDNINVVEGNEQFISASKSIVHPSY----DSNTLNN
EZEC58          ----------KGREITVILGAHDVRKAESTQQKIKVEKQIIHESYN-SAPNL---H
HPHB            LFL-NHSENATAKDIAPTLTLYVGKK-----QLVEIEKVVLHPNY--------SQV
KFBO            I---KPGVKITV-----VAGEHNTEKPEPTEQKRNVTRAIPYHSYN--ASINKYSH
PLHU            -LEKSP--RPSSYKVIL--GAHQEVNLEPHVQEIEV-----SRL-----FLEPTRK
TBBO            LLYPPWDKNFTVDDLLVRIGKHSRTRYERKVEKISMLDKIYIHPRY--NWKENLDR
N1CHG           --------GVTTSDVVV--AGEKIQKLKI-AKVFKN-----SKY-----NSLTINN
N1SGT           VSGSGNNTSITATGGVVDLQSGAAVKVRST--------KVLQAPGYN------GTGK
N3EST           -VDRELTFRVVVGEHNL--NQNNGTEQYVGVQKIVV-----HPYWN--TDDVAAGY
EXBO            L---HQAKRFTV-----RVGDRNTEQEEGNEMAHEVEMTVK-HSRF--VK-ETYDF
N2PKA           ----------KNDNYEVWLGRHNLFENENTAQFFGVTADFPHPGFNLSADGKDYSH
GGBSM.DATA2     htTttttbbbtthbbbtt..thhttttttthhhbh.hbbttttt...hhhhhhh
CF.DATA2        BbbTttbbbbbtBBBBbttthhhhhhhhhhhhhhhhhhhh.ttTT.....ht.tt
GOR.DATA2       .hTH..hbb.b.bbbbbbhHHHHHhhhhhHHhhhhhhhbbbttT..TTttThHh
GOR2.DATA2      ..tTttthtb.bt.bbbbhttttt...hhhHHhhhhhhhbbbttTT..Ttttthh
GOR3.DATA2      bB.TttbbbbbbbbbbbbbttttthhhhhhhhhhhhhhhbbbbbtTt..t....bbb


LCR number:   2 from alignment number:---->  28
Total number of gaps in LCR :          113
Total number of identities in LCR :            0
Names of seqs &
Struct. Pred. Methods.
                1                                                    54
N1NTP           Y-----KSGIQVRLGEDNINVVEGNRQFISASKSIVHP----SY---DSNTLNN
EZEC58          ----------KGREITVILGAHDVRKAESTQQKIKVEKQIIHESYNSAPNL--H
HPHB            LFLN-HSENATAKDIAPTLTLYVGKKQLVEIEKVVLH----------PNYSQV
KFBO            I---------KPGVKITVVAGEHNTEKPEPTEQKRNVIRAIPYHSYNASINKYSH
PLHU            L-----EKSPRPSSYKVILGAHQEVNLEPHVQEIEVSRLFLEPT---RK-----
TBBO            LLYPPWDKNFTVDDLLVRIGKHSRTRYERKVEKISMLDKIYIHPRYNWKENLDR
N1CHG           G-----VTT---SDVVVAGEKIQKLKIAKVFKNSKYNSITT------NN------
N1SGT           VSGSGNNTSITATGGVVDL----QSGAAVKVRSTKVLQAPGYNGT-------GK
N3EST           V-----DRELTFRVVVGEHNLNQNNGTEQYVGVQKLVVHPYWNT---DLVAAGY
EXBO            L---------HQAKRFTVRVGDRNTEQEEGNEMAHEVEMTVKHSRFVK--ETYDF
N2PKA           K-----NDNYRVWLGRHNLFENENTAQFFGVTADFPHPGFNLSA---DGKDYSH
GGBSM.DATA2     hTTT.Ttthhtthbbbbt..ttttttt..tthbb.hhttttTttHHttthhhhh
CF.DATA2        bBBT..t..bbbbbBbbBb.hhhhhhhhhhhhhhh...hhHTtTT.tt.hhtt
GOR.DATA2       ......Ttth.tbbBBbbb.HHhHHHHhhhhhhhhh...bbbt.hTTTTtTT.HH
GOR2.DATA2      .B.T.TTtt.bbbbbbbb.ttt.thhhhhhhhhhkb...bbttTTTTTttttthh
GOR3.DATA2      bB.T.Tttbbbbbbbbbbbb.ttthhhhhhhhhhbbbbbbb.tTTtttttt.b.b
```

```
LCR number:   2 from alignment number:  --->  35
Total number of gaps in LCR :          113
Total number of identities in LCR :            0
Names of seqs &
Struct. Pred. Methods.
                1                                                    54
N1NTP           ~~~~~~~YKSGIQVRLGEDNINVVEGNEQFISASKSIVHP----SYDS-NTLNN
EZEC58          -------KGREITVILGAHDVRKAESTQQKIKVEKQIIH----ESYNS-APNLH
HPHB            LFLN---HSENATAKDIAPTLTLYVGKKQLVEIEKVVLHPNYSQV--------
KFBO            I--------KPGVKITVVAGEHNTEKPEPTEQKRNVIRAIPYHSYNASINKYSH
PLHU            L-----EKSPRPSSYKVILGAHQEVNLEPHVQEIEV-SRLFLEPTRK-------
TBBO            LLYPPWDKNFTVDDLLVRIGKHSRTRYERKVEKISMLDKIYIHPRYNWKENLDR
N1CHG           G-----VTT---SDVVVAGEKIQKLKIAKVFKNSKY-NSLTI---NN-------
N1SGT           V--SGSGNNTSITATGGVVDLQS--GAAVKVRSTKVLQAPGYNGT-------GK
N3EST           V-----DRELTFRVVVGEHNLNQNNGTEQYVGVQKI-VVHPYWNTDDVAAGY
EXBO            L--------HQAKRFTVRVGDRNTEQEEGNEMAHEVEMTVKHSRFVK--ETYDF
N2PKA           -------KNDNYEVWLGRHNLFENENTAQFFGVTADFPHPGFNLSADG-KDYSH
GGBSM.DATA2     h..T..t.Tth..bbbbbtt.tt.ttttt.hhhh.bbb.tttTttth.IhhIHHI
CF.DATA2        b..T....ttthbbbbBbb.hh.hhhhhhhhhhhhhh..bt.tttttt.ht.tt
GOR.DATA2       h.....TTtt,.bbbbbbbb.IHIH.hHHHhhhhhhhhbbbttt.tttT.TThHH
GOR2.DATA2      h..T..TTtt.bbbbbbb.ttttt.hhhHhhhhhhhhbbbbttttttTTThhh
GOR3.DATA2      b.....tttbbbbbbbbbb...ttth.hhhhhhhhhhbbbbbbt.bttttBtIh..


LCR number:   2 from alignment number:  --->  36
Total number of gaps in LCR :          113
Total number of identities in LCR :            0
Names of seqs &
Struct. Pred. Methods.
                1                                                    54
N1NTP           --------YKSGIQVRLGEDNINVVEGNEQFISASKSIVHP----SYDSNTLNN
EZEC58          --------KGREITVILGAHDVRKAESTQQKIKVEKQTIH----RSYNSAPNLH
HPHB            LFL------NHSENATAKDIAPTLTLYVGKKQLVEIEKVVLHPNYSQ------V
KFBO            I--------KPGVKITVVAGEHNTEKPEPTEQKRNVIRAIPYHSYNASINKYSH
PLHU            L--------EKSPRPSSYKVILGAHQEVNLEPHVQEIEVSRLFLEPTRK-----
TBBO            LLYPPWDKNFTVDDLLVRIGKHSRTRYERKVEKISMLDKIYIHPRYNWKENLDR
N1CHG           G--------VTTSD------VVVAGEKIQKLKIAKVFKNSKYNSLTIEN-----
N1SGT           V--------SGSGNNTSITATGGVVDLQSGAAVKVRSTKVLQAPGYNGT----GK
N3EST           V---------DRELTFRVVVGEHNLNQNNGTEQYVGVQKIVVHPYWNTDDVAAGY
EXBO            L--------HQAKRFTVRVGDRNTEQEEGNEMAHEVEMTVKHSRFVK--ETYDF
N2PKA           --------KNDNYEVWLGRHNLFENENTAQFFGVTADFPHPGFNLSADGKDYSH
GGBSM.DATA2     h.......ttttthbbbb.ttttt.ttttchhhhhhhhbbbttttttt.hhHhh
CF.DATA2        b.......tttbbbbbbb..bb.hhhhhhhhhhhhhhhhh.btt.ttt.h.htt
GOR.DATA2       h.......Tttt.bbbbbbBBBb..hhHHHhhhhhhhhbbb.ttttttTTThHH
GOR2.DATA2      h.......Tttt.bbbbbhtt.ttthhhhhhhhhhhh.bbbttttttTTTThhh
GOR3.DATA2      b.........t.bbbbbbhbbb.t.hhhhhhhhhhhbbbbbbbttttttthh.b


LCR number:   2 from alignment number:  --->  37
Total number of gaps in LCR :          113
Total number of identities in LCR :            0
Names of seqs &
Struct. Pred. Methods.
                1                                                    54
N1NTP           YKSGIQVRLGEDNINVV--------EGNEQFISASKSIVHPSY----DSNTLNN
EZEC58          KGREITV----------ILGAHDVRKAESTQQKIKVEKQIIHESYNSAPN--LH
HPHB            LF------LNHSENATAKDIAPTLTLYVGKKQLVEIEKVVLHPNY------SQV
KFBO            IKPG---VKITV-----VAGEHNTEKPEPTEQKRNVIRAIPYHSYNASINKYSH
PLHU            LEK--SPRPSSYKVILG--------AEQEVNLEPHVQEIEVSRLFL---EPTRK
TBBO            LLYPPWDKNFTVDDLLVRIGKHSRTRYERKVEKISMLDKIYTHPRYNWKENLDR
N1CHG           GVT--TSD------VVV--------AGEKIQKLKIAKVFKNSKYNS---LTINN
N1SGT           VSGSGNNTSITATGGVVDL-------QSGAAVKVRSTKVLQAPGY----NGTGK
N3EST           VDRELTFRVVVGEHNLN--------QNNGTEQYVGVQKIVVHPYWNTDDVAAGY
EXBO            LHQA---KRFTV-----RVGDRNTEQEEGNEMAHEVEMTVKHSRFVK--ETYDF
N2PKA           KNDNYRVWLGRHNLFRN--------RNTAQFFGVTADFPHPGFNLSADGKDYSH
GGBSM.DATA2     htt..t...bbtth.bbBtTtTTt.ttttthhhhhhh.hb.tttth...hhhhh
CF.DATA2        bttTbbbbBBbb..bbbBB..Tt.hhhhhhhhhhhhhhhh..tttt.tt.t.tt
GOR.DATA2       tttt.bbbbbBB..bbhbh.....hhHHEhhhhhhhhhhh.ttttTTTtThHh
GOR2.DATA2      ttttbbbbbbbbtbbb.bh.ttttthhhhEhhhhhhhhh.bbttttttTTTtthh
GOR3.DATA2      bbtt.bbbbbbbbbbbbbh..ttt..hhhhhhhhhh..bbbbbttttttt.b.b


LCR number:   2 from alignment number:  --->  39
Total number of gaps in LCR :          113
Total number of identities in LCR :            0
Names of seqs &
Struct. Pred. Methods.
                1                                                    54
N1NTP           YKSGI---------QVRLGEDNINVVEGNEQFISASKSIVHPSY----DSNTLNN
EZEC58          KGR-----------EITVILGAHDVRKAESTQQKIKVEKQIIHESYNSAPNL--H
HPHB            LFLN------HSENATAKDIAPTLTLYVGKKQLVEIEKVVLHPNY------SQV
KFBO            IKPGVK---------ITVVAGEHNTEKPEPTEQKRNVIRAIPYHSYNASINKYSH
PLHU            LEKSP----------RPSSYKVILGAHQEVNLEPHVQEIEVS------RLFLEPTRK
TBBO            LLYPPWDKNFTVDDLLVRIGKHSRTRYERKVEKISMLDKIYIHPRYNWKENLDR
N1CHG           GVTTS----------D-------VVVAGEKIQKLKIAKVFKNS-----KYNSLTINN
N1SGT           VSGSGNNTSITATGGVVDL-------------QSGAAVKVRSTKVLQAPGYNGTGK
N3EST           VDREL-------------CFRVVVGEHNLNQNNGTEQYVGVQKIVVHPYWNTDDVAAGY
EXBO            LHQAKR---------FTVRVGDRNTEQEEGNEMAHEVEMTVKHSRFVK--ETYDF
N2PKA           KNDNY---------EVWLGRHNLFENENTAQFFGVTADFPHPGFNLSADGKDYSH
GGBSM.DATA2     httttt....BTThbbbbtttttt.ttttttthhhhhbb...tttttt.hhhhh
CF.DATA2        bttttbT.....B.bbbbBb..bbhhhhhhhhhhhhhhhhbbbttttttt...tt
GOR.DATA2       ttttthB.....Hbbhbbbb2BB.HhhhHHHhhhhhhhhhhBb.ttttTTTthhHh
GOR2.DATA2      tttttT....BBH.bbbbhttttttthhhhhhhhhhhhhhbb.ttttttTttthh
GOR3.DATA2      bbtt.t....BBHbbbbbbbb.btthhhhhhhhhhhh..bbbbbttttttthbbb
```

```
LCR number:   2 from alignment number:---->  44
Total number of gaps in LCR :          113
Total number of identities in LCR :            0
Names of seqs &
Struct. Pred. Methods.
                1                                                    54
N1NTP           YKSGI--------QVRLGEDNINVVEGNEQFISASKSIVHPSY----DSNTLNN
EZEC58          KGREI--------TVILGAHDVRKAESTQQKIKVEKQIIHESYNSAPNL----H
HPHB            LF--LNHSENATAKDIAPTLTLYVGKKQLVEIEKVVLHPNYSQV----------
KFBO            IKPG---VKITV-----VAGEHNTEKPEPTEQKRNVIRAIPYHSYNASINKYSH
PLHU            LEK--SPR--------PSSYKVILGAHQEVNLEPHVQEIEVSRLFL---EPTRK
TBBO            LLYPPWDKNFTVDDLLVRIGKHSRTRYERKVEKISMLDKIYIHPRYNWKENLDR
N1CHG           GVT--TSD--------------VVVAGEKIQKLKIAKVFKNSKYNS---LTINN
N1SGT           VSGSGNNTSITATGGVVDLQSGAAVKVRSTKVLQAPGYN----------GTGK
N3EST           VDRELTFRV--------VVVGEHNLNQNNGTEQYVGVQKIVVHPYWNTDDVAAGY
EXBO            LHQA---KRFTV--    RVGDRNTEQEEGNEMAHEVEMTVKHSRFVK--ETYDF
N2PKA           KNDNY--------EVWLGRHNLFENENTAQFFGVTADFPHPGFNLSADGKDYSH
GGBSM.DATA2     htt.httt.hB.Habbbbtttt.,ttttttthhhh,..h,ttttth.hhhhhhh
CF.DATA2        bttTbb.bBBBB,bBBbBbb.,b,hhhhhhhhhhhhhhhhh,.tt..tthht.tt
GOR.DATA2       tttt...bbbBBHbbhbbb...bbhhhHHHhhhhhhh..bhtttt.TTTTThHH
GOR2.DATA2      tttttttt.BBBHbbbbbttttttthhhhhhhhhhhh.bbbtttttTTttthhh
GOR3.DATA2      bbtt.ttbbBBBHbBbbbb.t.tbhhhhhhhhhhhbbbbbb..ttt.tt....b


LCR number:   2 from alignment number:---->  45
Total number of gaps in LCR :          113
Total number of identities in LCR :            0
Names of seqs &
Struct. Pred. Methods.
                1                                                    54
K1NTP           YKSGI--------QVRLGSDNINVVEGNEQFISASKSIVHPSY----DSNTLNN
EZEC58          KGREI--------TVILGAHDVRKAESTQQKIKVEKQIIHESYNSAPNL----H
HPHB            LF-------------LNHSENATAKDIAPTLTLYVGKKQLVEIEKVVLHPNYSQV
KFBO            IKPGVK--------ITVVAGEHNTEKPEPTEQKRNVIRAIPYHSYNASINKYSH
PLHU            LEKSPRPSSYKV--------KVILGAHQEVNLEPHVQEIEVS-----RLFLEPTRK
CBBO            LLYPPWDKNFTVDDLLVRIGKHSRTRYERKVEKISMLDKIYIHPRYNWKENLDR
N1CHG           GVTTSD--------------VVVAGEKIQKLKIAKVFKNS-----KYNSLTINN
N1SGT           VSGSGNNTSITATGGVVDLQSGAAVKVRSTKVLQAPGYNC----------TGK
N3EST           VDRELTFRVV--------VGEHNLNQNNGTEQYVGVQKIVVHPYWNTDDVAAGY
EXBO            LHQAKR--------FTVRVGDRNTEQEEGNEMAHEVEMTVKHSRFVK--ETYDF
N2PKA           KNDNY--------EVWLGRHNLFENENTAQFFGVTADFPHPGFNLSADGKDYSH
GGBSM.DATA2     httttttT.B....bbbbttt.hhhttttt..bh.hhb.tttttt..hhhhhhh
CF.DATA2        btttb.TBBB...BBBBBbt..hhhBhhhhhhhhhhhhhhhhhhTTttttthh.,tt
GOR.DATA2       ttttbbTT.B...bbbbbbHhhHHHhhhhhhhhhhhh,.bbtt.t.TttTThHh
GOR2.DATA2      tttt.tTT.B...BBbb.hhttt.thhhhhhhhhhhhh.bbttttttttttthh
GOR3.DATA2      bbttb.T.BB...Bbbb.bb.ttthhhhhhhhb.hhhbb.bb..ttt.ttt.hhh


LCR number:   2 from alignment number:---->  46
Total number of gaps in LCR :          135
Total number of identities in LCR :            0
Names of seqs &
Struct. Pred. Methods.
                1                                                    56
N1NTP           YKSG-----IQV-----RLGEDNINVVEGNEQFISASKSIVHPSY----DSNTLNN
EZEC58          ----------KGREITVILGAHDVRKAESTQQKIKVEKQIIHESYNSAPNL----H
HPHB            LFLNHSENA-TAKDIAPTLTLYV-----GKKQLVEIEKVVLHPNYS-------QV
KFBO            I---KPGVKITV-----VAGEHNTEKPEPTEQKRNVIRAIPYHSYNAS--INKYSH
PLHU            LEKSPRPSSYKV-----ILGAHQEVNLEPHVQEIEVSRLFLEPT------RK----
TBBO            LLYPPWDKNFTVDDLLVRIGKHSRTRYERKVEKISMLDKIYIHPRYNW--KENLDR
N1CHG           GVTTS-----DV-----VVAGEK-------IQKLKIAKVFKNSKYNSLTINN----
N1SGT           VSGSGNNTSITATGGVVDLQS-------GAAVKVRSTKVLQAPGY------NGTGK
N3EST           VD---RELTFRV-----VVGEHNLNQNNGTEQYVGVQKIVVHPYWNTDDVAAGY--
EXBO            L---HQAKRFTV-----RVGDRNTEQEEGNEMAHEVEMTVKHSRFVK----ETYDF
N2PKA           KNDN-----YEV-----WLGRHNLFENENTAQFFGVTADFPHPGFNLSADGKDYSH
GGBSM.DATA2     httttttthbbb.RtbBBtt.tthhttTttttchhh.hhhbbtTttt.ht..hhhhh
CF.DATA2        b..ttTTbbBBb..BBBBBB.hhhhhhhhhhhhhhhhhhhhhhhttttthht,.t.tt
GOR.DATA2       .ttT...hbbbbh.BBBbbhHHH..HHHHHHHhhhhhhhhbbbbtTTTT.TttThHH
GOR2.DATA2      .tTTttt.tbbbhHBBBb.ttttttt.hHHHhhhhhhhhbbbbtTTTT..ttTt.h
GOR3.DATA2      bbttttttbbBbbHHBBBBbb.ttt.hhhhhhhhh.hhhbbbbbtTt..b..ttttb


LCR number:   2 from alignment number:---->  53
Total number of gaps in LCR :          135
Total number of identities in LCR :            0
Names of seqs &
Struct. Pred. Methods.
                1                                                    56
N1NTP           ----------YKSGIQVRLGEDNINVVEGNEQFISASKSIVHPSY----DSNTLNN
EZEC58          ----------KGREITVILGAHDVRKAESTQQKIKVEKQIIHESYNSAPNL----H
HPHB            LFLNHSENA-TAKDIAPTLTLYV-----GKKQLVEIEKVVLHPNYS--------QV
KFBO            I---KPGVKITV-----VAGEHNTEKPEPTEQKRNVIRAIPYHSYNASIN--KYSH
PLHU            LEKSPRPSSYKV-----ILGAHQEVNLEPHVQEIEVSRLFLEPT--------RK--
TBBO            LLYPPWDKNFTVDDLLVRIGKHSRTRYERKVEKISMLDKIYIHPRYNWKE---NLDR
N1CHG           GVTTS-----DV-----VVAGEK-------IQKLKIAKVFKNSKYNSLTI--NN--
N1SGT           VSGSGNNTSITATGGVVDLQS-------GAAVKVRSTKVLQAPGY------NGTGK
N3EST           VD---RELTFRV-----VVGEHNLNQNNGTEQYVGVQKIVVHPYWNTDDV--AAGY
EXBO            L---HQAKRFTV-----RVGDRNTEQEEGNEMAHEVEMTVKHSRFVK--E--TYDF
N2PKA           ----------KNDNYEVWLGRHNLFENENTAQFFGVTADFPHPGFNLSADGKDYSH
GGBSM.DATA2     htttttttthbbb.h.bBBtt.tthhttTttttthhh.hhhbbtTttt.hhhTThhhh
CF.DATA2        bbb.tTTbbBbbTTBBBBBB.hhhhhhhhhhhhhhhhhhhhttttthh..TTthht
GOR.DATA2       htT....hbbbbbt.bbbbbbHHH..HHHHHHHhhhhhhhhbbbbtTTTTTTT.ThHH
GOR2.DATA2      htTTttt.tBbbthbBBb.tttttttt.hHHHhhhhhhhhbbbbtTTTTttTTt.hh
GOR3.DATA2      bB.ttttbbBbbhhhBBBBbb.ttt.hhhhhhhhh.hhhbbbbbtTt..b.TT...b
```

```
LCR number:   2 from alignment number:---->  55
Total number of gaps in LCR :          179
Total number of identities in LCR :            0
Names of seqs &
Struct. Prod. Methods.
                1                                                        60
N1NTP           ------YK---S----GIQVRLGEDNINVVEGNEQFI-SASKSIVHPSY----DSNTLNN
EZEC58          ------KG---R----EITVILGAHDVRKAESTQQKI-KVEKQIIHESYN----SAPNLH
HPHB            LFLNHSENATAKDIAPTLTLYVGKKQLV----------EIEKVVLHPNYSQV--------
KFBO            I---KPGV---KIT------VVAGEHNTEKPEPT-EQKRNVIRAIPYHSYNAS--INKYSH
PLHU            -LEKSPRP---S----SYKVILGAHQEVNLEPHVQEI-EVSRLFLEP------TRK----
TBBO            LLYPPWDK---NFTVDDLLVRIGKHSRTRYERKVEKISMLDKIYIHPRYN-W--KENLDR
N1CHG           GVTTS-------------DVVVAGEK-------IQKL-KIAKVFKNSKYNSLTINN----
N1SGT           VSGSGNNT---SITATGGVVDLQSGAAVKVRST--------KVLQAPGYN------GTGK
N3EST           V----DRE---L----TFRVVVGEHNLNQNNGTEQYV-GVQKIVVHPYWNTDDVAAGY--
EXBO            L----IQAK---RFT  ..   VRVGDRNTEQEEGN-EMAHEVEMTVKHSRF  V  KETYDF
N2PKA           ------KN---D----NYEVWLGRHNLFENENTAQFF-GVTADFPHPGFNLSADGKDYSH
GGBSM.DATA2     htttttttt....hbTT.bbBBbthhthhttttTTtthhHh.hh.bbtTttt.h.tthhhHH
CF.DATA2        Bbb.tTtt...bBBB.tBBBBB3hhhhhhhhhhhhhhhhHhhhhhhhhttttbh.t.httt
GOR.DATA2       h.TT..tt...tbB.8.bbbbbBHHHHHHhHhHHHHHhHhhhhhhbbb.tTTTt.TTtThHH
GOR2.DATA2      h.TTtttt...tBBB8.b8Bbhttttttttt.hhHHhHHhhhhhhbbbbtTTTTtRtttT.hh
GOR3.DATA2      bB.ttttt...bBBB3bbBBBbbhttthhhhhhhhhhhHhhhhhbbbbbtTtBbB.tt....


LCR number:   2 from alignment number:---->  56
Total number of gaps in LCR :          135
Total number of identities in LCR :            0
Names of seqs &
Struct. Prod. Methods.
                1                                                      56
N1NTP           -----YK-----SGIQVRLGEDNINVVEGNEQFISASKSIVHPSY----DSNTLNN
EZEC58          -----KG-----REITVILGAHDVRKAESTQQKIKVEKQIIHESYN----SAPNLH
HPHB            LFLNHSE------NATAKDIAPTLTLYVGKKQLVEIEKVVLHPNYSQV--------
KFBO            IKPGVK---------ITVVAGEHNTEKPEPTEQKRNVIRAIPYHSYNASIN--KYSH
PLHU            LEKSPRP-----SSYKVILGAHQEVNLEPHVQEIEVSRLFLEPTRK----------
TBBO            LLYPPWDKNFTVDDLLVRIGKHSRTRYERKVEKISMLDKIYIHPRYNWKE--NLDR
N1CHG           ------G-----VTTSDVVVAGEKIQKLKIAKVFKNSKYNSLTINN----------
N1SGT           VSGSG--------NNTSITATGGVVDLQSGAAVKVRSTKVLQAPGYN------GTGK
N3EST           VD---RE-----LTFRVVVGEHNLNQNNGTEQYVGVQKIVVHPYWN--TDDVAAGY
EXBO            LHQAKR--------FTVRVGDRNTEQEEGNEMAHEVEMTVKHSRF--VKE--TYDF
N2PKA           ------KN-----DNYEVWLGRHNLFENENTAQFFGVTADFPHPGFNLSADGKDYSH
GGBSM.DATA2     htttt.t.....thbbBB.tttthhttttttthhhb..hbbttttthhhhT.hhHH
CF.DATA2        b..tTTT.....ttBBBHBt.hhhhhhhhhhhhhhhhhh...tttt.hHhTTt.tt
GOR.DATA2       hhtT.tt.....t.bbbbbb.....hHhHHHHhhhhhhhhbbbttttTtTTT.ThHH
GOR2.DATA2      httTttt.....ttbbbb.tttttt.thHHHhhhhhhhhhbbbttttTtTTTTT.hh
GOR3.DATA2      hbttttt.....thbbhhbhh.ttt.hhhhhhhhhh.hhhbbbbbbttt...tTT...b


LCR number:   2 from alignment number:---->  58
Total number of gaps in LCR :          135
Total number of identities in LCR :            0
Names of seqs &
Struct. Prod. Methods.
                1                                                      56
N1NTP           -----YKSGIQ-----VRLGEDNINVVEGNEQFISASKSIVHPSY----DSNTLNN
EZEC58          -----KGREIT-----VILGAHDVRKAESTQQKIKVEKQIIHESYN----SAPNLH
HPHB            LFLNHSENATAKDI------APTLTLYVGKKQLVEIEKVVLHPNYS--------QV
KFBO            I---KPGVKIT-----VVAGEHNTEKPEPTEQKRNVIRAIPYESYNASIN--KYSH
PLHU            LEKSPRPSSYK-----VILGAHQEVNLEPHVQEIEVSRLFLEPTRK----------
TBBO            LLYPPWDKNFTVDDLLVRIGKHSRTRYERKVEKISMLDKIYIHPRYNWKE--NLDR
N1CHG           G-----VTTSD-----VVVAGEK-------IQKLKIAKVFKNS--KY--NSLTINN
N1SGT           VSGSGNNTSITATGGVVDLQS-------GAAVKVRSTKVLQAPGYN------GTGK
N3EST           VD---RELTFR-----VVVGEHNLNQNNGTEQYVGVQKIVVHPYWNT--DDVAAGY
EXBO            L---HQAKRFT-----VRVGDRNTEQEEGNEMAHEVEMTVKHSRF--VKE--TYDF
N2PKA           -----KNDNYE-----VWLGRENLFENENTAQFFGVTADFPHPGFNLSADGKDYSH
GGBSM.DATA2     htTTttttthbhHHT..BB.tttt.htttttttthhh.hhhhbbtTttttHHht.hhhh
CF.DATA2        bbbTTtt.bBBB....BBBBhhhhhhhhhhhhhhhhhhhhhhhhttttTHHhttt.tt
GOR.DATA2       h.T..tttbbb.Hh..bbbHHHH..hHhHHHHhhhhhhhhbbbbttTTTTTT.Thhh
GOR2.DATA2      h.TTTttttbBBHB..BbhttttttttthhhhhhhhhhhhbbbtTtTTTTTTtt.hh
GOR3.DATA2      bDtTttt.bbBDHB..BBbb.ttt.hhhhhhhhh.hhhbbbbbttttttt.tt..bb


LCR number:   2 from alignment number:---->  61
Total number of gaps in LCR :          135
Total number of identities in LCR :            0
Names of seqs &
Struct. Prod. Methods.
                1                                                      56
N1NTP           YK-------------SGIQVRLGEDNINVVEGNEQFISASKSIVHPSY------DSNTLNN
EZEC58          KGR----------EITVILGAHDVRKAESTQQKIKVEKQIIHESYN----SAPNLH
HPHB            L------FLNHSENATAKDIAPTLTLYVGKKQLVEIEKVVLHPNYSQV--------
KFBO            IKPGVK--------ITVVAGEHNTEKPEPTEQKRNVIRAIPYHSYNAS--INKYSH
PLHU            LEKSPRP-----SSYKVILGAHQEVNLEPHVQEIEVSRLFLEP------TRK----
TBBO            LLYPPWDKNFTVDDLLVRIGKHSRTRYERKVEKISMLDKIYIHPRYNW--KENLDR
N1CHG           ----------------GVTTSDVVVAGEKIQKLKIAKVFKNSKYNSLTINN----
N1SGT           VSGSG-------KNTSITATGGVVDLQSGAAVKVRSTKVLQAPGYN------GTGK
N3EST           VD---RE-----LTFRVVVGEHNLNQNNGTEQYVGVQKIVVHPYWNTDDVAAGY--
EXBO            LHQAKR--------FTVRVGDRNTEQEEGNEMAHEVEMTVKHSRF--V--KETYDF
N2PKA           KN----------DNYEVWLGRHNLFENENTAQFFGVTADFPHPGFNLSADGKDYSH
GGBSM.DATA2     htTTT.T.....thbbBB.tttt..ttttttthhh.hhhbbtTttt.h.tthhhHH
CF.DATA2        bt.TTT.....TTB3BBBt..b...hhhhhhhhhhhhhhttthh.t.htttt
GOR.DATA2       .ttT.bT.....,bbbbbBBBBBbhhHHHHHhhhhhhhhbbbbTTTTt.TTtThHH
GOR2.DATA2      .ttTttT.....t.bbbbhttttttt.hhHHHhhhhhhhhhbbbtTTTTtBtttT.hh
GOR3.DATA2      .btTt.T.....t.bbbbbb.tttb..hhhhhhh.hhhbbbbbttTtbbB.tt....
```

```
LCR number:    2 from alignment number:---->  62
Total number of gaps in LCR :            135
Total number of identities in LCR :             0
Names of seqs &
Struct. Pred. Methods.
             1                                                      56
N1NTP        YK------------SGIQVRLGEDNINVVEGNEQFISASKSIVIIPSY    -DSNTLNN
EZEC58       KGR-----------EITVILGAHDVRKAESTQQKIKVEKQIIHESYN----SAPNLH
HPHB         LFLNHSE------NATAKDIAPTLTLYVGKKQLVEIEKVVLHPNYS--------QV
KFBO         IKPGVK--------ITVVAGEHNTEKPEPTEQKRNVIRAIPYHSYNASIN--KYSH
PLHU         LEKSPRP-----SSYKVILGAHQEVNLEPHVQEIEVSRLFLEPTRK----------
TBBO         LLYPPWDKNFTVDDLLVRIGKHSRTRYERKVEKISMLDKIYIHPRYNWKE--NLDR
N1CHG        -----------------GVTTSDVVVAGEKIQKLKIAKVFKNS--KY--NSLTINN
N1SGT        VSGSG-------NNTSITATGGVVDLQSGAAVKVRSTKVLQAPGYN------GTGK
N3EST        VD---RR-----LTFRVVVGEHNLNQNNGTRQYVGVQKIVVHPYWNT--DDVAAGY
EXBO         LHQAKR--------FTVRVGDRNTEQEEGNEMAHEVEMTVKHSRF--VKE--TYDF
N2PKA        KN-----------DNYEVWLGRHNLFENENTAQFFGVTADFPHPGFNLSADGKDYSH
GGBSM.DATA2  htttttT......hbbBB.tttt..tttttthhh.hhhbbtTttttHHht.hhhh
CF.DATA2     bt.tTT......TTBBBBBt..b....hhhhhhhhhhhhhhhttttTHHhttt.tt
GOR.DATA2    .ttT.b......t.bbbbbBBBBBbhHhHHHHhhhhhhhhbbbttTTTTTT.ThHh
GOR2.DATA2   .ttTtt......t.bbbbhttttttt.hhHHHhhhhhhhbbbtTtTTTTTTtt.hh
GOR3.DATA2   .b.tt.t......t.bbbbbb.tttb..hhhhhhh.hhhbbbbbttttttt.tt..bb
```

```
LCR number:    2 from alignment number:---->  65
Total number of gaps in LCR :            135
Total number of identities in LCR :             0
Names of seqs &
Struct. Pred. Methods.
             1                                                      56
N1NTP             YK ----SGIQVRLGEDKINVVEGNEQFISASKSIVIIPSY-----DSNTLNN
EZEC58       -----KG-----REITVILGAHDVRKAESTQQKIKVEKQIIHESYN----SAPNLH
HPHB         LFLNHSE------NATAKDIAPTLTLYVGKKQLVEIEKVVLHPNYSQV--------
KFBO         IKPGVK--------ITVVAGEHNTEKPEPTEQKRNVIRAIPYHSYNASIN--KYSH
PLHU         LEKSPRP-----SSYKVILGAHQEVNLEPHVQEIEVSRLFLEPTRK----------
TBBO         LLYPPWDKNFTVDDLLVRIGKHSRTRYERKVEKISMLDKIYIHPRYNWKE--NLDR
N1CHG        ------G-----VTTSDVVVAGEKIQKLKIAKVFKNSKYNSLTINN----------
N1SGT        VSGSGNNTSITATGGVVDLQSGAAVKVRSTKVLQAP-------------GYNGTGK
N3EST        VD---RE-----LTFRVVVGEHNLNQNNGTRQYVGVQKIVVHPYWN--TDDVAAGY
EXBO         LHQAKR--------FTVRVGDRNTEQEEGNEMAHEVEMTVKHSRF--VKE--TYDF
N2PKA        ------KN-----DNYEVWLGRHNLFENENTAQFFGVTADFPHPGFNLSADGKDYSH
GGBSM.DATA2  htttttt.....thbbBB.ttthhttttttthhhbhhhhbbttttthhhhTthhHH
CF.DATA2     b..tTtT.....ttBBBBBb..hhhhhhhhhhhhhhhhhhhhhtttt.hH.TTt.tt
GOR.DATA2    hhtT.tt.....t.bbbbbhHHHHhhhhhHhhhhhhhhhbbbbtttTtTTT.ThHH
GOR2.DATA2   httTttt.....t.bbbb.ttttttttthhhhhhhhhhhhbbb.tttTtTTTTT.hh
GOR3.DATA2   bbtttttt......bbbbbbb.ttt.hhhhhhhhh.hhh.bbbbttt...tTT...b
```

These are the different alignments found for loosely conserved region   3

```
LCR number:    3 from alignment number:---->   1
Total number of gaps in LCR :            36
Total number of identities in LCR :             0
Names of seqs &
Struct. Pred. Methods.
             1                            29
N1NTP        ASLDSRVASISLPT---SCASAG--TQCL
EZEC58       VELTPAVNVVPLPS---PSDFIHPGAMCW
HPHB         VSVNERVMPICLPSKDYAEV----GRVGY
KFBO         LELNSYVTPICIADRDYTN-IFSKFGYGY
PLHU         AVITDKVIPACLPS---PNYVVADRTECF
TBBO         IELSDYIHPVCLPDKQTAAKLLHAGFKGR
N1CHG        ASFSQTVSAVCLPS---ASDDFAAGTTCV
N1SGT        INQ----------PTLKIATTTAYNQGTFT
N3EST        VTLNSYVQLGVLPR---AGTILANNSPCY
EXBO         IRFRRNVAPACLPEKDWAEATLMTQKTGI
N2PKA        AKITDAVKVLELPT---QEPELG--STCE
GGBSM.DATA2  tttt.bbbbbtTTTT.htttttttTTtbb
CF.DATA2     hhhhtbbbbbbbtt...hhhhb.ttt.bB
GOR.DATA2    hbhtttbBBBBBtthh.tbb..tttbbbb
GOR2.DATA2   hh.tttbBBbB.tT...t.bbbttttbbb
GOR3.DATA2   bbbbbbBBBBBbbt.h.bbbbbbttbbbB
```

```
LCR number:    3 from alignment number:---->   8
Total number of gaps in LCR :            36
Total number of identities in LCR :             0
Names of seqs &
Struct. Pred. Methods.
             1                            29
N1NTP        ASLDSRVASISLPTSC---A--SAGTQCL
EZEC58       VELTPAVNVVPLPSPS---DFIHPGAMCW
HPHB         VSVNERVMPICLPSKDYAE----VGRVGY
KFBO         LELNSYVTPICIADRDYTN-IFSKFGYGY
PLHU         AVITDKVIPACLPS---PYVVADRTECF
TBBO         IELSDYIHPVCLPDKQTAAKLLHAGFKGR
N1CHG        ASFSQTVSAVCLPS---ASDDFAAGTTCV
N1SGT        INQ----------PTLKIATTTAYNQGTFT
N3EST        VTLNSYVQLGVLPR---AGTILANNSPCY
EXBO         IRFRRNVAPACLPEKDWAEATLMTQKTGI
N2PKA        AKITDAVKVLELPTQE---P--ELGSTCE
GGBSM.DATA2  tttt.bbbbbtTTTTthttt..ttTTtbb
CF.DATA2     hhhhtbbbbbbbtttt.hhhhb.ttt.bB
GOR.DATA2    hbhtttbBBBBBtttt.bb.bbttttbbbb
GOR2.DATA2   hh.tttbBBbB.tTtt..b.bbttttbbh
GOR3.DATA2   bbbbbbBBBBBbbt...bbbbbbttbbbB
```

```
LCR number:   3 from alignment number:---->  19
Total number of gaps in LCR :          69
Total number of identities in LCR :             0
Names of seqs &
Struct. Pred. Methods.
                1                              32
N1NTP           ASLDSRVASISLP-----TSCASAG--T-QCL
EZEC58          VELTPAVNVVPLPSPSDFIHPGAM------CW
HPHB            VSVNERVMPICLPSK-DY------AEVGRVGY
KFBO            LELNSYVTPICIADR-DYTN--IFSKFGY-GY
PLHU            AVITDKVIPACLP-----SPNYVVADRT-ECF
TBBO            IELSDYIHPVCLPDK-Q-TAAKLL-HAGFKGR
N1CHG           ASFSQTVSAVCLP------SASDDFAAGT-TCV
N1SGT           ------INQPTL----KIATTTAYNQGTFT--
N3EST           VTLNSYVQLGVLP-----RAGTILANNS-PCY
EXBO            IRFRRNVAPACLPEK-DWAEATLMTQKT--GI
N2PKA           AKITDAVKVLELP-----TQEPELG--S-TCE
GGBSM.DATA2     tttt.bbbbbtTTTT.t.ttttt.ttTTTtbb
CF.DATA2        hhhht.bbbbbbbtTt.tB.hhh....t...bB
GOR.DATA2       hhhttttbBBBBBtT.....bt..tttbb.bb.
GOR2.DATA2      hh.tttbbbbBbtTt.tbttt..htt..bbb
GOR3.DATA2      bbbbbbbBBBBb....hbbbbbbbb.bb.bbB


LCR number:   3 from alignment number:---->  21
Total number of gaps in LCR :          36
Total number of identities in LCR :             0
Names of seqs &
Struct. Pred. Methods.
                1                          29
N1NTP           ASLDSRVASISLP---TSCASAG--TQCL
EZEC58          VELTPAVNVVPLP---SPSDFIHPGAMCW
HPHB            VSVNERVMPICLPSKDYAEV----GRVGY
KFBO            LELNSYVTPICIADRDYTNIFSKFGYGY-
PLHU            AVITDKVIPACLP---SPNYVVADRTECF
TBBO            IELSDYIHPVCLPDKQTAAKLLHAGFKGR
N1CHG           ASFSQTVSAVCLP---SASDDFAAGTTCV
N1SGT           IN---------QPTLKIATTTAYNQGTFT
N3EST           VTLNSYVQLGVLP---RAGTILANNSPCY
EXBO            IRFRRNVAPACLPEKDWAEATLMTQKTGI
N2PKA           AKITDAVKVLELP---TQEPELG--STCE
GGBSM.DATA2     tttt.bbbbbbTTTT.tttttttTTTtbb
CF.DATA2        hhhhtbbbbbbbtt..tahhh..ttt.bB
GOR.DATA2       hhhttttbBBBBBtThhhtbb..tttbbbb
GOR2.DATA2      hh.tttbBBbBttT..tt.bbbttttbbB
GOR3.DATA2      bbbbbbbBBBBBbb..htbbbbbbttbbbB


LCR number:   3 from alignment number:---->  26
Total number of gaps in LCR :          58
Total number of identities in LCR :             0
Names of seqs &
Struct. Pred. Methods.
                1                            31
N1NTP           ASLDSRVASISLP----TSCA--SAGT-QCL
EZEC58          VELTPAVNVVPLP----SPSDFIHPGA-MCW
HPHB            VSVNERVMPICLPSKDYAE------VGRVGY
KFBO            LELNSYVTPICIADRDYTN--IFSKFGY-GY
PLHU            AVITDKVIPACLP----SPNYVVADRT-ECF
TBBO            IELSDYIHPVCLPDKQ TAAKLL-HAGFKGR
N1CHG           ASFSQTVSAVCLP----SASDDFAAGT-TCV
N1SGT           ------INQPTL---KIATTTAYNQGTFT--
N3EST           VTLNSYVQLGVLP----RAGTILANNS-PCY
EXBO            IRFRRNVAPACLPEKDWAEATLMTQKT--GI
N2PKA           AKITDAVKVLELP----TQEP--ELGS-TCE
GGBSM.DATA2     tttt.bbbbbtTTTT.htttt.bttTTTtbb
CF.DATA2        hhhhtbbbbbbbtT..B.hhhbbbtt...bB
GOR.DATA2       hhhttttbBBBBBtTThhbbbt.bbtttb.bb.
GOR2.DATA2      hh.tttbbbbBbtT..bttt.bbttt..hbb
GOR3.DATA2      bbbbbbbBBBBb.hhhb.bbbbbbttb.bbB


LCR number:   3 from alignment number:---->  27
Total number of gaps in LCR :          36
Total number of identities in LCR :             0
Names of seqs &
Struct. Pred. Methods.
                1                          29
N1NTP           ASLDSRVASISLP---TSCA--SAGTQCL
EZEC58          VELTPAVNVVPLP---SPSDFIHPGAMCW
HPBB            VSVNERVMPICLPSKDYAE----VGRVGY
KFBO            LELNSYVTPICIADRDYTNIFSKFGYGY
PLBU            AVITDKVIPACLP---SPNYVVADRTECF
TBBO            IELSDYIHPVCLPDKQTAAKLLHAGFKGR
N1CHG           ASFSQTVSAVCLP---SASDDFAAGTTCV
N1SGT           INQPTLKIATTTAYNQG---------TFT
N3EST           VTLNSYVQLGVLP---RAGTTLANNSPCY
EXBO            IRFRRNVAPACLPEKDWAEATLMTQKTGI
N2PKA           AKITDAVKVLELP---TQEP--ELGSTCE
GGBSM.DATA2     tttttbbbbbbtTTTTtttttbbbTTTtbb
CF.DATA2        hhhhtbbbbbbbtt.tthhhhhhttt.bB
GOR.DATA2       hhhttttbBBBBBtT....t..bbtttbbbb
GOR2.DATA2      hh.tttbBBbBbtTttttt.bbttttbbB
GOR3.DATA2      bbbbbbBBBBBBbb..ht.bbbbbtt.bbB
```

```
LCR number:   3 from alignment number:---->  37
Total number of gaps in LCR :             36
Total number of identities in LCR :        0
Names of seqs &
Struct. Pred. Methods.
              1                           29
N1NTP         ASLDSRVASISLPTSCAS---A--GTQCL
EZEC58        VRLTPAVNVVPLPSPSDF---IHPGAMCW
HPHB          VSVNERVMPICLPSKDYAEV----GRVGY
KPBO          LELNSYVTPICIADRDYTNI-FSKFGYGY
PLHU          AVITDKVIPACLPSPNYV---VADRTECF
TBBO          IELSDYIHPVCLPDKQTAAKLLHAGFKGR
N1CHG         ASFSQTVSAVCLPSASDD---FAAGTTCV
N1SGT         INQ---------PTLKIATTTAYNQGTFT
N3EST         VTLNSYVQLGVLPRAGTI---LANNSPCY
EXBO          IRFRRNVAPACLPEKDWAEATLMTQKTGI
N2PKA         AKITDAVKVLELPTQEPE---L--GSTCE
GGBSM.DATA2   tttt.bbbbbtTTTTtt..HttttTtbb
CF.DATA2      hhhhthbbbbhbtt..hhhhHb.ttt.bB
GOR.DATA2     hhhtttbBBBBBtttt.tbbH.ttbbbbb
GOR2.DATA2    hh.tttbBBbB.tTtt.bbbHb.tttbbb
GOR3.DATA2    bbbbbbBBBBBBbtt.bbbbHbbttbbbB


LCR number:   3 from alignment number:---->  39
Total number of gaps in LCR :             36
Total number of identities in LCR :        0
Names of seqs &
Struct. Pred. Methods.
              1                           29
N1NTP         ASLDSRVASISLPTSCAS---A--GTQCL
EZEC58        VRLTPAVNVVPLPSPSD---FIHPGAMCW
HPHB          VSVNERVMPICLPSKDYAEV----GRVGY
KPBO          LELNSYVTPICIADRDYTNI-FSKFGYGY
PLHU          AVITDKVIPACLPSPNYV---VADRTECF
TBBO          IELSDYIHPVCLPDKQTAAKLLHAGFKGR
N1CHG         ASFSQTVSAVCLPSASDD---FAAGTTCV
N1SGT         INQ---------PTLKIATTTAYNQGTFT
N3EST         VTLNSYVQLGVLPRAGTI---LANNSPCY
EXBO          IRFRRNVAPACLPEKDWAEATLMTQKTGI
N2PKA         AKITDAVKVLELPTQEPE---L--GSTCE
GGBSM.DATA2   tttt.bbbbbtTTTTtt..tttttTtbb
CF.DATA2      hhhhthbbbbhbtt..hhhhHb.ttt.bB
GOR.DATA2     hhhtttbBBBBBtttt..bbh.ttbbbbb
GOR2.DATA2    hh.tttbBBbB.tTtt.bbb.b.tttbbb
GOR3.DATA2    bbbbbbBBBBBBbtt.bbbb.bbttbbbB


LCR number:   3 from alignment number:---->  44
Total number of gaps in LCR :             36
Total number of identities in LCR :        0
Names of seqs &
Struct. Pred. Methods.
              1                           29
N1NTP         ASLDSRVASISLP---TSCA--SAGTQCL
EZEC58        VRLTPAVNVVPLP---SPSDFIHPGAMCW
HPHB          VSVNERVMPICLPSKDYA----EVGRVGY
KPBO          LELNSYVTPICIADRDYTNI-FSKFGYGY
PLHU          AVITDKVIPACLP---SPNYVVADRTECF
TBBO          IELSDYIHPVCLPDKQTAAKLLHAGFKGR
N1CHG         ASFSQTVSAVCLP---SASDDFAAGTTCV
N1SGT         INQ---------PTLKIATTTAYNQGTFT
N3EST         VTLNSYVQLGVLP---RAGTILANNSPCY
EXBO          IRFRRNVAPACLPEKDWAEATLMTQKTGI
N2PKA         AKITDAVKVLELP---TQEP--ELGSTCE
GGBSM.DATA2   tttt.bbbbbtTTTT.ttttt.ttTTtbb
CF.DATA2      hhhhtbbbbbbbtt..thhhhbhttt.bB
GOR.DATA2     hhhttttbBBBBBtThhbt.bbbttbbbb
GOR2.DATA2    hh.tttbBBbB.tT..tttbbbttttbbb
GOR3.DATA2    bbbbbbBBBBBBbb..htbbbbbbttbbbB


LCR number:   3 from alignment number:---->  55
Total number of gaps in LCR :             36
Total number of identities in LCR :        0
Names of seqs &
Struct. Pred. Methods.
              1                           29
N1NTP         ASLDSRVASISLPT---SCASAG--TQCL
EZEC58        VELTPAVNVVPLPS---PSDFIHPGAMCW
EPHB          VSVNERVMPICLPSKDYAEV----GRVGY
KPBO          LELNSYVTPICIADRDYTN-IFSKFGYGY
PLHU          AVITDKVIPACLPS---PNYVVADRTECF
TBBO          IELSDYIHPVCLPDKQTAAKLLHAGFKGR
N1CHG         ASFSQTVSAVCLPS---ASDDFAAGTTCV
N1SGT         IN---------QFTLKIATTTAYNQGTFT
N3EST         VTLNSYVQLGVLPR---AGTILANNSPCY
EXBO          IRFRRNVAPACLPEKDWAEATLMTQKTGI
N2PKA         AKITDAVKVLELPT---QEPELG--STCE
GGBSM.DATA2   tttt.bbbbbtTTTT.htttttttTTtbb
CF.DATA2      hhhhtbbbbbbbtt...hhhhb.ttt.bB
GOR.DATA2     hhhttttbBBBBBtthh.tbb..tttbbbb
GOR2.DATA2    hh.tttbBBbBttT...t.bbbttttbbb
GOR3.DATA2    bbbbbbBBBBBbbt.b.bbbbbbttbbbB
```

```
LCR number:   3 from alignment number:---->  16
Total number of gaps in LCR :            36
Total number of identities in LCR :               3
Names of seqs &
Struct. Pred. Methods.
                1                          29
N1NTP           ASLDSRVASISLPT---SCASAG--TQCL
EZEC58          VELTPAVNVVPLPS---PSDFIHPSAMCW
HPHB            VSVNERVMPICLPSKDYAE----VGRVGY
KFBO            LELNSYVTPICIADRDYTN-IFSKFGYGY
PLHU            AVTTDKVIPACLPS---PNYVVADRTECF
TBBO            IELSDYIHPVCLPDKQTAAKLLHASFKGR
N1CHG           ASFSQTVSAVCLPS---ASDDPAASTTCV
N1SGT           IN---------QPTLKIATTTAYNQGTFT
N3EST           VCLNSYVQLCVLPR---ACTILANNSDCY
EXBO            IRFKRNVAPACLPEKDWAEATLMTQKTGI
N2PKA           AKTTDAVKVLSLPT---QEPEIG--STGR
GGBSM.DATA2     tttt.bbbbbtTTTT.httttttttTtbb
CF.DATA2        hhhhtbbbbbbbtt...hhhhb..tt.bB
GOR.DATA2       hhhtttbBBBBBtthh.tb...tttbbbb
GOR2.DATA2      hh.tttbBBbBttT...t..bbttttbbb
GOR3.DATA2      bbbbbbBBBBBbbt.h.bbbbbbttbbbB


These are the different alignments found for loosely conserved region    4

LCR number:   4 from alignment number:---->   1
Total number of gaps in LCR :            68
Total number of identities in LCR :               0
Names of seqs &
Struct. Pred. Methods.
                1                  20
N1NTP           NTK---SSGTSY--PDVLKC
EZEC58          KT------GVRDPTSYTLRE
HPHB            RN--------ANFKFTDHLKY
KFBO            KVFNR---------GRSASILQY
PLHU            --E---TQGT-FG-AGLLKE
TBBO            NRRETWTTSVAEVQPSVLQV
N1CHG           --R--------------LQQ
N1SGT           ANRE------GGSQQRYLLK
N3EST           LTR---TNGQ-LA-QTL-QQ
EXBO            RTHEK------GRLSSTLKM
N2PKA           SIE---PGPDDPEFPDEIQC
GGBSM.DATA2     TTTTT.TTTttTtTtthhhh
CF.DATA2        Tt.tT.tTTTtttt.tbbbb
GOR.DATA2       ...TT.....TT.BBbbbb
GOR2.DATA2      TTTTT.TTTTTttt.bbbb
GOR3.DATA2      tTtTT.TTTt.Ttbbbbbb


LCR number:   4 from alignment number:---->   8
Total number of gaps in LCR :            68
Total number of identities in LCR :               0
Names of seqs &
Struct. Pred. Methods.
                1                  20
N1NTP           NTK-SSGTSY----PDVLKC
EZEC58          RTGVRDPTSYT--LRE----
HPHB            R--------NANFKFTDHLKY
KFBO            KVFNR------GRSASILQY
PLHU            --ETQ-----CTFGAGLLKE
TBBO            NRRETWTTSVAEVQPSVLQV
N1CHG           --R--------------LQQ
N1SGT           ANRE------GGSQQRYLLK
N3EST           LTRTN-----GQLAQTL-QQ
EXBO            RTHEK-------GRLSSTLKM
N2PKA           SIE-PGPDDPE--FPDEIQC
GGBSM.DATA2     TTTTTTTTT.tTTtt.hhhh
CF.DATA2        Tt.TTTTTTTttthttbbbb
GOR.DATA2       ...TT.....TT..bbbbb
GOR2.DATA2      TTTTTTTTTTTTthbbbbb
GOR3.DATA2      tTttTTTTTbtTtb.bbbbb


LCR number:   4 from alignment number:---->  19
Total number of gaps in LCR :            68
Total number of identities in LCR :               0
Names of seqs &
Struct. Pred. Methods.
                1                  20
N1NTP           NTK-SSGTS-Y---PDVLKC
EZEC58          KT------GVRDPTSYTLRE
HPHB            --RNAN----FKFT-DHLKY
KFBO            K---VFNRGRS---ASILQY
PLHU            --E-TQGT--FG--AGLLKE
TBBO            NRRETWTTSVAEVQPSVLQV
N1CHG           --R--------------LQQ
N1SGT           ANREGGSQQR------YLLK-
N3EST           LTR-TNGQ--L---AQTLQQ
EXBO            R---THEKGRL---SSTLKM
N2PKA           SIE-PGPDD-FEF-PDEIQC
GGBSM.DATA2     TTTTttTTTttTTTttthhhh
CF.DATA2        TttTTTTTt.tHHHTtbbbb
GOR.DATA2       ....TTTTTT.....Bbbbb
GOR2.DATA2      TTTTTTTTTttT.Tt.bb.b
GOR3.DATA2      tTTTTTTTtbtTb.bbbbbb
```

```
LCR number:   4 from alignment number:---->  21
Total number of gaps in LCR :           68
Total number of identities in LCR :              0
Names of seqs &
Struct, Pred, Methods.
              1                   20
N1NTP         NTKS--SGTSY---PDVLKC
EZEC58        KT------GVRDPTSYTLRE
HPHB          RN------ANFKFT-DHLKY
KFBO          K---VFNRGRS---ASILQY
PLHU          --ET--QGT-FG--AGLLKE
TBBO          NRRETWTTSVAEVQPSVLQV
N1CHG         --R--------------LQQ
N1SGT         ANREGGSQQR------YLLK-
N3EST         LTRT--NGQ-L---AQTLQQ
EXBO          R---THEKGRL---SSTLKM
N2PKA         SIRP--GPDDFEF-PDEIQC
GGBSM.DATA2   TTTTTTTTtttGTTtthhhh
CF.DATA2      Tt.T..TttttHHHTtbbbb
GOR.DATA2     ...T..TTTT.....Bbbbb
GOR2.DATA2    TTTTTTTTTtttt.bb.b
GOR3.DATA2    tTTTTTTTtbtTb.bbbbbb


LCR number:   4 from alignment number:---->  26
Total number of gaps in LCR :           68
Total number of identities in LCR :              0
Names of seqs &
Struct, Pred, Methods.
              1                   20
N1NTP         NT-K-SSGTSY---PDVLKC
EZEC58        KT-GVRDPTSYTL-RE-----
HPHB          --RN----ANFKF-TDHLKY
KFBO          K---VFNRGRS---ASILQY
PLHU          -------ETQGTFGAGLLKE
TBBO          NRRETWTTSVAEVQPSVLQV
N1CHG         -------R---------LQQ
N1SGT         ANRE----GGSQ-QRYLLK-
N3EST         -----LTRTNGQL-AQTLQQ
EXBO          R---THEKGRL---SSTLKM
N2PKA         SI-R-PGPDDFEF--PDEIQC
GGBSM.DATA2   TTTTTTTTttttttT.thhhh
CF.DATA2      TTTT.ttttTThh.ttbbbb
GOR.DATA2     ......TTTT.....bbbbb
GOR2.DATA2    TTTT.TTTTTtTtTbbbbbb
GOR3.DATA2    TTTT.tTttttttB.bbbbb


LCR number:   4 from alignment number:---->  27
Total number of gaps in LCR :           68
Total number of identities in LCR :              0
Names of seqs &
Struct, Pred, Methods.
              1                   20
N1NTP         N---TKSSGTSY--PDVLKC
EZEC58        K---T---GVRDPTSYTLRE
HPHB          RN-------ANFKFTTHLKY
KFBO          K---VFNRGRS---ASILQY
PLHU          -------ETQGIPGAGLLKE
TBBO          NRRETWTTSVAEVQPSVLQV
N1CHG         -------R---------LQQ
N1SGT         ANREG     GSQQRYLLK-
N3EST         -----LTRTNGQL-AQTLQQ
EXBO          R---THEKGRL---SSTLKM
N2PKA         S---IEPGPDDFEFPDEIQC
GGBSM.DATA2   TT..TTTTTttttTTtthhhh
CF.DATA2      TT..T.ttTTTt.httbbbb
GOR.DATA2     ......TTTT....BBbbbb
GOR2.DATA2    TT..TTTTTTTTTtt.bb.b
GOR3.DATA2    TT..TtTTTttT.tbbbbbb


LCR number:   4 from alignment number:---->  37
Total number of gaps in LCR :           68
Total number of identities in LCR :              0
Names of seqs &
Struct, Pred, Methods.
              1                   20
N1NTP         NTKSS--GTSY--PDVLKC
EZEC58        KT------GVRDPTSYTLRE
HPHB          RNANF-------KFTDHLKY
KFBO          KVFNRGRSA---------SILQY
PLHU          ETQ---GTFG----AGLLKE
TBBO          NRRETWTTSVAEVQPSVLQV
N1CHG         R----------------LQQ
N1SGT         ANREGGSQQRY-------LLK
N3EST         LTR---TNGQL---AQTLQQ
EXBO          RTHEKGRLS-------STLKM
N2PKA         SIEPGPDDFEF---PDEIQC
GGBSM.DATA2   T?tTTTTTttt.TTtthhhh
CF.DATA2      ttttTTTttb..HHttbbbb
GOR.DATA2     ...TTTT.......Bbbbb
GOR2.DATA2    TTTTTTTTttt.TTttbbbb
GOR3.DATA2    tTTTTTTttbb.B.bbbbbb
```

```
LCR number:   4 from alignment number:---->  44
Total number of gaps in LCR :            68
Total number of identities in LCR :       0
Names of seqs &
Struct. Pred. Methods.
              1                   20
N1NTP         NTKSS--GTSY---PDVLKC
EZEC58        KT-GVRDPTSY---T--LRE
HPHB          RNANFKFT-------DHLKY
KFBO          KVFNRGRSA-------SILQY
PLHU          ETQGTFG-A------GLLKE
TBBO          NRRETWTTSVAEVQPSVLQV
N1CHG         R-----------------LQQ
N1SGT         ANREGGSQQRY-------LLK
N3EST         LTRTNGQLA------QTLQQ
EXBO          RTHEKGRLS------STLKM
N2PKA         SIEPGPDDFRF---PDRIQC
GGBSM.DATA2   TTtTTTTttt....Ttbhhh
CF.DATA2      tttTttttttt....Ttbbbb
GOR.DATA2     ...TTTT.B......Bbbbb
GOR2.DATA2    TTTTTTTttt....t.bbbb
GOR3.DATA2    tTTtTttttbB...Bbbbbb


LCR number:   4 from alignment number:---->  48
Total number of gaps in LCR :            68
Total number of identities in LCR :       0
Names of seqs &
Struct. Pred. Methods.
              1                   20
N1NTP         NTK---SSGTSY--PDVLKC
EZEC58        KT------GVRDPTSYTLRE
HPHB          RN-------ANFKFTDHLKY
KFBO          KVFNR-------GRSASILQY
PLHU          ETQ - - - - GTFG AGLLKE
TBBO          NRRETWTTSVAEVQPSVLQV
N1CHG         R-----------------LQQ
N1SGT         ANRE------GCSQQRYLLK
N3EST         LQR------TNGQLAQTLQQ
EXBO          RTHEK------GRLSSTLKM
N2PKA         SIE---PGPDDFEFPDEIQC
GGBSM.DATA2   TTTTT.TTTtttTTTtthhhh
CF.DATA2      tttTT.TTTTTtt.ttbbbb
GOR.DATA2     ...TT......TT.BBbbbb
GOR2.DATA2    TTTTT.TTTTTTTtt.bbbb
GOR3.DATA2    tTTTT.TTTttTttbbbbbb


LCR number:   4 from alignment number:---->  53
Total number of gaps in LCR :            68
Total number of identities in LCR :       0
Names of seqs &
Struct. Pred. Methods.
              1                   20
N1NTP         NTK-SSGTSY----PDVLKC
EZEC58        KTGVRDPTSYT--LRE----
HPHB          RN-----ANF--KFTDHLKY
KFBO          KVFNR------GRSASILQY
PLHU          --ETQ-----GTFGAGLLKE
TBBO          NRRETWTTSVAEVQPSVLQV
N1CHG         --R--------------LQQ
N1SGT         ANRE------GGSQQRYLLK
N3EST         LTRTN-----GQLAQTL-QQ
EXBO          RTHEK------GRLSSTLKM
N2PKA         SIE-PGPDDFE--FPDEIQC
GGBSM.DATA2   TTTTTTTtt.tTTtt.hhhh
CF.DATA2      TL.TTTTTTHttthttbbbb
GOR.DATA2     ...TT......TT..bbbbb
GOR3.DATA2    TTTTTTTTTTTTTTbbbbbb
GOR3.DATA2    tTtTTTTTTbTTb.bbbbb


LCR number:   4 from alignment number:---->  54
Total number of gaps in LCR :            68
Total number of identities in LCR :       0
Names of seqs &
Struct. Pred. Methods.
              1                   20
N1NTP         NTKSS---GTSY--PDVLKC
EZEC58        KTGVR---DPT---SYTLRE
HPHB          R-------NANFKFTDHLKY
KFBO          KVFNR ------GRSASILQY
PLHU          E------TQGTFG-AGLLKE
TBBO          NRRETWTTSVAEVQPSVLQV
N1CHG         R----------------LQQ
N1SGT         ANRE------GGSQQRYLLK
N3EST         L------TRTNGQLAQTLQQ
EXBO          RTHEK------GRLSSTLKM
N2PKA         SIEPG---PDDFEFPDEIQC
GGBSM.DATA2   TTTTT..TTttTTTthhhhh
CF.DATA2      tTTTT..tTTt..ttbbbb
GOR.DATA2     ...TT......TT.BBbbbb
GOR2.DATA2    TTTTT..TTTTTTtt.bbbb
GOR3.DATA2    tTTttT..TTttTttbbbbbb
```

```
LCR number:   4 from alignment number:---->  55
Total number of gaps in LCR :            68
Total number of identities in LCR :               0
Names of seqs &
Struct. Pred. Methods.
              1                   20
N1NTP         NTKSS---GT--SYPDVLKC
EZEC58        KTGVR---DPTSY---TLRE
HPHB          RNAN--------FKFTDHLKY
KFBO          KVFNR------GRSASILQY
PLHU          --ETQ---GTFG--AGLLKE
TBBO          NRRETWTTSVAEVQPSVLQV
N1CHG         --R--------------LQQ
N1SGT         ANREG------GSQQRYLLK
N3EST         LTRTN---GQL---AQTLQQ
EXBO          RTHEK------GRLSSTLKM
N2PKA         SIEPG---PDDFEFPDEIQC
GGBSM.DATA2   TTtTT...TTtTTtTthhhh
CF.DATA2      TttTT...Ttttt.tbbbb
COR.DATA2     ...TT.....TT.BBbbbb
GOR2.DATA2    TTTTT...TTTTT..bbbbb
GOR3.DATA2    tTttT...TtTT.bbbbbbb


LCR number:   4 from alignment number:---->  56
Total number of gaps in LCR :            68
Total number of identities in LCR :               0
Names of seqs &
Struct. Pred. Methods.
              1                   20
N1NTP         NTKSSG------SYPDVLKC
EZEC58        KTGVRDPT------SYTLRE
HPHB          RNAN--------FKFTDHLKY
KFBO          KVFNR . . . GRSASILQY
PLHU          ETQ---GTF---G-AGLLKE
TBBO          NRRETWTTSVAEVQPSVLQV
N1CHG         R----------------LQQ
N1SGT         ANREG------GSQQRYLLK
N3EST         LTR---TNG---QLAQTLQQ
EXBO          RTHEK-------GRLSSTLKM
N2PKA         SIEPGPDDF---EFFDEIQC
GGBSM.DATA2   TTtTTTTTT..TTTtthhhh
CF.DATA2      ttbTTTTTT..tt.tbbbb
COR.DATA2     ...TT.....TT.BBbbbb
GOR2.DATA2    TTTTTTTTT..TTtt.bbbb
GOR3.DATA2    tTttTTTTT..Tt.bbbbbb


These are the different alignments found for loosely conserved region    5

LCR number:   5 from alignment number:---->   1
Total number of gaps in LCR :            17
Total number of identities in LCR :               0
Names of seqs &
Struct. Prod. Methods.
              1
N1NTP         KSA--YPGQITSNM
EZEC58        VDYGYYEYKFQ---
HPHB          IRH-YEGSTVPELT
KFBO          LRSTKF--SIYSHM
PLHU          NRYEFLNGRVQSTE
TBBO          KASTRI--RITDNM
N1CHG         KKY--WGTKIKDAM
N1SGT         RS-AYGNELVANEE
N3EST         SSSSYWGSTVKNSM
EXBO          KLSSSF--SFTITPNM
N2PKA         ADA--HPDKVTESM
GGBSM.DATA2   hht.htTtbbtttt
CF.DATA2      tttt.bTt.b..hb
COR.DATA2     tTTTTTTTtbtt.t
GOR2.DATA2    ttTTtTtTTttttbb
GOR3.DATA2    bb.tttt.bbbbbb


LCR number:   5 from alignment number:---->   8
Total number of gaps in LCR :            17
Total number of identities in LCR :               0
Names of seqs &
Struct. Pred. Methods.
              1
N1NTP         KSA--YPGQITSNM
EZEC58        VD---YGYYEYKFQ
HPHB          IRH-YEGSTVPEHT
KFBO          LRST--KFSIYSHM
PLHU          NRYEFLNGRVQSTE
TBBO          KAST--RIRITDNM
N1CHG         KKY--WGTKIKDAM
N1SGT         RSA-YGNELVANEE
N3EST         SSSSYWGSTVKNSM
EXBO          KLSS--SFTITPNM
N2PKA         ADA--HPDKVTESM
GGBSM.DATA2   hht.tTttb.ttt.
CF.DATA2      tttTTttbb.h..b
GOR.DATA2     tTTTTTTTtttttb.
COR2.DATA2    ttTTTTttttttbb
GOR3.DATA2    bbttTtttt.bbbb
```

```
LCR number:   5 from alignment number:---->  19
Total number of gaps in LCR :           17
Total number of identities in LCR :              0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       K-SAY-PGQITSNM
EZEC58      VDYGYYEYKFQ---
HPHB        IR-EYEGSTVPEHT
KFBO        LRSTKF--SIYSHM
PLHU        NRYEFLNGRVQSTE
TBBO        KASTRI--RITDNM
N1CHG       KKY-W-GTKIKDAM
N1SGT       -RSAYGNELVANEE
N3EST       SSSSYWGSTVKNSM
EXBO        KLSSSF--TITPNM
N2PKA       A DAH-PDKVTESM
GGBSM.DATA2 hh.tt.Ttbbtttt
CF.DATA2    ttt..bTt.b..hb
GOR.DATA2   TTTTTTTTtbtt.t
GOR2.DATA2  ttTTtTtTttttbb
GOR3.DATA2  .bttttt.bbbbbb


LCR number:   5 from alignment number:---->  21
Total number of gaps in LCR :           17
Total number of identities in LCR :              0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       K--SAYPGQITSNM
EZEC58      VDYGYYEYKFQ---
HPHB        IRHYE-GSTVPEHT
KFBO        LRSTKF--SIYSHM
PLHU        NRYEFLNGRVQSTE
TBBO        KASTRI--RITDNM
N1CHG       KKY--WGTKIKDAM
N1SGT       -RSAYGNELVANEE
N3EST       SSSSYWGSTVKNSM
EXBO        KLSSSF--TITPNM
N2PKA       A--DAHPDKVTESM
GGBSM.DATA2 hhthhtTtbbtttt
CF.DATA2    ttttbTt.b..hb
GOR.DATA2   TTTTTTTTtbtt.t
GOR2.DATA2  ttTTtTtTttttbb
GOR3.DATA2  .bbtttt.bbbbbb


LCR number:   5 from alignment number:---->  25
Total number of gaps in LCR :           17
Total number of identities in LCR :              0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       K --SAYPGQITSNM
EZEC58      V--D-YGYYEYKFQ
HPHB        IRHYE-GSTVPEHT
KFBO        LRSTKF--SIYSHM
PLHU        NRYEFLNGRVQSTE
TBBO        KASTRI--RITDNM
N1CHG       KKY--WGTKIKDAM
N1SGT        RSAYGNELVANEE
N3EST       SSSSYWGSTVKNSM
EXBO        KLSSSF--TITPNM
N2PKA       A  DAHPDKVTESM
GGBSM.DATA2 hhthhtTtb.ttt.
CF.DATA2    tTTttbTtb.h..b
GOR.DATA2   TTTTTTTTttttb.
GOR2.DATA2  ttTTtTtTttttbb
GOR3.DATA2  .b.tt.ttt.bbbb


LCR number:   5 from alignment number:---->  26
Total number of gaps in LCR :           28
Total number of identities in LCR :              0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       -KSAY-PGQITSNM-
EZEC58      VDYGY  YE -YKFQ
5PHB        IR-EYEGSTVPEHT-
KFBO        LRSTKF--SIYSHM-
PLHU        NRYEFLNGRVQSTE-
TBBO        KASTRI--RITDNM-
N1CHG       KKY-W-GTKIKDAM-
N1SGT       -RSAY-GNELVANEE
N3EST       SSSSYWGSTVKNSM-
EXBO        KLSSSF--TITPNM-
N2PKA       -ADAH-PDKVTESM-
GGBSM.DATA2 hh.tt.Tt..ttt..
CF.DATA2    ttt..bTt..h.hb.
GOR.DATA2   TTTTTTTTT.ttt..
GOR2.DATA2  ttTTtTtTtttt.b.
GOR3.DATA2  bbttt.tt..bbbb.
```

```
LCR number:    5 from alignment number:---->  37
Total number of gaps in LCR :           17
Total number of identities in LCR :      0
Names of seqs &
Struct. Pred. Methods.
           1
N1NTP      KSAYD--GQITSNM
EZEC58     VDYGYYEYKF---Q
HPHB       IR-EYEGSTVPEHT
KFBO       LRSTKF--SIYSHM
PLEU       NRYEFLNGRVQSTE
TBBO       KASTRI--RITDNM
N1CHG      KKYWGT--KIKDAM
N1SGT      R-SAYGNELVANEE
N3EST      SSSSYWGSTVKNSM
EXBO       KLSSSF--TITPNM
N2PKA      ADAIP--DKVTESM
GGBSM.DATA2 hh.tt.Ttbbttt.
CF.DATA2   ttt.tbTT.bh.hb
GOR.DATA2  tTTTTTTTtbtt..
GOR2.DATA2 ttTTttTTttttbb
GOR3.DATA2 bb.tttTtbbbbbb


LCR number:    5 from alignment number:---->  44
Total number of gaps in LCR :           17
Total number of identities in LCR :      0
Names of seqs &
Struct. Pred. Methods.
           1
N1NTP      KSA--YPGQITSNM
EZEC58     VD---YGYYEYKFQ
HPHB       IR-EYEGSTVPEHT
KFBO       LRSTKF--SIYSHM
PLEU       NRYEFLNGRVQSTE
TBBO       KASTRI--RITDNM
N1CHG      KKYWGT--KIKDAM
N1SGT      R-SAYGNELVANEE
N3EST      SSSSYWGSTVKNSM
EXBO       KLSSSF--TITPNM
N2PKA      ADA--HPDKVTESM
GGBSM.DATA2 hhtt.tTtb.ttt.
CF.DATA2   ttTttbTtb.h..b
GOR.DATA2  tTTTTTTTtttttb.
GOR2.DATA2 ttTTttTTttttbb
GOR3.DATA2 bbttttbttt.bbbb


LCR number:    5 from alignment number:---->  46
Total number of gaps in LCR :           17
Total number of identities in LCR :      0
Names of seqs &
Struct. Pred. Methods.
           1
N1NTP      KSAY--PGQITSNM
EZEC58     VDYGYYEYKFQ---
HPHB       IRH-YEGSTVPEHT
KFBO       LRSTKF--SIYSHM
PLHU       NRYEFLNGRVQSTE
TBBO       KASTRI--RITDNM
N1CHG      KKY--WGTKIKDAM
N1SGT      RS-AYGNELVANEE
N3EST      SSSSYWGSTVKNSM
EXBO       KLSSSF--TITPNM
N2PKA      ADAH--PDKVTESM
GGBSM.DATA2 hhtthtTtbbtttt
CF.DATA2   tttt.bTt.b..hb
GOR.DATA2  tTTTTTTTtbtt.t
GOR2.DATA2 ttTTttTtTttttbb
GOR3.DATA2 bb.tttt.bbbbbb


LCR number:    5 from alignment number:---->  55
Total number of gaps in LCR :           17
Total number of identities in LCR :      0
Names of seqs &
Struct. Pred. Methods.
           1
N1NTP      KSA--YPGQITSNM
EZEC58     VDYGYYEYKFQ---
HPHB       IRH YEGSTVPEHT
KFBO       LRSTKF--SIYSHM
PLHU       NRYEFLNGRVQSTE
TBBO       KASTRI--RITDNM
N1CHG      KKY--WGTKIKDAM
N1SGT      -RSAYGNELVANEE
N3EST      SSSSYWGSTVKNSM
EXBO       KLSSSF--TITPNM
N2PKA      ADA--HPDKVTESM
GGBSM.DATA2 hht.htTtbbtttt
CF.DATA2   tttt.bTt.b..hb
GOR.DATA2  TTTTTTTTtbtt.t
GOR2.DATA2 ttTTtTtTttttbb
GOR3.DATA2 .bbtttt.bbbbbb
```

```
LCR number:    5 from alignment number:---->  58
Total number of gaps in LCR :            17
Total number of identities in LCR :       0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       KSA--YPGQITSKM
EZEC58      VDYGYYEYKFQ---
HPHB        IR-HYEGSTVPEHT
KFBO        LRSTKF--SIYSHM
PLHO        NRYEFLNGRVQSTE
TBBO        KASTRI--RITDNM
N1CHG       KK--YWGTKIKDAM
N1SGT       -RSAYGNELVANEE
N3EST       SSSSYWGSEVKNSM
EXBO        KLSGSF--DTTPNM
N2PKA       ADA--HPDKVTESM
GGBSM.DATA2 hhtthtTtbbtttt
CF.DATA2    ttt..bTt.b..Hb
GOR.DATA2   TTTTTTTTtbtt.t
GOR2.DATA2  ttTTtTtTttttbb
GOR3.DATA2  .bbtttt.bbbbbb


These are the different alignments found for loosely conserved region    6

LCR number:    6 from alignment number:---->   1
Total number of gaps in LCR :            32
Total number of identities in LCR :       0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       YLEGGK---DSC
EZEC58      ---SPTTLRAAF
HPHB        M---SKYQEDTC
KFBO        YHEGGK---DSC
PLHO        HLAGGT---DSC
TBBO        YKPGEGKRGDAC
N1CHG       --ASGV---SSC
N1SGT       YPDTGGV -DTC
N3EST       G-DGVR---SGC
EXBO        YDTQPE---DAC
N2PKA       YLPGGK---DTC
GGBSM.DATA2 TTTTTTTTTttT
CF.DATA2    b.TTtthHHttt
GOR.DATA2   T.TTTT.T.ttt
GOR2.DATA2  tTTTTT.TCbbb
GOR3.DATA2  .tTTTt.BBbBB


LCR number:    6 from alignment number:---->   3
Total number of gaps in LCR :            32
Total number of identities in LCR :       0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       YLEGGK---DSC
EZEC58      ---SPTTLRAAF
HPHB        M---SKYQEDTC
KFBO        YHEGGK---DSC
PLHO        HLAGGT---DSC
TBBO        YKPGEGKRGDAC
N1CHG       --ASGV---SSC
N1SGT       YPDTGG--VDTC
N3EST       G-DGVR---SGC
EXBO        YDTQPE---DAC
N2PKA       YLPGGK---DTC
GGBSM.DATA2 TTTTTTTTTttT
CF.DATA2    b.TttHHbttt
GOR.DATA2   T.TTTTTBttt
GOR2.DATA2  tTTTTTT.bbb
GOR3.DATA2  .tTTTtTBBbBB


LCR number:    6 from alignment number:---->   8
Total number of gaps in LCR :            32
Total number of identities in LCR :       0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       YLPGGK---DSC
EZEC58      SPTTLR---AAF
HPHB        M---SKYQEDTC
KFBO        YHEGGK---DSC
PLHO        HLAGGT---DSC
TBBO        YKPGEGKRGDAC
N1CHG       --ASGV---SSC
N1SGT       YPDTGG--VDTC
N3EST       G-DGVR---SGC
EXBO        YDTQPE---DAC
N2PKA       YLPGGK---DTC
GGBSM.DATA2 TTTTTt..Tttf
CF.DATA2    btTttt...ttt
GOR.DATA2   T.TTTT...ttt
GOR2.DATA2  tTTTTt..Tbbb
GOR3.DATA2  ttTTtt..BbBB
```

```
LCR number:    6 from alignment number:---->  19
Total number of gaps in LCR :             32
Total number of identities in LCR :        0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       YLEGGKDS---C
EZEC58      S-PTTLRAAF--
HPEB        ---MSKYQEDTC
KFBO        YHEG-GK--DSC
PLHU        HLAGGTDS---C
TBBO        YKPGEGKRGDAC
N1CHG       --ASGVSS---C
N1SGT       YPDTG---GVDTC
N3EST       --GDGVRSG--C
EXBO        YDTQ-PE--DAC
N2PKA       YLPGGKDT---C
GGBSM.DATA2 TTTTTttttTTTT
CF.DATA2    B.Ttttt.htT
GOR.DATA2   T.TTTTtt..tt
COR2.DATA2  tTTTTTtbbbbb
GOR3.DATA2  ttTTTtbBBbBB


LCR number:    6 from alignment number:---->  21
Total number of gaps in LCR :             32
Total number of identities in LCR :        0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       YLEGGK---DSC
EZEC58      S--PTTLRA-AF
HPEB        ---MSKYQEDTC
KFBO        YHEGGK- ---DSC
PLHU        HLAGGT---DSC
TBBO        YKPGEGKRGDAC
N1CHG       --ASGV---SSC
N1SGT       YPDTGGV--DTC
N3EST       G-DGVR---SGC
EXBO        YDTQPE---DAC
N2PKA       YLPGGK---DTC
GGBSM.DATA2 TTTTTTTTTTtTT
CF.DATA2    b.TtttbHttt
GOR.DATA2   T.TTTT.T.ttt
GOR2.DATA2  tTTTTt.TTbbb
GOR3.DATA2  ttTTTt.BBbBB


LCR number:    6 from alignment number:---->  23
Total number of gaps in LCR :             32
Total number of identities in LCR :        0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       YLEGGK---DSC
EZEC58      SPTTLR---AAF
HPHB        ---MSKYQEDTC
KFBO        YHEGGK---DSC
PLHU        HLAGGT---DSC
TBBO        YKPGEGKRGDAC
N1CHG       --ASGV---SSC
N1SGT       YPDTGG--VDTC
N3EST       G-DGVR---SGC
EXBO        YDTQPE---DAC
N2PKA       YLPGGK---DTC
GGBSM.DATA2 TTTTTt..TttT
CF.DATA2    btTttt...ttt
GOR.DATA2   T.TTTT...ttt
GOR2.DATA2  tTTTTt..Tbbb
GOR3.DATA2  ttTttt..BbBB


LCR number:    6 from alignment number:---->  26
Total number of gaps in LCR :             43
Total number of identities in LCR :        0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       -YLEGG---KDSC
EZEC58      -SPTTL---RAAF
HPHB        MSKYQR----DTC
KFBO        -YHEG-GK--DSC
PLHU        --HLAGGTDS--C
TBBO        -YKPGEGKRGDAC
N1CHG       ----ASGVSS--C
N1SGT       -YPDTGGV--DTC
N3EST       ----GDGVRSG-C
EXBO        -YDTQ-PE--DAC
N2PKA       -YLPGG---KDTC
GGBSM.DATA2 .TTTTTTTTtTTT
CF.DATA2    .bttttTtTTttt
GOR.DATA2   .TTTTTTTTtt.t
GOR2.DATA2  .tTTTTTT.bbbb
GOR3.DATA2  .ttTTttTtBbbBB
```

```
LCR number:   6 from alignment number:---->  37
Total number of gaps in LCR :           32
Total number of identities in LCR :              0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       YLEGGK---DSC
EZEC58      SPTTLR---AAF
HPHB        M---SKYQEDTC
KFBO        YHEGGK---DSC
PLHU        HLAGGT---DSC
TBBO        YKPGEGKRGDAC
N1CHG       --ASGV---SSC
N1SGT       YPDTGG--VDTC
N3EST       GD-GVR---SGC
EXBO        YDTQPE-   DAC
N2PKA       YLPGGK---DTC
GGBSM.DATA2 TTTTTt..TtTT
CF.DATA2    btTttt...ttt
GOR.DATA2   TTTTTT...ttt
GOR2.DATA2  tTTTTt..Tbbb
GOR3.DATA2  ttTTtt..BbBB


LCR number:   6 from alignment number:---->  41
Total number of gaps in LCR :           32
Total number of identities in LCR :              0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       YLEGGK---DSC
EZEC58      SPTTLR---AAF
HPHB        M---SKYQEDTC
KFBO        YHEGGK---DSC
PLHU        HLAGGT--- DSC
TBBO        YKPGEGKRGDAC
N1CHG       --ASGV---SSC
N1SGT       YPDTGGV  DTC
N3EST       GD-GVR---SGC
EXBO        YDTQPR---DAC
N2PKA       YLPGGK---DTC
GGBSM.DATA2 TTTTTtT..ttT
CF.DATA2    btTttt...ttt
GOR.DATA2   TTTTTTT..ttt
GOR2.DATA2  tTTTTtT..bbb
GOR3.DATA2  ttTTttT..bBB


LCR number:   6 from alignment number:---->  44
Total number of gaps in LCR :           32
Total number of identities in LCR :              0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       YLEGGK---DSC
EZEC58      SPTTLR---AAF
HPHB        MS---KYQEDTC
KFBO        YHEGGK---DSC
PLHU        HLAGGT---DSC
TBBO        YKPGECKRGDAC
N1CHG       --ASGV---SSC
N1SGT       YPDTGGV--DTC
N3EST       GD-GVR---SGC
EXBO        YDTQPE---DAC
N2PKA       YLPGGK---DTC
GGBSM.DATA2 TTTTTtT..ttT
CF.DATA2    btTttt...ttt
GOR.DATA2   TTTTTTT..ttt
GOR2.DATA2  tTTTTtT..bbb
GOR3.DATA2  ttTTttT..bBB


LCR number:   6 from alignment number:---->  55
Total number of gaps in LCR :           32
Total number of identities in LCR :              0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       YLEGG--KDS-C
EZEC58      S-PTT--LRAAF
HPHB        M---SKYQEDTC
KFBO        YHEGG---KDSC
PLHU        HLAGG--TDS-C
TBBO        YKPGEGKRGDAC
N1CHG       --ASG--VSS-C
N1SGT       YPDTGGV--DTC
N3EST       --GDG--VRSGC
EXBO        YDTQP---EDAC
N2PKA       YLPGG--KDT-C
GGBSM.DATA2 TTTTTTTttttT
CF.DATA2    b.TTt..ttttt
GOR.DATA2   T.TTTTTTtttt
GOR2.DATA2  tTTTTTTTtbbb
GOR3.DATA2  .tTTTTT,bbBB
```

```
LCR number:   6 from alignment number:---->  56
Total number of gaps in LCR :          32
Total number of identities in LCR :             0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       YLEGG---KDSC
EZEC58      SPTTL---RAAP
HPHB        M---SKYQEDTC
KFBO        YHEGG---KDSC
PLHU        HLAGG---TDSC
TBBO        YKPGEGKRGDAC
N1CHG       --ASG---VSSC
N1SGT       YPDTGG--VDTC
N3EST       Q-DGV---RSGC
EXBO        YDTQP---EDAC
N2PKA       YLPGG---KDTC
GGBSM.DATA2 TTTTTI..tttT
CF.DATA2    bttttt...ttt
GOR.DATA2   T.TTTT..ttt
GOR2.DATA2  tTTTTI..bbb
GOR3.DATA2  ttTTtT..bBB


These are the different alignments found for loosely conserved region    7

LCR number:   7 from alignment number:---->   1
Total number of gaps in LCR :          29
Total number of identities in LCR :             0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       C----SGKL--
EZEC58      CAG------VA
HPHB        VHDLEENTWYA
KFBO        TE--VEGTSFL
PLHU        CPE--KDKYIL
TBBO        MKSPYNNRWYQ
N1CHG       CKK--NGAWTL
N1SGT       RKDN-ADEWIQ
N3EST       CLV--NGQYAV
EXBO        TR--FKDTYFV
N2PKA       C----NGMW--
GGBSM.DATA2 bhhHTtTtbBB
CF.DATA2    bb.Hbttb8BB
GOR.DATA2   bb..bttttBB
GOR2.DATA2  BbtTbttLLbB
GOR3.DATA2  BbbTtttbbB


LCR number:   7 from alignment number:---->   8
Total number of gaps in LCR :          29
Total number of identities in LCR :             0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       --CSGK----L
EZEC58      --CAGV----A
HPHB        VHDLEENTWYA
KFBO        TE--VEGTSFL
PLHU        --CPEKDKYIL
TBBO        MKSPYNNRWYQ
N1CHG       --CKKNGAWTL
N1SGT       RK-DNADEWIQ
N3EST       --CLVNGQYAV
EXBO        TR---FKDTYFV
N2PKA       --CNGM----W
GGBSM.DATA2 ..b.ttTtbBb
CF.DATA2    HHBbb.TbBBB
GOR.DATA2   BBbb.tttbBb
GOR2.DATA2  Bbbbtttt.bb
GOR3.DATA2  BBBbbtt.bbb


LCR number:   7 from alignment number:---->  19
Total number of gaps in LCR :          29
Total number of identities in LCR :             0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       --C--SGKL--
EZEC58      --C----AGVA
HPHB        VHDLEENTWYA
KFBO        TE--VEGTSFL
PLHU        --CPEKDKYIL
TBBO        MKSPYNNRWYQ
N1CHG       --CKKNGAWTL
N1SGT       RK-DNADEWIQ
N3EST       --CLVNGQYAV
EXBO        TR--FKDTYFV
N2PKA       --C--NGMW--
GGBSM.DATA2 ..bhttTtbBB
CF.DATA2    HHB.ttLbBBB
GOR.DATA2   BBb.btttbBB
GOR2.DATA2  Bbb..ttt.bB
GOR3.DATA2  BBB..tt.bbB
```

```
LCR number:   7 from alignment number:---- >  21
Total number of gaps in LCR :            29
Total number of identities in LCR :               0
Names of seqs &
Struct. Pred. Methods.
          1
N1NTP     --C--SGKL--
EZEC58    C - - - AGVA
HPHB      VHDLEENTWYA
KFBO      TE--VEGTSFL
PLHU      --CFEKDKYIL
TBBO      MKSPYNNRWYQ
N1CHG     --CKKNGAWTL
N1SGT     RK-DNADEWIQ
N3EST     --CLVNGQYAV
EXDO      TR--FKDTYFV
N2PKA     --C--NGMW--
CGBSM.DATA2 b.bhttTtbBB
CF.DATA2  hHB.htthBBB
GOR.DATA2 BBh.btttbBB
GOR2.DATA2 Bbb..ttt.bB
GOR3.DATA2 BBB..tt.bbb


LCR number:   7 from alignment number:----->  23
Total number of gaps in LCR :            29
Total number of identities in LCR :               0
Names of seqs &
Struct. Pred. Methods.
          1
N1NTP     --C-----SGKL
EZEC58    C------AGVA
HPHB      VHDLEENTWYA
KFBO      TE--VEGTSFL
PLHU      --CFEKDKYIL
TBBO      MKSPYNNRWYQ
N1CHG     --CKKNGAWTL
N1SGT     RK-DNADEWIQ
N3EST     --CLVNGQYAV
EXDO      TR--FKDTYFV
N2PKA     --C----NGMW
GGBSM.DATA2 b.bhttTtbbb
CF.DATA2  hHB.h.TbbBB
GOR.DATA2 BBb.btttbbb
GOR2.DATA2 Bbb...tt.bb
GOR3.DATA2 BBB..tt.bbb


LCR number:   7 from alignment number:----->  25
Total number of gaps in LCR :            29
Total number of identities in LCR :               0
Names of seqs &
Struct. Pred. Methods.
          1
N1NTP     CSGK------L
EZEC58    CAGV- - - - -A
HPHB      VHDLEENTWYA
KFBO      TE--VEGTSFL
PLHU      --CFEKDKYIL
TBBO      MKSPYNNRWYQ
N1CHG     --CKKNGAWTL
N1SGT     RKD-NADEWIQ
N3EST     --CLVNGQYAV
EXDO      TR  FKDTYFV
N2PKA     CKGM------W
GGBSM.DATA2 bCbttttTtbBb
CF.DATA2  ..bbh.TbBBB
GOR.DATA2 bbbbbtttbBb
GOR2.DATA2 Bb....tt.bb
GOR3.DATA2 Bbbb.tt.bbb


LCR number:   7 from alignment number:----->  26
Total number of gaps in LCR :            29
Total number of identities in LCR :               0
Names of seqs &
Struct. Pred. Methods.
          1
N1NTP     --C--SGK--L
EZEC58    --C--AGV--A
HPEB      VHDLEENTWYA
KFBO      TE--VEGTSFL
PLEU      --CFEKDKYIL
CBBO      MKSPYNNRWYQ
N1CHG     --CKKNGAWTL
N1SGT     RK-DNADEWIQ
N3EST     --CLVNGQYAV
EXDO      TR--FKDTYFV
N2PKA     --C--NGM--W
GGBSM.DATA2 ..bhttttbBb
CF.DATA2  HHB.htthBBB
GOR.DATA2 BBb.btttbBb
GOR2.DATA2 Bbb..ttt.bb
GOR3.DATA2 BBB..tt.bbb
```

```
LCR number:   7 from alignment number:---->  44
Total number of gaps in LCR :            29
Total number of identities in LCR :               0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       CSG------KL
EZEC58      CAG------VA
HPHB        VHDLEENTWYA
KFBO        TE--VEGTSFL
PLHU        CF--EKDKYIL
TBBO        MKSPYNNRWYQ
N1CHG       CK--KNGAWTL
N1SGT       RKD-NADEWIQ
N3EST       CL--VNGQYAV
EXBO        TR--FKDTYFV
N2PKA       CNG------MW
GGBSM.DATA2 btt.bbTbbb
CF.DATA2    bb..h.TbBBB
GOR.DATA2   bbt.btttbbb
GOR2.DATA2  Bbt...tt.bb
GOR3.DATA2  Bbb..tt.bbb


LCR number:   7 from alignment number:---->  48
Total number of gaps in LCR :            29
Total number of identities in LCR :               0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       CSG--K----L
EZEC58      CAG------VA
HPHB        VHDLEENTWYA
KFBO        TE--VEGTSFL
PLHU        CFE--KDKYIL
TBBO        MKSPYNNRWYQ
N1CHG       CKK--NGAWTL
N1SGT       RKDN-ADEWIQ
N3EST       CLV--NGQYAV
EXBO        TR--FKDTYFV
N2PKA       CNG--M----W
GGBSM.DATA2 bttHTtTbBb
CF.DATA2    bbbHh.TbBBB
GOR.DATA2   bbt.btttbBb
GOR2.DATA2  BbtTbttt.bb
GOR3.DATA2  BbbTttt.bbb


LCR number:   7 from alignment number:---->  55
Total number of gaps in LCR :            29
Total number of identities in LCR :               0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       --C--SGKL---
EZEC58      --CAGVA-----
HPHB        VHDLEENTWYA
KFBO        TE--VEGTSFL
PLHU        --CFEKDKYIL
TBBO        MKSPYNNRWYQ
N1CHG       --CKNGAWTL
N1SGT       RK-DNADEWIQ
N3EST       --CLVNGQYAV
EXBO        TR--FKDTYFV
N2PKA       --C--NGMW--
GGBSM.DATA2 ..bhtttbBB
CF.DATA2    HH8b.ttbBBB
GOR.DATA2   BBbbbttbBB
GOR2.DATA2  Bbbbb.tttbB
GOR3.DATA2  BBBbbtttbbB


LCR number:   7 from alignment number:---->  61
Total number of gaps in LCR :            29
Total number of identities in LCR :               0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       CSG--K----L
EZEC58      CAG--VA-----
HPHB        VHDLEENTWYA
KFBO        TE--VEGTSFL
PLHU        CFE--KDKYIL
TBBO        MKSPYNNRWYQ
N1CHG       CKK--NGAWTL
N1SGT       RKDN-ADEWIQ
N3EST       CLV--NGQYAV
EXBO        TR--FKDTYFV
N2PKA       CNG--M----W
GGBSM.DATA2 bttHTtttbBb
CF.DATA2    bbbHh.tbBBB
GOR.DATA2   bbt.btttbBb
GOR2.DATA2  BbtTbttt.bb
GOR3.DATA2  BbbTttt.bbb
```

```
LCR number:    7 from alignment number:---->  62
Total number of gaps in LCR :           29
Total number of identities in LCR :              0
Names of segs &
Struct. Pred. Methods.
              1
N1NTP         CSG--K----L
EZEC58        CAG--VA----
HPHB          VHDLEENTWYA
KFBO          TE--VEGTSFL
PLHU          CFE--KDKYIL
TBBO          MKSPYNNRWYQ
N1CHG         CKK--NGAWIL
N1SGT         RED-NADEWIQ
N3EST         CLV--NGQYAV
EXBO          TR--FKDTYFV
N2PKA         CNG--M----W
GGBSM.DATA2   btt.TtttbBb
CF.DATA2      bbb.h.tbBBB
GOR.DATA2     bbt.bttlbBb
GOR2.DATA2    Bbt..ttt.bb
GOR3.DATA2    Bbb.ttt.bbb


These are the different alignments found for loosely conserved region    8

LCR number:    8 from alignment number:---->   1
Total number of gaps in LCR :           22
Total number of identities in LCR :              0
Names of segs &
Struct. Pred. Methods.
              1
N1NTP         --SGCAQK
EZEC58        HP----DA
HPHB          K---CSAV
KFBO          E--ECAMK
PLHU          --LGCARP
TBBO          E--GCDRD
N1CHG         --SSTCST
N1SGT         --YGCARP
N3EST         SRLGCNVT
EXBO          E--GCARK
N2PKA         -HTPCGSA
GGBSM.DATA2   tTTTtttT
CF.DATA2      ..TTtttt
GOR.DATA2     t.TTTTtt
GOR2.DATA2    tTTTTTtt
GOR3.DATA2    hTTttttt


LCR number:    8 from alignment number:---->   8
Total number of gaps in LCR :           22
Total number of identities in LCR :              0
Names of segs &
Struct. Pred. Methods.
              1
N1NTP         -S-GCAQK
EZEC58        HP----DA
HPHB          K---CSAV
KFBO          E--ECAMK
PLHU          --LGCARP
TBBO          E--GCDRD
N1CHG         --SSTCST
N1SGT         --YGCARP
N3EST         SRLGCNVT
EXBO          E--GCARK
N2PKA         HT-PCGSA
GGBSM.DATA2   tTTTtttT
CF.DATA2      ...Tttt
GOR.DATA2     t.TTTTtt
GOR2.DATA2    tTTTTTtt
GOR3.DATA2    .TTttttt


LCR number:    8 from alignment number:---->  19
Total number of gaps in LCR :           22
Total number of identities in LCR :              0
Names of segs &
Struct. Pred. Methods.
              1
N1NTP         --SGCAQK
EZEC58        HPDA----
HPHB          K---CSAV
KFBO          --EGCAMK
PLHU          L--GCARP
TBBO          --EGCDRD
N1CHG         S--STCST
N1SGT         --YGCARP
N3EST         SRLGCNVT
EXBO          --EGCARK
N2PKA         -HTPCGSA
GGBSM.DATA2   tTTTtttT
CF.DATA2      ..TTtttt
GOR.DATA2     T.TTTTtt
GOR2.DATA2    TTTTTTtt
GOR3.DATA2    .Tttttt
```

```
LCR number:   6 from alignment number:---->  21
Total number of gaps in LCR :           22
Total number of identities in LCR :              0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       --SGCAQK
SZEC58      H----PDA
HPHB        K---CSAV
KFBC        E--ECAMK
PLHU        L--GCARP
TBBC        E--GCDRD
N1CHG       S--STCST
N1SGT       Y--GCARP
N3EST       SRLGCNVT
EXBO        E--GCARK
N2PKA       -HTPCGSA
GGBSM.DATA2 t.TTttt?
CF.DATA2    t..Ttttt
GOR.DATA2   t..TTTtt
GOR2.DATA2  t.TTTTtt
GOR3.DATA2  t.Tttttt


LCR number:   8 from alignment number:---->  25
Total number of gaps in LCR :           22
Total number of identities in LCR :              0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       --SGCAQK
EZEC58      -HP---DA
HPHB        K---CSAV
KFBC        E--ECAMK
PLHU        L--GCARP
TBBC        E--GCDRD
N1CHG       S  STCST
N1SGT       Y--GCARP
N3EST       SRLGCNVT
EXBO        E--GCARK
N2PKA       -HTPCGSA
GGBSM.DATA2 tTTTtttT
CF.DATA2    t..Ttttt
GOR.DATA2   t..TTTtt
GOR2.DATA2  tTTTTTtt
GOR3.DATA2  .TTttttt


LCR number:   8 from alignment number:---->  26
Total number of gaps in LCR :           22
Total number of identities in LCR :              0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       --SGCAQK
EZEC58       H   PDA
HPHB        K---CSAV
KFBC        E--ECAMK
PLHU        L--GCARP
TBBC        E--GCDRD
N1CHG       S--STCST
N1SGT       Y--GCARP
N3EST       SRLGCNVT
EXBO        E--GCARK
N2PKA       -HTPCGSA
GGBSM.DATA2 tTTTtttT
CF.DATA2    t..Ttttt
GOR.DATA2   t..TTTtt
GOR2.DATA2  tTTTTTtt
GOR3.DATA2  .TTttttt


LCR number:   8 from alignment number:---->  44
Total number of gaps in LCR :           22
Total number of identities in LCR :              0
Names of seqs &
Struct. Pred. Methods.
            1
N1NTP       --SGCAQK
EZEC58      -HP---DA
HPHB        ---KCSAV
KFBC        E--ECAMK
PLHU        --LGCARP
TBBC        E--GCDRD
N1CHG       --SSTCST
N1SGT       Y--GCARP
N3EST       SRLGCNVT
EXBO        E--GCARK
N2PKA       -HTPCGSA
GGBSM.DATA2 TTTTtttT
CF.DATA2    T.Ttttt
GOR.DATA2   t.TTTTtt
GOR2.DATA2  tTTTTTtt
GOR3.DATA2  .TTttttt
```

```
LCR number:   8 from alignment number:----->  54
Total number of gaps in LCR :          22
Total number of identities in LCR :             0
Names of seqs &
Struct. Pred. Methods.
          1
N1NTP     --SGCAQK
EZEC58    H----PDA
HPHB      K---CSAV
KFBO      E--ECAMK
PLHU      --LGCARP
TBBO      E--GCDRD
N1CHG     --SSTCST
N1SGT     --YGCARP
N3EST     SRLGCNVT
EXBO      E--GCARK
N2PKA     -HTPCGSA
GGBSM.DATA2 t.TTtttT
CF.DATA2  ..TTtttt
GOR.DATA2 t.TTTTtt
GOR2.DATA2 t.TTTTtt
GOR3.DATA2 h.tttttt


LCR number:   8 from alignment number:     >  58
Total number of gaps in LCR :          22
Total number of identities in LCR :             0
Names of seqs &
Struct. Pred. Methods.
          1
N1NTP     -S-GCAQK
EZEC58    HP----DA
HPHB      K---CSAV
KFBO      E--ECAMK
PLHU      --LGCARP
TBBO      E--GCDRD
N1CHG     SSTCS--T
N1SGT     --YGCARP
N3EST     SRLGCNVT
EXBO      E--GCARK
N2PKA     HT-PCGSA
GGBSM.DATA2 tTTTtttT
CF.DATA2  tT.Tttt
GOR.DATA2 tTTTTTtt
GOR2.DATA2 tTTTTTtt
GOR3.DATA2 tTTtt.tt


These are the different alignments found for loosely conserved region:    9

LCR number:   9 from alignment number:----->   1
Total number of gaps in LCR :          19
Total number of identities in LCR :             0
Names of seqs &
Struct. Pred. Methods.
          1
N1NTP     TIASN--
EZEC58    VVN----
HPHB      TIAEN--
KFBO      --KTKLT
PLHU      VMRNN--
TBBO      VIDRLGS
N1CHG     TLAAN--
N1SGT     AARTL--
N3EST     VIASN--
EXBO      IMKARAG
N2PKA     TITEN-P
GGBSM.DATA2 b.tTTTT
CF.DATA2  bbhhH..
GOR.DATA2 b.Hhh..
GOR2.DATA2 .......
GOR3.DATA2 .......


LCR number:   9 from alignment number:----->   7
Total number of gaps in LCR :          19
Total number of identities in LCR :             0
Names of seqs &
Struct. Pred. Methods.
          1
N1NTP     TIASN--
EZEC58    VVN----
HPHB      TIAEN--
KFBO      --KTKLT
PLHU      VMRN--N
TBBO      VIDRLGS
N1CHG     TLAA--N
N1SGT     --AARTL
N3EST     VIAS--N
EXBO      IMKARAG
N2PKA     TITEN-P
GGBSM.DATA2 bbttTTT
CF.DATA2  bbhhhH.
GOR.DATA2 BBHh.H.
GOR2.DATA2 .......
GOR3.DATA2 .......
```

```
LCR number:   9 from alignment number:---->  10
Total number of gaps in LCR :           19
Total number of identities in LCR :      0
Names of seqs &
Struct. Pred. Methods.
                1
N1NTP           TIASN--
EZEC58          VVN----
HPHB            TIAEN--
KFBO            --KTKLT
PLHU            VMRNN--
TBBO            VIDRLGS
N1CHG           TLAAN--
N1SGT           --AARTL
N3EST           VIASN--
EXBO            IMKARAG
N2PKA           TITENP-
GGBSM.DATA2     bbttTTT
CF.DATA2        bbhhHH.
GOR.DATA2       BBHttH.
GOR2.DATA2      .......
GOR3.DATA2      .......


LCR number:   9 from alignment number:---->  16
Total number of gaps in LCR :           19
Total number of identities in LCR :      0
Names of seqs &
Struct. Pred. Methods.
                1
N1NTP           TIASN--
EZEC58          VVN----
HPHB            TIAS--N
KFBO            --KTKLT
PLHU            VMRN--N
TBBO            VIDRLGS
N1CHG           TLAA--N
N1SGT           --AART.
N3EST           VIAS--N
EXBO            IMKARAG
N2PKA           TITENP-
GGBSM.DATA2     bbttTTT
CF.DATA2        bbhhHH.
GOR.DATA2       BBHt.H.
GOR2.DATA2      .......
GOR3.DATA2      .......


LCR number:   9 from alignment number:---->  19
Total number of gaps in LCR :           30
Total number of identities in LCR :      0
Names of seqs &
Struct. Pred. Methods.
                1
N1NTP           TIAS--N-
EZEC58          VVN-----
HPBR            TIAEN---
KFBO            --KTKLT-
PLHU            VMRN--N-
TBBO            VID-RLGS
N1CHG           TLAA--N-
N1SGT           --AAR-TL
N3EST           VIAS--N-
EXBO            IMKARAG-
N2PKA           TITE--NP
GGBSM.DATA2     bbttTTTT
CF.DATA2        bbhHHH..
GOR.DATA2       BBH...T.
GOR2.DATA2      ........
GOR3.DATA2      ........


LCR number:   9 from alignment number:---->  21
Total number of gaps in LCR :           19
Total number of identities in LCR :      0
Names of seqs &
Struct. Pred. Methods.
                1
N1NTP           TIASN--
EZEC58          VVN----
HPHB            TIAEN--
KFBO            --KTKLT
PLHU            VMRNN--
TBBO            VIDRLGS
N1CHG           TLAAN--
N1SGT           --AARTL
N3EST           VIASN--
EXBO            IMKARAG
N2PKA           TITEN-P
GGBSM.DATA2     bbttTTT
CF.DATA2        bbhhHH.
GOR.DATA2       BBHttH.
GOR2.DATA2      .......
GOR3.DATA2      .......
```

```
LCR number:   9 from alignment number:----->  26
Total number of gaps in LCR :           30
Total number of identities in LCR :              0
Names of seqs &
Struct. Pred. Methods.
           .
N1NTP      TIAS--N-
EZEC58     VV----N-
HPHB       TIAE---N
KFBO       --KTKLT-
PLHU       VMRN---N
TBBO       VID-RLGS
N1CHG      TLAA---N
N1SGT      - AAR TL
N3EST      VIAS---N
EXBO       IMKARAG-
N2PKA      TITE--NP
GGRSM.DATA2 bbttTTTT
CF.DATA2   bbhHHHH.
GOR.DATA2  HHH.....
GOR2.DATA2 ........
GOR3.DATA2 ........


LCR number:   9 from alignment number:----->  55
Total number of gaps in LCR :           19
Total number of identities in LCR :              0
Names of seqs &
Struct. Pred. Methods.
           1
N1NTP      TIASN--
EZEC58     VVN----
HPHB       TIAEN
KFBO       KTKLT--
PLHU       VMRNN--
TBBO       VIDRLGS
N1CHG      TLAAN--
N1SGT      AARTL--
N3EST      VIASN--
EXBO       IMKARAG
N2PKA      TITENP-
GGRSM.DATA2 b.tTTT.
CF.DATA2   ..bbH..
GOR.DATA2  b.h....
GOR2.DATA2 .......
GOR3.DATA2 .......
```

Bibliography

1. Haslam, E., "The shikimate pathway", (1974), London, U.K.: Butterworths.

2. Weiss, U. and Edwards, J.M., "The biosynthesis of aromatic compounds", (1980), New York, USA.: John Wiley and Sons.

3. Bentley, R., "The shikimate pathway - a metabolic tree with many branches.", *CRC Critical Reviews in Biochemistry and Molecular Biology.*, (1990). **25**(5): p.307-384.

4. Singh, J. and Thornton, J.M., "The interaction between phenylalanine rings in proteins.", *FEBS Letters*, (1985). **191**: p.1-6.

5. Haslam, E., "Shikimic acid: metabolism and metabolites.", (1993), Chichester: John Wiley and Sons.

6. Boocock, M.R. and Coggins, J.R., "Kinetics of 5-enolpyruvylshikimate 3-phosphate synthase inhibition by glyphosate", *FEBS Letters*, (1983). **154**: p.127-133.

7. Steinrucken, H.C. and Amrhein, N., "The herbicide glyphosate is a potent inhibitor of 5-enolpyruvyl-shikimic acid 3-phosphate synthase", *Biochem. Biophys. Res. Comm.*, (1980). **94**: p.1207-1212.

8. Amrhein, N., Schab, J., and Steinrucken, H.C., "The mode of action of the herbicide glyphosate", *Naturwissenschaften*, (1980). **647**: p.356-357.

9. Herrmann, K.M., "The shikimate pathway: early steps in the biosynthesis of aromatic compounds", *The Plant Cell*, (1995). **7**: p.907-919.

10. Berlyn, M.B. and Giles, N.H., "Organization of enzymes in the polyaromatic synthetic pathway: separability in bacteria.", *Journal of Bacteriology*, (1969). **99**: p.222-230.

11. Lumsden, J. and Coggins, J.R., "The subunit structure of the arom multienzyme complex of *Neurospora crassa*.", *Biochemical Journal*, (1977). **161**: p.599-607.

12. Patel, V.B. and Giles, N.H., "Purification of the arom multienzyme aggregate from *Euglena gracilis*.", *Biochemica et Biophysica Acta*, (1979). **567**(24-34).

13. Charles, I.G., Keyte, J.W., Brammer, W.J., Smith, M., and Hawkins, A.R., "The isolation and nucleotide sequences of the complex *AROM* locus of *Aspergillus nidulans*.", *Nucleic Acid Research*, (1986). **14**: p.2201-2213.

14. Nakanishi, N. and Yamamoto, M., "Analysis of the structure and transcription of the *aro3* cluster gene in *Schizosaccharomyces pombe*.", *Mol. Gen. Genet.*, (1984). **195**: p.164-169.

15. Duncan, K., Edwards, R.M., and Coggins, J.R., "The pentafunctional arom enzyme of *Saccharomyces cerevisiae* is a mosaic of monofunctional domains.", *Biochemical Journal*, (1987). **246**: p.375-386.

16. Stallings, W.C., Abdel-Meguid, S.S., Lim, L.W., Shieh, H., Dayringer, H.E., Leimgruber, N.K., Stegeman, R.A., Anderson, K.S., Sokorski, J.A., Padgette, S.R., and Kishore, G.M., "Structure and topological symmetry of the glyphosate target 5-enol pyruvylshikimate-3-phosphate synthase: a distinctive protein fold", *Proceedings of the National Academy of Science, USA.*, (1991). **88**: p.5046-5050.

17. Boys, C.W.G., Fawcett, S.M., Sawyer, L., Moore, J.D., Charles, I.G., Hawkins, A.R., Deka, R., Kleanthous, C., and Coggins, J.R., "The crystallization of a type I 3-dehydroquinase from *Salmonella typhimurium*", *Journal of Molecular Biology*, (1992). **227**: p.352-355.

18. Gourley, D.G., Coggins, J.R., Isaacs, N.W., Moore, J.D., Charles, I.G., and Hawkins, A.R., "Crystallization of a type II dehydroquiase from *Mycobacterium tuberculosis*.", *Journal of Molecular Biology*, (1994). **241**: p.488-491.

19. Frost, J.W., Bender, J.L., Kadonaga, J.T., and Knowles, J.R., "Dehydroquinate synthase from *Escherichia Coli*: purification, cloning and construction of overproducers of the enzyme.", *Biochemistry*, (1984). **23**: p.4470-4475.

20. Millar, G. and Coggins, J.R., "The complete amino acid sequence of 3-dehydroquinate synthase of *Escherichia coli*.", *FEBS Letters*, (1986). **200**: p.11-17.

21. Gaertner, F.H. and Cole, K.W., "A cluster gene: evidence for 1 gene, 1 polypeptide, 5 enzymes.", *Biochem. Biophys. Res. Commun.*, (1977). **75**: p.259-264.

22. Hanson, K.R. and Rose, I.A., "The absolute stereochemical course of citric acid biosynthesis.", *Proceedings of the National Academy of Science, USA.*, (1963). **50**: p.981-988.

23. Deka, R.K., Kleanthous, C., and Coggins, J.R., "Identification of the essential histidine residue at the acvtive site of *Escherichia coli* dehydroquinase.", *Journal of Biological Chemistry*, (1992). **267**: p.22237-22242.

24. Chaudhuri, S., Duncan, K., Graham, L.D., and Coggins, J.R., "Identification of the active-site lysine residue of two biosynthetic 3-dehydroquinases.", *Biochemical Journal*, (1991). **275**: p.1-6.

25. Kleanthous, C. and Coggins, J.R., "Reversible alkylation of an active site methionine residue in dehydroquinase.", *Journal of Biological Chemistry,* (1990). **265**(7): p.10935-10939.

26. Kleanthous, C., Campbell, D.G., and Coggins, J.R., "Active site labelling of the shikimate pathway enzyme dehydroquinase: evidence for a common substrate binding site within dehydoquinase and dehydroquinate synthase.", *Journal of Biological Chemistry,* (1990). **265**(7): p.10929-10934.

27. Chaudhuri, S., Lambert, J.M., McColl, L.A., and Coggins, J.R., "Purification and characterization of 3-dehydroquinase from *Escherichia coli.*", *Biochemical Journal,* (1986). **239**: p.699-704.

28. Mousedale, D.M. and Coggins, J.R., "3-Phosphoshikimate 1-carboxyvinyltransferase from *Pisum sativum.*", in *Methods in Enzymology*, R.F. Doolittle, Editor. (1987), Academic Press Inc: San Diego, California, USA. p.348-354.

29. Giles, N.H., Case, M.E., Baum, J., Geever, R., Huiet, L., Patel, V.B., and Tyler, B., "Gene organization and regulation in the *qa* (quinic acid) gene cluster of *Neurospora crassa.*", *Microbiology Review,* (1985). **49**: p.338-358.

30. Hawkins, A.R., Giles, N.H., and Kinghorn, J.R., "Genetical and biochemical aspects of quinate breakdown in the filamentous fungus of *Aspergillus nidulans*", *Biochem. Genet.,* (1982). **20**: p.271-286.

31. Da Silva, A.J.F., Whittington, H., Clements, J., Roberts, C., and Hawkins, A.R., "Sequence analysis and transformation by the catabolic 3-dehydroquinase (*qute*) gene from *Aspergillus nidulans.*", *Biochemical Journal,* (1986). **240**: p.481-488.

32. Polley, L.D., "Purification and characterization of 3-dehydroquinate hydrolase and shikimate oxidoreductase.", *Biochemica et Biophysica Acta,* (1978). **526**: p.259-266.

33. Koshiba, T., "Purification of two forms of the associated 3-dehydroquinate hydro-lyase and shikimate NADP oxidoreductase in phaseous mungo seedlings.", *Biochemica et Biophysica Acta,* (1978). **522**: p.10-18.

34. Chaudhuri, S. and Coggins, J.R., "The purification of shikimate dehyrogenase from *Escherichia coli*.", *Biochemical Journal,* (1985). **226**(1): p.217-223.

35. Anton, I.A. and Coggins, J.R., "Sequencing and overexpression of the *Escherichia coli aroE* gene encoding shikimate dehydrogenase.", *Biochemical Journal,* (1988). **249**(2): p.319-326.

36. Pittard, A.J., "Biosynthesis of the aromatic aminno acids", in *Escherichia coli and Salmonella typhimurium: cellular and molecular biology*, F.C. Neidhardt, Editor. (1987), American Society for Microbiology: Washington D.C. p.368-394.

37. Whipp, M.J. and Pittard, A.J., "A reassessment of the relationship between *aroK* and *aroL* encoded shikimate kinase enzymes of *Escherichia coli*.", *Journal of Bacteriology*, (1995). **177**: p.1627-1629.

38. Hawkes, T.R., Lewis, T., Coggins, J.R., Mousedale, D.M., Lowe, D.J., and Thorneley, R.N.F., "Chorismate synthase, pre-steady state kinetics of phosphate release from 5-enolpyruvylshikimate 3-phosphate.", *Biochemistry*, (1990). **265**: p.899-902.

39. Balasubramanian, S., Abell, C., and Coggins, J.R., "Observation of an isotope effect in the chorismate synthase reaction.", *Journal of American Chemistry Society*, (1990). **112**: p.8581-8583.

40. Blundell, T.L., Sibanda, B.L., Sternberg, M.J.E., and Thornton, J.M., "Knowledge-based prediction of protein structures and the design of novel molecules.", *Nature*, (1987). **326**: p.347-352.

41. Ripka, W.C., "Computer-assisted model building.", *Nature*, (1986). **321**: p.93-94.

42. Jarvis, A.J., Munro, S.L.A., and Craik, D.J., "Homology model of thyroxine binding globulin and elucidation of the thyroid hormone binding site.", *Protein Engineering*, (1992). **5**: p.61-67.

43. Browne, W.J., North, A.C.T., Phillips, D.C., Brew, K., Vanaman, T.C., and Hill, R.L., "A possible three-dimensional structure of bovine α-lactalbumin based on that of hen's egg-white lysozyme.", *Journal of Molecular Biology*, (1969). **42**: p.65-86.

44. Rees, A.R. and de la Paz, A., "Investigating antibody specificity using computer graphics and protein engineering", *Trends in Biochemical Science*, (1986). **11**: p.144-148.

45. Bajaj, M. and Blundell, T.L., "Evolution and the tetiary structure of proteins.", *Annual Review of Biophysics and Bioengineering*, (1984). **13**: p.453-492.

46. Kamphuis, I.G., Drenth, J., and Baker, E.N., "Thiol Proteases. Comparative studies based on the high resolution structures of papain and actinidin, and on amino acid sequence information for cathepsins B and H, and Stem Bromelain.", *Journal of Molecular Biology*, (1985). **182**: p.317-329.

47. Ohlsson, I., Nordstrom, B., and Branden, C.I., "Structural and functional similarities within the coenzyme binding domains of dehydrogenases.", *Journal of Molecular Biology*, (1974). **89**: p.339-354.

48. Greer, J., "Comparative model-building of the mammalian serine proteases.", *Journal of Molecular Biology*, (1981). **153**: p.1027-1042.

49. Greer, J., "Comparative modelling methods: application to the family of the mammalian serine proteases.", *Proteins: Structure, Function and Genetics.*, (1990). **7**: p.317-334.

50. Milner-White, E.J., Coggins, J.R., and Anton, I.A., "Evidence for an ancestral core structure in nucleotide-binding proteins with a type-A motif.", *Journal of Molecular Biology,* (1991). **221**: p.751-754.

51. Schulz, G.E., Schiltz, E., Tomasselli, A.G., Frank, R., Brune, M., Wittinghofer, A., and Schirmer, R.H., "Structural relationships in the adenylate kinase family.", *Journal of Biochemistry,* (1986). **161**: p.127-132.

52. Pascarella, S. and Argos, P., "Analysis of insertions/deletions in protein structures.", *Journal of Molecular Biology,* (1992). **224**: p.461-471.

53. Perutz, M.F., Kendrew, J.C., and Watson, H.C., "Structure and function of haemoglobins: II. some relations between polypeptide chain configuration and amino acid sequence.", *Journal of Molecular Biology,* (1965). **13**: p.669-678.

54. Altschuh, D., Lesk, A.M., Bloomer, A.C., and Klug, A., "Correlation of co-ordinated amino acid substitutions with function in viruses related to tobacco mosaic virus.", *Journal of Molecular Biology,* (1987). **193**: p.693-709.

55. Altschuh, D., Vernet, T., Berti, P., Moras, D., and Nagai, K., "Coordinated amino acid changes in homologous protein families.", *Protein Engineering,* (1988). **2**: p.193-199.

56. Chothia, C. and Lesk, A.M., "Evolution of proteins formed by β-sheets. I. Plastocyanin and azurin.", *Journal of Molecular Biology,* (1982). **160**: p.309-323.

57. Lesk, A.M. and Chothia, C., "Evolution of proteins formed by β-sheets. II. The core of the immunoglobulin domains.", *Journal of Molecular Biology,* (1982). **160**: p.325-342.

58. Lesk, A.M. and Chothia, C., "How different amino acid sequences determine similar protein structures: the structure and evolution dynamics of the globins.", *Journal of Molecular Biology,* (1980). **136**: p.225-270.

59. Chothia, C., Lesk, A.M., Levitt, M., Amit, A.G., Mariuzza, R.A., V., P.S.E., and Poljak, R.J., "The predicted structure of immunoglobin D1.3 and its comparison with the crystal structure.", *Science,* (1986). **233**: p.755-758.

60. Struthers, R.S., Kitson, D.H., and Hagler, A.T., "Predicted three-dimensional structure of the protease inhibitor domain of the Alzheimer's disease β-Amyloid precursor.", *Proteins: Structure, Function and Genetics..* (1991). **9**: p.1-11.

61. Greer, J., "Model structure for the inflammatory protein C5a.", *Science,* (1985). **228**: p.1055-1060.

62. Reid, L.S. and Thornton, J.M., "Rebuilding flavodoxin from C_α coordinates: a test study.", *Proteins: Structure, Function and Genetics,* (1989). **5**: p.170-182.

63. Delbaere, L.T.J., Brayer, G.D., and James, M.N.G., "Comparison of the predicted model of α-lytic protease with the X-ray structure.", *Nature,* (1979). **279**: p.165-168.

64. Henneke, C.M., Danson, M.J., Hough, D.W., and Osguthorpe, D.J., "Sequence alignment of citrate synthase proteins using a multiple sequence alignment algorithm and multiple scoring matrices.", *Protein Engineering,* (1989). **2**: p.597-604.

65. O'Neil, K. and De Grado, W.F., "A predicted structure of calmodulin suggests an electrostatic basis for its function.", *Proceedings of the National Academy of Science, USA.,* (1985). **82**: p.4954-4958.

66. Sutcliffe, M. and Haneef, I., "Knowledge based modelling of sperm whale myoglobin.", *Information Quarterly for Protein Crystallography,* (1986). **18**: p.11-18.

67. Venkatachalam, C.M., "Stereochemical criteria for polypeptides and proteins. V. Conformation of a system of three linked peptide units.", *Biopolymers,* (1968). **6**: p.1425-1436.

68. Lewis, P.N., Momany, F.A., and Scherenga, H.A., "Chain reversals in proteins.", *Biochimca et Biophysica Acta,* (1973). **303**: p.211-229.

69. Richardson, J.S., "The anatomy and taxonomy of protein structure.", *Advances in Protein Chemistry,* (1981). **34**: p.167-339.

70. Wilmot, C.M. and Thornton, J.M., "β-turns and their distortions: a proposed new nomenclature.", *Protein Engineering,* (1990). **3**: p.479-493.

71. Smith, J.A. and Pease, L.G., "Reverse turns in peptides and proteins.", *CRC Critical Review of Biochemistry and Molecular Biology,* (1980). **8**: p.315-399.

72. Milner-White, E.J. and Poet, R., "Loops, bulges, turns and hairpins in proteins.", *Trends in Biochemical Science,* (1987). **12**: p.189-192.

73. Milner-White, E.J. and Poet, R., "Four classes of β-hairpins in proteins.", *Biochemical Journal,* (1986). **240**: p.289-292.

74. Sibanda, B.L., Blundell, T.L., and Thornton, J.M., "Conformation of β-hairpins in protein structures: a systematic classification with applications to modelling by homology, electron density fitting and protein engineering.", *Journal of Molecular Biology,* (1989). **206**: p.759-777.

75. Sibanda, B.L. and Thornton, J.M., "β-hairpin families in globular proteins.", *Nature,* (1985). **316**: p.170-174.

76. Blundell, T.L., Barlow, D., Sibanda, B.L., Thornton, J.M., Taylor, W.R., Tickle, I.J., Sternberg, M.J.E., Pitts, J.E., Haneef, I., and Hemmings, A.M., "Three-dimensional structural aspects of the design of new protein molecules.", *Phil. Trans. Royal. Soc. (London),* (1986). **A317**: p.333-344.

77. Efimov, A.V., "Pseudo-homology of protein standard structures formed by two consectutive β-strands.", *FEBS Letters,* (1987). **224**: p.372-376.

78. Richardson, J.S., Getzoff, E.D., and Richardson, D.C., "The β bulge: a common small unit of non-repetitive protein structure.", *Proceedings of the National Academy of Science, USA.,* (1978). **75**: p.2574-2578.

79. Poet, R. and Milner-White, E.J., "Displaying relevant features of protein molecules.", *Computer Graphics Forum,* (1986). **5**: p.211-215.

80. Milner-White, E.J., "Beta-bulges within loops as recurring features of protein structure.", *Biochemica et Biophysica Acta,* (1987). **911**: p.261-265.

81. Milner-White, E.J., "Recurring loop motif in proteins that occur in right-handed and left-handed forms: its relationship with alpha-helices and beta-bulge loops.", *Journal of Molecular Biology,* (1988). **199**: p.503-511.

82. Schellman, C., "The α_L conformation at the end of helices.", in *Protein Folding*, R. Jaenicke, Editor. (1980), Elsevier, North Holland: Amsterdam. p.53-61.

83. Efimov, A.V., "A novel super-secondary structure of proteins and the relation between the structure and the amino acid sequence.", *FEBS Letters,* (1984). **166**: p.33-38.

84. Efimov, A.V., "Structure of α-α-hairpins with short connections.", *Protein Engineering,* (1991). **4**: p.245-250.

85. Edwards, M.S., Sternberg, M.J.E., and Thornton, J.M., "Structural and sequence patterns in the loops of βαβ units.", *Protein Engineering,* (1987). **1**: p.173-181.

86. Leszczynski, J.F. and Rose, G.D., "Loops in globular proteins: a novel category of secondary structure.", *Science,* (1986). **234**: p.849-855.

87. Claessens, M., Cutsem, E.V., Lasters, I., and Wodak, S., "Modelling the polypeptide backbone with 'spare parts' from known protein structures.", *Protein Engineering,* (1989). **2**: p.335-345.

88. Blundell, T.L., Carney, D., Gardner, S., Hayes, F., Howlin, B., Hubbard, T., Overington, J., Singh, D.A., Sibanda, B.L., and Sutcliffe, M., "Knowledge-based protein modelling and design.", *Eur. J. Biochem.,* (1988). **172**: p.513-520.

89. Jones, T.A. and Thirup, S., "Using known substructures in protein model building and crystallography.", *EMBO Journal,* (1986). **5**: p.819-822.

90. Moult, J. and James, M.N.G., "An algorithm for determining the conformation of polypeptide segments in proteins by systematic search.", *Proteins: Structure, Function and Genetics,* (1986). **1**: p.146-163.

91. Robson, B. and Platt, E., "Refined models for computer calculations in protein engineering; calibration and testing of atomic potential functions compatible with more efficent calculations.", *Journal of Molecular Biology,* (1986). **188**: p.259-281.

92. Crippen, G.M. and Havel, T.F., "An algorithm for determining the conformation of polypeptide segments in proteins by systematic search.", *Acta Crystallographica,* (1978). **A34**: p.282-284.

93. Tramontano, A., Chothia, C., and Lesk, A.M., "Structural determinants of the conformations of medium-sized loops in proteins.", *Proteins: Structure, Function and Genetics.,* (1989). **6**: p.382-394.

94. McGregor, M.J., Islam, S.A., and Sternberg, M.J.E., "Analysis of the relationship between side-chain conformation and secondary structure in globular proteins.", *Journal of Molecular Biology,* (1987). **198**: p.295-310.

95. Janin, J., Wodak, S., Levitt, M., and Maigret, B., "Conformation of amino acid side-chains in proteins.", *Journal of Molecular Biology.,* (1978). **125**: p.357-386.

96. Ponder, J.W. and Richards, F.M., "Tertiary templates for proteins; use of packaging criteria in the enumeration of allowed sequences for different structural classes.", *Journal of Molecular Biology,* (1987). **193**: p.775-791.

97. Blundell, T.L., Barlow, D., Borkakoti, N., and Thornton, J., "Solvent-induced distortions and the curvature of α-helices.", *Nature,* (1983). **306**: p.281-283.

98. Sutcliffe, M.J., Hayes, F.R.F., and Blundell, T.L., "Knowledge based modelling of homologous proteins, part II: rules for the conformation of substituted sidechains.", *Protein Engineering,* (1987). **1**: p.385-392.

99. Taylor, W.E., "Protein structure prediction.", in *Nucleic acid and protein sequence analysis.*, M.J. Bishop and C.J. Rawlings, Editors. (1987), I.R.L. Press: Oxford. p.285-322.

100. Hemmings, A.M., Foundling, S.I., Sibanda, B.L., Wood, S.P., Pearl, L.H., and Blundell, T.L., "Energy calculations on aspartic proteinases: human renin, endothiapepsin and its complex with an angiotensinogen fragment analogue, H-142.", *Biochemical Society Transactions,* (1985). **13**: p.1036-1041.

101. Shih, H.L., Brady, J., and Karplus, M., "Structure of proteins with single-site mutations: a minimum perturbation approach.", *Proceedings of the National Academy of Science, U.S.A.,* (1985). **82**: p.1697-1700.

102. Hagler, A.T. and Moult, J., "Computer simulation of the solvent structure around biological macromolecules.", *Nature,* (1978). **272**: p.222-226.

103. Chothia, C. and Lesk, A.M., "The relation between the divergence of sequence and structure in proteins.", *Embo Journal,* (1986). **5**: p.823-826.

104. Novotny, J., Rashin, A.A., and Bruccoleri, R.E., "Criteria that discriminate between native proteins and incorrectly folded models.", *Proteins: Structure, Functions and Genetics,* (1988). **4**: p.19-30.

105. Novotny, J., Bruccoleri, R., and Karplus, M., "An analysis of incorrectly folded protein models; implications for structure predictions.", *Journal of Molecular Biology,* (1984). **177**: p.787-818.

106. Milik, M., Kolinski, A., and Skolnick, J., "Neural network system for the evaluation of side-chain packing in protein structures.", *Protein Engineering,* (1995). **8**(3): p.225-236.

107. Laskowski, R.A., MacArthur, M.W., Moss, D.S., and Thornton, J.M., "Procheck: a program to check the stereochemical quality of protein structure.", *Journal of Applied Crystallography,* (1993). **26**: p.283-291.

108. Morris, A.L., MacArthur, M.W., Hutchinson, E.G., and Thornton, J.M., "Stereochemical quality of protein structure coordinates.", *Proteins: Structure, Function and Genetics,* (1992). **12**: p.345-364.

109. La Paz, P., Sutton, B.R., Darsley, M.J., and Rees, A.R., "Modelling of the combining sites of three anti-lysozyme monoclonal antibodies and of the complex between one of the antibodies and its epitope.", *EMBO Journal,* (1986). 5: p.415-425.

110. Bairoch, A., "Prosite: a dictionary of sites and patterns in proteins.", *Nucleic Acid Research,* (1991). **19**: p.2241-2245.

111. Hodgman, T.C., "The elucidation of protein function by sequence motif analysis.", *Computer Applications in the Biosciences,* (1989). **5**: p.1-13.

112. Kyte, J. and Dolittle, R.F., "A simple method for displaying the hydropathic character of a protein.", *Journal of Molecular Biology,* (1982). **157**: p.105-132.

113. Eisenberg, D., Sweet, R.M., and Terwilliger, T.C., "The hydrophobic moment detects periodicity in protein hydrophobicity.", *Proceedings of the National Academy of Science, USA.,* (1984). **81**: p.140-144.

114. Hopp, T.P. and Woods, K.R., "Prediction of protein antigenic determinants from amino acid sequences.", *Proceedings of the National Academy of Science, USA.*, (1981). **78**: p.3824-3828.

115. Karplus, P.A. and Schulz, G.E., "Prediction of chain flexibility in proteins.", *Naturwissenschaften*, (1985). **72**: p.212-213.

116. Winter, G. and Fersht, A.R., "Engineering enzymes.", *Trends in Biotechnology*, (1984). **2**: p.115-119.

117. Argos, P., "Analysis of sequence-similar pentapeptides in unrelated protein tertiary structures. Strategies for protein folding and a guide for site-directed mutagenesis.", *Journal of Molecular Biology*, (1987). **197**: p.331-348.

118. Seto, Y., Ikeuchi, Y., and Kanehisa, M., "Fragment peptide library for classification and functional prediction of proteins.", *Proteins: Structure, Function and Genetics*, (1990). **8**: p.341-351.

119. Bugg, T.D.H., Alefounder, P.R., and Abell, C., "An amino acid sequence motif observed amongst enzymes of the shikimate pathway.", *Biochemical Journal*, (1991). **276**: p.841-843.

120. Benner, S.A. and Gerloff, D., "Patterns of divergence in homologous proteins as indicators of secondary and tertiary structure: a prediction of the structure of the catalytic domain of protein kinase.", *Advances in Enzyme Regulation*, (1990). **31**: p.121-181.

121. Bowie, J.U., Clarke, N.D., Pabo, C.O., and Sauer, R.T., "Identification of protein folds: matching hydrophobicity patterns of sequence sets with solvent accessibility patterns of known structures.", *Proteins: Structure, Function and Genetics.*, (1990). **7**: p.257-264.

122. Niermann, T. and Kirschner, K., "Improving the prediction of secondary structure of 'TIM-barrel' enzymes.", *Protein Engineering*, (1990). **4**: p.137-147.

123. Crawford, I.P., Niermann, T., and Kirschner, K., "Prediction of secondary structure by evolutionary comparison: application to the α-subunit of tryptophan synthase.", *Proteins: Structure, Function and Genetics*, (1987). **1**: p.118-129.

124. Nishikawa, K. and Ooi, T., "Amino acid sequence homology applied to the prediction of protein secondary structures, and joint predictions with existing methods.", *Biochemica et Biophysica Acta.*, (1986). **871**: p.45-54.

125. Nelson, D.R. and Strodel, H.W., "Secondary structure predictions of 52 membrane-bound cytochromes P450 shows a strong structural similarity to P450cam.", *Biochemistry*, (1989). **28**: p.656-660.

126. Benner, S.A., "Patterns of divergence in homologous proteins as indicators of tertiary and quaternary structure.", *Advances in Enzyme Regulation,* (1989). **28**: p.219-236.

127. Zvelebil, M.J., Barton, G.J., Taylor, W.R., and Sternberg, M.J.E., "Prediction of protein secondary structure and active sites using the alignment of homologous sequences.", *Journal of Molecular Biology,* (1987). **195**: p.957-961.

128. Zvelebil, M.J. and Sternberg, M.J.E., "Analysis and prediction of the location of catalytic residues in enzymes.", *Protein Engineering,* (1988). **2**(2): p.127-138.

129. Czelusniak, J., Goodman, M., Moncrieff, N.D., and Kehoe, S.M., "Maximum parsimony approach to construction of evolutionary trees from aligned homologous sequences.", in *Methods in Enzymology*, R.F. Doolittle, Editor. (1990), Academic Press Inc.: San Diego, California, USA. p.601-615.

130. Feng, D.-F. and Doolittle, R.F., "Progressive alignment and phylogenetic tree construction of protein sequences.", in *Methods in Enzymology.*, R.F. Doolittle, Editor. (1990), Academic Press Inc.: San Diego, California, USA. p.375-387.

131. Hein, J., "Unified approach to alignment and phylogenies.", in *Methods in Enzymology*, R.F. Doolittle, Editor. (1990), Academic Press Inc.: San Diego, California, USA. p.626-645.

132. Chou, P.Y. and Fasman, G.D., "Prediction of proteins conformation.", *Biochemistry,* (1974). **13**: p.222-245.

133. Chou, P.Y. and Fasman, G.D., "Conformational paramters for amino acids in helical, β-sheet and random coil regions calculated from proteins.", *Biochemistry,* (1974). **13**: p.211-222.

134. Chou, P.Y. and Fasman, G.D., "Prediction of the secondary structure of proteins from their amino acid sequence.", *Advances in Enzymology,* (1978). **47**: p.45-148.

135. Garnier, J., Osguthorpe, D.J., and Robson, B., "Analysis of the accuracy and implications of simple methods for predicting the secondary structure of globular proteins.", *Journal of Molecular Biology,* (1978). **120**: p.97-120.

136. Ralph, W.W., Webster, T., and Smith, T.F., "A modified Chou and Fasman protein structure algorithm.", *Computer Applications in the Biosciences,* (1987). **3**: p.211-216.

137. Gibrat, J.F., Garnier, J., and Robson, B., "Further developments of protein secondary structure prediction using information theory: new parameters and consideration of residue pairs.", *Journal of Molecular Biology,* (1987). **198**: p.425-443.

138. Mrazek, J.A. and Kypr, J., "Computer program Jamsek combing statistical and stereochemical rules for the prediction of protein secondary structures.", *Computer Applications in the Biosciences,* (1988). **4**(2): p.297-302.

139. Quian, N. and Sejnowski, T.J., "Predicting the secondary structure of globular proteins using neural network models.", *Journal of Molecular Biology,* (1988). **202**: p.865-884.

140. Pascarella, S. and Bossa, F., "PRONET: a microcomputer program for predicting the secondary structure of proteins with a neural network.", *Computer Applications in the Biosciences,* (1989). **5**: p.319-320.

141. Holley, H.L. and Karplus, M., "Protein secondary structure prediction with a neural network.", *Proceedings of the National Academy of Science, USA.,* (1989). **86**: p.152-156.

142. Kneller, D.G., Cohen, F.E., and Langridge, R., "Improvements in protein secondary structure prediction by an enhanced neural network.", *Journal of Molecular Biology,* (1990). **214**: p.171-182.

143. Sasagawa, F. and Tajima, K., "Prediction of protein structures by a neural network", *Computer Applications in the Biosciences,* (1993). **9**(2): p.147-152.

144. Rost, B. and Sander, C., "Combining evolutionary information and neural networks to predict protein secondary structure.", *Proteins: Structure, Function and Genetics,* (1994). **19**: p.55-77.

145. Deleage, G. and Roux, B., "An algorithm for protein secondary structure prediction based on class prediciton.", *Protein Engineering,* (1987). **1**: p.289-294.

146. Cohen, F.E., Abarbanel, R.M., Kuntz, I.D., and Fletterick, R.J., "Secondary structure assignment for α/β proteins by a combinatorial approach.", *Biochemistry,* (1983). **22**: p.4895-4904.

147. Lim, V.I., "A mechanism for protein folding. A stereochemical theory of the tertiary structure of globular proteins.", *Journal of Molecular Biology,* (1974). **88**: p.857-872.

148. Ptitsyn, O.B. and Finkelstein, A.V., "Prediction of protein secondary structure based on physical theory. Histones.", *Protein Engineering,* (1989). **2**: p.443-447.

149. Biou, V., Gibrat, J.F., Levin, J.M., Robson, B., and Garnier, J., "Secondary structure prediction: combination of three different methods.", *Protein Engineering,* (1988). **2**: p.185-191.

150. Levin, J.M., Robson, B., and Garnier, J., "An algorithm for secondary structure determination in proteins based on sequence similarity.", *FEBS Letters,* (1986). **205**: p.303-308.

151. Asai, K., Hayamizu, S., and Handa, K., "Prediction of protein secondary structure by the hidden Markov model.", *Computer Applications in the Biosciences,* (1993). **9**(2): p.141-146.

152. McGregor, M.J., Flores, T.P., and Sternberg, M.J.E., "Prediction of β-turns in proteins using neural networks.", *Protein Engineering,* (1989). **2**: p.521-526.

153. Chou, P.Y. and Fasman, G.D., "Prediction of β-turns.", *Biophysical Journal,* (1979). **26**: p.367-384.

154. Cohen, F.E., Abarbanel, R.M., Kuntz, I.D., and Fletterick, R.J., "Turn prediction in proteins using a pattern-matching approach.", *Biochemistry,* (1986). **25**: p.266-275.

155. Kabsch, W. and Sander, S., "How good are predictions of protein secondary structure?", *FEBS Letters,* (1983). **155**: p.179-182.

156. Nishikawa, K., "Assessment of secondary-structure prediction of proteins comparison of computerized Chou-Fasman method with others.", *Biochemica et Biophysica Acta.,* (1983). **748**: p.285-299.

157. Yada, R.Y., Jackman, R.L., and Nakai, S., "Secondary structure prediction and determination of proteins - a review.", *International Journal of Peptide and Protein Research.,* (1988). **31**: p.98-109.

158. Klein, P., "Prediction of protein structural class by discriminant analysis.", *Biochemica et Biophysica Acta,* (1986). **874**: p.205-215.

159. Klein, P. and Delisi, C., "Prediction of protein structural class from the amino acid sequence.", *Biopolymers,* (1986). **25**: p.1659-1672.

160. Nakashima, H., Nishikawa, K., and Ooi, T., "The folding type of a protein is relevant to the amino acid composition.", *Journal of Biochemistry (Tokyo),* (1986). **99**: p.153-162.

161. Reczko, M. and Bohr, H., "The DEF database of sequence based protein fold class predictions.", *Nucleic Acid Research,* (1994). **22**: p.3616-3619.

162. Clark, D.A., Shirazi, J., and Rawlings, C.J., "Protein topology prediction through constraint-based search and the evaluation of topological folding rules.", *Protein Engineering,* (1991). **4**(7): p.751-760.

163. Sun, S., Thomas, P.D., and Dill, K.A., "A simple protein folding algorithm using binary code and secondary structure constraints.", *Protein Engineering,* (1995). **8**(8): p.769-778.

164. Cohen, F.E., Sternberg, M.J.E., and Taylor, W.R., "Analysis and prediction of the packing of α-helices against a β-sheet in the tertiary structure of globular proteins.", *Journal of Molecular Biology,* (1982). **156**: p.821-862.

165. Chothia, C., "The classification and origins of protein folding patterns.", *Annual Reiew of Biochemistry,* (1990). **59**: p.1007-1039.

166. Lesk, A.M., Branden, C.I., and Chothia, C., "Structural principles of α/β barrel proteins: the packing of the interior of the sheet.", *Proteins: Structure, Function and Genetics.*, (1989). **5**: p.139-148.

167. Rice, P.A., Goldman, A., and Steitz, T.A., "A helix-turn-strand structural motif common in α-β proteins.", *Proteins: Structure, Function and Genetics*, (1990). **8**: p.334-340.

168. Chou, K., Carlacci, L., and Maggiora, G.G., "Conformational and geometrical properties of idealized β-barrels in proteins.", *Journal of Molecular Biology*, (1990). **213**: p.315-326.

169. Lifson, S. and Sander, C., "Specific recognition in the tertiary structure of β-sheets of proteins.", *Journal of Molecular Biology*, (1980). **139**: p.627-639.

170. Segrest, J.P., De Loof, H., Dohlman, J.G., Brouillette, C.G., and Anantharamaiah, G.M., "Amphipathic helix motif: classes and properties.", *Proteins: Structure, Function and Genetics*, (1990). **8**: p.103-117.

171. Zhu, Z., "A new approach to the evaluation of protein secondary structure predictions at the level of the elements of secondary structure.", *Protein Engineering*, (1995). **8**(2): p.103-108.

172. Serrano, L., Neira, J.-L., Sancho, J., and Fersht, A.R., "Effect of alanine versus glycine in alpha-helices on protein stability.", *Nature*, (1992). **356**: p.453-455.

173. Chakrabartty, A., Schellman, J.A., and Baldwin, R.L., "Large differences in the helix propensities of alanine and glycine.", *Nature*, (1991). **351**: p.586-588.

174. Richardson, J.S. and Richardson, D.C., "Amino acid preferences for specific locations at the end of α helices.", *Science*, (1988). **240**: p.1648-1652.

175. Rooman, M.J., Rodriguez, J., and Wodak, S.J., "Relations between protein sequence and structure and their significance.", *Journal of Molecular Biology*, (1990). **213-337**: p.350.

176. Wierenga, R.K., Terpstra, P., and Hol, W.G.J., "Prediction of the occurence of the ADP-binding βαβ-fold in proteins, using an amino acid sequence fingerprint.", *Journal of Molecular Biology*, (1986). **187**: p.101-107.

177. Kneale, G.G. and Bishop, M.J., "Nucleic acid and protein sequence databases.", *Computer Applications in the Biosciences Rev.*, (1985). **1**: p.11-17.

178. Devereaux, J., Haeberli, P., and Smithies, O., "A comprehensive set of sequence and programmes for the VAX.", *Nucleic Acids Research*, (1984). **12**(1): p.387-395.

179. Akrigg, D., Bleasby, A.J., Dix, N.J.M., Findlay, J.B.C., North, A.C.T., Parry-Smith, D.J., Wootton, J.C., Blundell, T.L., Gardner, S.P., Hayes, F., Islam, S., Sternberg, M.J.E., Thornton, J.M., Tickle, I.J., and Murray-Rust, P., "A protein sequence/structure database.", *Nature*, (1988). **345**: p.745-746.

180. Taylor, W.R., "Pattern matching methods in protein sequence comparison and structure prediction.", *Protein Engineering*, (1988). **2**: p.77-86.

181. Doolittle, R.F., Feng, D.F., Johnson, M.S., and McClure, M.A., "Relationship of human protein sequences to those of other organisms.", *Cold Spring Harbor Symposia on Quantitative Biology*, (1986). **LI**: p.447-455.

182. Cheever, E.A., Overton, G.C., and Searls, D.B., "Fast Fourier transform-based correlation of DNA sequences using complex plane encoding.", *Computer Applications in the Biosciences*, (1991). **7**(2): p.143-154.

183. Streletc, V.B., Shindyalov, I.N., Kolchanov, N.A., and Milanesi, L., "Fast, statistically based alignment of amino acid sequences on the base of diagonal fragments of dot-matrices.", *Computer Applications in the Biosciences*, (1992). **8**(6): p.529-534.

184. Needleman, S.B. and Wunsch, C.D., "A general method applicable to the search for similarities in the amino acid sequence of two proteins.", *Journal of Molecular Biology*, (1970). **48**: p.443-453.

185. Sellers, P.H., "On the theory and computation of evolutionary distances.", *SIAM Journal on Applied Mathematics*, (1974). **26**: p.787-793.

186. Waterman, M.S., Smith, T.F., and Beyer, W.A., "Some biological sequence metrics.", *Advances in Mathematics.*, (1976). **20**: p.367-387.

187. Murata, M., "Three-way Needleman-Wunsch algorithm.", in *Methods in Enzymology*, R.F. Doolittle, Editor. (1990), Academic Press Inc.: San Diego, California, USA. p.365-375.

188. Smith, T.F. and Waterman , M.S., "Identification of Common Molecular Subsequences.", *Journal of Molecular Biology*, (1981). **147**: p.195-197.

189. Waterman, M.S. and Eggert, S., "A new algorithm for best subsequence alignments with application to tRNA-rRNA comparisons.", *Journal of Molecular Biology*, (1987). **197**: p.723-728.

190. Gotoh, O., "Pattern matching of biological sequences with limited storage.", *Computer Applications in the Biosciences*, (1987). **3**(1): p.17-20.

191. Barton, G.J., "An efficient algorithm to locate all locally optimal alignments between two sequence allowing for gaps.", *Computer Applications in the Biosciences,* (1993). **9**(6): p.729-734.

192. Boswell, D.R. and McLachlan, A.D., "Sequence comparison by exponetially-damped alignment.", *Nucleic Acid Research,* (1984). **12**(1): p.457-465.

193. Goad, W.B. and Kanehisa, M.I., "Pattern recognition in nucleic acid sequences. I. A general method for finding local homologies and symmetries.", *Nucleic Acid Research,* (1982). **10**: p.247-263.

194. Hall, J.D. and Myers, E.W., "A software tool for finding locally optimal alignments in protein and nucleic acid sequences.", *Computer Applications in the Biosciences.,* (1988). **4**(1): p.35-40.

195. Sellers, P.H., "Pattern recognition in genetic sequences by mismatch density.", *Bulletin of Mathematical Biology,* (1984). **46:** p.501-514.

196. Fitch, W.M. and Smith, T.F., "Optimal sequence alignments.", *Proceedings of the National Academy of Science, USA.,* (1983). **80**: p.1382-1386.

197. Waterman, M.S., "General methods of sequence comparison.", *Bulletin of Mathematical Biology,* (1984). **46:** p.473-500.

198. Gotoh, O., "An improved algorithm for matching biological sequences.", *Journal of Molecular Biology,* (1982). **162**: p.705-78.

199. Lesk, A.M., Levitt, M., and Chothia, C., "Alignment of the amino acid sequences of distantly related proteins using variable gap penalties.", *Protein Engineering,* (1986). **1**: p.77-78.

200. Barton, G.J. and Sternberg, M.J.E., "Evaluation and improvements in the automatic alignment of protein sequences.", *Protein Engineering,* (1987). **1**: p.89-94.

201. Taylor, P., "A fast homology program for aligning biological sequences.", *Nucleic Acid Research,* (1984). **12**(1): p.447-455.

202. Altschul, S.F. and Erickson, B.W., "Optimal sequence alignment using affine gap costs.", *Bulletin of Mathematical Biology.,* (1986). **48**(5/6): p.603-616.

203. Myers, E.W. and Miller, W., "Optimal alignments in linear space.", *Computer Applications in the Biosciences,* (1988). **4**: p.11-17.

204. Watanabe, K., Urano, Y., and Tamaoki, T., "Optimal alignments of biological sequences on a microcomputer.", *Computer Applications in the Biosciences,* (1985). **1**(2): p.83-87.

205. Ukkonen, E., "Algorithms for approximate string matching.", *Inform. Control,* (1985). **64**: p.10-118.

206. Fickett, J.W., "Fast optimal alignment.", *Nucleic Acid Research,* (1984). **12**: p.175-180.

207. Spounge, J.L., "Fast optimal alignment.", *Computer Applications in the Biosciences,* (1991). **7**(1): p.1-7.

208. Bacon, D.J. and Anderson, W.F., "Multiple sequence alignment.", *Journal of Molecular Biology,* (1986). **191**: p.153-161.

209. Feng, D.F., Johnson, M.S., and Doolittle, R.F., "Aligning amino acid sequences: comparison of commonly used methods.", *Journal of Molecular Evolution,* (1985). **21**: p.112-125.

210. Maclachlan, A.D., "Tests for comparing related amino-acid sequences. Cytochrome c and Cytochrome c551.", *Journal of Molecular Biology,* (1971). **61**: p.409-424.

211. Risler, J.L., Delorme, M.O., Delacroix, H., and Henaut, A., "Amino acid substitutions in structurally related proteins. A pattern recognition approach. Determination of a new and efficient scoring matrix.", *Journal of Molecular Biology,* (1988). **204**: p.1019-1029.

212. George, D.G., Barker, W.C., and Hunt, L.T., "Mutation data matrix and its uses.", in *Methods in Enzymology*, R.F. Doolittle, Editor. (1990), Academic Press Inc.: San Diego, California, USA. p.333-351.

213. Argos, P., "A sensitive procedure to compare amino acid sequences.", *Journal Molecular Biology,* (1987). **193**: p.385-396.

214. Kidera, A., Konishi, Y., Oka, M., Ooi, T., and Scheraga, H.A., "Statistical analysis of the physical properties of the twenty naturally occuring amino acids.", *Journal of Protein Chemistry,* (1985). **4**: p.23-55.

215. Nakai, K., Kidera, A., and Kanehisa, M., "Cluster analysis of amino acid indices for prediction of protein structure and function.", *Protein Engineering,* (1988). **2**(2): p.93-100.

216. Dayhoff, M.O., Schwartz, R.M., and Orcutt, B.C., "A model of evolutionary change in proteins.", in *Atlas of protein sequence and structure.*, M.O. Dayhoff, Editor. (1978), National Biomedical Research Foundation: Washington, DC. p.345- 352.

217. Dayhoff, M.O., Barker, W.C., and Hunt, L.T., "Establishing homologies in protein sequences.", in *Methods in Enzymology*, R.F. Doolittle, Editor. (1983), Academic Press Inc.: San Diego, California, USA. p.524-545.

218. Dayhoff, M.O. and Eck, R.V., "A model of evolutionary change in proteins.", in *Atlas of Protein Sequence and Structure*, M.O. Dayhoff, Editor. (1968), National Biomedical Research Foundation: Silver Spring, Maryland. p.33 - 41.

219. Jones, D.T., Taylor, W.R., and Thornton, J.M., "The rapid generation of mutation data matrices from protein sequences.", *Computer Applications in Biosciences*, (1992). **8**: p.275-282.

220. Kubota, Y., Nishikawa, O., Takahashi, S., and Ooi, T., "Correspondence of homologies in amino acid sequences and tertiary structure of protein molecules.", *Biochimica et Biophysica Acta*, (1982). **701**: p.242-252.

221. Kubota, Y., Takahashi, S., Nishikawa, O., and Ooi, T., "Homology in protein sequences expressed by correlation coefficients.", *Journal of Theoretical Biology*, (1981). **91**: p.347-361.

222. Rechid, R., Vingron, M., and Argos, P., "A new interactive protein sequence alignment program and comparison of its results with widely used algorithms.", *Computer Applications in the Biosciences*, (1989). **5**: p.107-113.

223. Wilbur, W.J. and Lipman, D.J., "Rapid similarity searches of nucleic acid and protein data banks.", *Proceedings of the National Academy of Science, USA*, (1983). **80**: p.726-730.

224. Dumas, J.P. and Ninio, J., "Efficient algorithms for folding and comparing nucleic acid sequences.", *Nucleic Acid Research*, (1982). **10**: p.197-206.

225. Lipman, D.J. and Pearson, W.R., "Rapid and sensitive protein similarity searches.", *Science*, (1985). **227**: p.1435-1441.

226. Bleasby, A., "Sweep - an extension of the Lipman and Pearson FASTP algorithm (Science [1985] 227, 1435-1441) for database searching, available on SEQNET, Daresbury, U.K.", . (1989).

227. Pearson, W.R. and Lipman, D.J., "Improved tools for biological sequence comparison.", *Proceedings of the National Academy of Science, USA*, (1988). **85**: p.2444-2448.

228. Mott, R.F., Kirkwood, T.B.L., and Curnow, R.N., "A test for the statistical significance of DNA sequence similarities in data-bank searches.", *Computer Applications in the Biosciences.*, (1989). **5**: p.123-131.

229. Mott, R.F. and Kirkwood, T.B.L., "STATSEARCH: a GCG-compatible program for assessing statistical significance during DNA and protein databank searches.", *Computer Applications in the Biosciences.*, (1990). **6**(3): p.292-295.

230. Altschul, S.F., Gish, W., Miller, W., Myers, E.W., and Lipman, D.J., "Basic Local Alignment Search Tool.", *Journal of Molecular Biology*, (1990). **215**(403-410).

231. Karlin, S. and Altschul, S.F., "Methods for assessing the statistical significance of molecular sequences features using general scoring schemes.", *Proceedings of the National Academy of Science, U.S.A.,* (1990). **87**: p.2264-2268.

232. Hopcroft, J.E. and Ullman, J.D., "Introduction to automata theory, languages and computations.", (1979), Reading, MA. USA.: Addison-Wesley. 42-45.

233. Gribskov, M., McLachlan, A.D., and Eisenberg, D., "Profile analysis: detection of distantly related proteins.", *Proceedings of the National Academy of Science, USA,* (1987). **84**: p.4355-4358.

234. Higgins, D.G. and Gouy, M., "Interfacing similarity search software with the sequence retrieval system ACNUC.", *Computer Applications in the Biosciences.,* (1987). **3**: p.239-241.

235. Collins, J.F., Coulson, A.F.W., and Lyall, A., "The significance of protein sequence similarities.", *Computer Applications in the Biosciences,* (1988). **4**: p.647-71.

236. Coulson, A.F.W., Collins, J.F., and Lyall, A., "Protein and nucleic acid sequence database searching: a suitable case for parallel processing.", *Computer Journal,* (1987). **30**(420-424).

237. Deshpande, A.S., Richards, D.S., and Pearson, W.R., "A platform for biological sequence comparison on parallel computers.", *Computer Applications in the Biosciences,* (1991). **7**(2): p.237-247.

238. Edmiston, E.W., Gore, N.G., Saltz, J.H., and Smith, R.M., "Parallel processing of biological sequence comparison algorithms.", *Int. J. Parallel Program.,* (1988). **17**: p.259-275.

239. Lander, E., Mesirov, J.P., and Taylor, W., "Study of protein sequence comparison metrics on the Connection Machine CM-2.", *Journal of Supercomputing,* (1989). **3**: p.225-269.

240. Vogt, G. and Argos, P., "Searching for distantly related protein sequences in large databases by parallel processing on a transputer machine.", *Computer Applications in the Biosciences,* (1992). **8**(1): p.49-55.

241. Miller, P.L., Nadkarni, P.M., and Carriero, N.M., "Parallel computation and FASTA: confronting the problem of parallel database search for a fast sequence comparison algorithm.", *Computer Applications in the Biosciences,* (1991). **7**(1): p.71-78.

242. Barton, G.J., "Scanning protein sequence databanks using a distributed processing workstation network.", *Computer Applications in the Biosciences,* (1991). **7**(1): p.85-88.

243. Dickerson, R.E., Timkovich, R., and Almassy, R.J., "The cytochrome fold and the evolution of bacterial energy metabolism.", *Journal of Molecular Biology,* (1976). **100**: p.473-491.

244. Vingron, M. and Argos, P., "Determination of reliable regions in protein sequence alignments.", *Protein Engineering,* (1990). **3**: p.565-569.

245. Altschul, S.F., "Gap costs for multiple sequence alignment.", *Journal of Theoetical Biology,* (1989). **138**: p.297-309.

246. Schwartz, R.M. and Dayhoff, M.O., "Matrices for detecting distant relationships.", in *Atlas of Protein Sequence and Structure.*, M.O. Dayhoff, Editor. (1978), National Biomedical Research Foundation: Washington, D.C. p.353-358.

247. Bell, L.H., Coggins, J.R., and Milner-White, E.J., "Mix'n'Match: an improved multiple sequence alignment procedure for distantly related proteins using secondary structure predictions, designed to be independant of the choice of gap penalty and scoring matrix.", *Protein Engineering,* (1993). **6**: p.683-690.

248. Henneke, C.M., "A multiple sequence alignment algorithm for homologous proteins using secondary structure information and optionally keying alignments to functionally important sites.", *Computer Applications in the Biosciences.* (1989). **5**: p.141-150.

249. Schuler, G.D., Altschul, S.F., and Lipman, D.J., "A workbench for multiple alignment construction and analysis.", *Proteins: Structure, Function and Genetics,* (1991). **9**: p.180-190.

250. Doolittle, R.F., "Similar amino acid sequences: chance or common ancestry?", *Science,* (1981). **214**: p.149-159.

251. McLachlan , A.D. and Boswell, D.R., "Confidence limits for homology in protein or gene sequences. The c-myc oncogene and adenovirus E1a protein.", *Journal of Molecular Biology,* (1985). **185**(39-49).

252. Saroff, H.A., "The uniqueness of protein sequences. Uniqueness diagrams for the dayhoff file.", *Bulletin of Mathemathical Biology,* (1984). **46**: p.661-672.

253. Smith, T.F., Waterman , M.S., and Burks, C., "The statistical distribution of nucleic acid similarities.", *Nucleic Acid Research,* (1985). **13**(2): p.645-656.

254. Barton, G.J. and Sternberg, M.J.E., "A strategy for the rapid multiple alignment of protein sequences: Confidence levels from tertiary structure comparisons.", *Journal of Molecular Biology,* (1987). **198**: p.327-337.

255. Argos, P., Vingron, M., and Vogt, G., "Protein sequence comparison: methods and significance.", *Protein Engineering,* (1991). **4**: p.375-383.

256. McCaldon, P. and Argos, P., "Oligopeptide biases in protein sequences and their use in predicting protein coding regions in nucleotide sequences.", *Proteins:Structure, Function and Genetics,* (1988). **4**(99-122).

257. Kabsch, W. and Sander, S., "On the use of sequence homologies to predict protein structure: identical pentapeptides can have completely different conformations.", *Proceedings of the National Academy of Science, U.S.A.,* (1984). **81**: p.1075-1078.

258. Wilson, I.A., Haft, D.H., Getzoff, E.D., Tainer, J.A., Lerner, R.A., and Brenner, S., "Identical short peptide sequences in unrelated proteins can have different conformations: a testing ground for theories of immune recognition.", *Proceedings of the National Academy of Science, USA.,* (1985). **82**: p.5255-5259.

259. Sidman, K.E., George, D.G., Barker, W.C., and Hunt, L.T., "The Protein Identification Resource (PIR).", *Nucleic Acid Research,* (1988). **11**: p.1869-1871.

260. Walker, J.E., Saraste, M., Runswick, M.J., and Gray, N.J., "Distantly related sequences in the α- and β-subunits of ATP synthse, myosin, kinases and other ATP-requiring enzymes and a common nucleotide binding fold.", *EMBO Journal,* (1982). **1**(8): p.945-951.

261. Saraste, M., Sibbald, P.R., and Wittinghofer, A., "The P loop - a common motif in ATP and GTP binding proteins.", *TIBS,* (1990). **15**: p.430-434.

262. Vingron, M. and Argos, P., "A fast and sensitive multiple sequence alignment algorithm.", *Computer Applications in the Biosciences,* (1989). **5**(2): p.115-121.

263. Feng, D.-F. and Doolittle, R.F., "Progressive sequence alignment as a prerequisite to correct phylogenetic trees.", *Journal of Molecular Evolution,* (1987). **25**: p.351-360.

264. Murata, M., Richardson, J.S., and Sussman, J.L., "Simultaneous comparison of three protein sequences.", *Proceedings of the National Academy of Science, U.S.A.,* (1985). **82**: p.3073-3077.

265. Johnson, M.S. and Doolittle, R.F., "A method for the simultaneous alignment of three or more amino acid sequences.", *Journal of Molecular Evolution,* (1986). **23**: p.267-278.

266. Carillo, H. and Lipman, D., "The multiple sequence alignment problem in biology", *SIAM Journal on Applied Mathematics,* (1988). **48**: p.1073-1082.

267. Lipman, D.J., Altschul, S.F., and Kececioglu, J.D., "A tool for multiple sequence alignment.", *Proceedings of the National Academy of Science, USA,* (1989). **86**: p.4412-4415.

268. Altschul, S.F. and Lipman, D.J., "Trees, stars, and multiple biological sequence alignment.", *SIAM Journal on Applied Mathematics,* (1989). **49**: p.197-209.

269. Sobel, E. and Martinez, H., "A multiple sequence alignment program.", *Nucleic Acids Research,* (1986). **14**(1): p.363-374.

270. Santibanez, M. and Rohde, K., "A multiple alignment program for protein sequences.", *Computer Applications in the Biosciences,* (1987). **3**: p.111-114.

271. Wilbur, W.J. and Lipman, D.J., "The context dependent comparison of biological sequences.", *SIAM Journal on Applied Mathematics,* (1984). **44**: p.557-567.

272. Chappey, C., Danckaert, A., Dessen, P., and Hazout, S., "MASH: an interactive program for multiple alignment and consensus sequence construction for biological sequences.", *Computer Applications in the Biosciences,* (1991). **7**: p.195-202.

273. Karp, R.M., Miller, R.E., and Rosenberg, A.L. "*Rapid identification of repeated patterns in strings, trees and arrays.* ", in *Proceedings of the 4th Annual ACM Symposium on Theory of Computing*. 1972.

274. Depericux, E. and Feyntman, E., "Simultaneous and multivariate alignment of protein sequences: correspondence between physicochemical profiles and structurally conserved regions (SCR).", *Protein Engineering,* (1991). **4**: p.603-613.

275. Vihinen, M., "Simultaneous comparison of several sequences.", in *Methods in Enzymology.*, R.F. Doolittle, Editor. (1990), Academic Press Inc.: San Diego, California, USA. p.447-456.

276. Vihinen, M., Euranto, A., Luostarinen, P., and Nevalainen, O., "MULTICOMP: a program package for multiple sequence comparison.", *Computer Applications in the Biosciences,* (1992). **8**(1): p.35-38.

277. Vingron, M. and Argos, P., "Motif recognition and alignment for many sequences by comparison of dot-matrices.", *Journal of Molecular Biology,* (1991). **218**: p.33-43.

278. Bains, W., "MULTAN: a program to align multiple DNA sequences.", *Nucleic Acid Research,* (1986). **14**(1): p.159-177.

279. Bains, W., "Multan (2), a multiple string alignment program for nucleic acids and proteins.", *Computer Applications in the Biosciences,* (1989). **5**: p.51-52.

280. Taylor, W.R., "Multiple sequence alignment by a pairwise algorithm", *Computer Applications in the Biosciences,* (1987). **3**: p.81-87.

281. Higgins, D.M. and Sharp, P.M., "CLUSTAL: A Package for Performing Multiple Sequence Alignment on A Microcomputer.", *Gene.,* (1988). **73**: p.237-244.

282. Higgins, D.M. and Sharp, P.M., "Fast and sensitive multiple sequence alignments on a microcomputer.", *Computer Applications in the Biosciences,* (1989). **5**: p.151-153.

283. Bleasby, A., based on the CLUSTAL algorithm of Higgins, D. M. and Sharp, P. M. CLUSTAL: a package for performing multiple sequence alignment on a microcomputer., Gene 73 (1988), p237-244 and available on the U.K. molecular biology SEQNET service, Daresbury, Cheshire, U.K.

284. Berger, M.P. and Munson, P.J., "A novel randomized iterative strategy for aligning multiple protein sequences.", *Computer Applications in the Biosciences,* (1991). **7**: p.479-484.

285. Smith, R.F. and Smith, T.F., "Pattern-induced multi-sequence alignment (PIMA) algorithm employing secondary structure-dependant gap penalties for use in comparative protein modelling.", *Protein Engineering,* (1992). **5**: p.35-41.

286. Thirup, S. and Larsen, N.E., "Alma - an editor for large sequences alignments.", *Proteins: Structure, Function and Genetics.,* (1990). **7**(3): p.291-295.

287. Parry-Smith, D.J. and Attwood, T.K., "SOMAP: a novel interactive approach to multiple protein sequnces alignment.", *Computer Applications in the Biosciences,* (1991). **7**(2): p.233-235.

288. Stockwell, P.A. and Peterson, G.B., "HOMED: a homologous sequence editor.", *Computer Applications in the Biosciences,* (1987). **3**: p.37-43.

289. Cabot, E.L. and Beckenbach, A.T., "Simultaneous editing of multiple nucleci acid and protein sequences with ESEE.", *Computer Applications in the Biosciences,* (1989). **5**: p.223-234.

290. Faulkner, D.V. and Jurka, J., "Multiple aligned sequence editor.", *Trends in Biochemical Sciences,* (1988). **13**: p.321-324.

291. Gribskov, M. and Burgess, R.R., "Sigma factors from *E. coli*, *B. subtilis*, phage SP01 and phage T4 are homologous proteins.", *Nucleic Acids Research,* (1986). **14**(16): p.6745-6763.

292. Chou, P.Y. and Fasman, G.D., "Empirical prediction of protein conformation.", *Annual Review of Biochemistry,* (1978). **47**: p.251-276.

293. Gribskov, M., Burgess, R.R., and Devereux, J., "PEPPLOT: a protein secondary structure analysis program for the UWGCG sequence analysis software package.", *Nucleic Acids Research,* (1986). **14**(1): p.327-334.

294. Bernstein, F.C., Koetzle, T.F., Williams, G.J.B., Mayer, E.F., Bryce, M.D., Rodgers, J.R., Kennard, O., Simanouchi, T., and Tasumi, M., "The protein data bank: a computer based archival file for macromolecular structures", *Journal of Molecular Biology,* (1977). **112**: p.535-542.

295. Gascuel, O. and Golmard, J.L., "A simple method for predicting the secondary structure of globular proteins: implications and accuracy.", *Computer Applications in the Biosciences,* (1988). **4**: p.357-365.

296. Matsuo, Y. and Nishikawa, K., "Protein structural similarities predicted by a sequence-structure compatibility method.", *Protein Science,* (1994). **3**: p.2055-2063.

297. Orcutt, B.C., George, D.G., and Dayhoff, M.O., "Protein and nucleic acid sequence database systems.", *Annual Review of Biophysics and Bioengineering,* (1983). **12**: p.419-441.