The CHAIN Program

Forging evolutionary links to underlying mechanisms

User's Manual

Version 1.0.0; November 2007

http://www.chain.umaryland.edu/

Andrew F. Neuwald

The Institute for Genome Sciences

The University of Maryland School of Medicine

HSF-II, 20 Penn St., Baltimore, MD, 21201, USA

aneuwald@som.umaryland.edu

Notices

The CHAIN program is distributed in the hope that it will be useful, but without any warranty; without even the implied warranty of merchantability or fitness for a particular purpose.

Acknowledgements

Development of the CHAIN program was funded by National Institutes of Health, National Library of Medicine Grant LM06747. Application of the CHAIN program was funded by Grant LM06747 and by the National Institutes of Health Division of General Medicine Grant GM078541. This user's manual is being released in conjunction with a *Trends in Biochemical Sciences* (*TiBS*) article [1] describing the CHAIN program; portions of and excerpts from this article have been incorporated into this manual. The Bayesian statistical formulas underlying CHAIN analysis were developed in collaboration with Dr. Jun S. Liu (Department of Statistics, Harvard University). I thank Dr. Natarajan Kannan (University of California, San Diego) for structural biology discussions and advice, Dr. Aleksander Poleksic (Computer Science Department, University of Northern Iowa) for assistance in programming aspects of the BPPS routine (multinomial version) and in formulating Equation (A1-1), and Dr. Naoya Hata for implementing the surface accessibility routine used in CHAIN analysis.

Contents

1. Introduction

Contrast Hierarchical Alignment and Interaction Network (CHAIN) analysis [2] gathers information relevant to how proteins work at the atomic level through a statistical analysis of protein sequences—the cell's own language for encoding biological mechanisms—and interprets this information in the light of relevant structural and biochemical data. Indeed, proteins functionally diverge via mutations that modify and extend an ancestral protein's molecular machinery. Hence similarities and differences between two divergent proteins' functions and mechanisms will show up, to some degree, as similarities and differences between those proteins' sequences. Based on this principle, the CHAIN program [1] identifies divergent patterns associated with groups of evolutionarily-related proteins that have mechanistically diverged. Each of these divergent groups defines a distinct 'functional state' subject to somewhat different selective constraints. The CHAIN program identifies those divergent residues that are subject to the strongest constraints distinguishing one such functional state from another. When atomic coordinates are available, it also identifies atomic interactions associated with divergent residues. For a review of the statistical and scientific basis for CHAIN analysis, see [3].

1.1. CHAIN analysis

CHAIN analysis involves aligning a set of related sequences, optimally partitioning the alignment into two functionally divergent groups while concurrently defining a divergent sequence pattern that best distinguishes these two groups, and—when atomic coordinates are available—identifying associated structural interactions.

The specific algorithmic steps taken are shown in Figure 1. The standard input (red boxes in Figure 1) consists of: (i) a query sequence set, which contains the query plus a few related sequences of specific interest, (ii) a main sequence set, which typically contains many sequences related to the query, and (optionally) (iii) atomic coordinates corresponding to the query sequence. Ideally, the query set should contain representatives from organisms that have diverged at least a billion years ago, because this ensures that conserved patterns characterized in the analysis are not merely due to recent common descent. After creating a file for possible taxonomic annotation of sequences in the query set (step 1), the query is aligned against related sequences in both the query and main sets using the PSI-BLAST [4] algorithm (step 2).

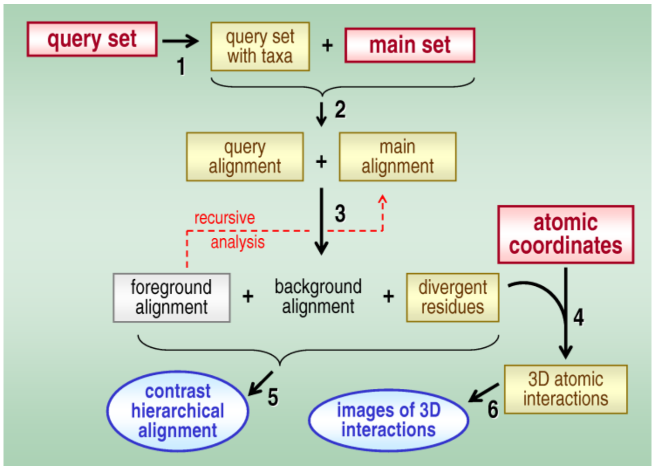

Figure 1. Flow chart of the algorithmic steps taken by the CHAIN program. Color scheme: red boxes, default input; yellow boxes, intermediate files that may also serve as alternative input points; blue ovals, output for graphic display. The foreground alignment (white box) can be used as the main alignment for recursive partitioning into divergent subgroups. This figure was adapted from reference [1].

Next, a Markov chain Monte Carlo[1] procedure [5], called Bayesian partitioning with pattern selection (BPPS) [2], optimally partitions the main alignment into two sets (termed the 'foreground' and the 'background') (step 3) such that sequences in the foreground set (which by design includes the query set) generally share a strikingly conserved pattern that is strikingly non-conserved in the background set. The residues specified by this pattern presumably play critical roles in the functional divergence of the foreground sequences away from the background sequences. A partitioned alignment of this kind is termed a Contrast Hierarchical (CH) alignment (Figure 2).

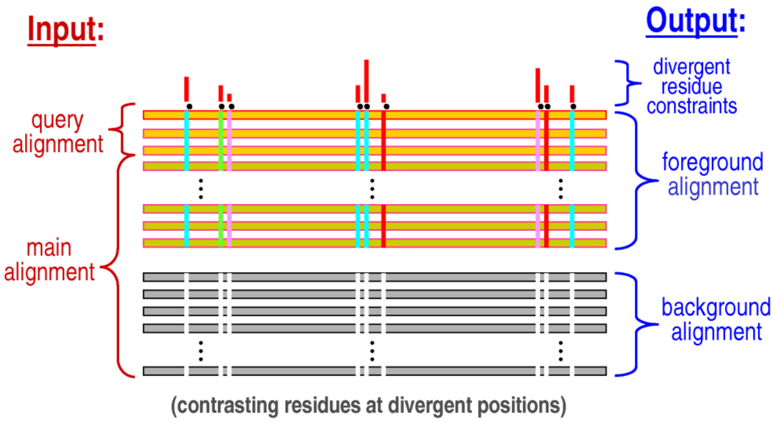

Figure 2. Schematic representation the BPPS procedure's input and output. Horizontal bars represent aligned sequences. Vertical bars represent foreground alignment conserved residues that strikingly diverge from the background residues at those positions; these bars are colored to reflect the types of residues conserved. The histogram above the alignment represents the relative selective constraints imposed on divergent residues. Adapted from reference [1]. The output is termed a 'contrast hierarchical' (CH) alignment (see below).

[1] **Markov chain Monte Carlo (MCMC) procedure**: A type of algorithm that iteratively samples random variables by transitioning from one random variable to another based on 'transition probabilities', which, in turn, are based on a specific probability distribution. A MCMC procedure has the property that, after a sufficiently large number of transitions, a sample from the actual probability distribution is obtained. Such a procedure is required for highly complex distributions in which direct calculations are intractable. Because MCMC samplers converge on the highest probability (or optimum) regions of a distribution, they are often used for optimization problems.

When atomic coordinates are available, atomic interactions between query residues within these various categories are identified (step 4). These interactions include weak [6] and strong [7] hydrogen bonds, and CH-π [8], cation-π [9], electrostatic [10] and aromatic-aromatic [11, 12] interactions. If not present in the atomic coordinate file, the locations of hydrogen atoms must be predicted prior to determining hydrogen bond interactions; this can be done using the Reduce program by J. Michael Word [13], which is available at http://kinemage.biochem.duke.edu/software/reduce.php.

Finally, two types of graphical output (blue ovals in Figure 1) are generated: (i) a contrast hierarchical (CH) alignment (step 5), which displays the selective constraints imposed on functionally divergent residues, and (ii) images of associated 3D structural interactions (step 6). A schematic representation of a CH-alignment is shown in Figure 2.

1.2. Getting started fast

To begin using the CHAIN program very quickly simply type the command "chain QuerySet MainSet", where the file 'QuerySet' contains a few very closely related sequences from diverse organisms and the file 'MainSet' contains lots of sequences related (though not necessarily closely related) to the query set. (Input sequences need to be in FASTA format, as defined below.) Using this command, the user can automatically generate a CH-alignment as a rich text format (*.rtf) file, which can be viewed in Microsoft Word. With this as a starting point, the various program options described below can be used to take an analysis progressively deeper.

1.3. Edited files as alternative input

The CHAIN program allows intermediate files corresponding to the yellow or white boxes in Figure 1 to be edited and used as alternative input corresponding to the yellow boxes in Figure 1. This allows the user to direct and refine an analysis so as to obtain information on specific aspects of protein functional divergence. Examples of how this is done are given in two *Trends in Biochemical Sciences* articles that describe both CHAIN analysis [3] and this program [1]. Such refinements, which are essential for obtaining biologically meaningful results, are described in detail below.

Several CHAIN program options (described below) allow edited intermediate files to be used as input; one key option in this regard is the following:

-restart	Skip over procedures for Steps 1-3 in Figure 1 for which output files already exist. This allows users to redo Step 2 or 3 by editing a particular output file and deleting output files generated after that file.

1.4. Examples

For users that prefer learning by example, sample input files are included with the CHAIN program. These correspond to various published analyses and include, for instance, an analysis of co-conserved residues distinguishing Ran, Ras, Rab, and Rho from other small GTPases [2]. These residues–along with other residues specifically conserved in Ran–appear to play critical roles in Ran’s C-terminal, basic patch, and nucleotide exchange mechanisms [2]. Other publications document CHAIN analyses of eukaryotic protein kinases [14], of CMGC [15] and AGC [16] protein kinases, of eukaryotic [17] and bacterial [18] clamp loaders and DNA clamps [19] and of the α subunit of heterotrimeric G proteins [20]. Users may examine these published examples to better understand how to apply the CHAIN program.

1.5. When is CHAIN analysis useful?

The CHAIN program is useful for the analysis of a class of protein domains that has functionally diverged into various subcategories and for which (as an aid to biological interpretation) abundant biochemical and structural data is available. The larger and more functionally diverse a protein class, the more useful CHAIN analysis will be. For such an analysis to be biologically meaningful, however, it is important that there be statistical support for the evolutionary relatedness of the input sequences. To ensure this, one should be able to align the sequences using a statistically based procedure (such as the PSI-BLAST algorithm), rather than multiple alignment procedures that will align unrelated sequences. One way to determine whether a multiple alignment program lacks a statistical basis is to apply it to a set of randomly generated sequences; statistically-based programs will not align such sequences.

2. Input sequences

The CHAIN program requires two FASTA format input files: one corresponding to the query sequence set, which contains the query plus a few related sequences, and another corresponding to the main sequence set.

2.1. FASTA format

A sequence in FASTA format begins with a single-line description, followed by lines containing only amino acid (single letter) residues (with optional space characters). The description line begins with a greater-than (">") symbol in the first position. The character string following this symbol (and terminated by a space character) contains the sequence identifier(s), whereas the rest of the line, which is optional, corresponds to the sequence description. There should be no space between the ">" and the first character of the identifier. The sequence ends either at the end of the file or at another line starting with a ">" symbol, which indicates the start of another sequence. The following is an example of a sequence in FASTA format:

```
>gi|73535739|pdb|1YZN|A Chain A, Gppnhp-Bound Ypt1p Gtpase
MGHHHHHHGSLVPRGSEYDYLFKLLLIGNSGVGKSCLLLRFSDDTYTNDYISTIGVDFKIKTVELDGKTVKLQIWDT
AGQERFRTITSSYYRGSHGIIIVYDVTDQESFNGVKMWLQEIDRYATSTVLKLLVGNKCDLKDKRVVEYDVAKEFAD
ANKMPFLETSALDSTNVEDAFLTMARQIKES
```

Aligned sequences in FASTA format are designed in the same way as unaligned sequences, except that gap (‘-‘) characters are used to indicate positions at which residues are deleted relative to other sequences in the alignment.

2.2. The query set

The query set input file contains the query along with a few other related sequences of interest. Ideally sequences in the query set should be taxonomically annotated. This section explains query set conventions and offers suggestions on how to select these sequences.

2.2.1. The query set file name

The file containing the query sequence set can take any legitimate Linux file name. The query file name will be denoted in this manual by the variable designation <*query_set*>. The names of the output files generated by the program use this designation as a prefix followed by various suffixes corresponding to the specific output file type.

2.2.2. The query sequence

The first sequence in the query set is the query sequence itself. Ideally, the query should correspond to a protein of known structure. Alternatively, it is sometimes helpful to use as a query a consensus sequence of a particular protein family inasmuch as a consensus sequence harbors every conserved residue characteristic of that family. Because CHAIN analysis is meaningful only over sequence regions that are evolutionarily related, it is often necessary to trim down the query sequence to correspond to a single conserved domain of interest.

2.2.3. Selecting the query set

To ensure that conserved patterns characterized in an analysis are not merely due to recent common descent, the query set should contain representatives from distinct phyla or kingdoms, which have diverged at least a billion years ago. When characterizing all of the constraints imposed on a specific protein family, the query set should include only those proteins from distantly related organisms that are likely to perform the same or a very similar function. When characterizing structural features conserved across multiple families, however, it is sometimes helpful to include representatives from each of the families.

2.2.4. Taxonomic annotation

In step 1 of the CHAIN algorithm (Figure 1) a source file for taxonomic annotation is created (output file name: <*query_set*>A). Each sequence in this file contains the pattern '{<unknown(U)>}' at the beginning of the description line. This pattern can be edited to include taxonomic information for each sequence matching the pattern "{<*taxon_string*>(*symbol*)>}", where *taxon_string* is the name of the organism's phylum or other taxonomic category and *symbol* is a single character representation of the organism's major taxonomic category. For example, phylum Chordata and major taxon Metazoa may be added to a sequence by changing the pattern '{<unknown(U)>}' on the FASTA description line to '{<Chordata(M)>}'. Single letter symbols (and the corresponding color schemes) for major taxonomic categories are as follows:

symbol:	'M'	'E'	'F'	'V'	'B'	'A'
taxon:	animals	protozoans	fungi	plants	eubacteria	archaea
Color code:	red	cyan	brown	green	violet	blue

The symbol 'U' denotes an unknown category. Both environmental samples and viral proteins should be classified as 'U'. By using the restart option (as described in Section 1.2 above) the CHAIN program will read taxonomic information given in an edited file (instead of creating a new one) and, as a result, will output a CH alignment (Figure 2) that uses this coloring scheme and the *taxon_string* to convey the taxonomic category of each query set sequence.

2.3. The main set

The second input sequence file consists of the main sequence set, which typically contains hundreds or thousands of FASTA sequences related to the query. This file also may take any legitimate Linux file name. In the initial stage of CHAIN analysis, the main set typically consists of all available sequences related to the query. In later stages of an analysis, however, it is important to curate this set carefully (based on information gleaned from previous stages of an analysis) so that biologically informative comparisons may be obtained. The protocols below describe how this may be done. For examples of highly informative comparisons see references

[3] and [20].

3. Multiple sequence alignment

The next step in CHAIN analysis is to obtain a multiple sequence alignment of the input sequences. This is done either by having the CHAIN program align the input sequences automatically or by having the user input the alignments directly. In either case, the program generates an output file containing the aligned query and main sequence sets in CHAIN program format. This file, which is named "<*query_set*>A.chn", serves as the input to step 3 of the CHAIN algorithm.

3.1. Using the built-in alignment method

By default the program assumes that input sequences in both the query and the main sets are unaligned, in which case both sets are aligned against the query using the PSI-BLAST algorithm (step 2 of the CHAIN algorithm Figure 1). For this step, the following two options may be used:

-max_iters=<*int*> set the maximum # of PSI-BLAST iterations to <*int*> (default=5).

-region=< *int1*>..<*int2*> align only those sequence regions corresponding to residues <*int1*> to <*int2*> of the query.

3.2. Using a third-party alignment method

Instead of relying on the program's built-in alignment procedure, pre-aligned sequence sets (obtained using either a sophisticated third-party alignment method or manual-curation) can be input directly. In this case, both the query and main sequence sets must be in FASTA alignment format and the following program option must be used:

-aligned Indicates that the input files contain aligned sequences.

With this option, the first sequence in the query set and the first sequence in the main set should be identical or at least both must be of *exactly* the same length.

4. Bayesian partitioning with pattern selection

A key statistical component of the CHAIN program is the Bayesian partitioning with pattern selection (BPPS) procedure, which is applied within step 3 of the CHAIN algorithm (Figure 1). Various input options can be used to influence both the statistical aspects of the BPPS procedure (via changes in prior probabilities, the form of the statistical model, and convergence criteria) and how these results are graphically displayed. The BPPS procedure's input and output is shown schematically in Figure 2—with the main output being a contrast hierarchical (CH) alignment in computer readable form. The procedure also generates four additional types of output files, which contain: (i) divergent pattern information; (ii) a set of divergent residues corresponding to the query; and (iii) foreground (see Figure 2) and (iv) query set multiple sequence alignments in FASTA format. The foreground and query set multiple alignments are useful for recursive analysis of functionally divergent protein subgroups.

4.1. A description of the BPPS procedure

Conceptually, the BPPS procedure assigns sequences to the foreground partition such that they optimally conserve a pattern that is optimally non-conserved in the remaining background sequences (see Figure 2). More specifically, the BPPS procedure uses an MCMC strategy [5] to sample two random variables: a conserved pattern and a set of indicators for assigning each sequence in an alignment to either the foreground partition (which includes the query sequence) or the background partition. It explores possible combinations of pattern-partition pairs, favoring selection of pairs where the pattern strikingly distinguishes the sequences in the foreground from those in the background. Mathematically, this corresponds to an optimum point in the corresponding probability distribution, which is defined (logarithmically) as:

$$\begin{aligned}\log P(\boldsymbol{X},\boldsymbol{R},\boldsymbol{C},\Theta,\alpha) = &\sum_{j=1}^{k}\sum_{i=1}^{n}\left\langle \log\boldsymbol{\theta}_j, x_{ij}\right\rangle + \sum_{j=1}^{k}\sum_{i=1}^{n} R_i C_j \left\langle \log\frac{\boldsymbol{\theta}_j^{\alpha}}{\boldsymbol{\theta}_j}, x_{ij}\right\rangle \\ &+\log p(\alpha)+\log p(\boldsymbol{\Theta})+\log p(\boldsymbol{R})+\log p(\boldsymbol{C})\end{aligned} \qquad (1)$$

where $\boldsymbol{X}$ is a matrix representing the sequence alignment, $\boldsymbol{R}$ is a vector indicating which rows in this matrix (i.e., sequences) belong to the foreground partition, $\boldsymbol{C}$ is a vector indicating which aligned columns are included in the pattern, Θ is an array of vectors representing the amino acid compositions at each column position for each partition, and α represents the small amount of 'contamination' allowed at pattern positions in the foreground partition. As defined, the pattern is required to include the residue observed in the query sequence at the positions defined by $\boldsymbol{C}$; hence the procedure requires that the pattern be present in (and thus characteristic of) the query. Degenerate residue patterns (i.e., where multiple residues at certain position are allowed) are modeled using a collapsing procedure that "merges" several residue types at specified positions into single "composite-residues". The BPPS procedure requires a MCMC sampling strategy to find a solution because it is initially unknown which sequences outside of the query set belong to the foreground partition, which positions correspond to pattern positions, and exactly which residues are conserved at each pattern position. Based on the probability distribution, the sampling procedure will converge on patterns that are both highly characteristic of conserved residues in the foreground sequences and highly uncharacteristic of the background sequences. For mathematical details see [2].

4.2. BPPS input options

Several input options can be used to influence the statistical model or convergence properties of the BPPS procedure or the appearance of the output. These include: (i) input of priors for $p(\alpha)$ in Equation (1), (ii) the use of a binary version of the BPPS statistical model, (iii) an option to search for multiple local optima in the posterior probability distribution, (iv) initialization of the BPPS procedure with seed patterns, and (v) altering the contrast of the CH alignment.

4.2.1. Setting the stringency with which a pattern is conserved

One BPPS input option influences how stringently pattern residues are conserved within the foreground partition by changing the prior probability for $p(\alpha)$ in Equation (1). The usage for this input option is:

-alpha=<*real1*>,<*real2*>	Set hyperparameters a_0 = <*real1*> and b_0 = <*real2*> (default: a_0 =1.0, b_0 =5.0).

The input values a_0 and b_0 are hyperparameters for the following prior probability distribution:

$$p(\alpha)=\frac{\Gamma(a_0+b_0)}{\Gamma(a_0)\Gamma(b_0)}\alpha^{a_0-1}(1-\alpha)^{b_0-1}. \qquad (2)$$

Equation (2) corresponds to the Beta distribution, which is defined in the interval between 0.0 and 1.0, and which in this case specifies the prior probability that an arbitrary residue in a foreground sequence will match the pattern. Note that a_0 > b_0 favors the conservation of canonical residues at pattern positions in the foreground, whereas a_0< b_0 disfavors conservation of canonical residues. (Recall that the parameter α represents the degree of departure from canonical residues that is to be tolerated at pattern positions.)

In general, one may interpret the Beta distribution as defining the probability of a specific coin coming up heads based on an observed number of heads ("successes") and tails ("failures") in a certain total number of coin flips. As applied here, Equation (2) can be viewed as defining the probability of a residue match, $p(\alpha)$, based on a fictitiously observed number (termed "pseudocounts") of a_0 foreground residue pattern matches (successes) and b_0 foreground residue pattern mismatches (failures). These pseudocounts influence α's posterior probability distribution, which is based both on these pseudocounts and on the observed pattern matches and mismatches. The default setting of $a_0 = 1$ residue match and $b_0 = 5$ mismatches at first might appear to highly favor the inclusion of mismatches in foreground sequences. However, the number of actual observed counts is typically quite large (often > 100) relative to the total number of pseudocounts, so that, in practice, such priors do not unduly influence p(α)'s posterior distribution.

4.2.2. Using an alternative (binary) statistical model

By default, the BPPS procedure applies Equation (1) so as to model each type of amino acid residue that fails to match the pattern explicitly, but it collapses all of the residue types that match the pattern into a single (abstract) residue type. For example, if the foreground sequences typically conserve either an aspartate or a glutamate at a particular pattern position, then, by default, the BPPS procedure will model each of the 18 types of residues that fail to match this pattern explicitly, but will collapse the matching residue types (aspartate and glutamate) into a single (acidic amino acid) residue type.

As a program option, pattern positions can also be modeled using an alternative, binary statistical model for pattern positions where all twenty amino acid residues at pattern positions are collapsed into just two residue types: one type consisting of all types of amino acid that match the pattern and another type consisting of the other types of amino acids. The usage for turning on this option is:

-binary	Use a binary model for partitioning with pattern selection.

Returning again to our previous example, if the foreground sequences tend to conserve an aspartate or a glutamate at a particular pattern position, then this binary model will collapse all twenty amino acids into only two types at that position, namely an acidic residue (corresponding to the foreground) and a non-acidic residue (corresponding to the background). Such binary statistical models tend to detect subtler constraints than the originally formulated multinomial model.

4.2.3. *Identifying multiple divergent categories concurrently*

When two groups of proteins functionally diverge, one group may harbor specific structural features (and a corresponding residue pattern) that the rest of the sequences clearly lack. The BPPS procedure often readily detects these contrasting features by converging on a globally optimum point in the posterior probability distribution specified by Equation (1). What more typically happens, however, is that functional divergence leads to three or more groups: one harboring particular functional features, others that lacking them, and one or more groups that, due to further evolution, harbor additional divergent features. Although such branching patterns of functional divergence can provide additional mechanistic insights, detecting them may require that the BPPS search for both globally and locally optimum pattern-partition pairs. This can be done using the following program option:

-local	Report locally optimal pattern-partition pairs (default: report the global optimum only).

When local optima are present, this option can result in the creation of multiple output files, as described below. Note, however, that, although the BPPS procedure will converge on either a local or global optimum, there is no guaranteed that it will find all such optima in a particular run.

4.2.4. *Initialization with seed patterns*

All MCMC procedures must start out in some initial state. For the BPPS procedure, this "state" corresponds to a (by default, arbitrary) pattern-partition pair associated with the input multiple alignment. In order to speed up convergence to a global optimum or to a local optimum of interest, the sampler can be seeded with a characteristic pattern using the following input option:

-P=<*char_set*><*int*>,[<*char_set*><*int*>]	Initialize sampler with a seed pattern where each <*char_set*> and <*int*> correspond to pattern residues and positions, respectively.

For example, this option can be used to initialize the sampler with the pattern [YF]….[ST][YF] starting at position 92 of the query by specifying "-P=YF92,ST97,YF98" on the command line. Note, however, that the BPPS procedure will not converge on a seed pattern when that pattern fails to correspond to a locally optimum point in the posterior probability distribution, as specified by Equation (1). In other words, using this option cannot force the procedure to obtain results lacking statistical support.

The following option performs a similar function by initializing the BPPS procedure with a seed pattern closely matching residues conserved within the query sequence set:

-close Favor convergence on foreground sequences closely related to the query.

This option favors convergence on the protein group that most closely corresponds to the sequences used in the query set.

4.2.5. Adjusting the contrast

It is often helpful to fine-tune the appearance of a CH-alignment (Figure 2). One way to do this is to adjust the contrast. If the contrast is set too low many marginally significant divergent residues in the CH-alignment will be identified and highlighted (see below); this tends to obscure the most important features. But if the contrast is set too high, some important divergent residues may not show up. To specify how many of the highest contrast residue positions that the BPPS procedure should report, the following option may be used:

-contrast=<*int*> <*int*> = number of the most divergent residue positions to report.

This option is used to adjust the contrast—rather than modifying prior probabilities associated with the pattern position vector $\boldsymbol{C}$ in Equation (1)—because its effect is easier to predict and understand. Note that for this option, lower values result in higher contrast. This option also indirectly changes the highlighting of the human-readable version of the CH-alignment (see Section 6.1) based on the linear-to-logarithmic scaling scheme described in Appendix 1.

4.3. BPPS output files

Various output files provide information regarding each CH alignment generated by the BPPS procedure, as follows:

4.3.1. Contrast hierarchical (CH-) alignment (.cha) files

The main output consists of one or more computer-readable CH alignment files. If only one such file is generated (which is the default option) this file is named "<*query_set*>A_pps.cha". In this case, the output corresponds to the optimum CH alignment found by the program. I the '-local' option is used BPPS procedure may output additional (locally optimal) CH alignments corresponding to additional (suboptimal) functionally divergent categories. When multiple categories are found, the corresponding output files are named "<*query_set*>A1_pps.cha", "<*query_set*>A2_pps.cha", etc.

4.3.2. Pattern information (.ptn) files

The BPPS procedure identifies a divergent pattern associated with each CH alignment. The amino acid residues conserved at pattern positions and the associated statistical information are written either to a text file named “<*query_set*>A_pps.ptn” (with default option) or to multiple text files named “<*query_set*>A1_pps.ptn”, “<*query_set*>A2_pps.ptn”, etc. (with ‘–local’ option). A ptn file provides the amount of information (in nats) contributed by each pattern position to the total log-probability ratio. (Note, however, that the number of significant figures given for each information estimate is not adjusted according to the accuracy of that estimate.)

4.3.3. Alignment (.aln) files for recursive analysis

After the BPPS procedure partitions the main alignment into two divergent groups it outputs the foreground set and query set alignments in FASTA format using the file names “<*query_set*>_fg.aln” and “<*query_set*>_qs.aln”, respectively. These files may then be used as input for a recursive analysis in which the previous foreground alignment serves as the main alignment. In this way, subgroups of proteins that have further functionally diverged from other members of the previous foreground set may be identified and characterized.

4.3.4. Divergent residue (_3D) files

The BPPS procedure also creates a text file listing those residue positions in the query that—based on statistically rigorous criteria—have functionally diverged. In addition to listing these statistically defined divergent residues (which are color coded yellow), at this step the program uses heuristic criteria to tentatively assign the remaining conserved residue positions into three other categories: (i) those primarily conserved in the query set alone (color code: cyan), (ii) those generally conserved in the main set as a whole (color code: magenta), and (iii) those conserved both in the query set and in some, but not all of the sequences in the main set (color code: green). These tentative categories merely provide an initial context in which to interpret the statistically-defined divergent residues; later these tentative categories need to be replaced, however, by statistically-defined categories.

The output files names are “<*query_set*>_3D” (single file) or “<*query_set*>_3D1”, “<*query_set*>_3D2”, etc. (multiple files). The following is an example _3D file:

```
File1=query_set.pdb:B      // atomic coordinate file
K51A.M,D126.M,E127.M,R169.M// Main set; magenta
L140.Y,K141.Y,E145.Y,P146.Y,T165.Y     // BPPS defined; yellow
Y123.C,M130.C,F136.C       // query set alone; cyan
I125.G,S168.G         // “intermediate”; green
```

For the syntax and semantics of "_3D" and other vsi-format files see below.

Such files serve as one of two input files to Step 4 of the CHAIN algorithm (Figure 1) where the other input file contains the query protein's 3D atomic coordinates. This file may be and indeed should be edited by the user in order to obtain the best 3D structural perspective on the various categories of divergent residues identified by CHAIN analysis (see below). To rerun the CHAIN program using an edited "_3D" files the following options may be used:

-3D Based on file "<*infile_prefix*>_3D" create vsi files ("<*infile_prefix*>_3D<*int*>.vsi") and corresponding RasMol scripts ("<*infile_prefix*>_3D<*int*>.ras"), where each of one or more <*int*>'s correspond to 3D image IDs. This option, which allows input of edited _3D files.

5. Divergent residue interactions

After the BPPS procedure classifies query residues into functionally divergent categories and when atomic coordinates corresponding to the query protein are provided, the CHAIN program next identifies associated atomic interactions between divergent residues (Step 4 in Figure 1). This step requires that the query sequence in the alignment exactly correspond to at least one subunit in the atomic coordinate file and that atomic coordinates for hydrogen atoms be included. The output from Step 4 is a "visualization of structural interactions" (vsi) file that also serves as the input to Step 6 of the CHAIN algorithm. The output from Step 6 consists of RasMol scripts that allow visualization of atomic interactions involving divergent residues.

5.1. Input of atomic coordinates

In order for the CHAIN program to carry out Steps 4 and 6 in Figure 1, the following option, which specifies the input atomic coordinate file, must be used:

-pdb=<*pdb_file*> Determine 3D interactions base on the atomic coordinates data in file <*pdb_file*>.

From this input file the program generates 'cleaned up' atomic coordinates for further analysis and outputs these into a file named "<*query_set*>.pdb".

5.1.1. Correspondence between sequence and structure

Note that the atomic coordinate input file must *exactly* correspond to the query sequence used in the analysis. If the query protein atomic coordinates, for example, begin with residue

100 and end with residue 250, then the first sequence in the query set likewise needs to include only residues 100-250. Moreover, the numbering of residue positions between the coordinate and sequence files also need to be consistent. This is done by modifying the pattern "{<*taxon_name*(*symbol*)>}" within the FASTA description line of the query (i.e., the first sequence in file "<*query_set*>A", as explained above) by inserting the pattern "|N_{nt}(N_{ct})|" directly after the left squiggly bracket, to produce the pattern "{|N_{nt}(N_{ct})|<*taxon_name*(*symbol*)>}"—where the variables N_{nt} and N_{nt} are the number of residues trimmed from the N-terminus and C-terminus, respectively. However, because only N_{nt} need be specified for the query residue numbering to be consistent with the coordinate file, the value of N_{ct} also may be set to zero (resulting in the pattern "{|N_{nt}(0)|<*taxon_name*(*symbol*)>}"). Thus, if the example query protein just mentioned has a native length of 400 residues and comes from phylum Chordata, then its FASTA description line should be modified to incorporate the pattern "{|99(150)|<Chordate(M)>}" or "{|99(0)|<Chordate(M)>}".

5.1.2. Automatic identification of query chains

Maintaining consistency in the lengths and residue numbering between the query sequence and coordinate files allows the CHAIN program to identify automatically every chain within the pdb file that corresponds to the query. Thus the user can input coordinate files for which the query protein is but one component within a multiple component complex without having to specify which protein chains correspond to the query.

5.1.3. Adding hydrogen atoms to structural files

Protein structural coordinates that are obtained directly from the pdb database typically lack atomic coordinates for hydrogen atoms. Thus, in order for the CHAIN program to identify hydrogen bonds, atomic coordinates for predicted hydrogen atoms typically must be added to the pdb file. This may be done using the 'reduce' program by J. Michael Word; reduce may be downloaded at http://kinemage.biochem.duke.edu/software/reduce.php. Once reduce is on the user's path, the CHAIN program will add hydrogens automatically to pdb input files that lack hydrogens by making a system call to the reduce program. This will generate the intermediate file "<*pdb_file*>.reduced", from which the 'clean up' "<*query_set*>.pdb" file is then derived.

5.2. Creation of vsi files

In Step 4 of the CHAIN algorithm (see Figure 1) "visualization of structural interactions" (vsi) files are created—one for each _3D file created in Step 3. These files are named

<*query_set*>_3D.vsi (single category mode) or <*query_set*>_3D1.vsi, <*query_set*>_3D2.vsi, … (multiple category mode). These serve as intermediate files from which scripts (ras files) are generated (in step 6) for direct visualization of divergent residue atomic interactions using the program RasMol [21].

6. Output for graphic display

In steps 5 and 6 the CHAIN program converts the output generated in previous steps into human readable form. More specifically, in step 5 the program creates a rich text formatted file of each CH-alignment generated in step 3 and in step 6 the program creates a RasMol script of each vsi file generated in step 4.

6.1. Color scheme

Within these files, the CHAIN program uses a coloring scheme to distinguish between residues belonging to different functionally divergent categories as follows: (i) Divergent residues identified based on rigorous statistical criteria, **yellow**; (ii) tentative category 1 (residues primarily conserved in the query set alone), **cyan**; (iii) tentative category 2 (residues generally conserved in the main set as a whole), **magenta**, and (iv) tentative category 3 (residues inconsistently conserved across the main set), **green**. When these tentative categories are replaced by other statistically-defined categories a useful convention is the following:

superclass	*class*	*subclass*	*superfamily*	*family+*	*family*	*subfamily*
magenta	**red**	**orange**	**yellow**	**green**	**cyan**	**blue**

6.2. CH-alignment rich text format (.rtf) files

The rich text format (*.rtf) CH- alignment(s) created in step 5 can be viewed (and edited) in MicroSoft Word. A single output file is named “<*query_set*>A_pps.rtf”’, whereas multiple files are named “<*query_set*>A1_pps.rtf”, “<*query_set*>A2_pps.rtf”, etc. Conserved pattern positions within these CH alignments are highlighted as indicated in Table 1.

The following input options allow the user to modify the appearance of rtf files:

-colors=<*char*><*char*><*char*> Assign colors to CH-alignment categories (default: "MYC").
Colors allowed:
‘M’: magenta ‘R’: red ‘O’: orange ‘Y’: yellow
‘G’: green ‘C’: cyan ‘B’: blue

-conserve=<real> Require pattern residues to be ≥ <real> % conserved in query set where 0.0 ≤ <real> ≤ 100.0.

-font=<*int*> Set alignment font size: 4-24 points (default: <*int*> = 7)

-page=<*char*> MS Word page setup option (default: <*char*> = 'p'):
'p': 8.5"x11" portrait
'P': 11"x14" portrait
'l': 8.5"x11" landscape
'L': 11"x14" landscape
-reference=<*int*> Used the <*int*>th query set seq. for numbering alignment columns.
-rtf=<*int1*>..<*int2*> Print alignment from query position <*int1*> to position <*int2*> only.

Table 1. CHAIN analysis residue highlighting scheme. Chemically similar conserved residues are shaded similarly with the intensity of highlighting proportional to how strikingly foreground residues contrast with background residues.

	Level of contrast			
Residue type:	**none**	**weak**	**moderate**	**high**
aliphatic	ACILVM	ACILVM	ACILVM	ACILVM
aromatic	FWY	FWY	FWY	FWY
histidine	H	H	H	H
turns	G P	G P	G P	G P
basic	KR	KR	KR	KR
polar	NQST	NQST	NQST	NQST
acidic	DE	DE	DE	DE

6.3. Rasmol (.ras) scripts

In Step 6 of the CHAIN algorithm (Figure 1), Rasmol scripts are created. These are named <*query_set*>_3D.ras (for a single file) or <*query_set*>_3D1.ras, <*query_set*>_3D2.ras, etc. (for multiple files). These scripts can be viewed using the command: 'rasmol -script <*query_set*>_3D.ras'. RasMol can be downloaded at http://www.rasmol.org.

7. Refining CH alignments

The rtf and RasMol files that are created automatically by the CHAIN program merely serve as a first step by informing the user about certain categories of functionally divergent residues and by providing an initial glimpse of their associated atomic interactions. The ultimate goal, however, is to identify evolutionary links between a protein's sequence and its underlying molecular mechanisms. To achieve this CH-alignments and corresponding 3D images need to be adjusted and refined in the light of biochemical and structural information. This section describes how to adjust and refine CH alignments by improving the quality of the input multiple alignments and by selecting foreground and background sets that readily lend themselves to biological interpretation. The next section describes how to refine vsi files and thus the RasMol scripts derived from them.

7.1. Improving alignment quality

The multiple sequence alignment used as input to step 3 of the CHAIN algorithm may be improved in two ways: (i) by correcting misaligned regions, and (ii) by eliminating pseudogene products and other misleading sequences.

7.1.1. Correcting alignment errors

Alignment errors may be corrected by using a sophisticated 'third-party' multiple alignment procedure, manual curation, or both. The resulting alignment (in FASTA format) is input directly to the CHAIN program, thereby bypassing the program's built-in alignment procedure. We often rely on Bayesian multiple alignment procedures [22-28], which can improve an alignment by taking advantage of large numbers of available sequences to detect very subtle, yet clearly statistically significant sequence similarities. (These procedures currently are rather difficult to use, however; though a user friendly version is being developed.)

Unfortunately even the best alignment methods will fail to correctly align certain functionally important residues. Correctly aligning such regions requires manual curation, which, in turn, requires knowledge of structurally conserved features. Hence, an effective strategy is to iterate between improvements in the alignment, application of the BPPS procedure, and examination of 3D structural interactions involving functionally divergent residues; this strategy requires using the CHAIN program's alternative input options, which were described above.

7.1.2. Eliminating pseudogene products

Pseudogene products and other highly atypical sequences may be identified and eliminated by: (i) generating (from a specific group of proteins) simulated sequences that harbor random mutations at a specific percentage (say 25%) of their sites, (ii) combining these simulated sequences with the original sequences, and (iii) using the program's BPPS procedure to partition these sequences into 'wild type' and 'mutated' sets. The CHAIN program can perform all three of these operations automatically by using the option:

-mutate=<*fraction*>	Mutate a fraction of the sequence sites in the main set (where 0.0 < fraction <= 1.0) and merge mutated sequences into the main set.

Note, however, that this option is activated only when aligned sequences are given as input.

7.1.3. Eliminating misleading protein categories

Protein evolution often leads to three or more functional divergence groups: one harboring certain divergent features, another lacking them, and one or more groups that, due to further specialization, harbor additional variant features. For example, the eukaryotic DNA clamp loader Replication Factor C (RFC) complex, which consists of five evolutionarily-related subunits, performs three distinct cellular roles—one associated with DNA replication, another with sister chromatid cohesion and a third with DNA damage checkpoints. As a result, one of the five subunits, namely RFC-A, has further functionally diverged into three distinct RFC-A-like subunits—each of which is required for one of RFC's three cellular roles. Although such branching patterns of functional divergence can provide further insights, they can also obscure features important for a specific cellular role, in which case the CHAIN program's BPPS procedure can be applied (perhaps recursively) to separate out misleading divergent groups.

7.2. Strategy for linking residues to divergent functions

Applying the approaches described thus far still falls short of predicting specific functional or mechanistic roles for individual residues – a task requiring (i) a focus on one aspect of protein function at a time and (i) selection of background proteins that have functionally diverged in specific ways both along with, and away from, the foreground protein of interest. Moreover, in order to infer clear cut evolutionary links between protein sequences and underlying mechanisms, an analysis needs to be performed on well defined functionally divergent categories. It also requires choosing foreground and background sets that can provide biological insights given what is known about the biochemistry of each protein and its particular cellular function.

The following strategy seeks to address these issues:

(i) First, various categories of functionally divergent proteins manifesting subtle biochemical similarities and distinctions between them are identified. For instance, references [1] and [3], illustrate this process for bacterial and eukaryotic clamp loader complexes, respectively. Both types of complexes are composed of five subunits that are evolutionarily related, but that have functionally diverged so as to perform specialized roles in clamp loading. As a result, each subunit exhibits biochemical similarities and differences compared to other clamp loader subunits.

(ii) Second, several preliminary CHAIN analyses—each of which uses one of the divergent proteins defined in the previous step as a query— are performed so as to favor convergence on closely related sequences (see "–close" option). The resultant foreground sequence sets from each of these analyses are then merged. (Because most database sequences correspond to uncharacterized proteins, the BPPS procedure must be used in this way to identify sequences belonging to the families of interest.)

(iii) Finally, the CHAIN program is applied to the merged sequence set to obtain the intended CH-alignment foreground versus background comparison. Note, however, that the CHAIN program will only produce specific results that have some statistical support—so it is important to first confirm functional divergence of the intended foreground and background sequences through preliminary analyses.

It is important to perform multiple CHAIN analyses of a particular protein family in this way, as this allows a given residue's role to be fully appreciated by placing it in the context of functionally-associated residues, which often belong to distinct divergent categories. This strategy is explained and illustrated in greater detail in reference [3].

8. Refining images of 3D interactions

After characterizing multiple functionally divergent categories, it is important to identify atomic interactions between the corresponding divergent residues and to thereby recognize any obvious clues regarding associated underlying mechanisms.

8.1. Manual curation of vsi files

The ultimate goal of CHAIN analysis is to identify multiple categories of functionally divergent residues (through generation of CH alignments), to identify associated atomic interactions between divergent residues (through automatic generation of vsi files and Rasmol scripts), and to then combine these results into a single manually-curated vsi file that can best reveal the structural features most critical to protein function. RasMol scripts generated from curated vsi files display multiple categories of divergent residues concurrently by using a distinct color code for each category (see above). Examples of curated vsi files, from which such RasMol scripts can be generated, are included with the CHAIN program. To generate RasMol scripts from a curated vsi file named "<*infile_prefix*>.vsi", the user may use the following option:

-vsi=<int>	Translate the (typically edited) file <*infile_prefix*>.vsi into a RasMol script (output file = "<*infile_prefix*>_<*int*>.ras"); <*int*> specifies the 3D image ID

With this option, the chain program takes a single input file (i.e., the command line syntax is: "chain < *infile_prefix* > –vsi=<*int*>"). The <*int*> associated with the –vsi option specifies a "3D image ID" that identifies one among many possible 3D perspectives of the query protein specified in the vsi input file (see below). For example, given as input the (manually curated) vsi file "RanEdited.vsi" (which is included with the CHAIN program) the commands "chain RanEdited -vsi=3" and "chain RanEdited –vsi=4" generate two different RasMol scripts, each of which displays a distinct 3D perspective of functionally divergent residues in Ran GTPase.

8.1.1. Syntax underlying vsi files

Manual curation of vsi files requires some knowledge of the context-free grammar used to define the vsi language. Although a formal description of this language is beyond the scope of this manual, Appendix 2 provides (i) informal specification of the vsi syntactical patterns recognized by the CHAIN program, (ii) descriptions of the resulting RasMol graphical effects associated with these patterns, and (iii) examples of character strings matching these patterns (these strings are shown just as they would appear within an actual vsi file). Note that if a vsi file is edited using incorrect syntax, the CHAIN program will exit (without generating a RasMol output file) and will report the line on which the first syntactic error occurs.

*8.1.2. **File** and **view** descriptors*

To specify an atomic coordinate input file the vsi language uses the syntactic pattern "**File**<*int*>**=**<*name*>**.pdb:**<*chain*>" (see Appendix 2), where (i) <*int*> represents the 3D image ID, which attaches the input fle to a specific 3D image to be generated; (ii) <*name*> specifies the prefix of the input atomic coordinate file name (the suffix being ".pdb"); and (iii) <*chain*> specifies a default single letter chain designation. The atomic coordinate file needs to be in pdb format. The program associates the default chain with vsi syntactical phrases that require, but lack a chain designation. An example of a character string matching this "**File**" pattern is "File3=RAN/1I2MAHM.pdb:A".

The 3D image ID associates specific syntactic phrases within a vsi file with a specific 3D output image. This is necessary because a single vsi file typically specifies multiple 3D images by including a distinct "**File**" descriptor for each image to be generated. Although each "**File**" descriptor's 3D image ID must be unique, the <*name*> and <*chain*> variables need not be.

For each 3D image, the orientation of the atomic coordinates are specified using the syntactic pattern "**view**<*int*>**:R=**<x_r>,<y_r>,<z_r>**;T=**<x_t>,<y_t>,<z_t>**.**", where <*int*> is the 3D image ID; <x_r>,<y_r>, and <z_r> specify the angles of rotation about the *x*, *y* and *z* axes, respectively; and <x_t>,<y_t>, and <z_t> specify the distance the coordinates are translated along the *x*, *y* and *z* axes, respectively. A convenient way to obtain appropriate rotational and translational values is (i) to choose an appropriate structural orientation within RasMol, (ii) to save the settings to a script using RasMol's "write script" command, and (iii) to copy these parameters from the RasMol script. Note, however, that the distance of translation along the *z* axis is denoted by the ZOOM setting within RasMol scripts.

The following are examples of “**File**” and “**view**” descriptors within a vsi file:

```
# vsi file of Ran GTPase corresponding to the analysis in reference [2]
File3=RAN/1I2MAHM.pdb:A     // Ran-RCC1: focus on interface with RCC1 (Fig. 9E).
view3:R=-59,-73,-176;T=8,-20,348.

File4=RAN/1BYUAHM.pdb:A         // Ran-GDP: focus on interface with RCC1 (Fig. 9D).
view4:R=44,29,-58;T=-9,37,276.
```

Note that the “**view**” and other descriptors within the file must be placed *after* the corresponding “**File**” descriptor.

8.1.3. Adding comments to vsi files.

Comments are added to vsi files either by starting a line with the number character (‘#’), in which case the entire line is ignored, or by inserting the string “//” within a line, in which case text on the line from that point on is ignored. To have the program ignore an entire section of a vsi file, the user can start and end the section with the strings “**#off**” and “**#on**”, respectively; these strings need to be place at the start of a new line. If a line begins with the string “**text=**” then the remainder of the line is printed to the terminal when the vsi file is parsed by the CHAIN program; this feature informs the user at run time about the contents of a file.

8.1.4. Specifying backbone traces

The syntactic pattern “**(**<*int1*>**)**<*int2*>**-**<*int3*><*chain*>**.**<*color*><*int4*>” specifies a backbone trace within a vsi file where (i) “**(**<*int1*>**)**” specifies the 3D image ID; (ii) “<*int2*>**-**<*int3*>“ specifies the starting and ending residue positions to be traced out; (iii) “<*chain*>” specifies the single letter designation of the chain that is to be traced; (iv) “<*color*>” specifies the single letter designation for the color of the trace (see Table A2-2 in Appendix 2); and (v) “<*int4*>” specifies the backbone trace width. An example of a string matching this pattern is “(3)15-35A.R150”, which translates into the command: “For 3d image #3, trace the backbone for chain ‘A’ from residue 15 to residue 35 with a width of 150”. Items (iii) and (v) need not be specified, in which case default values for these missing variables are assumed. Item (i) also need not be specified, in which case the CHAIN program will translate matching strings into RasMol commands regardless of what 3D image ID is specified on the command line—that is, assuming that the input 3D image ID is *not* explicitly shut off (as described below). An example of a minimum matching string is “15-35.R”.

DNA and RNA chains can also be visualized using the backbone trace syntax pattern. For nucleic acids a useful variant syntactic pattern is to replace the character ‘.’ in the standard pattern with ‘+’ (e.g., “(3)15-35A+R150”); this causes all covalent bonds to be displayed as sticks, thereby making the molecular characteristics of DNA or RNA more evident.

8.1.5. Specifying residues and residue interactions

Two types of syntactic patterns are used for amino acid residues: one for displaying side chains and one for displaying individual atoms. The pattern “<*int1*><*aa*><*int2*><*chain*>**.**<*color*><*int3*>” specifies a residue side chain where (i) “<*aa*>” specifies the amino acid single letter code, and (ii) “<*int2*>” specifies the residue position. An example string matching this pattern is “3N100A.Y150”. The descriptors “<*chain*>”, “<*color*>”, and “<*int3*>” are as described in the previous section, and “<*int1*>” corresponds to the 3D image ID; “<*int1*>” “<*chain*>”, and “ <*int3*>” are optional. An example string matching the minimal pattern is “N100.Y”.

The syntactic pattern “<*int1*><*aa*><*int2*><*chain*>**_**<atom>**.**<*color*><*int3*>” specifies a residue hydrogen bond acceptor atom where “<*atom*>” specifies the atom in lowercase pdb notation. Other descriptors are the same as for residue side chains. The minimal pattern (i.e., with optional terms omitted) is “<*aa*><*int2*>**_**<atom>**.**<*color*>”. Dot clouds may be displayed around a residue atom (or side chain) by placing squiggly brackets around the color designator: “<*aa*><*int2*>**_**<atom>**.{**<*color*>**}**”.

The syntactic pattern “<*int1*><*aa*><*int2*><*chain*>_<atom>**-**<*hydrogen*>**.**<*color*><*int3*>” specifies residue hydrogen bond donor atoms where “<*hydrogen*>” specifies the hydrogen in lowercase pdb notation. Other descriptors are the same as for a residue acceptor atom. The corresponding minimal pattern is “<*aa*><*int2*>**_**<atom>**-**<*hydrogen*>**.**<*color*>”. An example string matching this minimal pattern is: “R106_nh2-2hh2.X”; the color designator ‘X’ used here indicates CPK coloring, which depicts carbon as black or grey, oxygen as red, nitrogen as blue, and hydrogen as white.

It is important to note that strings matching residue syntactical patterns are not translated unless they are preceded by a backbone trace string corresponding to those residues. This feature allows the user to turn off entire regions of a protein temporarily, while editing and improving a particular 3D image, by commenting out the backbone trace designators.

8.1.6. Specifying non-protein molecules and their interactions

The syntactic pattern “<*int1*>**![**<*molecule*>**]**<*int2*><*chain*>**.**<*color*><*int3*>” (or the corresponding minimal pattern: “**![**<*molecule*>**]**<*int2*>**.**<*color*>) specifies a non-protein molecule, for which "<*molecule*>" specifies the corresponding lowercase pdb notation. Other descriptors are the same as for residue side chains. An example string matching such a pattern is "![SO4]300C.X". The square brackets around "<molecule>" are *not* required when the pdb designation for the molecule consists only of alphabetical characters (i.e., lacks numerical characters)—as, for example, in the string "!ATP801.C". The syntactic pattern for non-protein molecule hydrogen bond acceptors and donors is obtained as for residues, namely by inserting the patterns "**_**<*atom*>" and "**_**<*atom*>**-**<*hydrogen*>", respectively, between "<*chain*>" and the period ("**.**") in the syntactic pattern above. Water molecules also may be specified using the "**HOH**" syntactic pattern given in Table A2-1 of Appendix 2.

8.1.7. Lists of residues, molecules, and atoms

Two or more residue and/or molecule designators may be incorporated into a single-line list by separating them with commas, as, for example, in the following string: "W28.M, E70A_oe2.X, HOH2C.X". Such lists are useful when constructing multiple 3D images of a particular protein, as explained in the next section.

8.1.8. 3D image IDs and image creation

To better comprehend the preceding informal description of the vsi syntax, a concrete example of an actual vsi file is given in Appendix 3. This vsi file illustrates a few additional syntactic features that relate to 3D image IDs and to the translation of a vsi file into a RasMol script and that warrant an explanation.

Recall that creating a RasMol script from an edited vsi file requires the command "chain <*vsi_filename_prefix*> -vsi=<*int*>", where <*int*> denotes the 3D image ID. This ID tells the CHAIN program which image is to be generated. Whenever a list of one or more syntactic patterns associated with a trace, a residue or a non-protein molecule begins with that ID, the CHAIN program translates that pattern into a RasMol script command corresponding to that image. Whenever an image ID is not specified for a syntactic pattern, the program translates the pattern unless the image ID has been inactivated. An image ID is inactivated beyond any point in the vsi file where the syntactic pattern "**(**<*int*>**-).**" occurs, where <*int*> is the ID. For example, the string "(3-)." inactivates image #3 from that point forward. An ID may be reactivated by placing the pattern "**(**<*int*>**+).**" at another point in the file. This corresponds to "(3+)." in the previous example. Alternatively, a range of image IDs may be inactivated or reactivated using the patterns "**(**<*int1*>**-**<*int2*>**-).**" and "**(**<*int1*>**-**<*int2*>**+).**", respectively.

8.2. Recognizing evolutionary links to underlying mechanisms

Once the user has characterized various functionally divergent categories associated with a particular protein of interest, the final step is to pour over the results in the light of published studies relevant to that protein. Because the Bayesian methodology underlying CHAIN analysis is empirically based and thus aims to let the data speak for itself, any biologically valid inferences regarding underlying mechanisms that can be made at this point ought to be obvious. Of course, CHAIN analysis cannot directly reveal actual mechanisms. Nevertheless, it can reveal aspects of those mechanisms by revealing the evolutionary constraints associated with them. At times, certain hypothetical mechanisms will seem highly probable in the light of these constraints. However, even when CHAIN analysis fails to point to a single, clear cut mechanism, it still can be useful for formulating plausible hypotheses either as an aid to experimental design or for interpreting newly published experimental results.

9. References

1. Neuwald, A.F. (2007) The CHAIN program: forging evolutionary links to underlying mechanisms. *Trends Biochem Sciences* 32, 000-000
2. Neuwald, A.F.*, et al.* (2003) Ran's C-terminal, basic patch and nucleotide exchange mechanisms in light of a canonical structure for Rab, Rho, Ras and Ran GTPases. *Genome Res* 13, 673-692
3. Neuwald, A.F. (2006) Bayesian shadows of molecular mechanisms cast in the light of evolution. *Trends Biochem Sciences* 31, 374-382
4. Altschul, S.F.*, et al.* (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res* 25, 3389-3402
5. Liu, J.S. (2001) *Monte Carlo Strategies in Scientific Computing*. Springer-Verlag
6. Weiss, M.S.*, et al.* (2001) More hydrogen bonds for the (structural) biologist. *Trends Biochem Sci* 26, 521-523.
7. Baker, E.N., and Hubbard, R.E. (1984) Hydrogen bonding in globular proteins. *Prog Biophys Mol Biol* 44, 97-179.
8. Brandl, M.*, et al.* (2001) C-H...pi-interactions in proteins. *J Mol Biol* 307, 357-377
9. Gallivan, J.P., and Dougherty, D.A. (1999) Cation-pi interactions in structural biology. *Proc Natl Acad Sci U S A* 96, 9459-9464.
10. Honig, B., and Nicholls, A. (1995) Classical electrostatics in biology and chemistry. *Science* 268, 1144-1149
11. Burley, S.K., and Petsko, G.A. (1985) Aromatic-aromatic interaction: a mechanism of protein structure stabilization. *Science* 229, 23-28.
12. Burley, S.K., and Petsko, G.A. (1986) Amino-aromatic interactions in proteins. *FEBS Lett* 203, 139-143.
13. Word, J.M.*, et al.* (1999) Asparagine and glutamine: using hydrogen atom contacts in the choice of side-chain amide orientation. *J Mol Biol* 285, 1735-1747.
14. Kannan, N., and Neuwald, A.F. (2005) Did protein kinase regulatory mechanisms

evolve through elaboration of a simple structural component? *J Mol Biol* 351, 956-972.
15. Kannan, N., and Neuwald, A.F. (2004) Evolutionary constraints associated with functional specificity of the CMGC protein kinases MAPK, CDK, GSK, SRPK, DYRK, and CK2{alpha}. *Protein Sci* 13, 2059-2077.
16. Kannan, N*., et al.* (2007) The hallmark of AGC kinase functional divergence is its C-terminal tail, a cis-acting regulatory module. *Proc Natl Acad Sci U S A*
17. Neuwald, A.F. (2005) Evolutionary clues to eukaryotic DNA clamp-loading mechanisms: analysis of the functional constraints imposed on replication factor C AAA+ ATPases. *Nucleic Acids Res* 33, 3614-3628.
18. Neuwald, A.F. (2006) Hypothesis: bacterial clamp loader ATPase activation through DNA-dependent repositioning of the catalytic base and of a trans-acting catalytic threonine. *Nucleic Acids Res* 34, 5280-5290
19. Neuwald, A.F. (2003) Evolutionary clues to DNA polymerase III beta clamp structural mechanisms. *Nucleic Acids Res* 31, 4503-4516.
20. Neuwald, A.F. (2007) Gα-Gβγ dissociation may be due to retraction of a buried lysine and disruption of an aromatic cluster by a GTP-sensing Arg-Trp pair. *Protein Science* 16, 000-000
21. Sayle, R.A., and Milner-White, E.J. (1995) RASMOL: biomolecular graphics for all. *Trends Biochem Sci* 20, 374
22. Lawrence, C.E*., et al.* (1993) Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science* 262, 208-214
23. Liu, J.S., and Lawrence, C.E. (1995) Statistical models for multiple sequence alignment: unifications and generalizations. *Proc. Amer. Statist. Assoc. - Statistical Computing Section*, 1-8
24. Liu, J.S*., et al.* (1995) Bayesian models for multiple local sequence alignment and Gibbs sampling strategies. *J Am Stat Assoc* 90, 1156-1170
25. Liu, J.S*., et al.* (1999) Markovian structures in biological sequence alignments. *JASA* 94, 1-15
26. Neuwald, A.F., and Liu, J.S. (2004) Gapped alignment of protein sequence motifs through Monte Carlo optimization of a hidden Markov model. *BMC Bioinformatics* 5, 157.
27. Neuwald, A.F*., et al.* (1997) Extracting protein alignment models from the sequence database. *Nucleic Acids Research* 25, 1665-1677
28. Neuwald, A.F*., et al.* (1995) Gibbs motif sampling: detection of bacterial outer membrane protein repeats. *Protein Sci* 4, 1618-1632
29. Levine, J*., et al.* (1995) *lex & yacc*. O'Reilly Media, Inc.

10. Appendix 1. Adjusting the contrast of CH alignments

To adjust the contrast, the CHAIN program uses the scaling parameter *L* in the following formula for computing *h*, the histogram bar height displayed above an given column in a CH alignment:

$$h = \frac{r^{1-L}}{1-L} \quad \text{for } 0 \le L < 1. \qquad \text{(A1-1)}$$

The variable *r* in Equation A1-1 represents the selective pressure associated with functional divergence of the foreground residues from the background residues at that position. More specifically, *r* corresponds to the number of random trials required to obtain, by chance alone, the observed difference between foreground and background residue compositions based on a ball-in-urn statistical model described in reference [2].

Adjusting the value of *L* up or down over the range $0 \le L < 1$ adjusts the contrast in the opposite direction—that is, down or up, respectively. To see why this is so, consider what happens to *h* at the extremes of this range: When *L* = 0, *h* is a linear function of *r* inasmuch as Equation A1 becomes *h*=*r*. As L → 1, however, the histogram height *h* approaches a logarithmic function of *r* inasmuch as the derivative of Equation A1-1,

$$h' = \frac{1}{r^L}, \qquad \text{(A1-2)}$$

approaches $h' = \frac{1}{r^1} = \frac{1}{r}$, which is the derivative of $h = \ln r$. In other words, as *L* goes from zero to nearly 1, the rate of change in *h* as a function of *r* goes from being linear to nearly logarithmic. Thus, setting *L* closer to zero raises the contrast (i.e., results in greater differences between histogram bar heights for distinct values of *r*) whereas setting *L* closer to 1.0 lowers the contrast.

When the program option "–contrast=<*int*>" is used to specify the number of divergent residue positions to highlight, the CHAIN program automatically adjusts the value of *L* so that only the specified number of columns (as input by the user) have a histogram bar height greater than 1; the CHAIN program only highlights those columns of a CH alignment where *h* > 1.

11. Appendix 2. Informal syntax for vsi files.

Informal syntactic specifications for the context free grammar used for vsi files are given in Table A2-1; carrot bracketed terms in column 1 of Table A2-1 are defined in Table A2-2.

Table A2-1.. Informal syntactic specifications for the vsi language The CHAIN program translates the patterns in the first column into corresponding RasMol scripts, using the Unix programs lex and yacc [29].

Syntactical pattern	RasMol graphic effect	Example string
File<int>**=**<name>**.pdb:**<chain>	Load pdf file[1]; set default chain[2]	File2=RAN/1BYUAHM.pdb:A
view<int>**:R=**<int>,<int >,<int >**;T=**< int>,<int >,<int>.	Rotation & translation[3]	view2:R=140,32,-28;T=-3,-21,262.
	Show backbone trace[4]	
<int>-<int>.<color>	Implicit chain, width	15-90.G (uses default chain,width)
<int>-<int><chain>.<color><int>	explicit chain, width[5]	15-90A.G200
(<int>)<int>-<int>.<color>	w/ 3D image ID	(2)15-90.G
	Show residue sidechain	
<aa><int>.<color>	implicit chain, width	W28.M
<aa><int><chain>.<color><int>	Explicit chain, width	W28B.M200
<aa><int>.{<color>}	w/ dot cloud	W28.{M}
	Show residue atom(s)	
<aa><int>_<atom>-<hydrogen>.<color>	hydrogen donor atom	G17_ca-1ha.X
<aa><int><chain>_<atom >.<color>	hydrogen acceptor	E70A_oe2.X
<aa><int><chain>_<atom >.{<color>}	w/ dot cloud	E70A_oe2.{X}
<aa><int>_**c-o.**<color>	–C=O group	G17_c-o.X
	Show non-protein molecule	
!<mol><int>.<color>	implicit chain, width	!ATP801.C
!<mol><int><chain>.<color><iint>	explicit width	!MG811C.X250
!<mol><int><chain>.{<color>}	w/ dot cloud	!MG811C.{X}
![<mol0>**]**<int><chain>.<color>	w/ numerals in name	![SO4]300C.X
	Show molecule atom(s)	
!<mol><int>_<atom>.<color>	w/ hydrogen acceptor	!ATP801_o1g.X
!<mol><int>_<atom>-<hydrogen>.<color>	w/ hydrogen donor atom	!GDP220A_n1-hn1.X
	Show water	
HOH<int><chain>.<color>	molecule	HOH2C.X
HOH<int><chain>_<atom>-< hydrogen>.<color>	w/ hydrogen donor atom	HOH2W_oh2-h1.X
<item>,…<item><nl>	Item list[6]	W28.M, E70A_oe2.X, HOH2C.X
<int><item>,…<item><nl>	… w/ file number[7]	2 E70A_oe2.X, ![SO4]300C.X

[1] The integer directly following ‘**File**’ is termed the 3D image ID.

[2] If the chain associated with a backbone trace, residue, molecule or atom is not given explicitly, then the default chain is assumed. Note that the pdb input file needs to have single letter labels for each chain.

[3] Specifies the rotation (in degrees) and translation of the structural coordinates. A convenient way to obtain appropriate values is to adjust the perspective within RasMol and then save a RasMol script, which will contain the corresponding *x*, *y*, *z* rotation and translational values; note, however, that translation along the *z* axis is denoted by the RasMol ZOOM setting. The integer following ‘file’ is the 3D image ID.

[4] The first and second integers in the pattern correspond to the start and end residue positions, respectively.

[5] Providing an explicit width changes the size of atoms or bonds.

[6] One or more items (as defined in Table A2-2) separated by commas.

[7] Items may be assigned explicitly to a specific output image by starting the line on which the item(s) occur with the 3D image ID.

Table A2-2. Descriptions of token variables used in the first column of Table A2-1.

Token	Description	Example
<int>	A positive integer	‘25’
<name>	A pdb filename (a .pdb suffix is assumed)	‘RAN/1BYUAHM’
<chain>	Subunit character (any uppercase letter)	‘A’ (chain A)
<color>	Atom color (magena, red, orange, yellow, green, cyan, blue, white) [1]	**‘M’,’R’,’O’,’Y’,’G’,’C’,’B’,’W’**
<aa>	Amino acid residue (single letter representation)	‘H’ (Histidine)
<atom>	Atom (any pdb hon-hydrogen atom designation)	‘ca’ (alpha carbon)
<hydrogen>	Hydrogen atom (pdb representation)	‘1ha’
<mol>	Small molecule with no numerals in designation (non-protein)	‘ATP’
<mol0>	Small molecule with numerals in designation	‘SO4’
<item>	Any residue, molecule, or water pattern in Table A2-1.	
<nl>	Newline	

[1] Lowercase and uppercase single letter color designations yield different shades.

12. Appendix 3. Example vsi file.

The following vsi file serves as a concrete example and as an aid to descriptions of the vsi language.

```
File3=RAN/1I2MAHM.pdb:A         // Ran-RCC1: focus on interface with RCC1 (Fig. 9E).
view3:R=-59,-73,-176;T=8,-20,348.
3![SO4]1250.X

File4=RAN/1BYUAHM.pdb:A         // Ran-GDP: focus on interface with RCC1 (Fig. 9D).
view4:R=44,29,-58;T=-9,37,276.
4!GDP220A.C,!GDP220A_o3b.X

15-35.R  // P-loop region
G17A.m
 3G17A_c-o.X
 4G17A_n-h.X,G17A_c-o.X,G17A_ca-1ha.X
D18A.Y
 4D18A_od1.X, D18A_od2.X,D18A_c-o.X
K23A.M  // PM1 Mg++/Phosphate site.
 K23A_nz-2hz.X,K23A_nz-3hz.X

91-106.Y  // [FY].[ILV]....[ST][FY] motif loop
R95A.G
 3R95A_n-h.X,R95A_ne-he.X,R95A_nh2-1hh2.X
3Y98A.{Y}  // FY; contact with I137, F82.
4Y98A.Y  // FY; contact with I137, F82.
K99A.C  // next to key Y98 residue.
 3K99A_c-o.X,K99A_nz-2hz.X
N100A.Y
 N100A_nd2-2hd2.X
V101A.G  // intermediate? VIL at superfamily level?
 3V101A_cb-hb.X
P102A.C
W104A.Y        // contacts G10 in P-loop region.
 4W104A_ne1-he1.X
R106A.C
 3R106A_nh1-2hh1.X,R106A_nh2-2hh2.X

#************************ RCC1 subunit ****************************
(3+). (1-2-). (4-1000-).  // turn off all files except File 3

90-99B.W
D95B.W  // DEN in RCC1 from metazoans, fungi, plants and protozoans.
 D95B_od1.X,D95B_od2.X

(3)127-131B.W
D128B.W  // D(r) in RCC1.
 D128B_od1.X,D128B_od2.X

(3)380-388B.W
D384B.W  // Highly conserved!
 D384B_od1.X,D384B_od2.X

(3)407-411B.W
Q409B.W  // contacts K99 backbone c=o; RCC1 Subfamily highly specific.
 Q409B_ne2-2he2.X
```