4D-CHAINS

User Manual and Tutorial

Thomas Evangelidis, Konstantinos Tripsianes

Contents

0.1 ABBREVIATIONS

CS chemical shift
AA amino acid
AAT amino acid type
AAIG amino acid index group
NH-mapping mapping of AAIGs to the protein sequence

Chapter 1

Theory

1.1 Backbone N-H chemical shift assignment workflow outline

The main AAIG-to-residue assignment algorithm is outlined in Figure 1.1 and consists of the following steps:

1. Make lists of peaks for the ^{15}N-H HSQC, 4D-TOCSY and 4D-NOESY spectra.

2. Assign an amino acid index group (**AAIG**) to each N and HN resonance pair of the ^{15}N-H HSQC (hereafter referred as **HSQC**).

3. Group the 4D-TOCSY and 4D-NOESY H and C resonances using the common N, HN resonances they both share with the root HSQC spectrum. In this way each AAIG will correspond to a spin system (protein residue) and will have a set of C and H resonance pairs in the 4D-TOCSY and another set of C and H resonance pairs in the 4D-NOESY. Each pair of C-H resonances corresponds to a hydrogen and a carbon that are covalently bonded. The electromagnetic pulse sequences can detect the α, β, γ, δ or ε hydrogen and carbon pairs. The number of C-H resonance pairs corresponding to an AAIG varies between the 4D-TOCSY and the 4D-NOESY, because

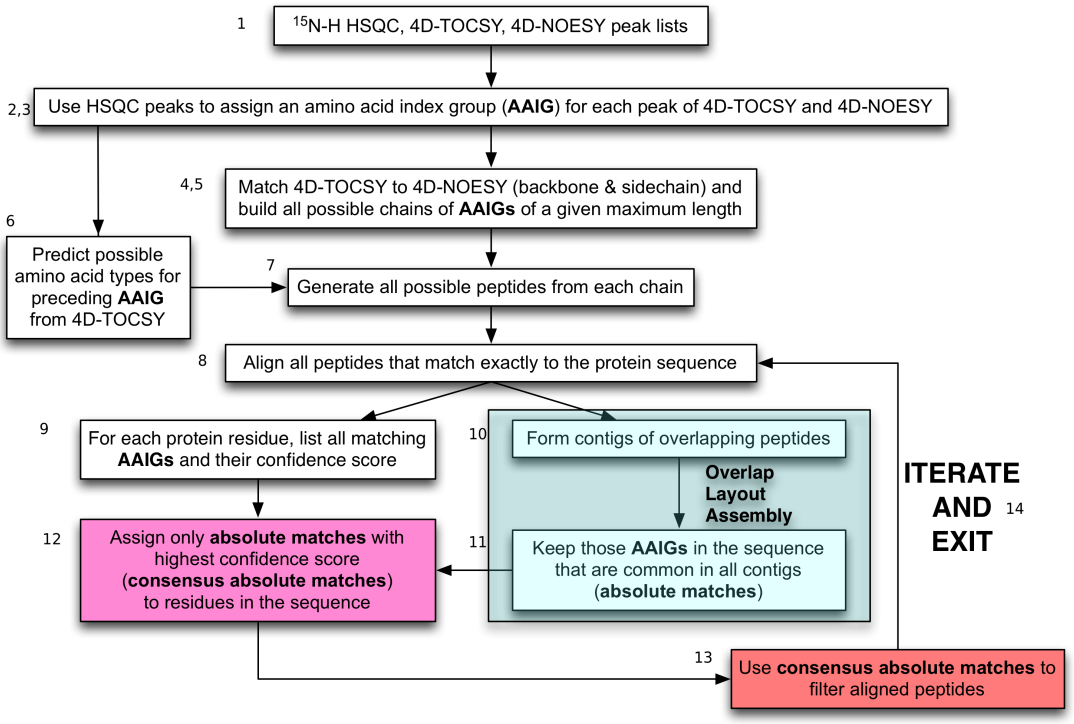


Figure 1.1: The backbone N-H chemical shift assignment algorithm workflow.

in 4D-TOCSY you get only the C-H pairs of the same spin system, but in 4D-NOESY you get C-H pairs of the neighboring spin systems too. The algorithm reduces gradually the tolerances for matching the HN resonance (**rtolHN**) and the N resonance (**rtolN**) of each AAIG in the HSQC until it finds a pair of rtolHN and rtolN for which the AAIG of the HSQC matches the C-H resonances of only one AAIG in the 4D-TOCSY and 4D-NOESY. If an AAIG in the HSQC matches more than one AAIG in the 4D-TOCSY or 4D-NOESY by any combination of rtolHN and rtolN, then the following empirical formula is used to decide which AAIG in the 4D-TOCSY or 4D-NOESY corresponds to the AAIG in the HSQC:

$$\delta_{CS} = \sqrt{(CS_{HN} - CS_{HN}^{HSQC})^2 + (\frac{CS_N - CS_N^{HSQC}}{6})^2}$$

where CS_{HN} and CS_N are the HN and N chemical shifts of an AAIG in the 4D-TOCSY or 4D-NOESY and CS_{HN}^{HSQC}, CS_N^{HSQC} the HN and N chemical shifts of an AAIG in the HSQC. The favored AAIG of 4D-TOCSY or 4D-NOESY is the one with the lower δ_{CS}.

4. Match each C-H resonance pair of every AAIG in 4D-TOCSY to C-H pairs of other AAIGs in 4D-NOESY. Recall that in 4D-TOCSY we "see" the i^{th} protein residue, whereas in 4D-NOESY we see residue $i-1$. In this way we find sequential **connectivities** (backbone and side chain).

5. Use the connectivities to build all possible chains of AAIGs of a given maximum length L.

6. Predict possible amino-acid (aa) types for each AAIG from its 4D-TOCSY C-H resonance pairs using 2-dimensional probability density histograms generated from VASCO database of chemical shifts [ref] (see section 1.1.1 for details).

7. Use the chains of AAIGs and the aa type predictions to build all possible peptides from each chain.

8. Align all peptides to the protein sequence using the Needleman-Wunsch

algorithm [ref] and keep only those that match completely (step 1. in Figure 1.2).

9. For each position in the protein sequence, list all predictions and their associated confidence scores (see section 1.1.2 for details).
10. Form clusters of overlapping peptides named **contigs** (step 2. in Figure 1.2).
11. Find those residues of the protein that are common in all contigs (**absolute matches**; step 3. in Figure 1.2).
12. Find AAIGs that have the highest confidence score and are also absolute matches (**consensus absolute matches**). Assuming that if both methods (scoring and aligning) agree the chances of getting a false positive are minimal.
13. Use the consensus absolute matches to discard contigs that do not agree (step 4. in Figure 1.2).
14. Repeat steps 9-12 6 times.
15. Print the final AAIG-to-residue assignment.

In the main algorithm several parameters can be adjusted, like:

- the maximum chain length L (step 5).
- the tolerances for matching H (**tolH**) and C (**tolC**) chemical shifts (step 4).
- the minimum value of the ratio of matched 4D-NOESY C-H resonance pairs divided by the total number of 4D-TOCSY C-H resonance pairs (**mcutoff**) for every AAIG (step 4).
- the minimum value of the ratio of the number of 4D-TOCSY C-H resonance pairs that matched an aa type (**occupancy**) divided by the total number of available 4D-TOCSY C-H resonance pairs (**acutoff**) for every AAIGs (step 6).

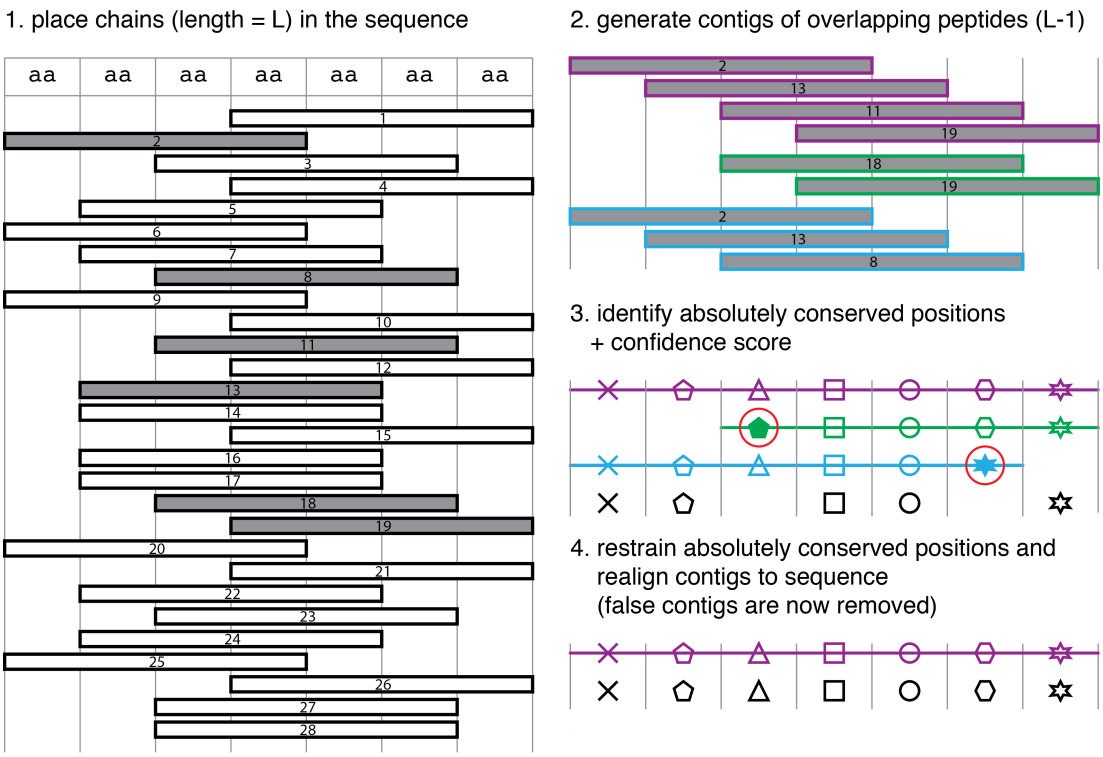


Figure 1.2: Outline of the alignment and scoring algorithm that maps the chains of AAIGs on the protein sequence.

- due to lacking C-H resonance pairs for each AAIG we usually have multiple aa type predictions. So all the ratios described in the previous bullet point are converted to Z-scores using the equation:

$$Zscore = \frac{r - \mu}{stdev}$$

, where r is the ratio for that aa type prediction, μ is the mean ratio for that particular AAIG and *stdev* the standard deviation of all ratios for that AAIG. Then only the prediction above a cutoff value (**zcutoff**) are used to build peptides from AAIG chains, assuming that the true aa type prediction does not have a low Z-score.

- a previous AAIG-to-residue assignment file may be used to discard the peptides in step 8 of the main algorithm that do not comply with the assignment. This way errors can be potentially reduced.

However, in real cases one round of the main algorithm is not sufficient to assign AAIGs to more than half of the protein residues. Therefore, a general case protocol has been devised that consists of 16 cycles, each of which consists of 3 to 4 rounds of the main algorithm, but each time lowering the maximum chain length L. To our experience this protocol is sufficient to ensure that no more AAIGs can be assigned. Typically each round starts with the main AAIG-to-residue assignment algorithm, which produces a pool of connectivities and aa type predictions and a AAIG-to-residue assignment file. That file is used to clean the seemingly wrong connectivities and aa type predictions from the pools. A self-correction mechanism is also applied in this stage. If the pool of connectivities and/or aa type predictions of a particular AAIG were cleaned in a previous round using a consensus absolute match that no longer exists, the original connectivities of that AAIG are recovered. These new connectivities and aa type prediction pools are used for the next round. Once all the rounds have finished a chain linker algorithm is implemented to bridge the chains of AAIGs in the AAIG-to-residue assignment file by utilizing only the remaining pool of connectivities. The chain linker algorithm works like this:

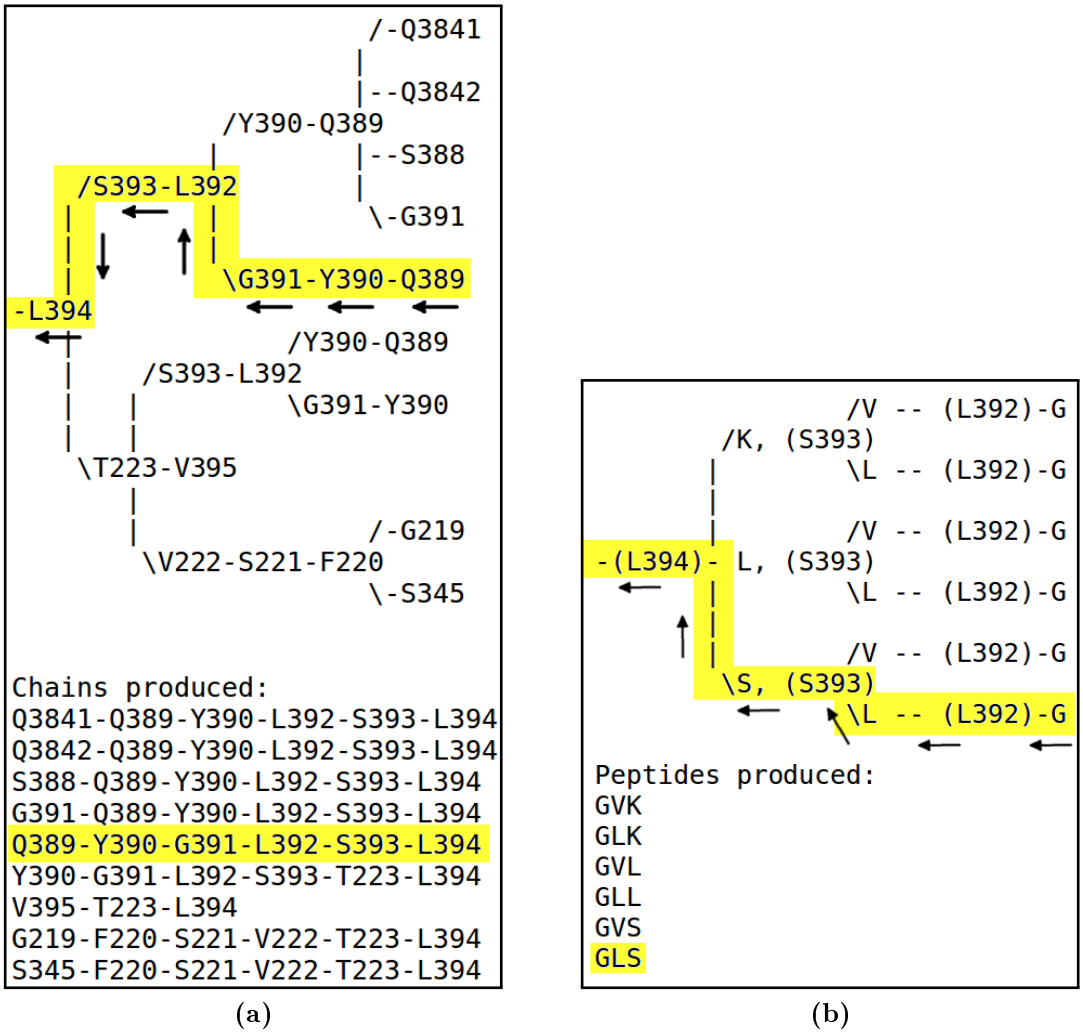
1. read in an AAIG-to-residue assignment file and a pool of connectivities.

2. keep only those connectivities that are above an mcutoff value.
3. make a list of all contigs in the AAIG-to-residue assignment file.
4. look for linkers of arbitrary length that connect the C-term with the N-term of two successive contigs. If only one such linker exists, add it it to the AAIG-to-residue assignment and modify the pool of connectivities accordingly.
5. repeat steps 3 and 4 until no more linkers can be found.
6. reduce mcutoff by a specified value (**mcincr**) and repeat steps 2-5.
7. when mcutoff has reached 0 and steps 2-5 are completed, print the final AAIG-to-residue assignment and exit.

For more details about the backbone N-H assignment protocol please refer to the tutorial (section 3).

The primary bottleneck of the main AAIG-to-residue assignment algorithm is the excessive memory consumption when building all possible peptides from the AAIG chains. The algorithm builds the chains using dendrograms as show in Figure 1.3. It creates a tree root from each AAIG and adds progressively edges and nodes using the information in the connectivities file, until it reaches the maximum chain length L. The last nodes are the leaves of the tree and correspond to the beginning of each chain, because chains are built backwards as the connectivities we have are $i \rightarrow (i-1)$. Then from each leaf the algorithm moves node-per-node until it reaches the root, thus assembling the chains sequentially from the end to the start. The peptides are build in a similar manner from the chains, albeit consuming more memory because the aa type predictions for each AAIG are usually much more than its connectivities. To avoid memory overflow, the parameters L, tolH, tolC, mcutoff, acutoff and zcutoff must be adjusted accordingly in each round. Also higher values of L are usually used in the first rounds due to the greater fidelity of the predictions (the longer the peptide the less likely to be aligned in a wrong position). These high fidelity predictions are utilized to clean the pools of connectivities and aa type predictions for the next round. The chain length L is progressively reduced throughout the protocol as

the pools of connectivities and aa type predictions are also progressively cleaned from wrong predictions. The short chain length L (3 or 4) is required to be able to align the peptides at short regions in the protein sequence that are flanked by gaps in the connectivities, or prolines in the sequence that inherently don't give connectivity information with our method.



(a) (b)

Figure 1.3: Example dendrogram created by the algorithm to find all possible chain combinations. Each node of the dedrogram is named after an amino acid index group (AAIG). The algorithm assembles chains by starting from the leafs of the tree and moving towards the root node-by-node.

1.1.1 2D density histogram generation

2-dimensional probability density histograms (Figure 1.4) were created for every type of C-H carbon pair in the proteins, using the data of VASCO database [ref]. The outliers were identified using the Benjamini-Hochberg p-value correction for multiple hypothesis testing [Boris Iglewicz and David Hoaglin (1993)] and those that had adjusted P-value $< 10^{-100}$ in the proton dimension and P-value $< 10^{-10}$ in the carbon dimension. Subsequently the 2D histograms were smoothed using a Gaussian kernel function as described below.

Let (x_1, x_2, ..., x_n) be an independent and identically distributed sample drawn from some distribution with an unknown density function f. We are interested in estimating the shape of this function f. Its kernel density estimator in one 1D is

$$\hat{f}(x_0) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x - x_i}{h}) \tag{1.1}$$

where K is the kernel function - a non-negative function that integrates to one and has mean zero - and h >0 is a smoothing parameter called the bandwidth. Intuitively one wants to choose h as small as the data allow, however there is always a trade-off between the bias of the estimator and its variance. Due to its convenient mathematical properties, the Gaussian kernel function is often used:

$$K(t) = \frac{1}{\sqrt{2\pi}} e^{\frac{t^2}{2}} \tag{1.2}$$

and the kernel density estimator at a point x_0 in a single dimension is:

$$\hat{f}(x_0) = \frac{1}{nh\sqrt{2\pi}}\sum_{i=1}^{n} e^{\frac{(x_i - x_0)^2}{2h^2}} \tag{1.3}$$

The kernel density estimator at a point $\vec{x_0}$ in 2 dimensions becomes:

$$\hat{f}(x_0) = \frac{1}{2\pi n h_1 h_2} \sum_{i=1}^{n} e^{\frac{1}{2}[\frac{(x_{i1}-x_{01})^2}{h_1^2} + \frac{(x_{i2}-x_{02})^2}{h_2^2}]} \tag{1.4}$$

where n is the data size, h_1 and h_2 are the bandwidth in the 1st and in the 2nd dimensions, respectively.

There are many methods for optimal bandwidth selection. We used Scott's rule of thump:

$$h = n^{-1/6} \tag{1.5}$$

Finally, the 1D probability density histograms were also generated only for Carbons for the cases where one of the 4D-TOCSY peaks has C and H resonance values with 0 2D-histogram probability. In such cases the 1D-probability of the carbon resonance is used instead (e.g. in amino-acid type predictions).

1.1.2 Calculation of confidence scores of the AAIG-to-residue assignments

Inspired by Marin et al., 2004, we have implemented a probabilistic model to calculate the probability of amino-acid types from a set of C-H chemical shift pairs, through the use of statistical properties obtained from BioMagResBank [ref]. In our model we denote by AA^u an amino-acid of type u, where u can be any of the 20 regular amino-acid types. By $AAIG^{CS}$ we denote the set of C-H resonance pairs (chemical shifts) of an AAIG in 4D-TOCSY, which hereafter we will call it an observation. Recall that if $AAIG^{CS}$ corresponds to residue $i-1$, then from these chemical shifts we can predict the AA^u of the preceding residue i, where $i \in [2, N]$ and N is the total number of protein residues. The prior probability of an amino-acid type u is written as $P(AA^u)$, and represents the probability of finding this amino-acid type independent of the observation $AAIG^{CS}$. This probability is given by:

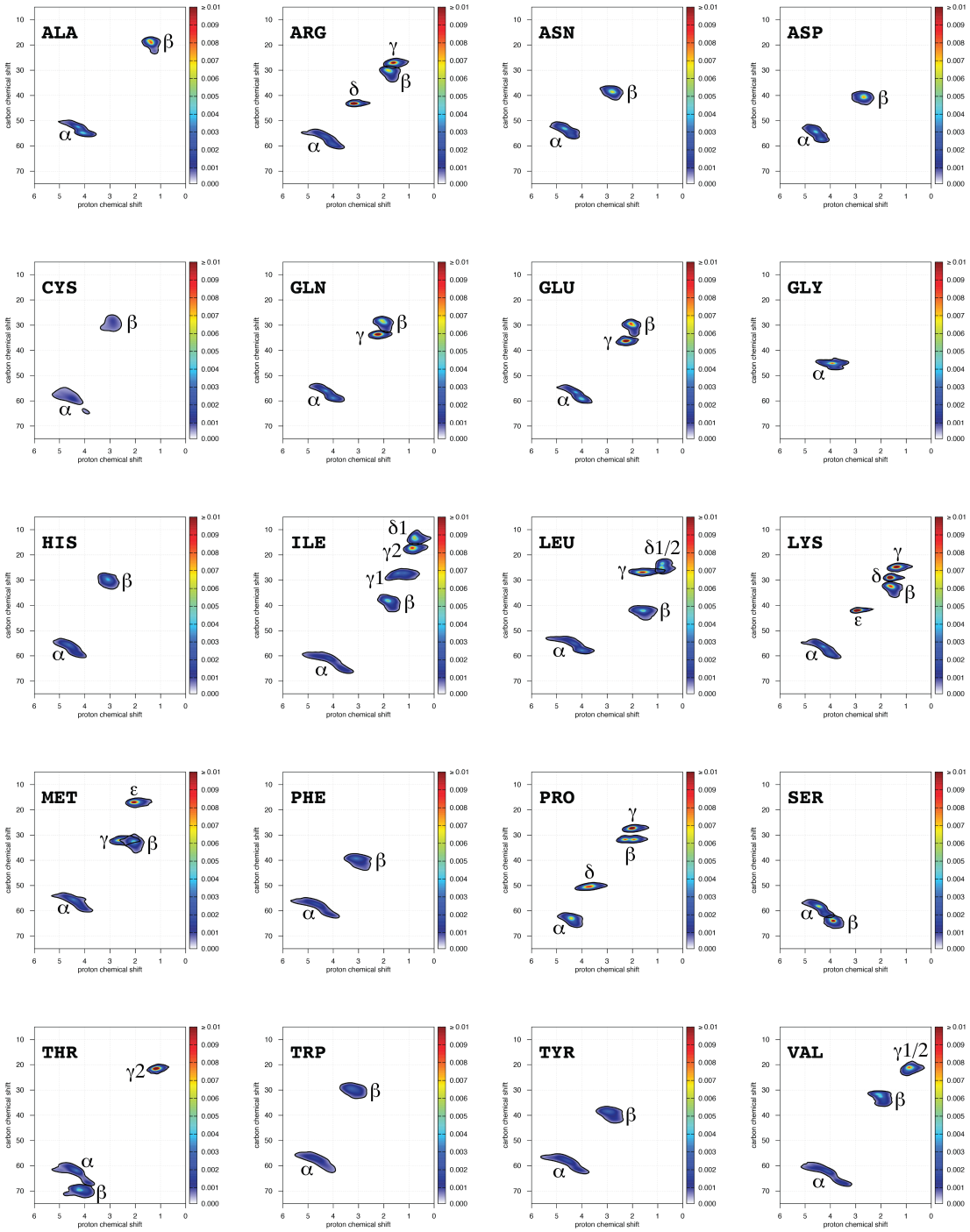


Figure 1.4: 2D probability densities for all the C-H pairs of the 20 amino acids. The densities were derived using the VASCO database, outliers were discarded and the densities were smoothed as described in the text.

$$P(AA^u) = \frac{N^u}{N} \tag{1}$$

where N^u is the number of occurrence of the amino-acid type u in the protein (excluding the N^{th} residue)

Our objective is to calculate the conditional probability $P(AA^u|AAIG^{CS})$, which quantifies how probable is to get an amino-acid of type u given a set of chemical shifts $AAIG^{CS}$. From the Bayes's theorem one can write:

$$P(AA^u|AAIG^{CS}) = \frac{P(AAIG^{CS}|AA^u)P(AA^u)}{P(AAIG^{CS})} \tag{2}$$

where $P(AAIG^{CS}|AA^u)$ is the conditional probability of the set of chemical shift values $AAIG^{CS}$ given the amino acid type u, and $P(AAIG^{CS})$ is the sum of the $P(AAIG^{CS}|AA^u)$ terms over all amino acid types:

$$P(AAIG^{CS}) = \sum^{all\ u} P(AAIG^{CS}|AA^u) \tag{3}$$

The computation of $P(AA^u|AAIG^{CS})$ requires to relate the n observed C-H chemical shift pair values of observation $AAIG^{CS}$ to the n_{max} amino-acid type u C-H nuclei pairs. The fact that we know not only the chemical shifts of each carbon, but also the chemical shifts of its covalently bonded proton, the assignment of the observed chemical shift values to the nuclei of the considered amino acid is usually unique. Nevertheless, there are cases where overlapping of chemical shift ranges produces multiple probable assignment, but for simplicity we consider only the most probable one. In this way we don't need to take into account all alternative assignments in the computation of $P(AAIG^{CS}|AA^u)$. If a number of nuclei larger than the number present in AA^u is experimentally observed ($n > nmax$), the probability $P(AAIG^{CS}|AA^u)$ is set to 0. Otherwise, if $n = n_{max}$, under the assumption that the observations of C-H chemical shift pair values are independent inside a spin system (an AAIG in our case), for a specific assignment of the C-H chemical shift pair values to the spin system nuclei, the probability

$P(AAIG^{CS}|AA^u)$ can be written as a product over the n observed C-H chemical shift pairs:

$$P(AAIG^{CS}|AA^u) = \prod_{j=1}^{n} P(AAIG^{CS_j}|AA^u) \tag{4}$$

where $P(AAIG^{CS_j}|AA^u)$ is the probability to observe the j^{th} C-H chemical shift pair with a values $AAIG^{CS_j}$ in the amino acid AA^u. $P(AAIG^{CS_j}|AA^u)$ can be estimated accurately using the chemical shift frequency histograms from Biological Magnetic Resonance Bank (BMRB) [ref]. For example, an AAIG may have the following pairs of (H,C) resonances: (0.624, 22.196), (0.743, 21.355), (1.583, 36.809), (3.997, 59.786). According to BMRB histograms, the probable aa types of this AAIG are VAL, ILE, LEU, ARG, LYS or PRO....

Finally, in the most frequent case where the number of observed C-H chemical shift pairs is smaller than the number of C-H nuclei pairs ($nnmax$) in a specific aa type u, we take also into account the probability $P(CS_j)$ of the presence of the j^{th} C-H chemical shift pair determined from the analysis of missing chemical shifts in the BMRB assignments. Equation 4 is then rewritten as:

$$P(AAIG^{CS}|AA^u) = \prod_{i=1}^{n_{max}} \frac{1}{2}\{w_H[1_{\{CS_i^H=NA\}}(1-P(CS_i^H))+1_{\{CS_i^H\neq NA\}}(1-P(CS_i^H))]$$

$$+ w_C[1_{\{CS_i^C=NA\}}(1-P(CS_i^C))+1_{\{CS_i^C\neq NA\}}(1-P(CS_i^C))]\}$$

$$\prod_{j=1}^{n} P(AAIG^{CS_j}|AA^u) \tag{5}$$

where $1_{\{CS_i^H=NA\}}$ and $1_{\{CS_i^C=NA\}}$ equal to 1 if the H and C chemical shift of pair i , respectively, are not available (NA) and 0 otherwise. Recall that we use only those resonance pairs for aa type prediction where both H and C chemical shifts are available, therefore if $CS_i^H = NA$ then also $CS_i^C = NA$. $1_{\{CS_i^H\neq NA\}}$ and $1_{\{CS_i^C\neq NA\}}$ equal to $1 - 1_{\{CS_i^H=NA\}}$ and $1 - 1_{\{CS_i^C=NA\}}$, respectively. w_H and w_C are weights for the proton and carbon contribution to the probability. Equation

5 has the advantage compared to Equation 4 that, if no chemical shift is observed (n = 0), a value for $P(AAIG^{CS}|AA^u)$ can be calculated, as the number n_{max} of nuclei in the spin system always differs from 0.

Equation 5 gives the probability of a residue corresponding to an AAIG to be preceded by an amino-acid of type u, but it does not include any position dependence. To incorporate the dependence of that probability to the position i of $AAIG_i$, we must consider both the probability of the chain of AAIGs that $AAIG_i$ belongs to, and the probability of that chain to be aligned in the right position, so that $AAIG_i$ is aligned with protein residue i.

The probability of occurrence of a chain X is given by the product of the probabilities of each connectivity in X:

$$P(X) = \sum_{k=1}^{L-1} P(X_{k \to k+1}) = \sum_{k=1}^{L-1} \frac{occupancy_{k+1}}{N_k^{C-H\ pairs}} \tag{6}$$

where $N_k^{C-H\ pairs}$ is number of C-H resonance pairs in the 4D-TOCSY or AAIG k and $occupancy_{k+1}$ is the number of matching C-H resonance pairs in the 4D-NOESY of AAIG $k+1$ (see step 4 in main AAIG-to-residue assignment algorithm).

In step 8 in the main AAIG-to-residue assignment algorithm we eliminate all AAIG chains that cannot be aligned to the protein sequence. For those that are aligned without any mismatch, we calculate an alignment score S_{align} using a BLOSUM similarity matrix [ref], which quantifies the importance of the alignment of the chain to a specific amino-acid sequence. Taking all the above into account, the probability of assigning an AAIG from chain X to residue i is given by:

$$P(AA_{i-1}^u|AAIG_i^{CS}, X) = P(AA^u|AAIG^{CS}) * P(X) * S_{align} \tag{7}$$

where $P(AA^u|AAIG^{CS})$ is given by Equations 2, 3, 5 and $P(X)$ by Equation 6. Since several different chains may be aligned at overlapping position in the protein

sequence, for each position i there may be multiple possible AAIGs. Therefore, to calculate the confidence score $P(AA^u_{i-1}|AAIG^{CS}_i)$ of the assignment of an AAIG to protein residue i, we sum over all the chains that contain that specific AAIG aligned at position i:

$$P(AA^u_{i-1}|AAIG^{CS}_i) = \sum^{all\ chains} P(AA^u_{i-1}|AAIG^{CS}_i, X) \tag{8}$$

1.2 Carbon and Proton chemical shift assignment

In the previous stage we mapped the AAIG on the protein sequence. In this stage we want to assign C-H types to every TOCSY peak of each AAIG, given the aa type. The TOCSY assignment algorithm briefly works like this.

1. Start reading the AAIGs that were mapped to the protein sequence in the 1st stage from the N-terminus to the C-terminus of the protein.

2. Do C-H type assignment for each AAIG i (recall that we know the aa type from the 1st stage). First the 2D-histogram probability is found for every peak and if it is 0, then the 1D-histogram probability of the carbon only is used instead. The probability of a peak to be of type $C_t - H_t$, where t is the type of the C-H pair, is derived from the respective 2D-histogram. If it is 0, then the 1D-histogram probability of the carbon only is used instead.

3. The final C-H type assignment is not conducted for every peak individually. First the peaks are clustered according to their carbon resonance using a progressively lower cutoff (starting from 0.200001 ppm) and Carbon groups (**C-groups**) consisting of 1 or 2 peaks are formed. The program calculates a **consensus score** for each C-group of being a particular Carbon type (CA, CB, CG, CD, CE, etc.), by default $\frac{prob1*prob2}{0.5*(prob1+prob2)}$ where $prob1$ and $prob2$ are the individual probabilities of the peaks encompassed by the C-group. This consensus score is used to find the most probable Carbon type prediction for each C-group. The best combination of C-H type assignments of the TOCSY peaks of a particular AAIG, is that which has the highest

possible product of consensus C-group scores. Give an example!!!

Since NOE-Rosetta reads NOESY resonances to identify the NOEs that will be used as restraints in structure calculations, it is important to match the assigned TOCSY peaks to the respective NOESY peaks. In addition, provided that 4D-NOESY contains more information than the 4D-TOCSY, it is possible to find the resonances of C-H types that could not be found in TOCSY (namely the missing C-H types for each residue). The NOESY assignment algorithm works in 3 iterations.

Iteration 1:

1. Start reading the AAIGs that were mapped to the protein sequence in the 1st stage from the C-terminus to the N-terminus of the protein.
2. Match the TOCSY assigned peaks to the NOESY peaks of the respective AAIG, using tolerance 0.4 ppm for carbon and 0.04 ppm for proton.
3. At each position i, match the peaks of AAIG i to those of AAIG $i+1$. For these kind of peaks (**i->i+1 matched**) we are more confident that they belong to the AAIG i.
4. Do C-H type assignment for each AAIG i with missing C-H types in TOCSY (recall that we know the aa type from the 1st stage) of these i->i+1 matched peaks. First the 2D-histogram probability is found for every i->i+1 matched peak and if it is below a percentile threshold value (i.e. percentile 0.8) then the peak is not used. NOESY peaks that have normalized intensity value below a specified threshold (0.1 by default) are not used either. The final probability of an i->i+1 matched peak to be of type $C_t - H_t$, where t is the type of the C-H pair, is given by the product of the 2D-histogram probability and the intensity of the NOESY peak transformed by an exponential function, e.g. $2Dprob * (100 * intensity^2)$.
5. The peaks are clustered according to their carbon resonance using a progressively lower cutoff (starting from 0.200001 ppm) and Carbon groups (**C-groups**) consisting of 1 or 2 peaks are formed. The program calculates

a consensus score for each C-group of being a particular Carbon type (CA, CB, CG, CD, CE, etc.), by averaging the individual probabilities of the peaks encompassed by the C-group. This consensus score is used to find the most probable Carbon type prediction for each C-group.

6. Once you iterate over all AAIGs, do C-H type assignment for each AAIG i which had absolutely no peaks in TOCSY (patched N-terminal residues in the 1st stage) using the same procedure as in steps 4-5.

7. Finally, copy from TOCSY all the assigned peaks that could not be matched with any NOESY peak.

Iteration 2:

1. Repeat the same procedure as in iteration 1, but this time using all the unassigned NOESY peaks of AAIG i to find the missing C-H resonances of each residue.

Iteration 3:

1. Repeat the same procedure as in iteration 2, but only for the CD-HD and CE-HE of TYR or PHE, as well as the missing C-H resonances of LYS and ARG (usually CD-HD or CE-HE).

Chapter 2

INSTALLATION AND USAGE

2.1 Download and Installation

The 4D-CHAINS code is available at https://github.com/tevang/4D-CHAINS. To download the code (clone the repository) from github:

git clone https://github.com/tevang/4D-CHAINS.git

Or to update an existing repository, enter the main directory and type:

git pull origin master

The next step is to install all Python dependencies. Enter the main directory and type:

python setup.py install

or if you need sudo privileges to install new python packages:

sudo python setup.py install

Place the bin/ folder in your PATH environment variable if you want to call 4D-CHAINS executables from every directory. E.g. place the following line in your .bashrc:

export PATH=<path to 4D-CHAINS/>/bin:$PATH

2.2 Usage

4D-CHAINS works by reading a protocol file that contains a list of directives and their respective values. The directives are divided into compulsory, which the user must define in order 4D-CHAINS to proceed to the analysis of the spectra, and optional, which the user must leave undefined, in which case the default values will be used.

2.2.1 COMPULSORY DIRECTIVES

The compulsory directives of 4D-CHAINS are categorized into "files" and execution "parameters".

FILES:

fasta template sequence file in fasta format.
E.g.
>prot
ANIVGGIEYSINNASLCSVGFSVTRGATKGFVTAGHCGTVNATA

HSQC 2D N-H HSQC root spectrum. E.g.
N201N-H 118.685 8.433
I202N-H 125.618 9.167
V203N-H 123.768 6.021
.....

4DTOCSY 4D TOCSY (HCTOCSYNH) file. E.g.
?-?-?-? 3.983 51.496 118.695 8.431
?-?-?-? 1.340 19.234 118.698 8.430
?-?-?-? 4.998 52.252 125.616 9.165
.....

4DNOESY 4D NOESY (HCNOENH) file. The 2nd column is the

proton resonance, the 3rd the carbon, the 4th the amide nitrogen, the 5th the amide proton and the last one the intensity of the peak (optional but gives better results if present). E.g.
?-?-?-? 4.276 57.681 133.465 8.951 19344416
?-?-?-? 3.887 64.875 133.466 8.951 115502000
?-?-?-? 3.658 68.443 133.472 8.952 10664100
.....

PARAMETERS:

doNHmapping do NH mapping ("True") or skip it ("False").

doassign4DTOCSY do assignment of 4D-TOCSY peak list ("True") or skip it ("False").

doassign4DNOESY do assignment of 4D-NOESY peak list ("True").

doassignonly4DNOESY skip the 4D-TOCSY assignments and do only 4D-NOESY assignment ("True"). The default is to do both 4D-TOCSY and 4D-NOESY assignments ("False").

The following compulsory directives define the number of NH mapping cycles. All of them must have a consistent number of values separated by ",". In each cycle, 4D-CHAINS does iterative NH mapping using each time peptides of different length. It starts the first iteration by building long peptides, does the NH mapping, and then uses the mapped amino acid index groups (AAIGs) as restraints for the next iteration, which is conducted using peptides shorter by one amino acid (AA). Each cycle consists of the number of iterations that is necessary to bring the peptide length form **first_length** to **last_length**. As such, the

following values

first_length=6,6,6,6,6
last_length=4,4,4,4,3

instruct 4D-CHAINS to run 5 cycles, the first 4 cycles consist of 3 iterations, one with 6mer peptides, one with 5mers and one with 4mers, whereas the 5th cycle consists of 4 iterations, one with 6mers, one with 5mers, 4mers and finally 3mers.

In each cycle you can control the values of the following parameters:

first_length the largest peptide length of each cycle, separated by ','.

last_length the shortest peptide length of each cycle, separated by ','.

mcutoff a floating point number between 0.0-1.0 (percentage), designating how many of TOCSY resonances of an AAIG should match with resonances in the NOESY of another AAIG in order for the two AAIGs to be considered sequential (namely a connectivity between them is added). E.g. 0.5 means that if at least half of the TOCSY peaks of an AAIGs match with NOESY peaks of another AAIGs, then the two of them are considered connected that controls the number of connectivities. The lower the mcutoff, the more connectivities are added, therefore caution should be practiced when selecting values for this directive.

zmcutoff a floating point number (Z-score) controlling how many connectivities satisfying the given mcutoff will be retained for chain formation. The lower the zmcutoff, the more connectivities are retained, therefore caution should be practiced when selecting values for this directive.

zacutoff a floating point number specifying the lower Z-score for an AA

type prediction to be considered as valid. For example, a value of -1.0 means that all AA type predictions with probability higher that then mean probability minus 1 standard deviation, will be considered when translating chains to peptides. The lower the zacutoff, the more AA type predictions are considered., therefore caution should be practiced when selecting values for this directive.
rst_from_prev_cycle whether to use the NH mappings of the last cycle as restraints for the upcoming cycle ("True") or not ("False").

As such, the following values:

mcutoff=1.0,1.0,0.8,0.6,0.6
zacutoff=-0.5,-1.0,-1.0,-0.5,-1.0
zmcutoff=0.0,0.0,-1.0,0.0,-1.0
rst_from_prev_cycle=False,True,True,True,True

mean that the 1st cycle is conducted without using any pre-existing NH mappings as restraints, the mcutoff will be 1.0, the zmcutoff 0.0, the zacutoff -0.5. The 2nd cycles will be conducted using the NH mappings of the 1st cycle as restraints, while only the zacutoff will be altered to -1.0. Likewise for the rest 3 cycles.

The user is able to define his own protocol for NH-mapping using the above directives. However, 4D-CHAINS provides you with 3 test protocols which varying exhaustiveness level. In level 1, 5 cycles are conducted, each one with lower cutoffs than the previous. In level 2, 10 cycles are conducted, while in level 3 16 cycles are carried out. To get a sample protocol file with the default parameters of each exhaustiveness level, use the "-writeprotocol" and "-exhaustiveness" arguments of "4Dchains.py" script. E.g.

4Dchains.py -writeprotocol -exhaustiveness 3

2.2.2 OPTIONAL DIRECTIVES:

Below are listed all the optional directives, categorized by the stage they are used in (peptide generation, peptide alignment, modify connectivities and amino acid types, chain linker, TOCSY assignment, NOESY assignment). The most important of them are written in black font, while the rest in gray.

Finally, there are some optional directives that give you more control over the assignment process. These are:

GENERIC OPTIONAL DIRECTIVES FOR ALL STAGES:

rstart is the number of the first residue in the protein sequence. 4D-CHAINS will try to guess the starting residue number from the HSQC file, but it is recommended the user to define it with this directive.

PEPTIDE GENERATION:

rst_file is a pre-computed or edited table with the N-H mapped to the protein sequence, or otherwise, AAIGs mapped to sequence. This file will be used in the very first cycle and iteration to clean the connectivities and predicted AA types that do not agree with the AAIG mappings. That will result in fewer connectivities and AA type predictions and therefore in few peptides. This directive is recommended for advanced users only.

con_file instead of a rst_file, one can provide a pre-computed or edited file with connectivities that will be used in the very first cycle and iteration of the NH mapping workflow.

aa_file like con_file, but with pre-computed or edited AA type predictions.

PG_tolH tolerance for the proton resonance when matching NOESY peaks to TOCSY peaks. (default 0.04)

PG_tolC tolerance for the carbon resonance when matching NOESY peaks to TOCSY peaks. (default 0.4)

PG_wH weight in (0.0-1.0] to apply on aa type prediction from aliphatic H resonances. (default 0.1)

PG_wC weight in (0.0, 1.0] to apply on aa type prediction from aliphatic C resonances. (default 1.0)

PG_resoncut the minimum number of C-H resonance pairs for a TOCSY index group to start predicting aa types from both C-H or C only resonances. (default 3)

PG_zmin minimum number of aa type predictions required to apply the Z-score cutoff. The fewer the predictions the more inaccurate is Z-score (default: 4)

PG_transform type of mathematical transform to apply on the probabilities P[TAAIG(i)|aatype(i-1)]. Allowed values are: "None", "log", "log10", "boxcox_pearson", "boxcox_mle". (default None)

PG_probprod select the best C-H type assignment combination based on the product of probabilities of the individual C-H assignments (default: True)

colorgray textbfPG_probmodel If '1' the probability of each peak will be given by [wH*1Dhist(H)+wC*1Dhist(C)]/(wH+wC). If '2' then by 1Dhist(H)*1Dhist(C). (default 2)

PG_cgrpprob The way to calculate the total score of a set of chemical shift assignment (default 2). Can be: 0: just multiply all the probabilities of the individual peaks The following values control how to calculate the consensus probability of each C-group. The total score will be the product of this consensus C-group probabilities. 1: average; 2: sqrt(prob1*prob2) ; geometric mean 3: (prob2-prob1)/(log(prob2)-log(prob1)) ; logarithmic mean 4: (prob1*prob2)/((prob1+prob2)/2.) ;composite average 5: prob1*prob2 ; product of probabilities When the C-group contains only one peak, the respective probability will be prob1 in all cases. (default 2)

PG_2dhist use 2D BMRB histograms for aa type prediction (default: True)

PG_log convert aa type prediction probabilities to logarithmic scale and then calculate Z-scores, if the min(probability)/max(probability) > 1000. (default: False)

PG_delpred delete aa type predictions with probabilities which are 1000, 10000 or 100000 times lower than the highest, if the highest is >10e-10, >10e-20, <=10e-20, respectively. (default False)

PEPTIDE ALIGNMENT:

PA_matrix matrix file for alignment. Allowed values: EBLOSUM30, EBLOSUM40, EBLOSUM50, EBLOSUM60, EBLOSUM62-12, EBLOSUM70, EBLOSUM80, EBLOSUM90, EBLOSUM35, EBLOSUM45, EBLOSUM55, EBLOSUM62, EBLOSUM65, EBLOSUM75, EBLOSUM85, EBLOSUMN. (default EBLOSUM90)

PA_gapopen gap open penalty. (default 100.0)

PA_gapextend gap entension penalty. (default 10.0)

PA_cpl characters per line in the consensus alignment file. (default 162)

PA_useall use all peptides (not only overlapping which is the default) for prediction score calculation. (default False)

PA_idcutoff sequence identity cutoff. (default 1.0)

PA_minoverlap minimum number of overlapping TOCSY amino acid indices. (default: maximum peptide length -1)

PA_compress compress intermediate files to save disk space with the cost of computation time. (default: False)

MODIFY CONNECTIVITIES AND AMINO ACID TYPES:

MC_keepgly if GLY is the top ranked prediction with a probability at least 2 orders of magnitude greater than the 2nd prediction, keep only

GLY. (default: True)

MC_keepala if ALA is the top ranked prediction with a probability at least 2 orders of magnitude greater than the 2nd prediction, keep only ALA. (default: True)

MC_nocorrect do not do self-correction. (default: True)

MC_addconn add missing connectivities. (default: False)

MC_bigchains keep only connectivities that form chains of length equal or greater of the specified value. This options is useful when you have a big protein and you want to reduce the number of possible peptides. (default: None)

MC_mcutoff number 0.0-1.0 saying how much of TOCSY resonances should match in the NOESY in order to consider it a possible match. (default: 0.8)

MC_zmcutoff a real number specifying the lower Z-score for a resonance match to be retained for chain building. (default: 0.0)

MC_maxocc keep only the connectivities of a TOCSY index group with the highest occupancy or those that differ from the maximum by -maxocc number (recommended: 0). (default: None)

CHAIN LINKER:

CL_patch place single TAAIG groups at the N-terminals if the respective connectivity exists. (USE WITH CAUTION!). (default: False)

CL_multicon allow linker extension even if multiple alternative connectivities (including the correct one) are present. (default: False)

CL_mcincr matching cutoff increment. (default: 1.0)

CL_mcmin minimum matching cutoff to be used. (default: 0.0)

GENERIC C-H ASSIGNMENT OPTIONAL DIRECTIVES

NHmap is pre-computed or edited table with the N-H mapped to the protein sequence, or otherwise, AAIGs mapped to sequence. This file must be provided in case doNHmapping=False (namely, skipp NH-mapping), otherwise 4D-CHAINS will look for a file named "4DCHAINS_NHmap" in the working directory.

4DTOCSY_assignedNH is the 4D-TOCSY peak list with N-H assignments. This file must be provided in case you want to skip NH mapping (doNHmapping=False) and proceed directly to 4D-TOCSY assignment.

4DTOCSY_assignedall is the fully assigned 4D-TOCSY peak list. This file must be provided in case you want to skip 4D-TOCSY assignment and proceed directly to 4D-NOESY assignment (doassign4DTOCSY=False and/or doassignonly4DNOESY=True).

4DNOESY_assignedNH is the 4D-NOESY peak list with N-H assign-

ments. This file must be provided in case you want to skip NH mapping (doNHmapping=False) and proceed directly to 4D-TOCSY assignment, or directly to 4D-NOESY assignment (doassign4DTOCSY=False and/or doassignonly4DNOESY=True).

probprod select the best C-H type assignment combination based on the product of probabilities of the individual C-H assignments. (default: True)

probmodel If '1' the probability of each peak will be given by [wH*1Dhist(H)+wC*1Dhist(C)]/(wH+wC). If '2' then by 1Dhist(H)*1Dhist(C). (default: 2)

TOCSY-SPECIFIC OPTIONAL DIRECTIVES:

TA_probmode The way to calculate the total score of a set of chemical shift assignment. Can be: 0: just multiply all the probabilities of the individual peaks The following values control how to calculate the consensus probability of each C-group. The total score will be the product of this consensus C-group probabilities. 1: average; 2: sqrt(prob1*prob2) ; geometric mean 3: (prob2-prob1)/(log(prob2)-log(prob1)) ; logarithmic mean 4: (prob1*prob2)/((prob1+prob2)/2.) ;composite average 5: (log(prob1)+log(prob2))/2. (default: 4)

NOESY-SPECIFIC OPTIONAL DIRECTIVES:

NA_wcthres1 The first Carbon type must have probability greater this ratio from the second one of the SAME CARBON GROUP in order to be kept in the final NOESY assignments. (default: 0.0)

NA_bcthres1 The same Carbon type of a Carbon group must have probability greater this ratio from the probability of the SAME CARBON TYPE in any other Carbon group in order to be kept in the final NOESY assignments. (default: 10.0)

NA_wcthres2 same as -wcthres1, but for the 2nd iteration. (default: 0.0)

NA_bcthres2 same as -bcthres1, but for the 2nd iteration. (default: 10.0)

NA_wcthres3 same as -wcthres1, but for the 3rd iteration. (default: 0.0)

NA_bcthres3 same as -bcthres1, but for the 3rd iteration. (default: 0.0)

NA_wcthres4 same as -wcthres1, but for the 4th iteration. (default: 0.0)

NA_bcthres4 same as -bcthres1, but for the 4th iteration. (default: 0.0)

NA_wcthres5 same as -wcthres1, but for the 5th iteration. (default: 0.0)

NA_bcthres5 same as -bcthres1, but for the 5th iteration. (default: 0.0)

NA_percentile1 This arguments is used as a theshold to discard low probability C-H type predictions in iteration 1. Can be 0.9, 0.85, 0.8, 0.75, or 0.0 (no filtering). A 0.9 percentile of value X means that the top 10 percent of the probability density values were above X. (default: 0.85)

NA_percentile2 This arguments is used as a theshold to discard low

probability C-H type predictions in iteration 2. Can be 0.9, 0.85, 0.8, 0.75, or 0.0 (no filtering). A 0.9 percentile of value X means that the top 10 percent of the probability density values were above X. (default: 0.9)

NA_percentile3 This arguments is used as a theshold to discard low probability C-H type predictions in iteration 3. Can be 0.9, 0.85, 0.8, 0.75, or 0.0 (no filtering). A 0.9 percentile of value X means that the top 10 percent of the probability density values were above X. (default: 0.8)

NA_percentile4 This arguments is used as a theshold to discard low probability C-H type predictions in iteration 4. Can be 0.9, 0.85, 0.8, 0.75, or 0.0 (no filtering). A 0.9 percentile of value X means that the top 10 percent of the probability density values were above X. (default: 0.8)

NA_percentile5 This arguments is used as a theshold to discard low probability C-H type predictions in iteration 5. Can be 0.9, 0.85, 0.8, 0.75, or 0.0 (no filtering). A 0.9 percentile of value X means that the top 10 percent of the probability density values were above X. (default: 0.8)

NA_int1 use peak intensities in iteration 1. (default: True)

NA_int2 use peak intensities in iteration 2. (default: True)

NA_int3 use peak intensities in iteration 3. (default: True)

NA_int4 use peak intensities in iteration 4. (default: True)

NA_int5 use peak intensities in iteration 5. (default: True)

NA_ithres1 use relative peak intensity threshold (default: 0.1)

NA_ithres2 use relative peak intensity threshold (default: 0.0)

NA_ithres3 use relative peak intensity threshold (default: 0.1)

NA_ithres4 use relative peak intensity threshold (default: 0.0)

NA_ithres5 use relative peak intensity threshold (default: 0.0)

NA_itrans1 transform the intensities in iteration1. Can be 1: do nothing; 2: (default: 4)

NA_itrans2 transform the intensities in iteration2. Can be 1: do nothing; 2: (default: 2)

NA_itrans3 transform the intensities in iteration3. Can be 1: do nothing; 2: (default: 3)

NA_itrans4 transform the intensities in iteration4. Can be 1: do nothing; 2: (default: 2)

NA_itrans5 transform the intensities in iteration5. Can be 1: do nothing; 2: (default: 1)

NA_probmode The way to calculate the total score of a set of chemical shift assignment (default: 5). The following values control how to calculate the consensus probability of each C-group. The total score will be the product of this consensus C-group probabilities. 1: average; 2: sqrt(prob1*prob2) ; geometric mean 3: (prob2-prob1)/(log(prob2)-log(prob1)) ; logarithmic mean 4: (prob1*prob2)/((prob1+prob2)/2.) ;composite average 5: prob1 + prob2 ; . (default: 5)

Chapter 3

Tutorial

3.1 Complete protein NMR assignment using 4D-TOCSY and 4D-NOESY data

To run 4D-CHAINS use the 4Dchains.py script. You can do all operations you wish with this script as long as you provide the appropriate protocol file. E.g.

4Dchains.py -protocol protocol.txt

You can find a tutorial for nEIt (248 residues) protein under "tutorials/nEIt/" folder. In that folder you will find the protocol file "nEIt_protocol.txt", which contains all the input parameters for the program, and the following input files:

nEIt.fasta is the fasta sequence of the protein

nEIt_HSQC.list is the N-H-HSQC peak list, which must consists of 3 columns: <label> <N resonance> <HN resonance>

nEIt_NOESY.list is the 4D-NOESY peak list, which must consist of the following 5 columns (the last column can be omitted, but will decrease the accuracy of the assignments): ?-?-?-? <H resonance> <C resonance> <N resonance> <HN resonance> <peak intensity>

nEIt_NOESY.list.curated this file is optional, it is the same as "nEIt_NOESY.list", but contains the supervised peak assignments for proofreading of 4D-CHAINS automatic assignments

nEIt_TOCSY.list is the 4D-TOCSY peak list, which must consist of the following 4 columns: ?-?-?-? <H resonance> <C resonance> <N resonance> <HN resonance> <peak intensity>

nEIt_TOCSY.list.curated this file is optional, it is the same as "nEIt_TOCSY.list", but contains the supervised peak assignments for proofreading of 4D-CHAINS automatic assignments

The protocol files consist of compulsory and optional directives. The compulsory directives must be defined in every protocol file in order 4D-CHAINS to run, whereas the optional can be omitted. For a detailed description of each directive, please refer to the manual. Briefly, the available protocol file "nEIt_protocol.txt" will do full chemical shift assignment of the protein using a 4D-TOCSY and a 4D-NOESY file. First, the N-H resonances are mapped to the protein sequence. These N-H resonances are used to assist the assignment of 4D-TOCSY peaks to aliphatic carbons. Finally, the assignments from 4D-TOCSY are transferred to the respective 4D-NOESY peaks, and the rest of them are assigned de novo. To run full protein assignment for the nEIt, do:

4Dchains.py -protocol nEIt_protocol.txt

At the end, you will find several output files and a directory named 4DCHAINS_workdir, which contains all the intermediate files for backtracking (only for advanced users). The output files in the working directory are the following:

nEIt_TOCSYnum.list is the original 4D-TOCSY with N-H assignments

nEIt_NOESYnum.list is the original 4D-NOESY with N-H assignments

nEIt_HSQCnum.list is the original N-H-HSQC with the correct existing labels preserved and the other lines labeled as X1N-H, X2N-H, etc.

4DCHAINS_Nhmap is the table with the N-H mapped to the protein sequence

4DCHAINS_assigned_NH_HSQC.sparky is the original N-H-HSQC with all the assignments made by 4D-CHAINS

4DTOCSY_assignedall.xeasy is the original 4D-TOCSY with all the assignments made by 4D-CHAINS, in xeasy format

4DTOCSY_assignedall.sparky is the original 4D-TOCSY with all the assignments made by 4D-CHAINS, in sparky format

4DNOESY_assignedall.sparky is the original 4D-NOESY with all the assignments made by 4D-CHAINS, in sparky format

4DNOESY_assignedall.proofread.xeasy is the original 4D-NOESY with all the assignments made by 4D-CHAINS, in xeasy format. Since the files with supervised 4D-TOCSY and 4D-NOESY assignments were provided, this file contains at the end of each line comments like "<CORRECT>" or "<WRONG>".

From all the output files, the most important is "4DNOESY_assignedall.proofread.xeasy", which will be passed as the input "chemical shift file" to autNOE-Rosetta for protein structure prediction, as described below.

Index