Modern triple resonance protein NMR backbone assignment, using AlphaFold and unlabelling to drive chemical shifts assignment in proteins

Hannah Morris

A Thesis to the University of Kent for the degree of

M.Sc. by Research in Biochemistry

DECLARATION:

No part of this thesis has been submitted in support of an application for any degree or other qualification of the University of Kent or any other University or Institution of learning.

ACKNOWLEDGEMENTS:

I would like to give thanks to everyone that has supported me throughout the 2 years of my Masters project. I would like to express my gratitude to my supervisor, Dr. Gary Thompson for his valuable expertise, guidance and support throughout my project – not to mention him having lots of patience with me learning to code!

I would also like to give thanks to Dr Alex Heyam from the University of Leeds for starting this whole project in the first place as well as providing vital support along the way.

In addition, I would also like to offer thanks to my secondary supervisor, Dr. Jose Ortega-Roldan for his insightful suggestions and encouragement to complete this thesis. I would also like to thank everyone in his lab team (2022-2024) as well as Cian Bartholomew for picking up from where I left off to continue developing this exciting project!

Lastly, I want to thank my friends and family for supporting me through this project.

Thank you ☺

ABSTRACT:

NMR is a powerful technique to study the structure, dynamics and interactions of proteins. However, to obtain atomic resolution data, NMR signals must first be correlated with specific chemical groups – a problem called assignment. The software AlphaFold has shown to be a great advancement in modern-day science. Until now, structural analysis of proteins had been bottlenecked by months/ years' worth of slower techniques that were traditionally used to determine a protein structure

In this project we have designed and applied a semi-automated assignment program called SNAPS (**S**imple **N**MR **A**ssignment using **P**redicted **S**hifts). This allows the user to go from a set of NMR spectra of a protein with a known 3D AlphaFold or X-ray crystallography structure to a fully assigned chemical shifts of the backbone resonances. In addition, unlabelling experimental data can be incorporated into the use of the program to generate more reliable assignment data by helping the program along in the mapping of the amino acids for assignment by providing places in the assignment where the amino acid it is known.

The program was largely written by Dr Alex Heyam from the university of Leeds but testing scripts and well as a NEF importer was written to ensure the program was user-friendly and ensured rigid, fool-proof datasets being imported for backbone assignment. The program also underwent large-scale testing on roughly 150 proteins with known 3D crystal structures as well as ^{15}N unlabelleing data being implemented to improve assignment. AlphaFold structures could also be implemented for use in the program also. Assignment was as good as 86-88% dependent upon parameters, with the unlabelling being slightly more effective with assignment (for PDB crystal structures).

Table of Contents

TABLE OF FIGURES:

TABLE OF TABLES:

1 INTRODUCTION:

1.1 The protein Folding problem and AlphaFold

This project aims to challenge the topic of whether there can be a fully automated method for assignment and structural determination of proteins using known structural information. Over the past few years there have been great scientific advancement achieve this – generating new working solutions to predict structure of proteins. The main root of the issue lies with the protein folding problem, this is one of the largest problems in structural biology today and can be split into three subsequent parts:

(i) What is the folding code?

(ii) What is the folding mechanism?

(iii) Can we predict the native structure of a protein from its amino acid sequence?(1)

However, the development of AlphaFold2 (AF) and its ability to predict 3D structure (and more or less answering the latter two parts of these questions) allows for new opportunity for programs to build on AF data presented. This project focuses on the use of AF to assist with the assignment of chemical shift data derived from NMR data.

All the important processes in a cell replication, defence and homeostasis involve proteins. We understand that it is very important to study and understand the dynamics, structure, and function of proteins. Proteins form biological pathways essential for survival. Research into proteins drives research for things such as oncology, immunology, structural biology etc (2). So, a way to predict their 3D structure as well as to rapidly and accurately map their dynamics is massively important to researchers across the biological sciences. In summary, this project assesses a new method to fully assign backbone resonances of protein from a 'known' 3D structure.

In recent years, new software tools AF and ROSETTAFOLD has demonstrated a rapid way of determining proteins 3D structure(3), (4). AF is believed to have made a significant impact across structural biology as well as research areas that require protein structure information such as drug discovery, protein design, prediction of protein function. AF can be referred to as one of the greatest contributions to AI in the scientific community(5). Figure 1 shows the increased publications and referencing of AlphaFold, AlphaFold2 and AlphaFold3 from 2019-2025.

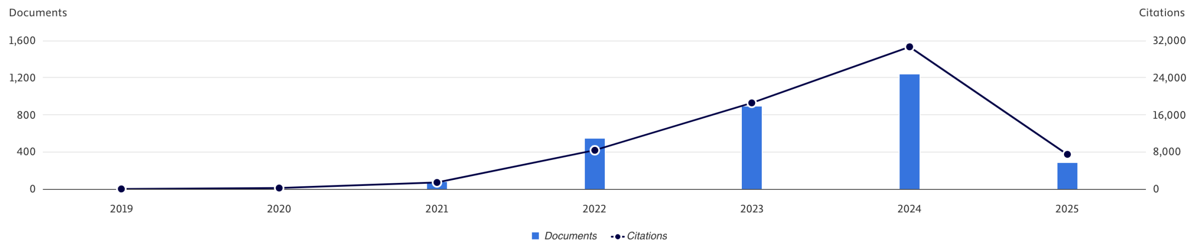
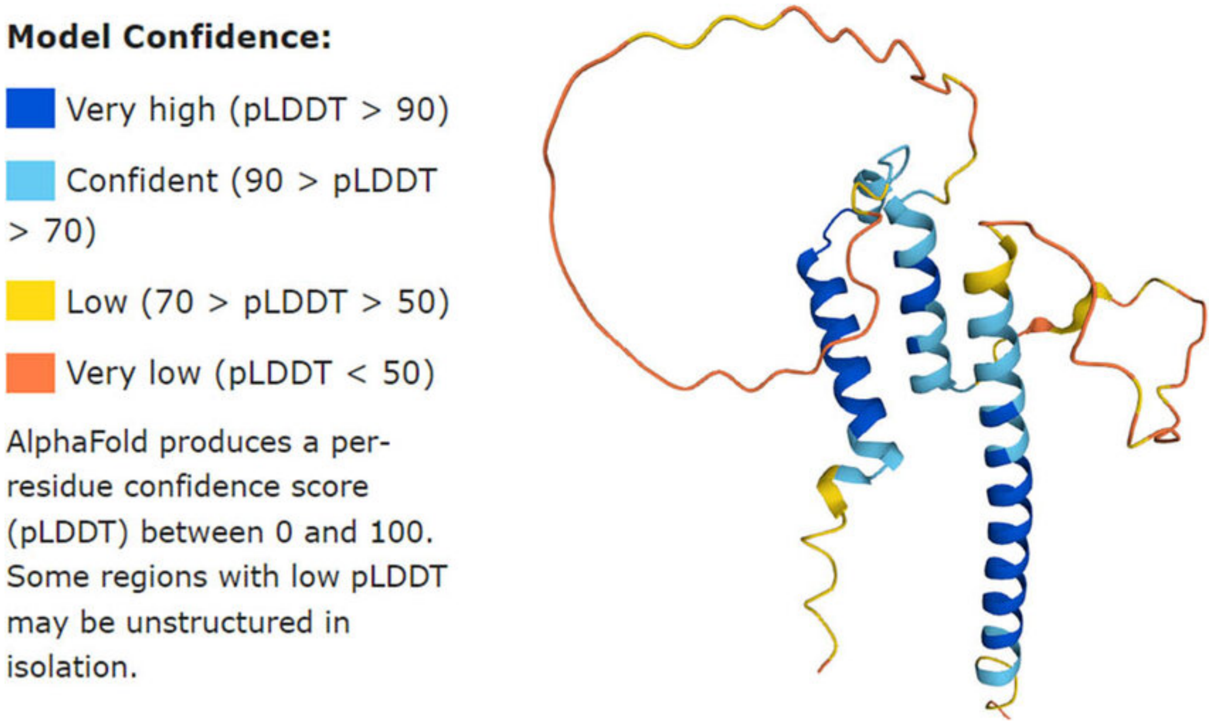


Figure 1- Number of AlphaFold, AlphaFold2 and Alphafold3 publications and citations over time. (6)

AF is a software developed by DeepMind; it works by applying an algorithm to predict the overall structure of a protein from its amino acid (AA) sequence based on statistics from many known structures. AF performed exceptionally well in the Critical assessment of structure prediction (CASP) 13 and 14, which is a community driven experiment held annually to determine the best methods of protein structure prediction (7).The CASP assessment is carried out by using recently solved protein structures that are not yet deposited in the PDB, so it is a blind test for all methods of structure determination. In short, AF has somewhat solved the issue of part 3 of the protein folding problem in that it is able to give an accurate prediction of the proteins native state from its amino acid sequence. In recent studies, AF models of small proteins have rivalled (and in some cases exceeded) the accuracy of solution NMR structures, this shown great success for the program and opens up the greater conversation for future developments of the program. AF structures of small single domain proteins in solution without structural templates can also display great accuracy(8).

However, this doesn’t mean to say that AF doesn’t have any shortcomings/ points for improvement and further research. It is notably poor at predicting structures of entirely novel proteins. Furthermore, given that proteins are moving, fluid structures, AF cannot predict all conformations, only the most likely structure and has varying confidence of how likely the given structure is (9). Figure 2 illustrates how AF can predict protein structure with various confidence scores for different regions.

Proteins can be involved in highly complex biochemical pathways and can undergo conformational changes to facilitate binding of other proteins, ligands etc., therefore, conformational stability is only marginal. In order to grasp a better understanding of protein structures and functions for research purposes (such as health and disease) improvements/ additional data needs to be added to increase the reliability of structure given(10).



Model Confidence:

🟦 Very high (pLDDT > 90)

🟦 Confident (90 > pLDDT > 70)

🟨 Low (70 > pLDDT > 50)

🟧 Very low (pLDDT < 50)

AlphaFold produces a per-residue confidence score (pLDDT) between 0 and 100. Some regions with low pLDDT may be unstructured in isolation.

Figure 2- An exemplar AlphaFold model which shows the varying confidence scores of the predicted 3D structures of the Protein. Yellow and orange regions, display a lower confidence, meaning that there is

less certainty that these regions exhibit the correct 3D structure. (Mid1-interacting protein 1, UniProt #Q9NPA3) (11)

AF has shown to be a great advancement in modern-day science. Until now, structural analysis of proteins had been bottlenecked by months/ years' worth of slower techniques that were traditionally used to determine a protein structure, these will be discussed in the next section. The development of AF consisted of computational methods based off two complementary paths that focused on either:

A) The physical interactions of the protein (Thermodynamics, kinetic simulation)
B) The proteins evolutionary history (Genomic sequencing)

The AF program also demonstrates how protein structure architecture is used to jointly embed sequence alignments (MSAs) alongside pairwise features and the network directly predicts the coordinates of heavy atoms for a given protein from its sequence and aligned sequences of homologues as inputs. (3)

AF is a great recent addition to the world of structural biology and beyond and had can have a myriad of different applications to better advance scientific discovery.

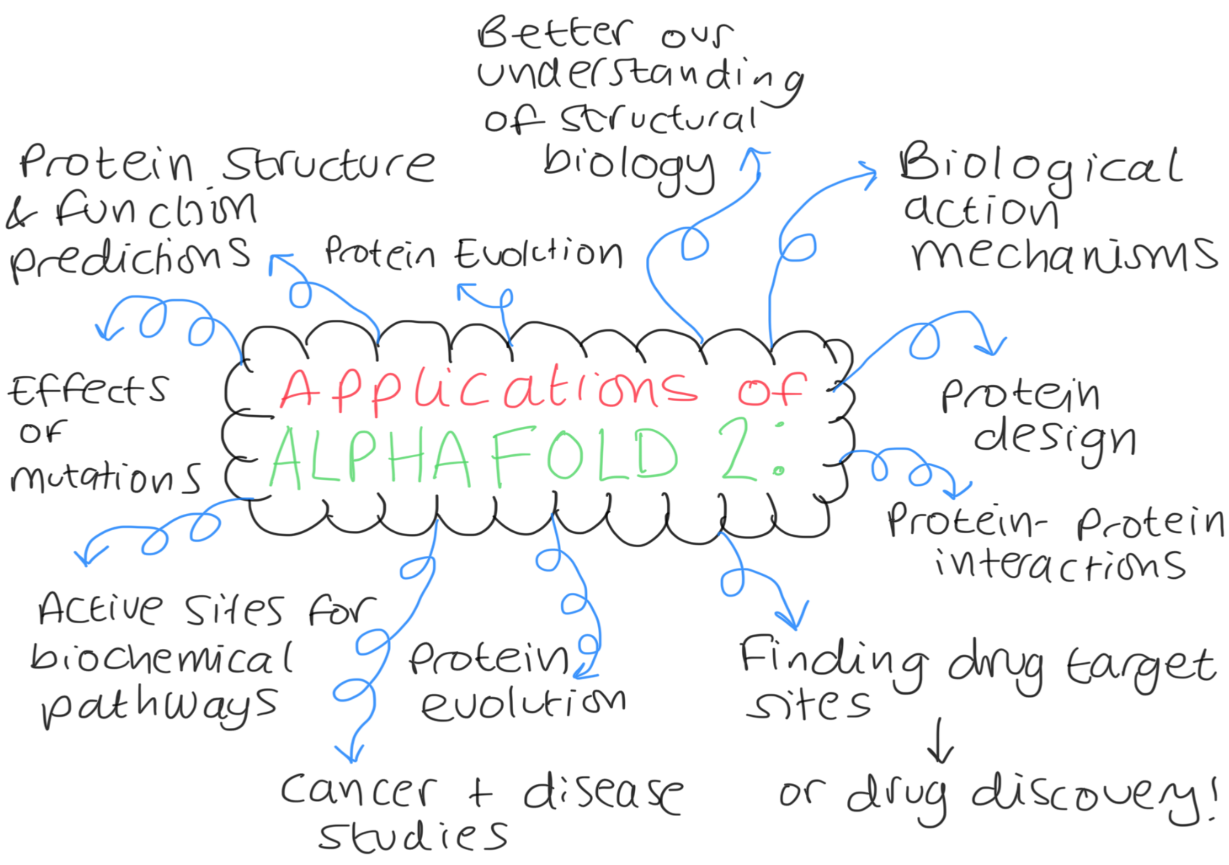

Applications of ALPHAFOLD 2:
Better our understanding of structural biology
Biological action mechanisms
Protein structure & function predictions
Protein Evolution
Effects of mutations
protein design
Protein-Protein interactions
Active Sites for biochemical pathways
Protein evolution
Finding drug target sites
or drug discovery!
Cancer + disease studies

Figure 3- Some of the applications and potential future applications of AlphaFold 2. This is a revolutionary development in the field of protein biochemistry and beyond with the potential to help develop new kinds of medicine as well as have a better understanding of biological processes and evolution!

AF has bought about a lot of questions for scientific advancement such as: how can this now be applied to the wider fields of biology and medicine? (Figure 3) In effect, AF has provided a good 'jumping off point' for a number of aspects of scientific research into structure and function of proteins and other biological molecules and the roles they play in biological pathways etc.

In summary, this project assesses a new method to fully assign backbone resonances of protein from a 'known' 3D structure.

1.2 Traditional protein structure prediction determination methods

Protein structure prediction began in 1961 with Anfinsen's dogma(12)

"A protein's native structure stands for a free energy minimum determined by its amino acid sequence, or in other words, the 3D structure of a protein is only determined by its amino acid sequence."(13)

This hypothesis is the theoretical foundation of protein structure prediction.

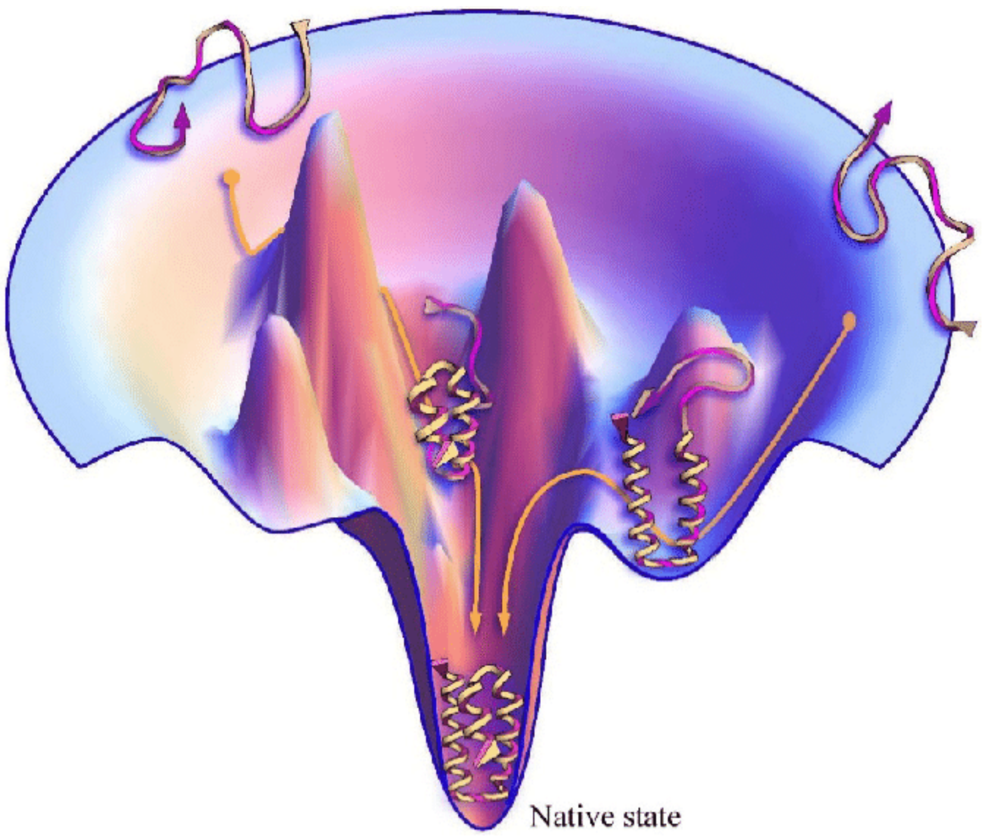


Figure 4- Protein folding is guided by a funnel-shaped energy landscape. The native state being the base of the funnel with different conformations (or misfolding, where the protein can become 'stuck')(14)

Modern-day protein structure determination can be put into three subsequent categories:

Homology modelling: Comparative modelling or template-based modelling is based off of the hypothesis that 3D structures are more conserved than their AA sequence. Similar AA sequences should have similar 3D structures.

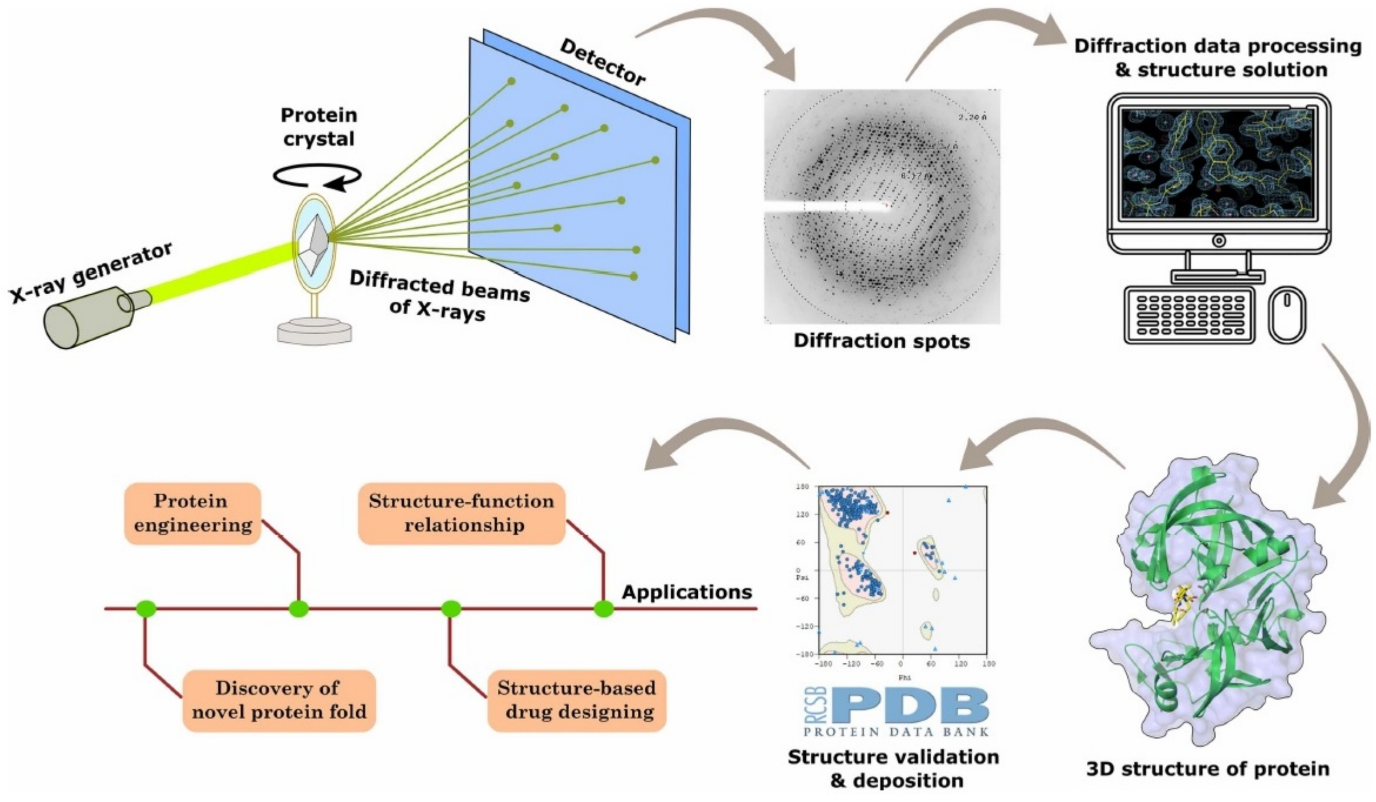
De novo modelling: De novo modelling is a protein structure prediction method based on the “first principles”. Unlike the homology modelling, the de novo modelling does not depend on the known protein structures but generating the 3D structure of a target protein only based on the established laws of physics (quantum mechanics).

Machine learning-based modelling: A strategy that utilises machine learning algorithms and known protein structures to predict the structures of other, ‘target’ proteins, such as AF. (15)

AF is a relatively new method for protein structure prediction; however, experimental techniques have been used over many years to deduce the 3D structures. These can vary in accuracy and their choice largely depend upon the type of protein being investigated. In comparison to AF these techniques all require much more manual work and take significantly longer to deduce a 3D structure as opposed to AF which is extremely rapid. Experimental approaches for determining the 3D structure of proteins typically involve 3 main methods at atomic resolution (Note SAXS isn’t atomic resolutions- so doesn’t appear on this list):

- X-ray Crystallography
- Cryo-electron microscopy
- Nuclear magnetic resonance

1.2.1 *X-ray Crystallography:*



Detector
Protein crystal
X-ray generator
Diffracted beams of X-rays
Diffraction spots
Diffraction data processing & structure solution
3D structure of protein
Structure validation & deposition
RCSB PDB PROTEIN DATA BANK
Applications
Protein engineering
Structure-function relationship
Discovery of novel protein fold
Structure-based drug designing

Figure 5- How the X-ray crystallography experiment works to allow scientists to determine 3D structures of proteins. The protein must be preserved in a crystal and then using X-ray generators. This causes detectable levels of diffraction. This can be processed to give a final 3D structure. (16)

X-ray crystallography has proven to be the most important method for protein structure determination and has assisted in the research of many complex systems. The first work on this technique began in 1912 by Lawrence Bragg, the youngest ever Nobel prize winner (age 25) and is still very commonly used today(17). The technique aims to obtain a 3D molecular structure of the protein of interest from a crystal. A purified protein sample of a high concentration is crystallised and then exposed to an X-ray beam. The beam causes diffraction patterns that can be processed and in turn provide information about crystal packing, symmetry as well as the size of the repeating unit that form the crystal(18). Protein crystallography is a very high-resolution technique, a cocktail of chemicals is added to the protein sample in the hope that an adequate crystal is formed. A single protein crystal can contain millions of molecules. From the spots in the diffraction patterns, computer software converts this into a 3D image of the protein. This process leaves a jigsaw-like puzzle that needs to be solved in order to solve for the correct 3D structure.

The atoms in the molecule are surrounded by a cloud of electrons which define the proteins overall shape. If the amino acid sequence of the protein is known, then this can be used to fit the amino acids within the electron density map(19).

However, there are a few limitations with this technique which make it not the best method for protein structure determination. For example, the sample must be crystallised; this can be a very long process as most proteins are not naturally crystalline. Furthermore, there are often ambiguities in the presented model due to the fact that it can be challenging to visualise the model as a whole because of the effect of crystal packaging (20). The protein must also be analysed outside of its usual cell environment and needs to be embedded in a crystal form at cryogenic temperatures, so may not account for all conformations and removes its natural context(21), especially for proteins with excited states. The protein data bank, or PDB is filled with crystal structures for many given proteins.

1.2.2 Cryo-electron Microscopy:

Cryo-electron microscopy (cryo-EM) is a newer technique than X-ray crystallography. This new approach was developed because of the shortfalls of the X-ray. Cryo-EM encompasses several applications: Single particle analysis (SPA) analyses and images a vast amount of protein sizes with often high throughput and resolution. Cryo tomography (CryoET) visualises structure within cells and is ideal for looking at cellular components- it combats one of the issues with X-ray by providing a cellular context to the macromolecular structures determined with SPA. Lastly, micro electron diffraction (MicroED) is used for the structural determination of a small unique class of crystals that cannot be analysed with traditional X-ray methods(21).

In recent years, addition to the Cryo-EM database (EMDB) has increased exponentially, especially because the resolution of Cryo-EM structures has recently radically improved to the point where atomic resolution structures can be obtained. The method entails flash freezing solutions of proteins/ other biomolecules and then blasting them with electrons to produce microscopic images of individual molecules. The structures produced by this technique are becoming more and more detailed due to continuous advancements in both software and hardware and can produce images of very large molecular machines(22).

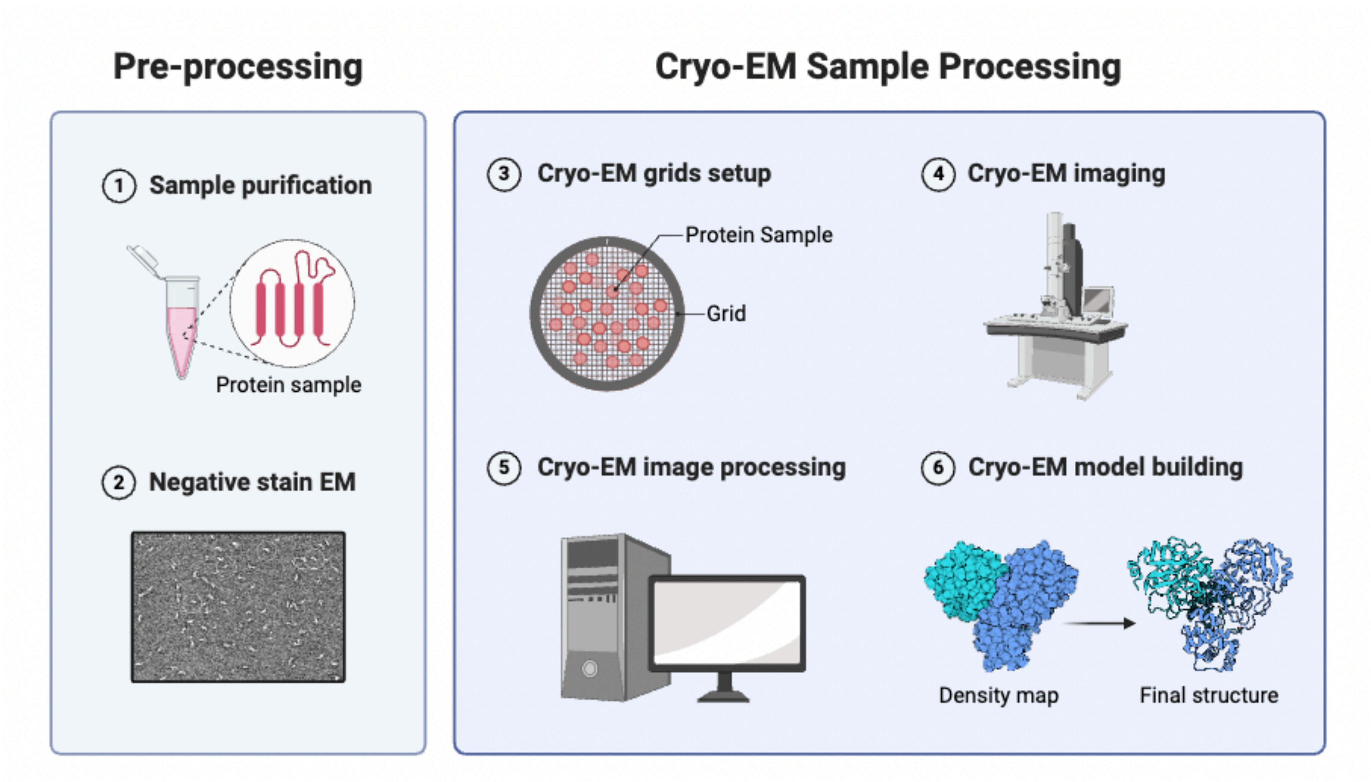


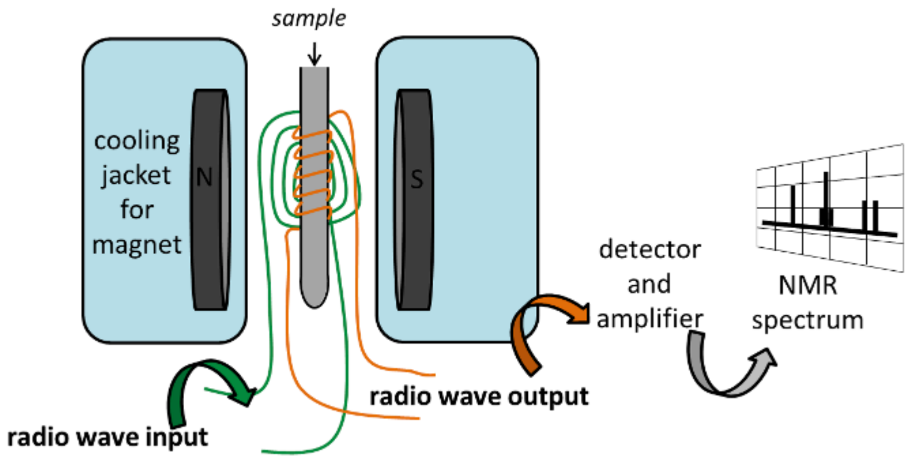
Figure 6- A summary of how data is obtained for Cryo-EM and how this is used to determine the 3D structure of proteins. Sample is collected and purified and a negative stain is applied. The protein is placed on a grid and imaged with the microscope and then processed to generate a 3D model. This figure was created with BioRender.com

Despite its growing successes, this technique is also not without its limitations. The key limiting factor to how much structural information that can be obtained from Cryo-EM lies with radiation damage from the electrons as well as poor signal-to-noise ratio (SNR). Naturally with continuing advancements of experimental and computational methods, this will be set to improve(23). In addition, given that this is a type of microscopy, Cryo-EM can currently only be used on larger protein complexes (usually requiring a minimum of 150kDa) otherwise the protein cannot be detected, and resolution may be poor. The sample also must have a high homogeneity and can be difficult to obtain accurate and reliable results if the protein is very flexible(24).

1.2.3 Nuclear magnetic resonance (NMR):

Nuclear magnetic resonance, NMR is another well-known method used to determine the 3D structure and dynamics of proteins. The key concept surrounding NMR is that many nuclei have a charge and have multiple spins (or intrinsic angular momentum). Because of these properties, an external magnetic field can drive energy transfers(25). The standard set up for an NMR experiment is shown in figure 7 where the sample is placed in between two large magnets and analysed on the console. NMR is reliant on specific types of nuclei that can absorb and then re-release electromagnetic energy at defined frequencies when in a fixed magnetic field. Spin state is also important for NMR. Atoms with a spin state of 0 or 1 (whole numbers) such as ^{14}N and ^{2}H are not compatible with NMR. Spin states must have a value of 1/2 such as ^{31}P or ^{13}C or ^{23}Na has a spin value of 3/2(26). It is because of this; protein labelling is very important for NMR. In order to see proteins on the spectrum, they must be ^{13}C / ^{14}N labelled (which is used in methods later). Any shift in the usual response frequency is due to the nuclei's surrounding environment such as nearby electrons and other magnetic nuclei. This makes it possible to deduce information about the molecular identity and structure of the molecule. NMR is a very advantageous technique as it provides a myriad of structural and dynamic information about a protein. Unlike X-ray, there is no need for any crystallisation, and relatively simple sample preparation. Samples are also not degraded as a result of this experiment; the same sample can be used for repeat experiments/ stay intact for extended periods (27). Despite its successes, analysis of large proteins using NMR can be very difficult and time consuming. Larger protein spectra can be near-impossible to assign. In addition, the sample needs to be of a high purity to have a good signal-to-noise ratio(28). This project concerns NMR data, its applications and advances in protein structure prediction and protein assignment, usage of different isotopes can change the appearance of the spectra and ‘mask’ certain residues, this is called unlabelling (Next chapter). It is a highly advanced technique that can demonstrate a myriad of information surrounding a protein of interest. Further features will be discussed in the next section.

Protein structure determination methods are time consuming and require very specific sample conditions, concentrations and resolution of the equipment used. Compared to AF, the amount of time to manually work out the overall 3D structure is significantly longer. AF can produce a rapid result. However, its accuracy is sometimes questionable. This debate of which method of protein structure determination works best is a critical question for structural biologists today.



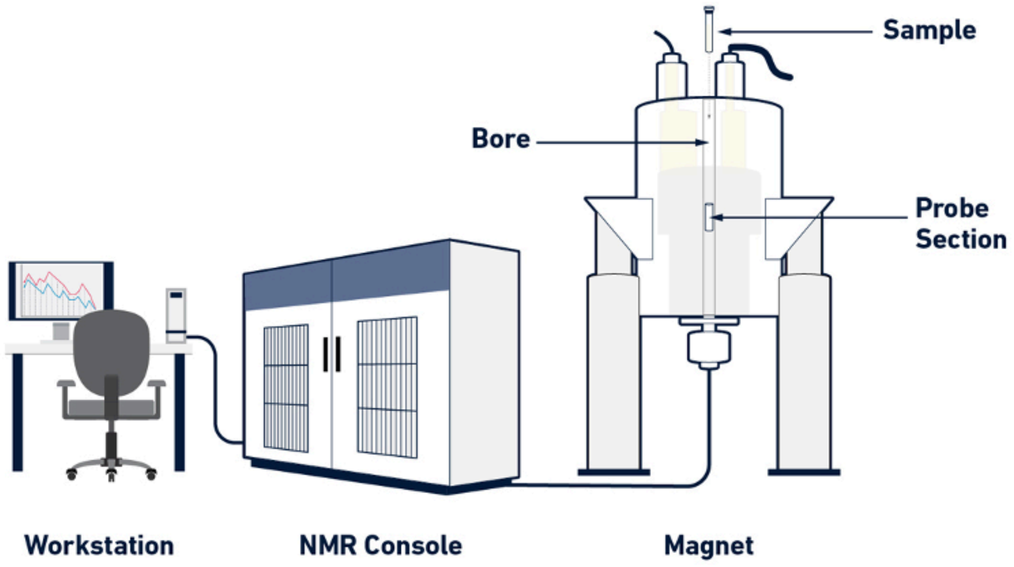


Figure 7-NMR set up. The NMR set up is essentially a workstation which is connected to a console and a spectrometer. Inside the spectrometer, the sample is placed between 2 large and very strong magnets. It is the magnetisation of the sample that is measured with radio wave output that generates a spectrum. The spectra can then be analysed and used to assess/ determine the 3D structures of different proteins and biological molecules.(29) **,** (30)

1.3 NMR, techniques and how assignment works:

1.3.1 NMR theory:

Following on from the very brief introduction to NMR in the previous section, a more detailed view of NMR techniques will be discussed. NMR analysis plays a centre role of this project. In summary, this project encompasses a combination of different NMR techniques as well as AF and applying programming and chemical shift prediction programs to automate the assignment process using a 3D AF or crystallography structure.

1.3.2 Pulses:

In a magnetic field (B_0), nuclear spins will either line up with the field or go against it. Those that go against the field are in a slightly higher energy state. Due to this small difference, more spins will be lined up with the field rather than against it. This creates a net macroscopic magnetisation (pointing upwards). NMR is an insensitive technique Meaning the sensitivity is dependent on the strength of the magnetic field.

A series of pulses on the sample can pass magnetisation from one nucleus to another via spin couplings. This can be predicted and makes it possible to record resonance frequencies of the nuclei involved. Not all nuclei of the same isotope will have the same resonance frequency- as this can be dependent on the chemical group and environment, this is what drives the differences in spectra of molecules that can be read and used to interpret and understand protein structure, function and dynamics.(31)

A simple 90° pulse experiment can be used to demonstrate how spins are aligned with the magnetic field and how applying a pulse can change the direction of magnetisation temporarily and how it can be measured. The 90° pulse experiment is outlined in figure 8 below.

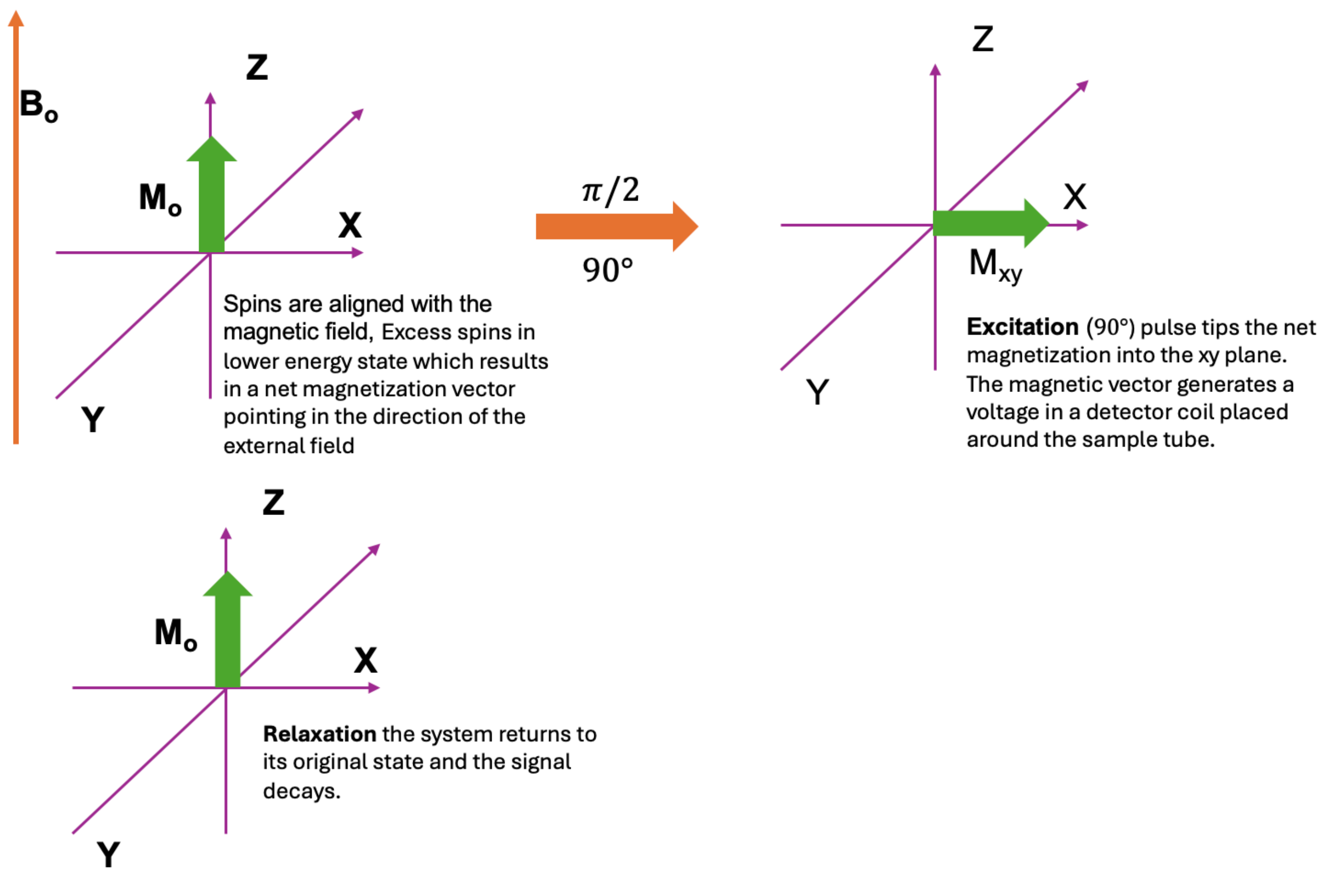

B$_o$
Z
M$_o$
X
Y
Spins are aligned with the magnetic field, Excess spins in lower energy state which results in a net magnetization vector pointing in the direction of the external field
$\pi/2$
90°
Z
X
M$_{xy}$
Y
Excitation (90°) pulse tips the net magnetization into the xy plane. The magnetic vector generates a voltage in a detector coil placed around the sample tube.
Z
M$_o$
X
Y
Relaxation the system returns to its original state and the signal decays.

Figure 8- A diagram of a 90° NMR pulse experiment showing alignment to the magnetic field and the effects of excitation to the spins in respect to the magnetic field

1.3.3 Fourier Transform:

In simple terms, NMR data is a collection of electromagnetic sine curves stacked on top of one another. There is one sine curve per 'active' nucleus. This data can be characterised to study specific atoms in proteins of interest. To convert the sine waves into readable data, the Fourier Transform is applied. This converts the X-axis (time) to frequency (Hz), where 1Hz = 1 per second. The Fourier transform takes a curve and finds the frequency of each sine wave present at that curve. If the curve contains a sine wave with a frequency of approximately 500MHz, then the transform gives this a positive value, if the sine wave is missing at the curve, then it has a value of zero for that frequency(32).

1.3.4 Free induction decay (FID)

The free induction decay (FID) refers to the exponential decay of the sine wave signal induced by an RF pulse. This exponential decay is caused by the loss of in-phase precession of M_{XY} as it spirals upwards from the transverse plane to its original alignment (B_0). (33)

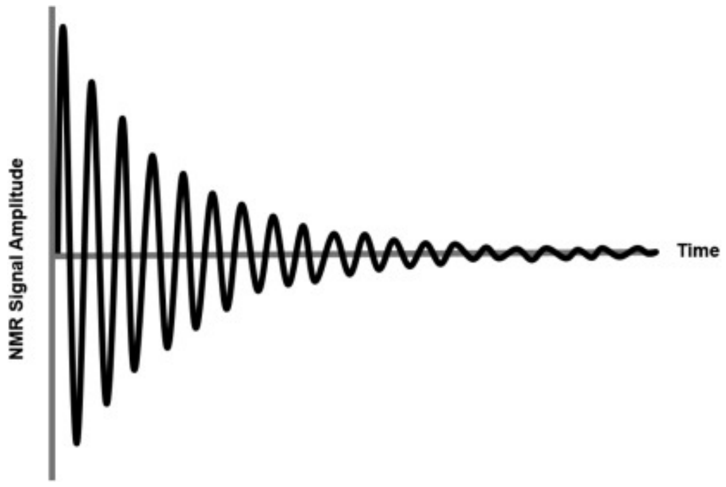


Figure 9- The exponential decay of the sine wave (FID signal) following an RF pulse (33)

1.3.5 Couplings:

During an NMR experiment, pulses are applied to the molecule. In turn, this allows for absorption and re-release of electromagnetic energy. This pulse drives the nuclei to generate the NMR signal. This signal can have excess energy and can transfer this surplus to its neighbours if they are connected by a covalent bond (32).

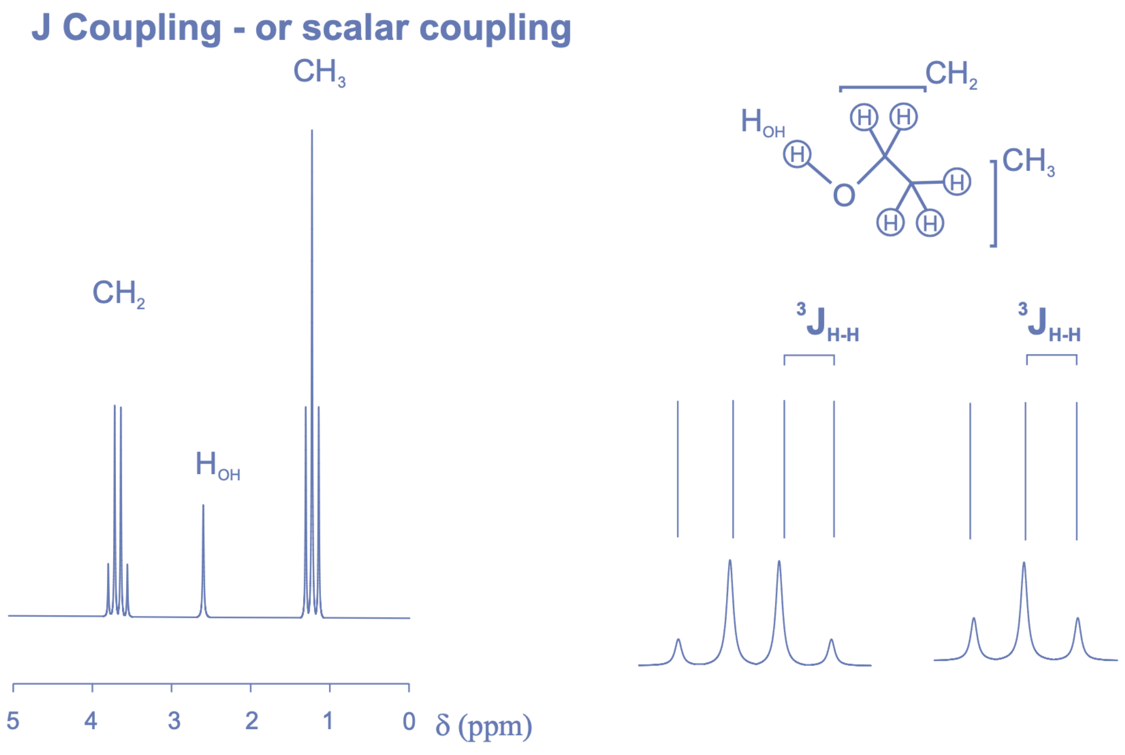


Figure 10- J-coupling in ethanol arises because of nuclear magnetic moments within the molecule, affecting each other through the shared electron cloud- through the covalent bond(34)**. Diagram provided by Dr Gary Thompson.**

The transfer of magnetisation (energy) through covalent bonds is known as J-coupling. This can provide a direct measure of bond strength(35) and provides information on covalent molecular structure. The other type of coupling is Dipole-dipole couplings. This results from the additional magnetic field felt by one spin due to the magnetic dipole moments of other spins in its vicinity (typically for nuclei that are within approximately 5Å of one another). These demonstrate a dipolar field that acts on top of B_0. This change in energy is the potential energy of the dipole, in the field of another magnetic dipole, so they're coupled!(36)

1.3.6 *Protein Backbone Assignment:*

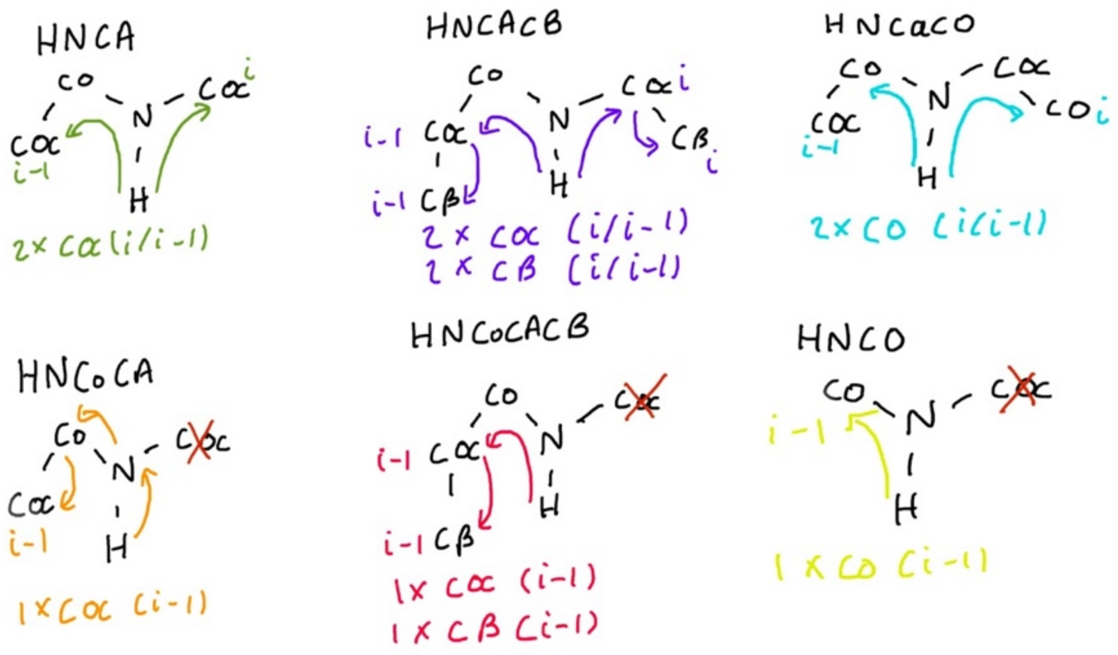


Figure 11- A diagram to show how each type of spectra follows a different electron path to show different results that can be interpreted for assignment. The arrows show the path taken, each of these can help with the puzzle that is assignment.

In order to understand the information in the NMR spectra, the chemical shifts of its atoms must be assigned. A simple backbone resonance assignment for a larger protein typically requires a ^{15}N ^{13}C labelled protein alongside the different spectrum types HNCA, HNCOCA and HNCACO. It may also be seen as beneficial to deuterate the sample as well as move to a higher field strength (TROSY- transverse relaxation optimized spectroscopy) to allow for a greater resolution when the degree of overlap is very large(37). Figure 11 displays what each spectrum shows in reference to each amino acid (six different types of spectra: HNCA, HNcaCB, HNcaCO, HNcoCA, HNcocaCB, HNCO referenced to an HSQC figure 12), each of these can be used in combination to assign any given protein. Assignment of resonances can be a long and challenging task and can often be seen as a bottleneck in the exploration of solution NMR spectroscopy in the study of protein dynamics, structure, and function(38).

Automated assignment is intended to improve this assignment process that is still often not sophisticated enough to ensure a speedy process.

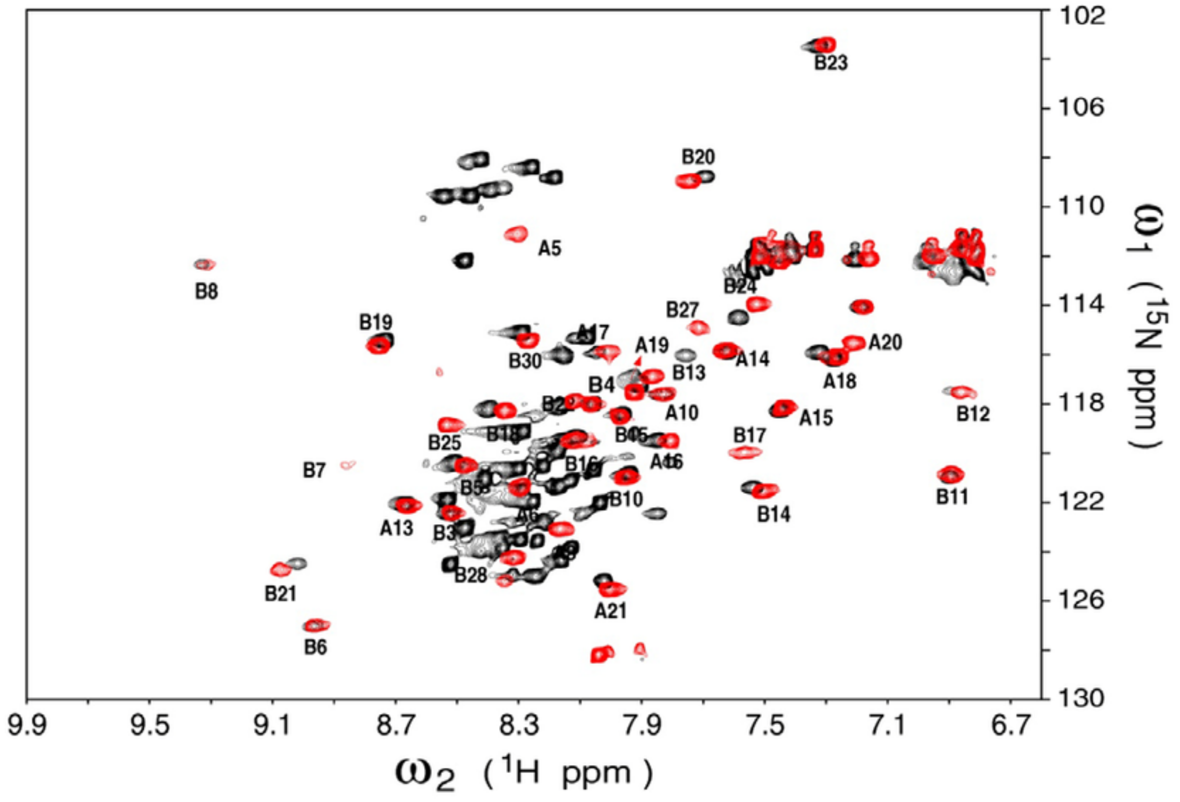


Figure 12- An HSQC spectrum for KP-proinsulin and DKP-insulin at pH 7.1. Each amino acid has a singular peak on an HSQC spectrum. The X axis shows the ^{1}H dimension and the Y axis shows the ^{14}N (in ppm) (39)

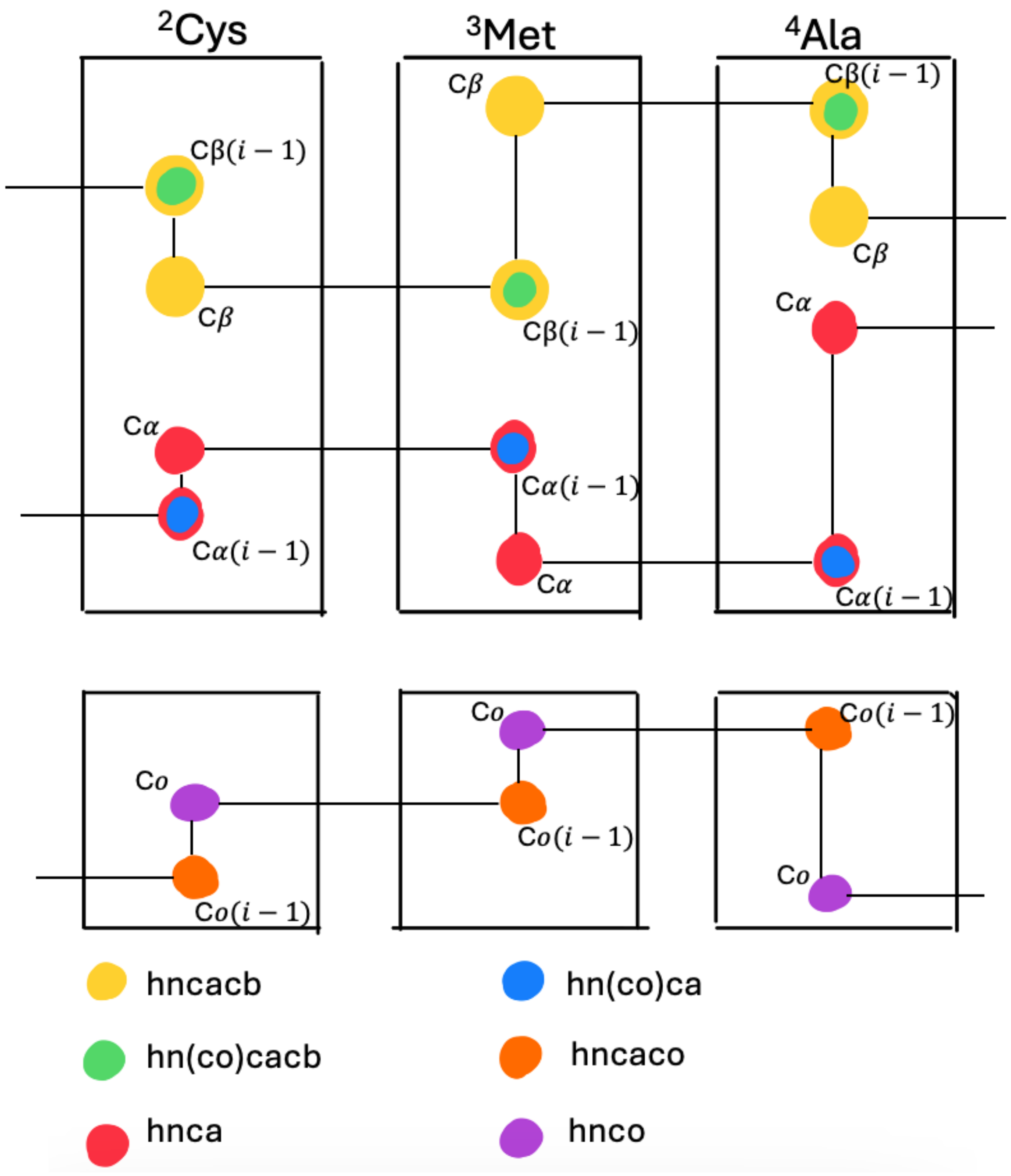


Figure 13- This is an example of the process of assignment with a strip plot. Cα peaks are correlated with the Cα-1 peaks like a jigsaw puzzle.

Each peak on the HSQC spectra represents one amino acid in the protein sequence. The whole point of assignment is to map each one of the peaks presented on the HSQC to each amino acid on the protein sequence. As mentioned above, six different types of spectra are needed to compare and correlate the Cα, Cβ and Co peaks uniquely. Assignment can be viewed as a puzzle, for smaller proteins it is much easier, as there are less pieces. For a large protein, there are significantly more peaks to assign, this can take much more time and the whole thing can be extremely difficult- especially when there are peaks missing.

For assignment, 3D spectra need to be visualised, a common way to do this is with strip plots. For example, for a particular Z-plane of a 3D HNCO spectrum, there would probably only be a few peaks in that z plane. Because of this, there isn't much point in looking at the complete ^{1}H width of the spectrum- the area can instead be trimmed into a strip. This corresponds to a particular ^{1}H AND ^{15}N part of the spectrum whilst showing the complete ^{13}C width(37). This can allow you to pair up the strips like a jigsaw puzzle! Figure 14 shows the $C\alpha$, $C\beta$ chemical shifts and how these can be mapped and demonstrates the use of CCPN assign. HNCO and HNCACO spectra show i-*1* spectra; this is the same information as HNCA and HNCOCA but without the $C\beta$ resonances(40). These are also used on the strip plots to correlate the data as shown by the black lines on Figure 13.

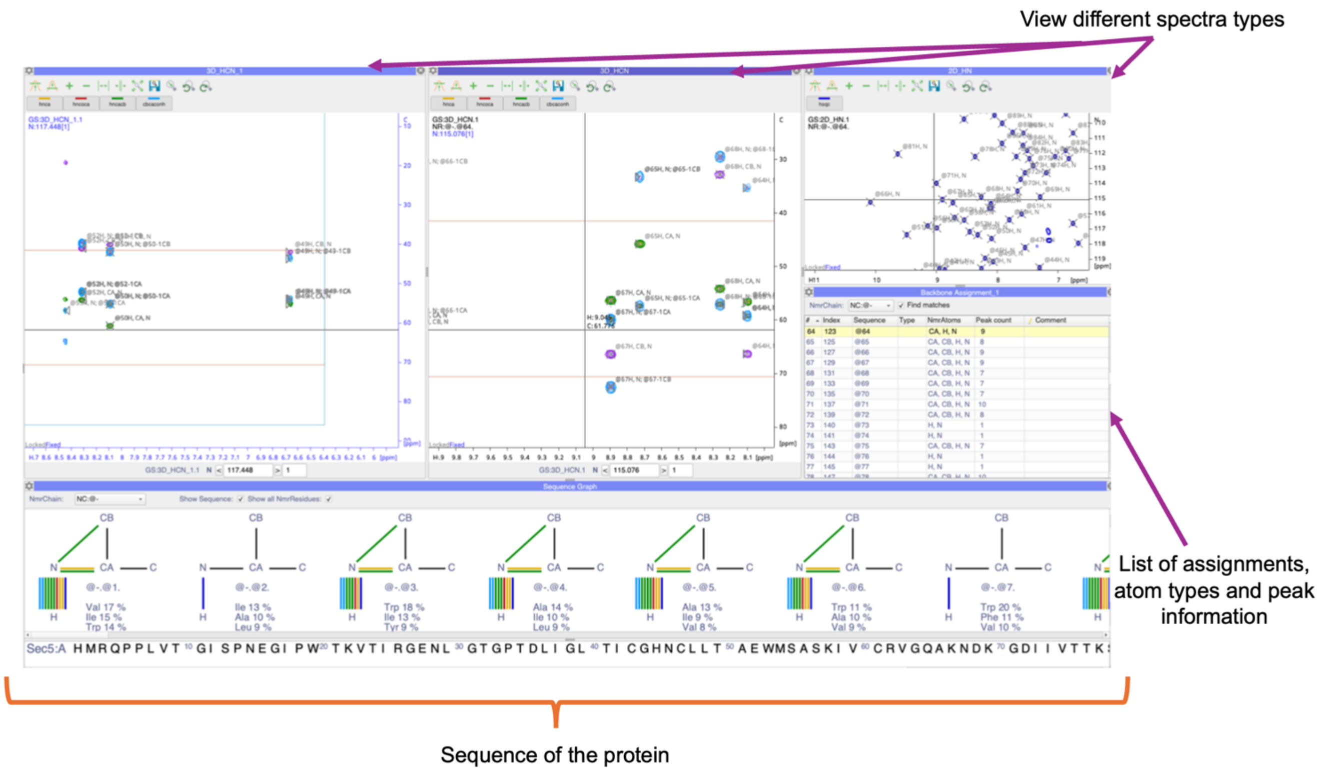

View different spectra types
List of assignments, atom types and peak information
Sequence of the protein
Sec5:A HMRQPPLVT GI SPNEGI PW TKVTI RGENL GTGPTDLI GL TI CGHNCLLT AEWMSASKI V CRVGQAKNDK GDI I VTTK

Figure 14- CCPN Assign set up used for triple resonance backbone assignment. The set up has the HSQC spectrum (with one peak per residue) as well as a peak assignment list. HNCA, HNcoCA, HNCACB and HNcoCACAB spectra is used here for assignment and are shown in the two spectra on the left-hand side. At the bottom there is a sequence and mapping of the residues to map the assignment process visually! (41)

Various software's have been designed to assist in the assignment process. One of which is CCPN assign which allows the user to map out the spectra and assignments on the computer in relation to the protein sequence, a CCPN Assign set up is displayed in figure 14.

1.3.6.1 Spectra size limitations:

With usage of NMR, it is also important to be mindful of which spectrum type to use TROSY is used for larger proteins – which was used in the case of this paper. Proteins with molecular masses under 10kDa be determined using simple homonuclear ^{1}H NMR experiments. Larger proteins (10-30kDa) require non-uniform sampling, ^{13}C and ^{15}N labelling, standard triple resonance. Larger proteins (30kDa+) require TROSY-based techniques and more complex methods(42).

One of the largest proteins assigned using TROSY-based triple resonance experiments is the protein malate synthase G (MSG). This is 723 residues and 81.4kDa in size! (43)NMR has size limitation for when it comes to large proteins due to the overlapping of residues, assignment becomes a near-impossible task.

1.3.7 NMR techniques:

A mentioned previously, for larger proteins, the assignment process can be a bit complex with lots of overlap that can make the task of assignment considerably harder. There are a few scientific innovations in this field that have assisted in making this process more manageable.

1.3.7.1 Residual Dipolar Couplings:

Residual dipolar couplings (or RDCs) are sensitive probes of molecular structure and dynamics in solution. They can provide valuable information for determining 3D structure for a protein of interest.

Dipolar coupling arises from nuclear systems containing adjacent pairs of magnetic nuclei in a magnetic field. The interactions give rise to dipole-dipole interaction (talked about previously in section 1.3.4). In turn, this causes splitting in the signal for each nucleus involved. The resultant dipolar couplings in the solid state are of the order of 20kHz for a ^{1}H-^{15}N nuclear pair with a bond length of 1.0Å. These strong couplings lead to very broad spectra in the solid state. As discussed above, the size of the coupling depends on the distance between 2 nuclei ($1/r^3$) and the direction of their internuclear direction in reference to $B_{o,}$ ($1\text{-}3\cos^2(\Theta)$) the main magnetic field. However, in the solution state this interaction averages to zero due to molecular tumbling.

However, if a molecule in solution is anisotropic in shape and partially / weakly aligned by adding a lyotropic liquid crystal (LLC) to the solution or confinement in a stretched / compressed polymer gel)(44). This leads to the dipolar coupling reappearing as an increase of a decrease in the scalar couplings observed in the spectra, with addition contribution scaled by a factor of 1000-10000 (2-20Hz) typically. This allows the measurement of these so-called dipolar couplings in the solution state.

Dipolar couplings are sued to provide useful long range orientational, and dynamic information about pairs of nuclei [e.g. ^{1}H-^{15}N bond vectors] in the protein / molecule of interest. However, their measurement is limited by the available alignment media and the design and implementation of experiments to measure them, especially in high molecular mass systems where relaxation can be limiting with respect to signal to noise and accuracy. RDCs can be positive or negative, depending on set range of angles that are sampled in the structure of analyte. (45)

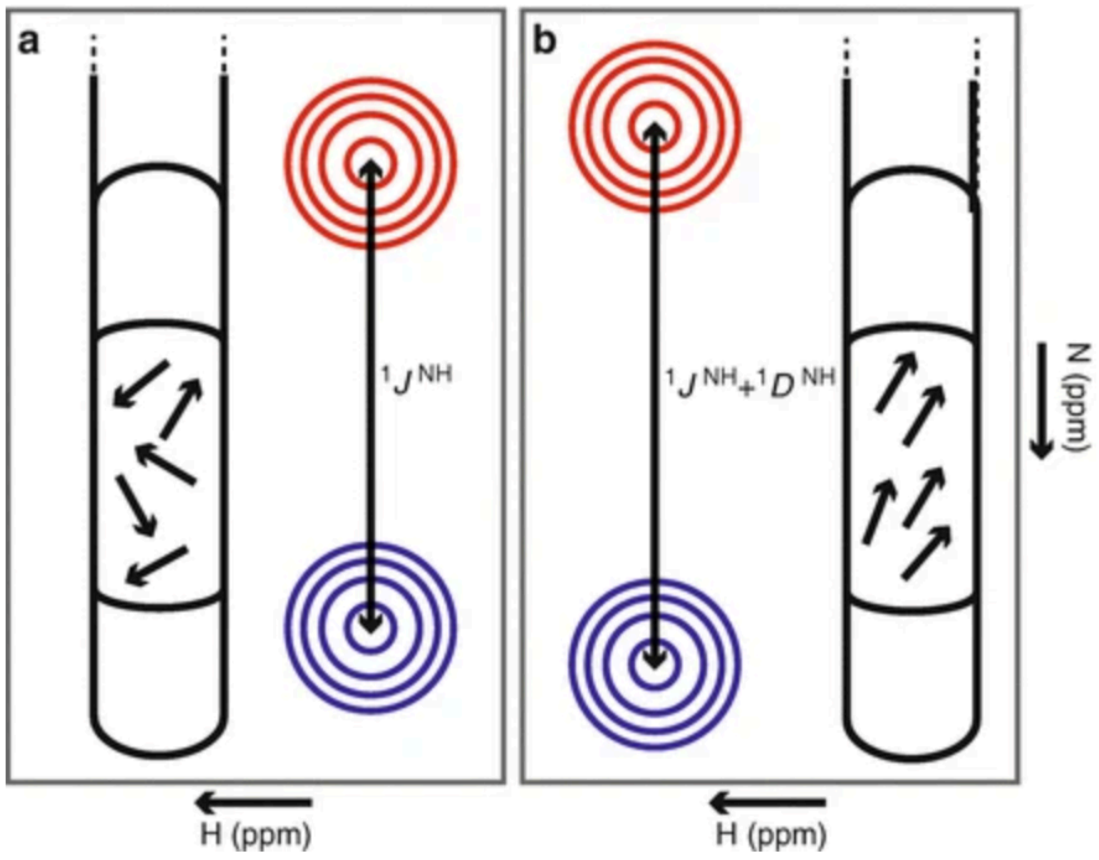


Figure 15- Residual dipolar couplings. The effect that alignment media has on the orientation of molecules in reference to the magnetic field, (a) is without the media and (b) is the addition of it. More spins are aligned with the direction of the magnetic field. (46)

1.3.7.2 Non-Uniform Sampling:

Non-uniform sampling (NUS) is an acquisition method for NMR experiments with at least 2 dimensions. It works by skipping a random fraction of data that would otherwise be measured normally. As a result of this, the experiment time is lower. It is particularly useful for multi-dimensional experiments in biomolecular NMR (protein analysis) where high quality experiments can have time savings up to a factor of 10! Because of this 4D and 5D experiments can be performed and take a reasonable amount of time and still yield high quality results with a good resolution. (47).

NUS is a well-established technique for spectra with no dynamic range problem such as triple resonance experiments. It allows for the recording of triple resonance spectra at high resolution is significantly less time (48).

1.3.7.3 Unlabelling:

Another important method to aid assignment is unlabeling. For traditional NMR analysis, samples that have been labelled with $^{13}C/^{15}N$ isotopes. As mentioned before with assignment, signal overlap can become a big problem with larger proteins. This is because as molecular size increases, the line widths of NMR signals broaden. Specific isotopic labelling 'turns on' the NMR signals at selective sites whilst the rest of the protein remains 'invisible'. As a result, it can simplify the NMR spectra, improve sensitivity and facilitate the assignment process(49). This can go either way: the protein can either be completely labelled with $^{13}C/^{15}N$ isotopes and then the selective $^{12}C/^{14}N$ labelled amino acids to make them not appear on the spectra and vice versa (however this way is considerably more expensive because specific amino acids are labelled). Figure 16 below is an example of the unlabelling experiment. This doesn't take amino acid scrambling into account.

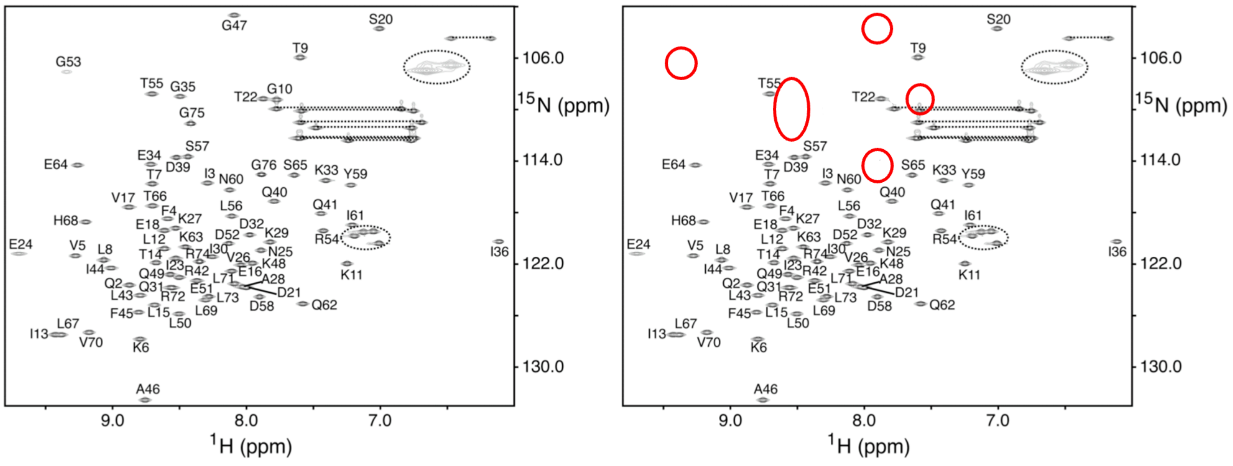
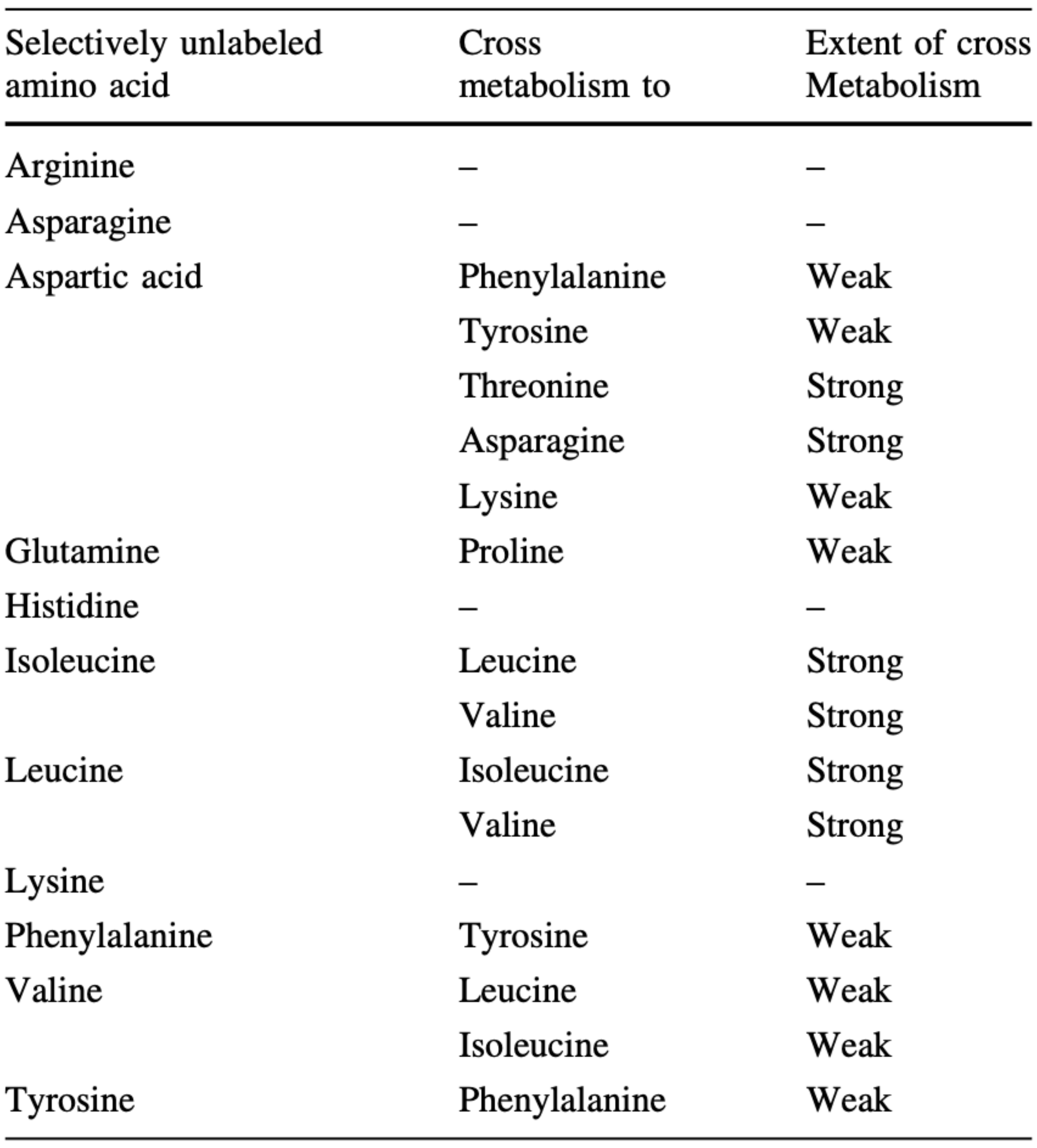


Figure 16- A Glycine unlabelled amino acid spectrum (perfect scenario). The spectrum on the left shows a section of ubiquitin HSQC and the spectrum shows a glycine unlabelled HSQC. The red circles indicate where the glycine's were originally, which are now absent. (50)

With the unlabelling process, amino acids have to be carefully selected. This is because in bacteria scrambling can take place due. Scrambling occurs because of common metabolic pathways from some types of amino acids; in these pathways the amino acids are interchanged and so the wrong amino acid can then also become ^{15}N unlabelled. Amino acid scrambling results in the blurring of the unlabelling effects making it harder to determine where the unlabelling has occurred. This means that if you can't determine where on the spectrum the unlabelling has occurred, then you cannot say that the unlabelling was a success.

Some amino acids such as isoleucine have a strong scrambling effect with leucine and valine. As a result, the unlabelling loses information. When performing an unlabelling experiment it is best to select amino acids that have no or a weak scrambling effect so that the results can be more easily interpreted. Amino acids such as arginine and phenylalanine have little to no scrambling effect.

Figure 17 displays which amino acids can become scrambled.



Selectively unlabeled amino acid	Cross metabolism to	Extent of cross Metabolism
Arginine	–	–
Asparagine	–	–
Aspartic acid	Phenylalanine	Weak
	Tyrosine	Weak
	Threonine	Strong
	Asparagine	Strong
	Lysine	Weak
Glutamine	Proline	Weak
Histidine	–	–
Isoleucine	Leucine	Strong
	Valine	Strong
Leucine	Isoleucine	Strong
	Valine	Strong
Lysine	–	–
Phenylalanine	Tyrosine	Weak
Valine	Leucine	Weak
	Isoleucine	Weak
Tyrosine	Phenylalanine	Weak

Figure 17- A table to display which amino acids are more or less likely to scramble. It is best to consider unlabelling with weak scrambling as results will be clearer and much easier to determine whether or not the unlabelling process was a success.(51)

1.3.8 Circular Dichroism and hydrogen bonding in tertiary structure:

In relation to assignment and how easy a protein can be to assign can relate to Circular Dichroism, or CD. This technique is used to give information about the chirality of molecular systems as well as being used to determine secondary structure of biological molecules such as proteins(52). A CD spectrum of the far UV region mainly is reflective of peptide bonds. For usual secondary structure of proteins, peptide bonds are highly regularly arranged. The orientation of arrangement determines the splitting of the peptide bond energy level transition. In other words, the values and positions of absorbance strengths are different with each type of secondary structure (53).

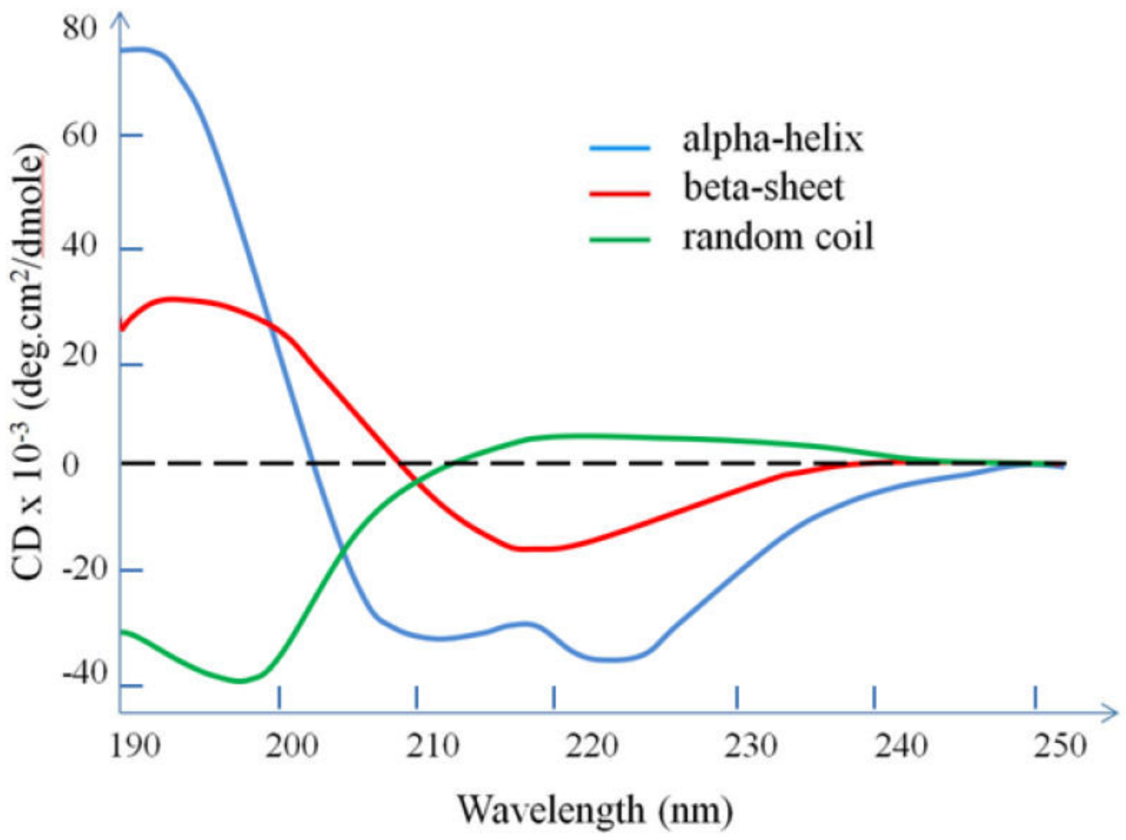


Figure 18- A Circular Dichroism (CD) spectrum displaying the different wavelengths of the different types of tertiary structures, it is also important to note that the random coil graph takes a completely opposite shape to the alpha-helix and beta-sheet due to their vastly different structures. (54)

In addition, there are significantly more intermolecular forces and hydrogen bonding is involved in beta sheets and alpha helices. This is what makes them more rigid and less likely to be affected by the presence of other molecules.

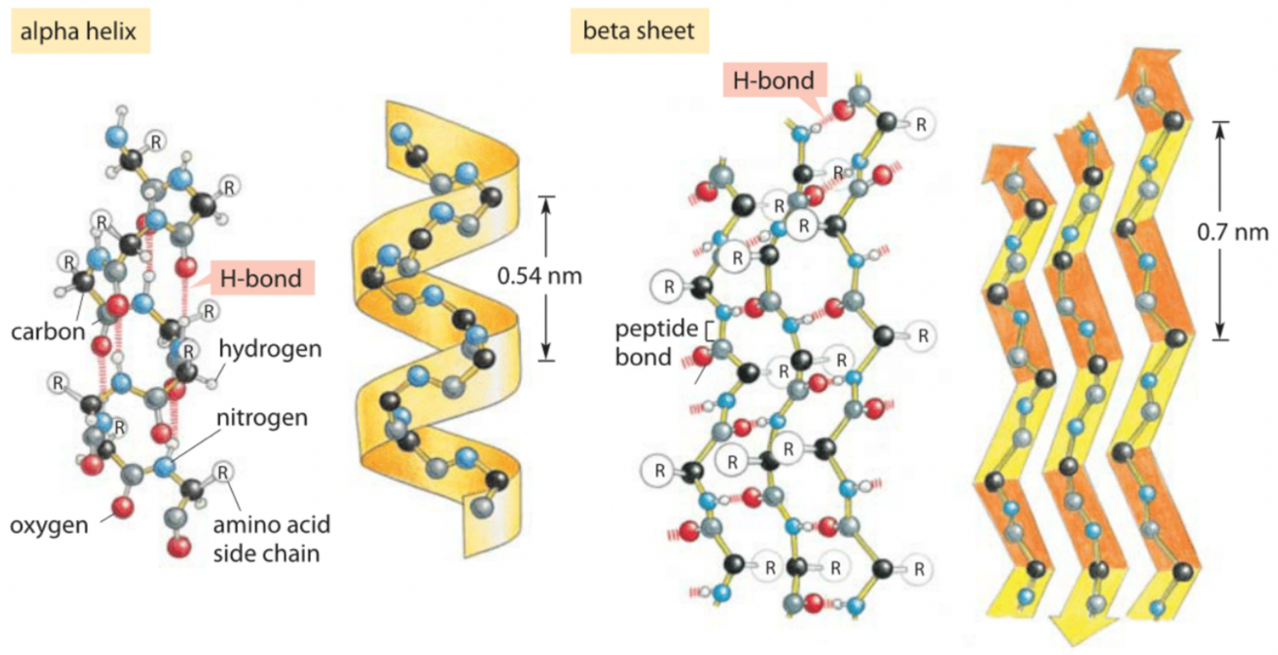


Figure 19- A diagram to highlight the importance of hydrogen bonding in tertiary protein structure and how this contributes to the stability and rigidity in alpha helices and beta sheets. Random coils do not have the same stable structures and so this is what makes them much more fluid and therefore much harder to assign correctly. (55)

In effect, some regions of proteins can be more difficult to assign because of their lack of structural rigidity. Amino acids are held in place by weaker intermolecular forces and so could take different shapes/ conformations (hence the name random coil).

1.4 Part IV: Automated assignment and an introduction to SNAPS:

1.4.1 What is SNAPS and how it works + project aims:

From part I, this core focus of this project asks the question: ‘Can we have a fully automated assignment pipeline for proteins?’ Because assignment is a very long process that can slow down the progression of scientific research, why can’t we make it easier? This is where SNAPS comes into play. SNAPS or **S**imple **N**MR **A**ssignment using **P**redicted **S**hifts is a program that gives predicted chemical shifts for proteins with ‘known’ 3D structure to aid assignment. This can be used hand-in-hand with AlphaFold (AF) or with proteins whose structures have already been determined with X-ray crystallography. AF can be used to quickly generate a good quality 3D structure and then assignment completed with SNAPS. As a result, a fully automated assignment approach! This is based on predicted shifts and can be used for any protein with a globular 3D fold. The chemical shift values are needed and triple resonance data, CA, CB, C, N, H, HA.

1.4.1.1 Project aims:

SNAPS was originally written by Alex Heyam from the University of Leeds. For this project, we modified some of the functions and focused on more large-scale robust testing. The aims of this project overall were:

1. Develop a system where the user can go from a known 3D structure of a protein and sufficient NMR data and without use of manual assignment be given an automated assignment suggestion.
2. Ensure the program is compatible with AlphaFold structures (where 3D structure is determined, but not with high certainty).
3. Perform small-scale unit tests to ensure that analysis aspects of the program are working as they should.

4. Perform large-scale testing on the program to ensure high quality backbone assignment predictions are produced on model systems (large protein datasets derived from crystal and AlphaFold structures and laboratory-made protein)

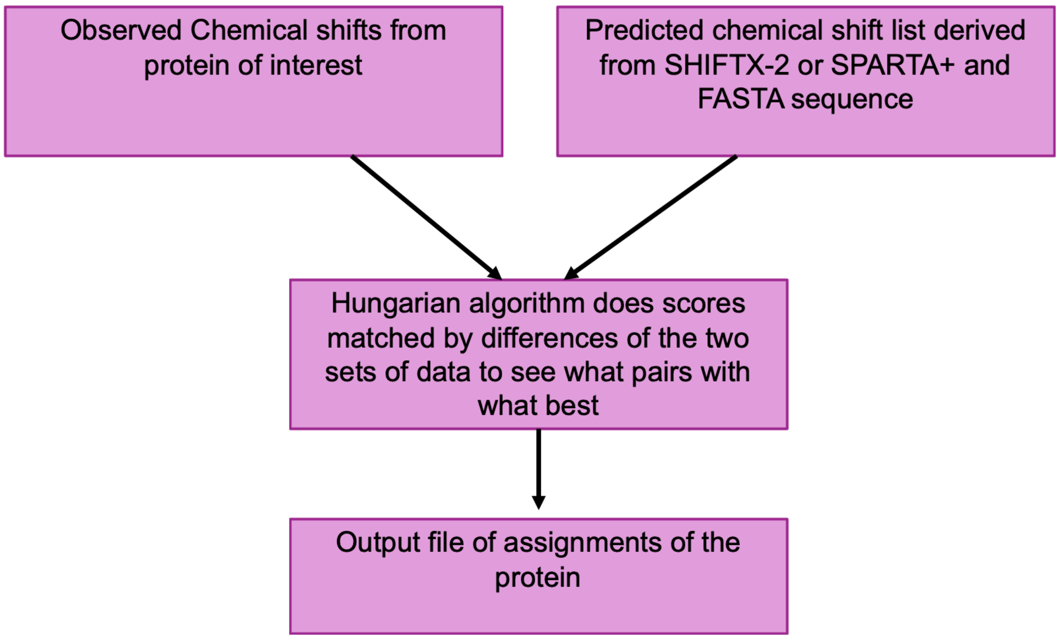

Observed Chemical shifts from protein of interest
Predicted chemical shift list derived from SHIFTX-2 or SPARTA+ and FASTA sequence
Hungarian algorithm does scores matched by differences of the two sets of data to see what pairs with what best
Output file of assignments of the protein

Figure 20- A flow diagram to show how SNAPS works. The program needs observed chemical shift list derived from NMR data. This is used in combination with a predicted chemical shift list derived from SHIFTX2 or SPARTA+ and FASTA sequence. Following this, an algorithm is applied to best match the chemical shifts to their optimum assignment to produce an output file.

The program SNAPS works by taking real experimental NMR chemical shift data that the user has measured for a particular protein and the predicted chemical shift data from prediction programs such as SHIFTX-2 or SPARTA+. These predicted shifts are used alongside one another to predict the assignment of the given protein. A matrix is applied to best match the two sets of chemical shift data to provide an assigned protein in accordance with the FASTA sequence (matrix explained in section 1.4.3). The flow diagram Figure 20 summarises the inputs and outputs required to make the program work. It should be noted that SNAPS currently doesn't use the correlation data present in the triple resonance data.

1.4.1.2 Structure of SNAPS:

SNAPS contains three main python scripts – SNAPS.py, SNAPS_assingner.py and SNAPS_importer.py; each of these carry out different functions which are fundamental to the running's of the program.

SNAPS.py:

This is the main program which has two functions:

1. **Get arguments: Mandatory arguments:** Shift table, predicted shift table, output file **Input and configuration options:** Shift types (CCPN, SNAPS, Sparky, mars, xeasy, NMR pipe, nef, test), predicted types (Shiftx2, SPARTA+), predicted sequence offset (integer, 0), config file, test_aa_classes
   control of output files: log file (file logging info written to), shift output file (assigned shift list will be written to Sparky (def), xeasy, nmrpipe), shift confidence (high, med, low, undefined), strip plot and hsqc plot
2. **Run SNAPS: set up logging**: create a logger - writes to a specific file which defines the default perameter for how the calculation is carried out. Set up assigner object – which uses the Hungarian algortihm (uses SNAPS_assigner.py), import + read config file, import obs and, pred shifts (SNAPS_importer.py), analysis: log probability matrix and mismatch matrix , output results (tabulate), write chemical shift lists, make plots, close

This is the central script that actually acts to get the arguments for the program and then run it using the other two scripts, SNAPS_importer.py and SNAPS_assigner.py.

In summary though; SNAPS_importer.py imports the chemical shift data supplied in the NEF files and formats it correctly. SNAPS_assigner.py uses the hungarian matrix and deals with the assignment of the chemical shift data.

1.4.2 Python and Pandas:

SNAPS consists of three python scripts and, each of these contains a myriad of functions. The program combines two datasets for the same protein, one of which is from a 'real' experimental test (real experiment meaning it can be an imported set of results from the BMRB or an experiment undertaken by the user), and the other derived from SHIFTX2 or SPARTA+ and is then applied to an algorithm (discussed in the next section). As a result, the program deals with large amounts of data. It is therefore critical to have an organised code and an efficient way of storing data in tables and lists that are easy to understand and access to relevant mathematical routines. This is achieved in SNAPS using the python library **pandas**. This python library that models data as a data table which contains a number of named columns (or series) each containing data in rows (displayed in figure 21). Pandas is typically imported at the top of the script like so:

```
    import pandas as pd
```

Following this, the library can be called at any point in the script using the short name pd. For example, this function is taken from the NEF reader.

```
def read_nef_shifts_to_pandas(file_handle,
shift_list_name=_DEFAULT_SHIFT_LIST):
    snaps_shifts = read_nef_shifts(file_handle, shift_list_name)

    pandas_columns = [_SEQUENCE_CODE, _ATOM_NAME, _SHIFT_VALUE,
_RESIDUE_NAME]
    return pd.DataFrame(snaps_shifts, columns=pandas_columns)
```

The script takes a handle to a file and the name of a NEF shift list, defines a list of column headings and then returns the dataframe in the pandas format with the given headings being part of the table. It refers to the PyNMRstar library to find the data for the shift list frame as a list of lists with each subset contains rows of the data.

```
[_SEQUENCE_CODE , _ATOM_NAME, _SHIFT_VALUE, _RESIDUE_NAME],
[_SEQUENCE_CODE , _ATOM_NAME, _SHIFT_VALUE, _RESIDUE_NAME],
[_SEQUENCE_CODE , _ATOM_NAME, _SHIFT_VALUE, _RESIDUE_NAME],
[_SEQUENCE_CODE , _ATOM_NAME, _SHIFT_VALUE, _RESIDUE_NAME] …
```

Simply, pandas can be used for organising data, creating Data Frames, reshaping data, summarising data and combining different datasets/ group data. It is an incredibly powerful tool and can facilitate with organising large datasets.

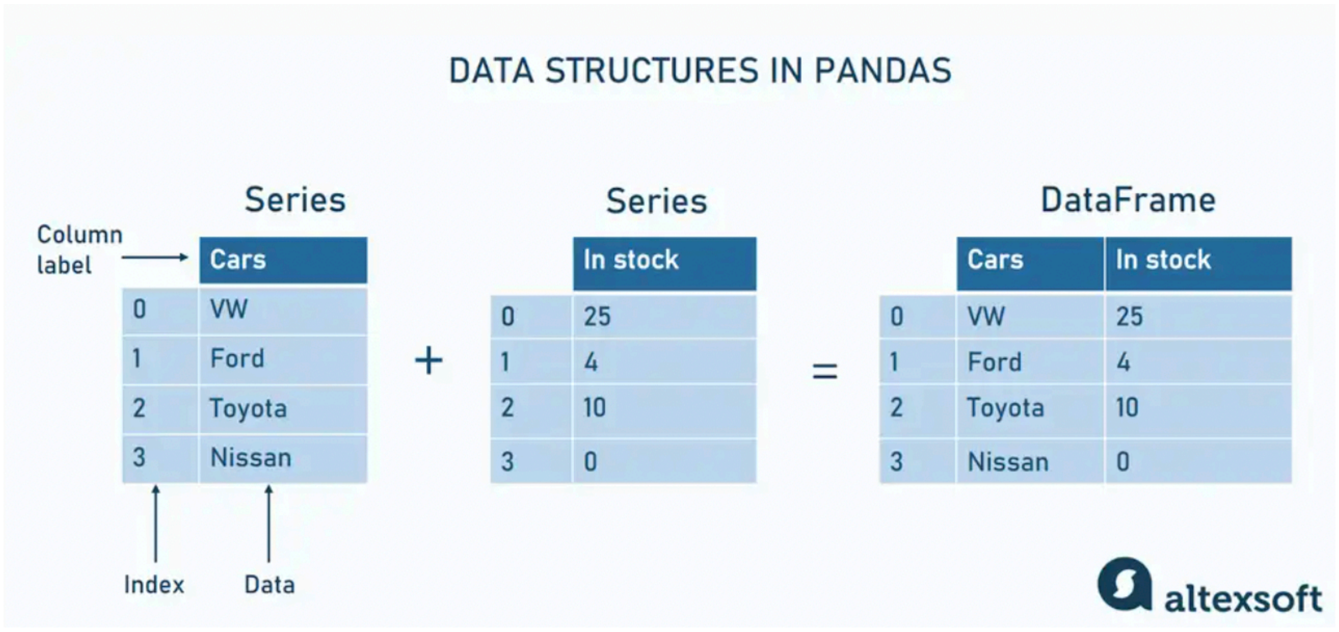


Figure 21- The data structures of Pandas. Data is arranged in DataFrames and Series. Series consist of a column heading and row index (or vice versa) and these are combined to make a total DataFrame that can be organised accordingly (56).

1.4.3 The Hungarian Algorithm:

The key step in the SNAPS program is the application of the Hungarian algorithm. This is applied to find the best correlations between the dataset of observed chemical shifts and the dataset of predicted chemical shifts derived from shift-prediction programs. The algorithm works by finding maximum weight matches and tackles what is known as the assignment optimisation problem (as opposed to the NMR assignment problem).

The assignment problem deals with assigning tasks with the optimal cost assignments. For example, to minimise costs and maximise outputs(57). This can be applied to SNAPS; we want to achieve the lowest cost (or in the case of SNAPS, the lowest error between observed and predicated shifts) which gives the highest accuracy of assignment.

	Cut Grass	**Walk Dog**	**Cook dinner**
Ben	£5	£10	£10
Adam	£10	£5	£10
Paul	£10	£10	£5

Figure 22- How the Hungarian algorithm can be applied to real-life scenarios to find the lowest cost solution to a problem.

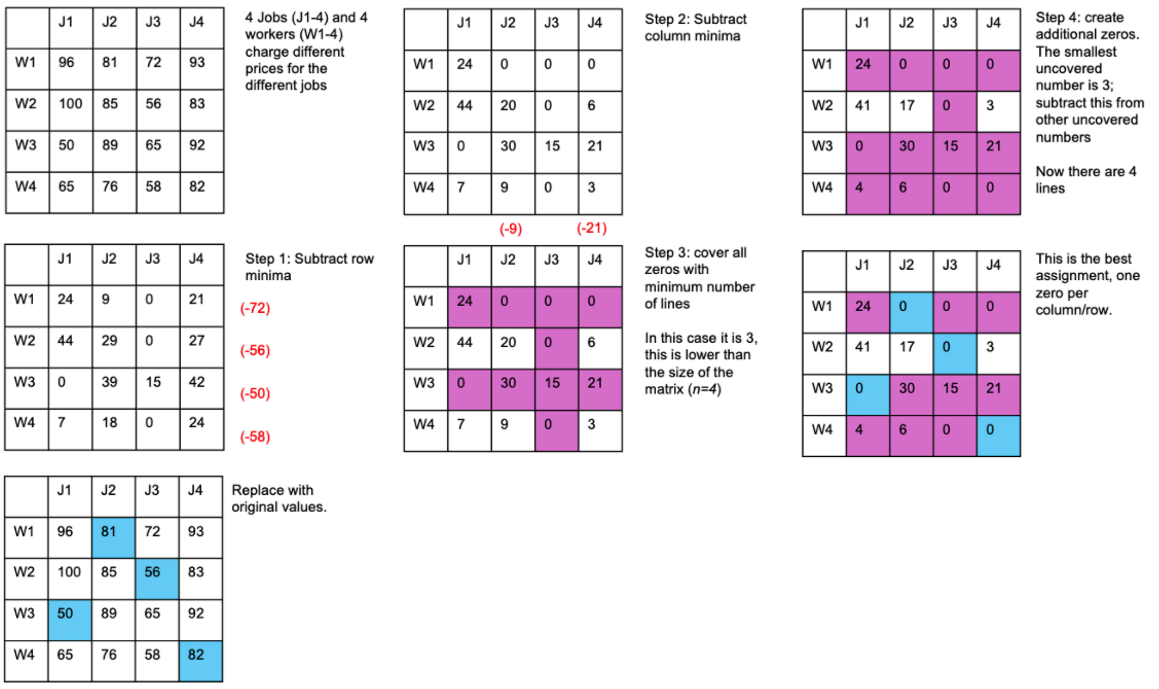
As an example, take the case of a homeowner who employs three workers, Ben, Adam, and Paul as they need some work done around their house. Each worker charges a different rate for different jobs. The owner wants to spend the least amount of money on paying these workers. From the table, it is clear that it would cost the least to get Ben to cut the grass, Adam to walk the dog and Paul to cook dinner and gives the minimum cost of £15, compared to a potential cost of £30 had the algorithm not been applied(58). Overall, the problem can be viewed as how to make the diagonal values of the table have the lowest cost solution.

In order to achieve this goal, the Hungarian algorithm works as follows:

1. Subtract the smallest entry in each row from all the other entries in the row. Now the smallest entry in the row will equal 0.
2. Repeat for each column
3. Draw lines through the rows and columns that have the 0 entries such that the fewest lines possible are drawn.
4. If there are *n* lines drawn, an optimal assignment of zeros is possible, and the algorithm is finished. If the number of lines is less than *n*, then the optimal number of zeroes is not yet reached. Go to the next step.
5. Find the smallest entry not covered by any line. Subtract this entry from each row that isn't crossed out and then add it to each column that is crossed out. Then, go back to Step 3

**n = matrix size= n x n (I.e., 3x3)*

Here is a worked example of exactly how the algorithm works to provide the best solution to the assignment problem. The method itself is simple but can quickly become complicated with large tables of data: (59)



	J1	J2	J3	J4
W1	96	81	72	93
W2	100	85	56	83
W3	50	89	65	92
W4	65	76	58	82

4 Jobs (J1-4) and 4 workers (W1-4) charge different prices for the different jobs

	J1	J2	J3	J4	
W1	24	9	0	21	(-72)
W2	44	29	0	27	(-56)
W3	0	39	15	42	(-50)
W4	7	18	0	24	(-58)

Step 1: Subtract row minima

	J1	J2	J3	J4
W1	96	81	72	93
W2	100	85	56	83
W3	50	89	65	92
W4	65	76	58	82

Replace with original values.

	J1	J2	J3	J4
W1	24	0	0	0
W2	44	20	0	6
W3	0	30	15	21
W4	7	9	0	3
		(-9)		(-21)

Step 2: Subtract column minima

	J1	J2	J3	J4
W1	24	0	0	0
W2	44	20	0	6
W3	0	30	15	21
W4	7	9	0	3

Step 3: cover all zeros with minimum number of lines

In this case it is 3, this is lower than the size of the matrix (*n*=4)

	J1	J2	J3	J4
W1	24	0	0	0
W2	41	17	0	3
W3	0	30	15	21
W4	4	6	0	0

Step 4: create additional zeros. The smallest uncovered number is 3; subtract this from other uncovered numbers

Now there are 4 lines

	J1	J2	J3	J4
W1	24	0	0	0
W2	41	17	0	3
W3	0	30	15	21
W4	4	6	0	0

This is the best assignment, one zero per column/row.

Figure 23- A worked example of how the Hungarian Algorithm works to solve the assignment problem. The output result of the algorithm will have the lowest possible score across the data provided. This is applied to the program SNAPS and acts to find the best matched observed chemical shift data versus the predicted data across vast amounts of data for given proteins (59)

This increasing complexity means a computer program is required to analyse chemical shift data. It should also be noted that the result here is deterministic. There is only one correct solution. The algorithm scale is *O*(|*V*|3) suggesting that the algorithm isn't a suitable choice for very large proteins.

1.4.4 *Generating chemical shift lists:*

1.4.4.1 SHIFTX-2:

As mentioned above, SNAPS currently relies on SHIFTX-2 to produce a predicted chemical shift list. SHIFTX-2 was at the time of writing one of the best programs for rapidly and accurate programs for calculating predicted chemical shifts for a protein structure. In the

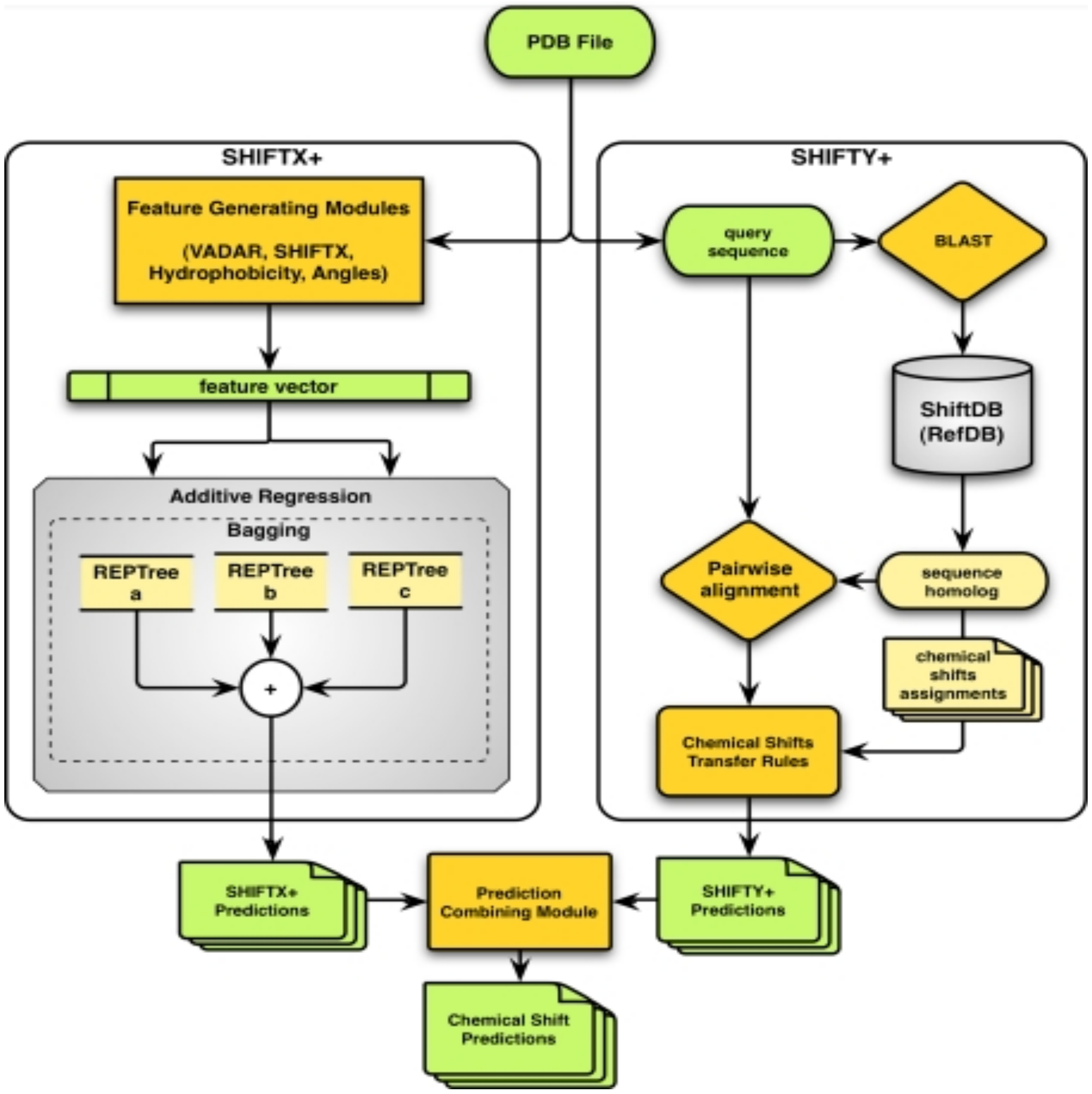


Figure 24- A flow diagram to show how SHIFTX-2 works to generate predicted chemical shifts (60)

case of SNAPS, CA, CB, C, H, N, and HA shifts. SHIFTX2 is able to achieve a very high-level accuracy because it is 'trained' on a large database of more than 190 different proteins. This database of proteins is used to produce data and includes advanced features such as solvent accessibility, hydrogen bond geometry, pH, and temperature also. SHIFTX2 is an AI tool and encompasses sophisticated machine learning techniques. Analyses state that SHIFTX2 can perform up to 26% better by correlation coefficient compared to the next best program available. Furthermore, the program is up to 8.5X faster than its competitors. The results from SHIFTX2 are impressive and show a very clear ability to predict chemical shifts, especially for nuclei other than 1H. However, it is believed that it is nearing the limit of how effective an AI chemical shift predictor can accurately depict a proteins true chemical shift list. Perhaps by using a combination of techniques such as SHIFTX2, SPARTA+ etc. the results could be more accurate. (60)

1.4.4.2 SPARTA+

At the time of writing, two other programs were available for shift prediction. One of which is SPARTA+; it is also based upon machine learning which is trained on a carefully selected database of around 580 proteins. The proteins provided for this database contained high resolution X-ray crystallography structures and almost-complete backbone and $^{13}C\beta$ chemical shifts were available. (33)

However, SPARTA+ predictions are known to be not quite as accurate as SHIFTX2 and UCBShift.

1.4.4.3 *UCBShift:*

This is a machine learning algorithm for protein chemical shift prediction that outperforms many exiting chemical shift calculator such as SPARTA+ and SHIFTX2. UCBShift works by implementing two modules:

- A transfer prediction module that employs sequence and structural alignment to select reference candidates for chemical shift replication
- A redesigned machine learning module based on a random forest regression which utilises mire feature extracted data.

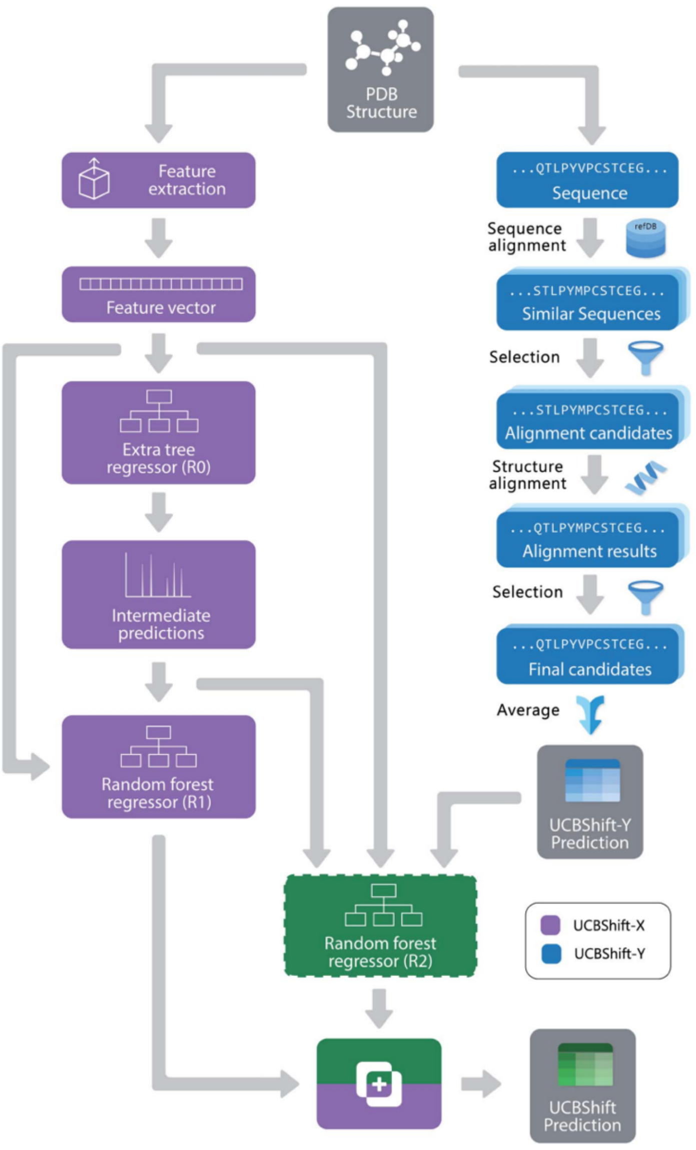


Figure 25- A flow diagram summarising how UCBShift works to generate a set of chemical shifts (62)

These features combined achieves a very high-level accuracy for prediction the chemical shift data(62).

1.4.5 Automated assignment systems:

There are a few programs available already that aim to automatically assign backbone resonances of proteins.

1.4.5.1 MARS:

MARS is one program used for automated backbone assignment of $^{13}C/^{15}N$ labelled proteins. It doesn't require tight thresholds and can work with a broad range of NMR spectral data. This program was tested on a 370 amino acid maltose binding protein and gave error-free assignment for 96% of this protein. MARS can also be used when there are gaps in the data given as well as with proteins that have a high chemical shift degeneracy as well as partially unfolded proteins(63).

MARS uses assignment tables containing pseudo residue spin system records, and a protein amino acid sequence to give the output of assigned pseudo residues to the sequence. Typically, run time of the program is dependent on the size of the protein i.e., 2-minute run time for >100 residues(64).

1.4.5.2 RASP:

RASP (rapid and robust backbone chemical shift assignments from protein structure) is another automated assignment program. The program works to assign backbone resonances of proteins in cases where the protein structure is known and from chemical shift predictions. Accurate assignments can be obtained from a minimal set of triple resonance experiments. A test set with 154 proteins; RASP assigned 88% of residues with a 99.7% accuracy from HNCO and HNCA spectra. The test set comprising of matched pairs of high-resolution crystal structures and assigned NMR chemical shifts from the database using TALOS (Torsion Angle Likeliness Obtained from Shift and Sequence Similarity(65)). A weighted distance measure to compare the set of experimental chemical shifts with the set of predicted shifts from ShiftY and then scoring and optimisation function is applied. For proteins where structural in formation is readily available RASP promises to significantly speed up the backbone assignment process by reducing the requirements for both data acquisition and manual analysis. Other structural parameters such as RDCs and NOEs can also be incorporated into the scoring function and can work to improve the accuracy of the algorithm(66).

1.4.5.3 ARTINA and NMRrtist

NMRrtist is a cloud computing service and can give a fully automated analysis of protein NMR spectra. This analysis ranges from peak picking, chemical shift assignment and protein structure determination. ARTINA is an application for protein structure determination via NMR- it looks at distance restraints and cross-peak positions etc. and was trained on more than 1300 2D-4D NMR spectra. ATRINA works to give a predicted overall 3D structure of proteins based off of NMR data. Combined, the two programs can automate peak detection in different spectrum types and chemical shift assignment(67).

ARTINA was simulated and tested using 100 different proteins and the process of how it works is displayed in figure 26.(68)

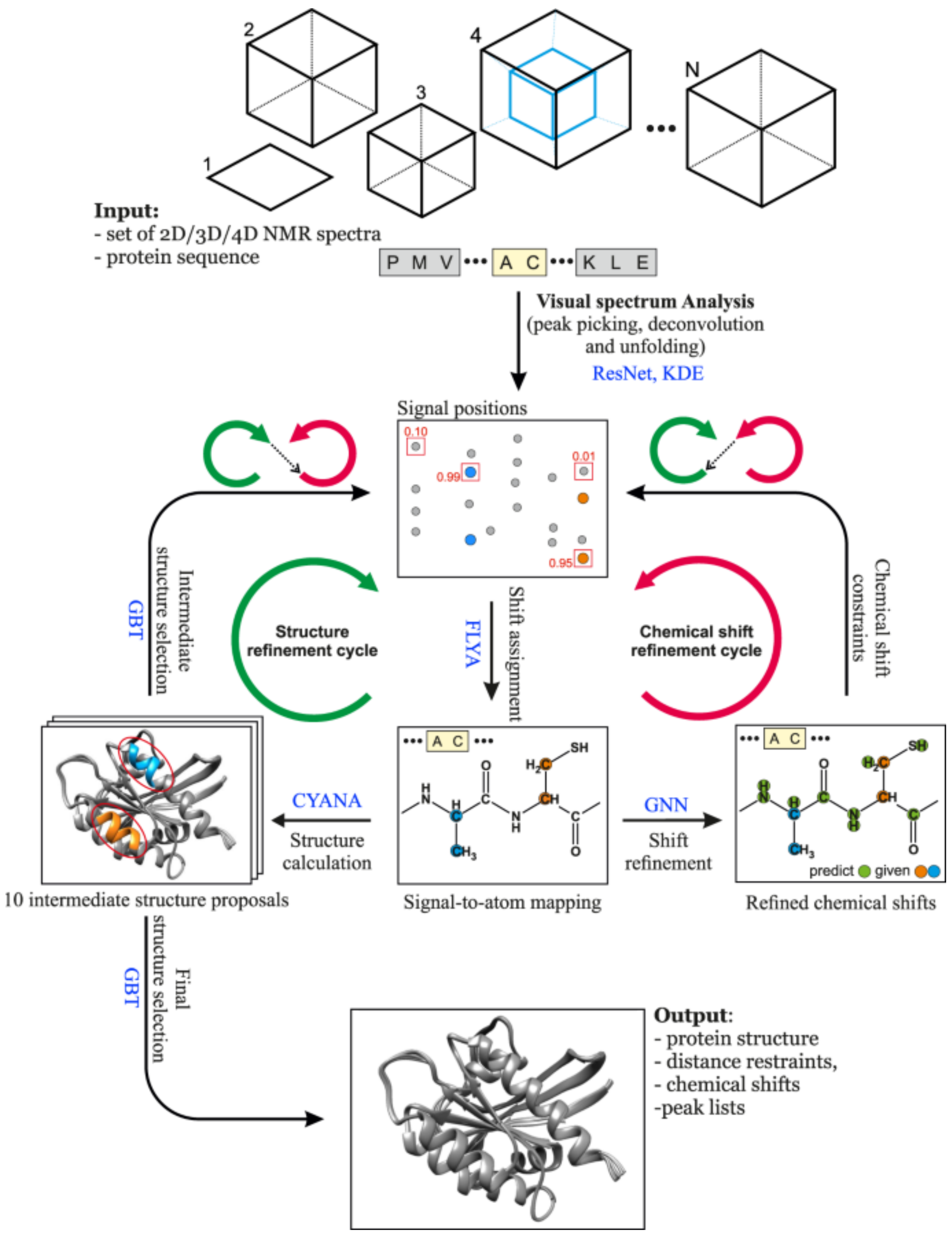


Figure 26- A visual representation of how ARTINA determines 3D structures of proteins derived from NMR data(68)

1.5 Model systems and application of SNAPS:

1.5.1 PctA:

PctA (methyl-accepting chemotaxis protein A) is found in the gram-negative, rod-shaped bacterium Pseudomonas aeruginosa. This bacterium is commonly found in water, on surfaces as well as in plants and humans. It is an opportunistic pathogen, meaning that it usually only causes disease in immunocompromised patients. For example, patients who are suffering from disease caused by antibiotic resistance or have autoimmune disease such as lupus, cystic fibrosis or rheumatoid arthritis. A pseudomonas infection can affect different areas of the body including the blood, ear, skin, lung urinary tract etc. and can cause a range of different symptoms(69). Its PDB model structure is displayed in figure 27.

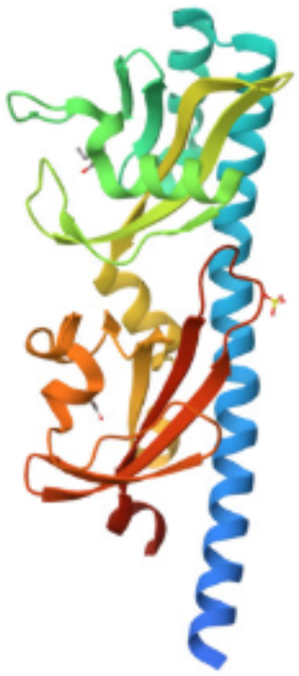


Figure 27- A PDB model of the ligand binding domain of PctA (PDB code, 6FU4)(70)

P. aeruginosa has a complex chemotaxis system that involves the protein PctA. This system allows the bacterium to propel itself towards nutrients like amino acids and away from poisons. The ligand binding domain of this protein is located in the periplasm and contains chemotactic signal transducers. These are able to sense ligands and allow the bacterium to react to changes within its environment(71).

Because of these dynamic properties, PctA was chosen as a model system to demonstrate the accuracy of SNAPS with how it is assigned. More specifically, the ligand binding domain of PctA, which is approximately around 30kDa in size, or 270 amino acids including a his-tag.

1.5.2 P3AL23R:

This is a 79-residue protein, Human PACT-D3 L273R, BMRB entry 27143(72). The protein is involved in the homodimerization of dicer-associated double-stranded RNA-binding proteins(73). Without going into too great detail of the mechanisms and pathways of this protein, it was studied by Alex Heyam who is the initial writer for SNAPS and was the protein they used for basic testing scripts. P3a_L273R is a segment of the protein Interferon-inducible double-stranded RNA-dependent protein kinase activator A. P3A_L273R was also used for this study for testing to drive consistency. There was also a plethora of NMR data readily available for this protein.

The pink region is the 10-residue fragment to be used for matrix testing for small scale tests and checking the matrix was working as predicted. This region is shown in pink in figure 28.

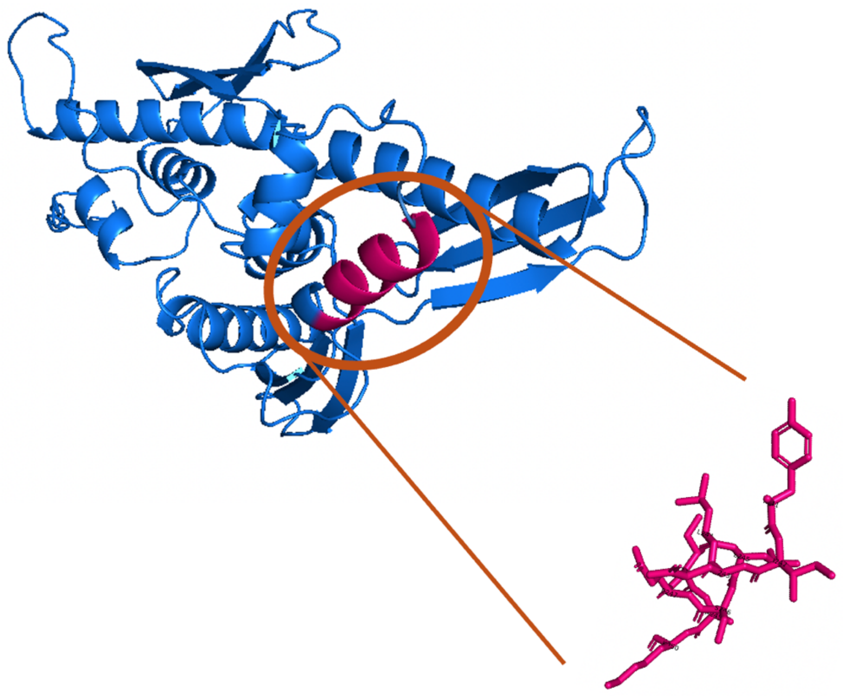


Figure 28- An AlphaFold model of the protein P3AL273R. The region highlighted in pink shows the 10 residues that were selected for testing with sequence 241-Y,I,Q,L,L,S,E,I,A,K

1.5.3 Unlabelling experiments:

As mentioned previously in the NMR techniques section, an unlabelling experiment was performed with the protein PctA. The amino acid Tyrosine was chosen and selectively unlabelled and henceforth 'removed' from the NMR spectrum. This should in theory produce good quality data as it is only known to have weak scrambling with phenylalanine. This unlabelling experiment can be applied to SNAPS.

The intention with adding unlabelling as a feature to SNAPS as a whole should mean that the algorithm should have to do less work because some of the residues are already assigned so should be exempt from the assignment process- because it is already assigned and would anchor assignment and remove unlikely assignments also.

The overall aim of the unlabelling experiment was to generate triple resonance data which could then in turn be used in the SNAPS program.

2 MATERIALS AND METHODS:

2.1 Python: Examples of Key Features:

All scripts were written using the language python. Python is a widely used programming language and has many important features that make it an excellent choice for programming.

Two notable features of python is that it has indentation levels to define program structure and it is also an interpreted, dynamic language.

2.1.1 Variables:

A variable is a memory location to store values/ information. It simply gives a name to a value so it may be used later on in the code. Variables can store numbers, lists, tuples, strings, and dictionaries. A variable is created as soon as a value is assigned to it. For example:

```
Project = “Thesis”
```

Now the string “Thesis” is stored inside the variable project. This variable can be used later, such as in a print statement:

```
print (project)
```

Prints to screen:

```
Thesis
```

There are 4 basic data types used in python(74):

Table 1- The different data types in Python and what they mean.

Data Type:	Description:
Integer (Int)	Any whole number 1, 20, 175 etc.
Float	Floating point number- any number including decimals 7.0, 3.147, 39.8
String (Str)	Any text that is wrapped in quotation marks “Hello”, “This is my masters project” “95”
Boolean (Bool)	Only 2 values, True or False

2.1.2 Loops:

Once values are stored in variables, or more specifically a list stored in a variable. Loops may be used for iterating over a sequence. The simplest type of loop used is a while loop. This is used to execute a block of statements continuously until a given condition is met. When the condition then becomes false, the line immediately after the loop in the program is executed.

For example:

```
count = 0

while (count < 5):

      count = count + 1

      print(“This is a while loop”)

output:

This is a while loop

This is a while loop
This is a while loop

This is a while loop

This is a while loop
```

As a result, it will print the value 5 times. (75)

However, the while loop can be error prone and so the for loop is more commonly used.

The for loop can be used to iterate over the contents of python containers including, lists, tuples, characters in strings, and dictionaries in python. For example,

```
weather = ["sunny", "cloudy", "stormy"]

for x in weather:

  print(x)
```

Output:

```
sunny

cloudy

stormy
```

(76) (77)

2.1.3 Decisions:

Decision making in python is essential for writing code, this uses the ‘if statement’. In an if statement, when certain condition is met (The statement is true) then an inner block of code is executed. If the statement isn’t true (False), the interpreter bypasses the indented block and instead executes statements in subsequent elif or else block(s) present. In summary, decision structures are dependent on true or false outcomes, the decision lies in which statements to execute for each outcome(78). Python assumes any non-zero value as true and 0/ null as false and has rules to define if strings and containers are true or false (empty strings and containers are false).

This is implemented throughout the SNAPS program as it determines which route the data goes to be processed/ raises certain exceptions to tell the user whether or not there is anything wrong with the data provided etc.

This is an example from the SNAPS_importer.py script and works with unit testing to give the user an error if there is anything is wrong with the data provided.

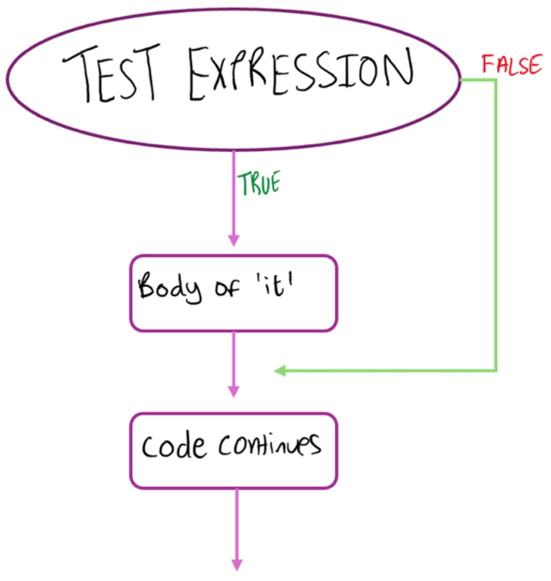
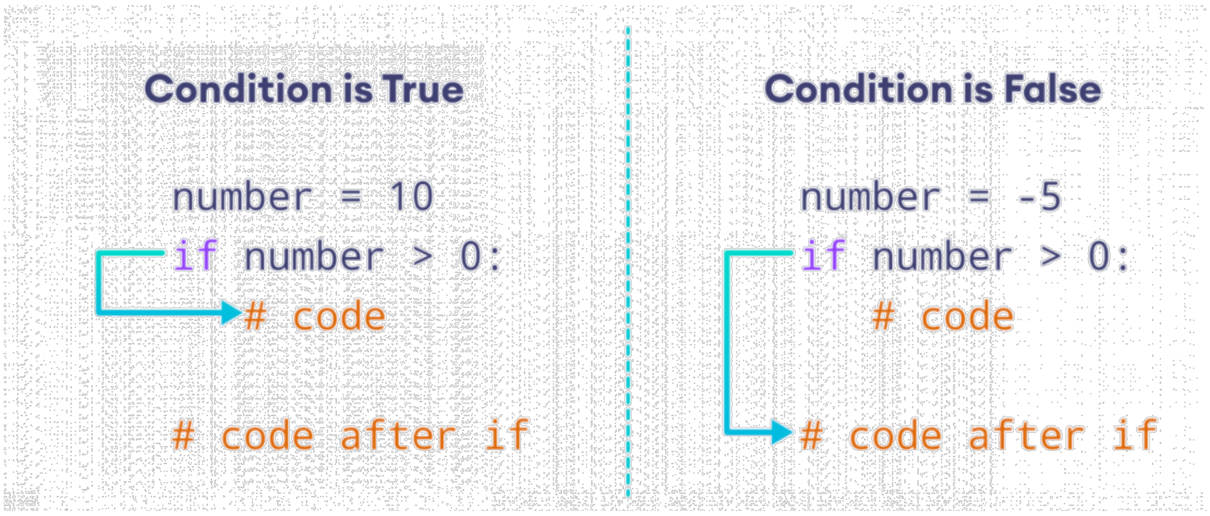


Figure 29- A flow diagram to demonstrate decision making in Python

IF: A Boolean expression followed by one or more statements

ELSE: Follows the 'If' statement with an optional else statement that is executed when the first if statement is false.

ELIF: If the 'if'' and 'else' statements are false then a third condition can be executed and so on…



```
number = 10
if number > 0:
    # code

# code after if
```

```
number = -5
if number > 0:
    # code

# code after if
```

Figure 30- An example of how a python if statement should be formatted. Elif is also something that can be used to determine extended decision structures with multiple decisions - as well as if and else. (79)

Decision making in python has played a very key role in the writing of code, in particular the unit tests. Essentially, from the unit tests the file needs to be checked whether or not the correct information is present, and *if* not, then a suitable error is raised (this will be discussed in more depth later.

2.1.4 Classes:

Classes provide a means of bundling data and functionality together. Creating a class creates a new type of object which allows new instances of that type to be made. Each class instance can have attributes attached to it for maintaining its state. In addition, class instances can also have methods which are defined by its class for modifying its state(80). A class is like an object constructor, or a “blueprint” for creating and manipulating objects.

Create a class:

```
class HannahsClass:

      x = 5
```

Create an object:

```
hello = HannahsClass()

print(hello.x)
```

The above example is a very simple application of a class and isn’t the useful for large-scale scripts/ projects such as SNAPS as it doesn’t contain stored data, only data associated with the class (x). There is a built-in function in python used to deal with classes called __init__(). All classes use this function and is always executed when an instance of a class is being initiated.

```
class Hannah:

      def __init__(self, name, age):

            self.name = name

            self.age = age

            self.paper = paper

p1 = Hannah(“Hannah”, 22, “Thesis”)

print(p1.name)

#prints “Hannah”
```

The __str__() function controls what should be returned when the class object is represented as a string, if this function isn’t set than the default string representation of the object is returned.

```
class Person:

  def __init__(self, name, age, paper):

    self.name = name

    self.age = age


  def __str__(self):

    return f"{self.name}({self.age})


p1 = Person("Hannah", 22)


print(p1)
```

Methods can also be applied to classes; these are functions that belong to an instance of class or object. Also, as you can see from the previous examples, there is a parameter that is called self. This is used to access variables that belong to the class and has to always be the first parameter of any function in the class(81).

Classes are used consistently throughout the SNAPS program; it is much more convenient for bundling data together. The two scripts SNAPS_importer.py and SNAPS_assigner.py have classes defined at the top of the scripts with embedded functions. The exception class SNAPSImportException() was also used to write unit tests and check for errors within the data provided to SNAPS.importer.py.

2.1.5 Stack traces:

Stack traces are used to display all of the function calls when an error occurs for debugging. The error is listed on the last line of the stack trace. The stack trace is important to show information and function calls related to the raised error to help users locate and solve the issues in their code(82). By definition, when an error occurs in the program/ code, python will raise an exception and when “caught” this will display a stack trace. This can be used to locate the error within the code.

The following information can be deduced from a stack trace:

1. ERROR LOCATION
2. ERROR TYPE
3. ERROR MESSAGE

An important tool that makes programs more user-friendly is to create your own exception and error message to tell the user in a better detail what the problem is with the program, or more specifically in the case of SNAPS, what the problem is with the data provided, and what needs to be changed.

Some common errors for stack traces are:

a) KeyError

   Trying to access a key that doesn’t exist in the dictionary will throw a KeyError. The user needs to input the correct key. For example, a quick fix can be using an if statement, if ‘x’ in dict… or using the keys() function to list the available keys.

b) NameError

   This happens when an attribute is called that isn’t defined. Attributes and functions should always be stored in a variable/ defined before they are called or used.

c) UnicodeError

   This is a result of python not being able to encode or decode from/ to Unicode. This can come from special characters, such as accents (é,ß,å etc.) and so the code will need to be put into ASCII letters format or try a codec that supports this- text format has to used.

d) TypeError

This error occurs when an object is the wrong type for an action that is being attempted. Such as trying to add a Boolean to an integer. To change a Boolean to an integer, use the int() function. (83)

e) ValueError

This happens when the function that is called has the correct argument type but it contains a wrong or invalid value. This can often occur during the unpacking of sequenced data and functions where a return statement is used. For example, for a string which can't be converted to a float, such as '123ab' (84).

Stack traces can be difficult to interpret, especially when the user doesn't have a great understanding of coding and how best to resolve the error. This is why SNAPS was integrated with its own Exceptions to tell the user in an easier way to understand what the error is. Here is an example of how to raise a custom Exception:

```
Class HannahsMastersException(Exception):

a=1
b=5

if a==b:
         print ("Yes")
else:
         msg= "this is a custom stack trace error message"
         raise HannahsMastersexception(msg)
-----------------------------------------------------------
Returns:
this is a custom stack trace error message
```

Figure 31- An example of a custom stack trace message raised as an exception.

Stack traces can be very complex and confusing to the inexperienced coder. Typically, the first error that is encountered in a program will cause it to die and then the user will go away and try to amend the imputed code to try and allow the program to run. This means that the code could be filled with errors throughout but this won't all be presented on one stack trace! Otherwise, it would be extremely difficult to read and understand- let alone be able to resolve the errors. Stack traces are very helpful in the sense that it allows the user to debug their code in chronological order of error- so a step-by-step method.

2.2 Writing a NEF importer:

The first work carried out on SNAPS was adding a NEF importer to make the program simpler and easier to use. It was important to pick a well-established file type, so NEF was chosen to be the primary importer used for SNAPS.

2.2.1 The structure of a NEF file

NEF (**N**MR **E**xchange **F**ormat) files are a new file type defined by the NMR community to transfer all NMR data between programs to improve data exchange as opposed to the BMRB NMRstar file which is designed for data archiving.

NEF uses the STAR (**S**elf-defining **T**ext **A**rchival and **R**etrieval file (85))file format; this has a clearly defined structure, and only certain terms are allowed. This ensures reliability and uniformity across platforms. In addition, it is a much more readable/. Understandable file type compared to others(86). NEF was generated to be a unified and easily adaptable file type due to the increased NMR-derived entries being added to the BMRB as well as to drive improvement in quality and integrity of the archive(87).

Key characteristics of NEF files:

1. All data names start with an underscore (these are called tags)
2. There are reserved keywords: data_, loop_, save_ and stop_.

3. NEF files always start with data_ with the project/ file description such as data_PctA_project as this indicates the start of the data block
4. Only have one data block per file

2.2.2 Components of a NEF file:

Saveframes: Data is organised into sections, known as saveframes which always start with the keyword _save and after that the framecode which is unique to the saveframe. For example, chemical shift data is stored in the saveframe (_save_nef_chemical_shift_list_default) and are identified by given tags. Nef save frame had a type component e.g., NEF_chemical_shift_list and an optional name, in this case default. Technically, each NEF file MUST contain the following saveframes:

- _nef_nmr_meta_data
- _nef_molecular_system
- _nef_chemical_shift_list

Tags: The tags in the file provide the information that is required for a particular section of the saveframe. NEF has a number of mandatory and optional tags for each saveframe. Figure 32 shows this.

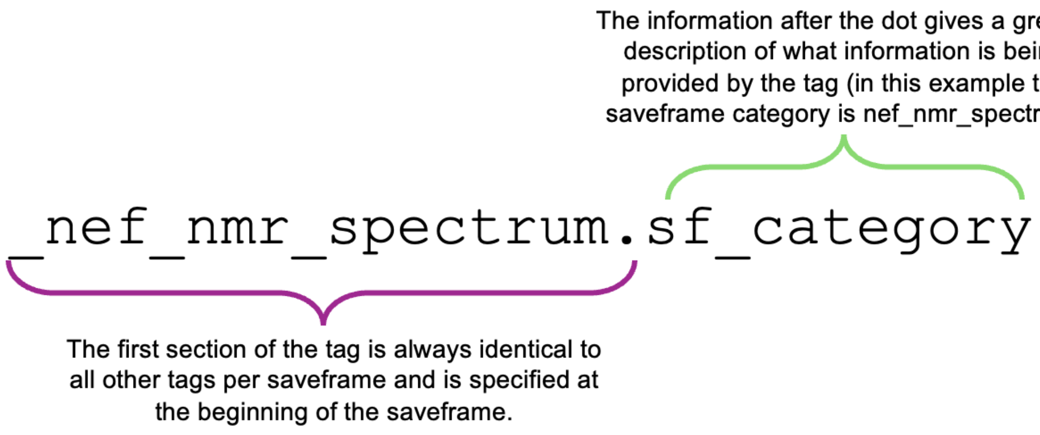
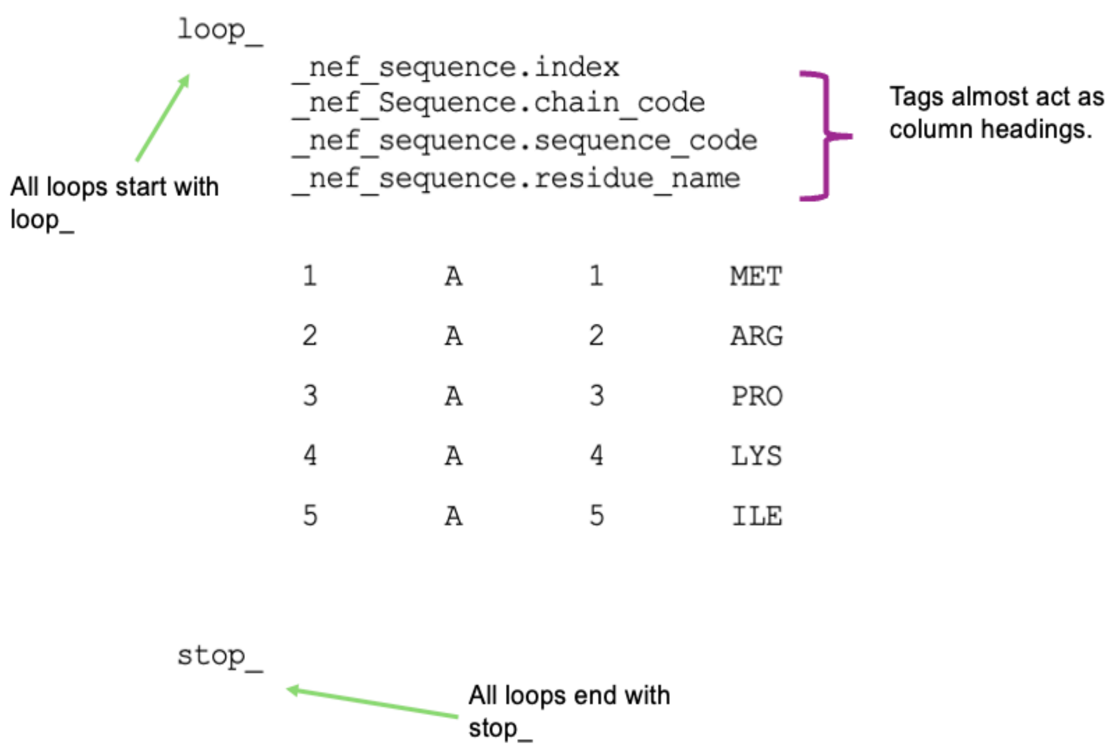

The information after the dot gives a greater description of what information is being provided by the tag (in this example the saveframe category is nef_nmr_spectrum.
_nef_nmr_spectrum.sf_category
The first section of the tag is always identical to all other tags per saveframe and is specified at the beginning of the saveframe.

Figure 32- A description of a tag in a NEF file. It is split into two parts, the beginning of the tag is identical to all others in that saveframe, the second part after the dot gives a further indication of what data is being stored here.

All tags will take the same format, but some will be universal to all NEF files produced whereas some can be user-specific depending on what the user is doing.

<u>Loops: </u>Because there can only be one of each tag per saveframe, it isn’t possible to include repetitive complex data, such as peak lists or chemical shift values in this format. This is because there are multiple values for each tag. Instead, they are outputted as a loop in the



```
loop_
    _nef_sequence.index
    _nef_Sequence.chain_code
    _nef_sequence.sequence_code
    _nef_sequence.residue_name

    1      A      1      MET

    2      A      2      ARG

    3      A      3      PRO

    4      A      4      LYS

    5      A      5      ILE


stop_
```

Tags almost act as column headings.
All loops start with loop_
All loops end with stop_

Figure 33- The structure of a loop in a NEF file, the tags are found at the top of the loop and act as column headings almost. The data within the loop is listed underneath in a table-style format. All loops start with loop_ and end with stop_. This tells the user the beginning and end of the loop.

saveframe. A loop is a set of tags with multiple values- used to store tabular data(88).

2.3 Writing the NEF importer, structure, and functions:

Functions of the NEF importer:

Read NEF shifts from file:

This opens the users file and stores it in the variable result_shifts

Read NEF shifts:

Acts to grab given information from the NEF file presented by the user

1. Sequence code - NEF loop heading for sequence

2. Atom name - NEF loop heading for Atom name (C, H, N…)
3. Shift values – NEF loop heading for chemical shift list value
4. Residue names – NEF loop heading for residue name (Arg, Met…)

And then arrange them in a list [ ]

Then, a variable, row_index uses the enumerate function is applied to allow you to keep track of the number of iterations within the loop, starting from 0 for the chemical shift data.

Offsets of the residues are also determined and can be used to indicate i and i-1 residues.

All data is stored in the variable output

Read NEF shifts from file to pandas:

This function reads shifts from a file using read_nef_shifts_from_file and converts the returned list to a pandas data frame. It gives the column the names sequence_code, SS_name, atom_name, Atom_type, value, and shift.

And then returns the new output as a pandas data frame.

Otherwise, output the SS_name as a string in addition to the residue name

Output is returned

Read NEF shifts to pandas:

Gives pandas columns under the variable pandas_columns (_SEQUENCE_CODE, _ATOM_NAME, _SHIFT_VALUE, _RESIDUE_NAME

And then returns a pd.DataFrame(snaps_shifts)

Tests:

A series of tests were also written into the script to ensure that it is clear and easy to use. Should there be any issue with the dataset provided, an error will flag and tell the user what the problem is. The tests written for this script are simply to make sure that the NEF file being supplied is in the correct file format (as in the correct structure, mentioned above).

The script checks for any errors where it cannot convert any strings to a float and tells the user which row number this problem occurs, as well as which loop and which file name. it can also tell the user when no chemical shift list loops are detected in a given file. Furthermore, the program will also tell the user if there aren't any chemical shift list frames in the data and in addition check the correct tags are being implemented.

The tests were written using f strings. This is a string formatting mechanism that is used to make string interpolation cleaner. To generate an f string the string is prefixed with the letter "f", with the variable inserted in curly brackets (89), { } for example:

```
name =  'Hannah'
age = '22'
project = 'Masters Thesis'
print(f"Hello, my name is {name} , I am {age} years old and am writing my
{project}.")
```

Result:

```
Hello, my name is Hannah , I am 22 years old and am writing my
Masters Thesis.
```

The validity checks and internal were defined in internal methods with names starting with an underscore for clarity of coding.

```
def _read_heading_index_or_error(save_frame, heading, loop, file_name):
    """

    :param save_frame: _CHEMICAL_SHIFT_LIST_FRAME
    :param heading: _SEQUENCE CODE
    :param loop: _CHEMICAL_SHIFT_LOOP
    :param file_name: file name
    :return: error if there is no sequence index/ heading  for {sequence
code} in file name. Otherwise, return sequence
    index for headings in the saveframe
    """
    sequence_index = loop.tag_index(heading)
    if sequence_index is None:
        msg = f"ERROR: couldn't find a sequence heading {_SEQUENCE_CODE} in
save frame {save_frame.name} in {file_name}"
        raise Exception(msg)
    return sequence_index
```

2.4 Writing Python tests:

The second key issue tackled was ensuring that the user knows exactly why the program has had an error. This is very important because standard stack traces in python can be very long and very difficult to understand for the untrained eye. By writing in a series of tests, the user can know exactly what the issue is with their dataset that they have provided as well as how to go about fixing it. In turn, this means a rigid, fool proof dataset is always provided to SNAPS and the user can have confidence in the workings of the program.

2.4.1 What does a test script look like?

For this step, tests were written using pytest. Pytest is used for writing unit tests and entails the Arrange, Act, Assert model:

Arrange –or, set up the conditions for the test

Act – by calling some function or method

Assert – that some end condition is true (90)

The tests were written to ensure that the user is providing the correct information in the observed (real) chemical shift list data. The tests check for:

- Correct column names
- Headings are correct- correct case and names
- Amino acid data provided is valid (letter code is correct)
- If an unexpected spin system is provided
- Incorrect offset
- Incorrect info provided in the columns

These are crucial to ensuring that the user can provide the correct data for the program to run. The test script tells the user if the data is correct and fits with the program by passing the test (no errors show). Otherwise, an error flags and tells the user precisely what the problem is with their data provided. It does this by raising 'SnapsImportException' and asserting an error message. For example:

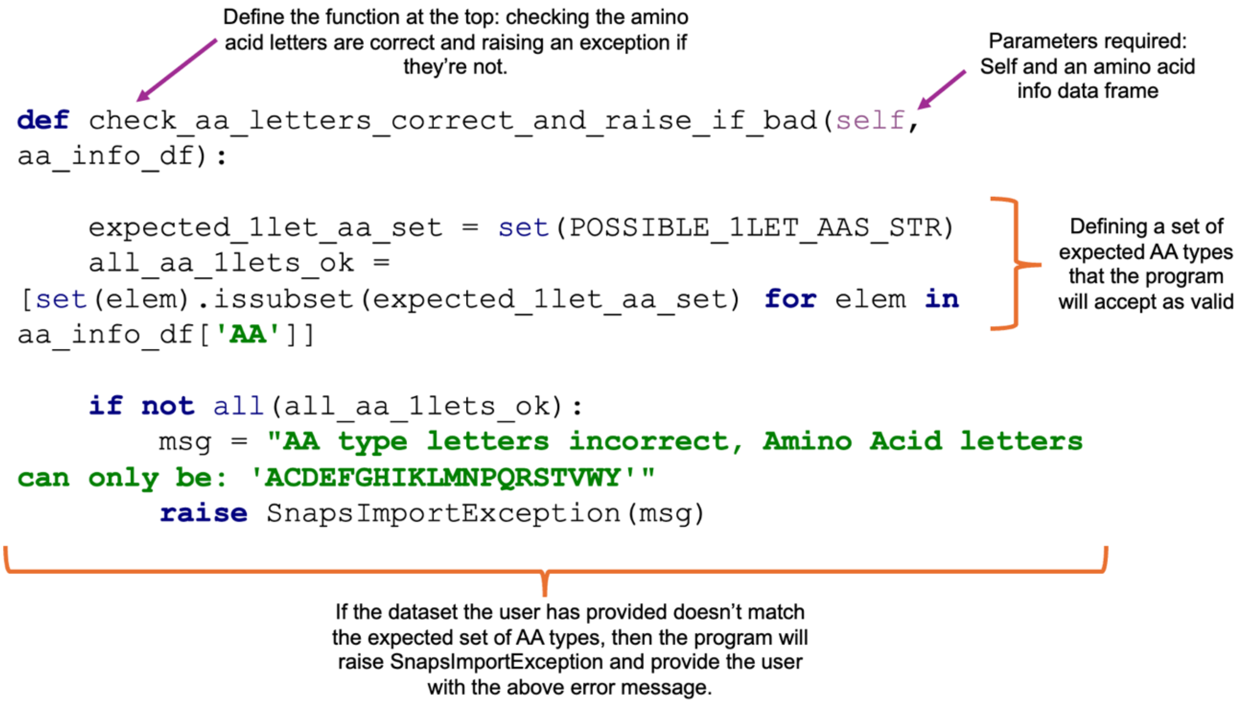
For SNAPS, data for residue type information is stored in a table like so:

Table 2- How data needs to be presented for the SNAPSImportException tests

SS_name	AA	Type
245Met	M	IN
01Dev	J	EX

245Met is filled out correctly and shouldn't flag any error. But 01Dev isn't an accepted amino acid, for testing purposes, an error would be raised.

The user has provided data that has an incorrect amino acid code in:

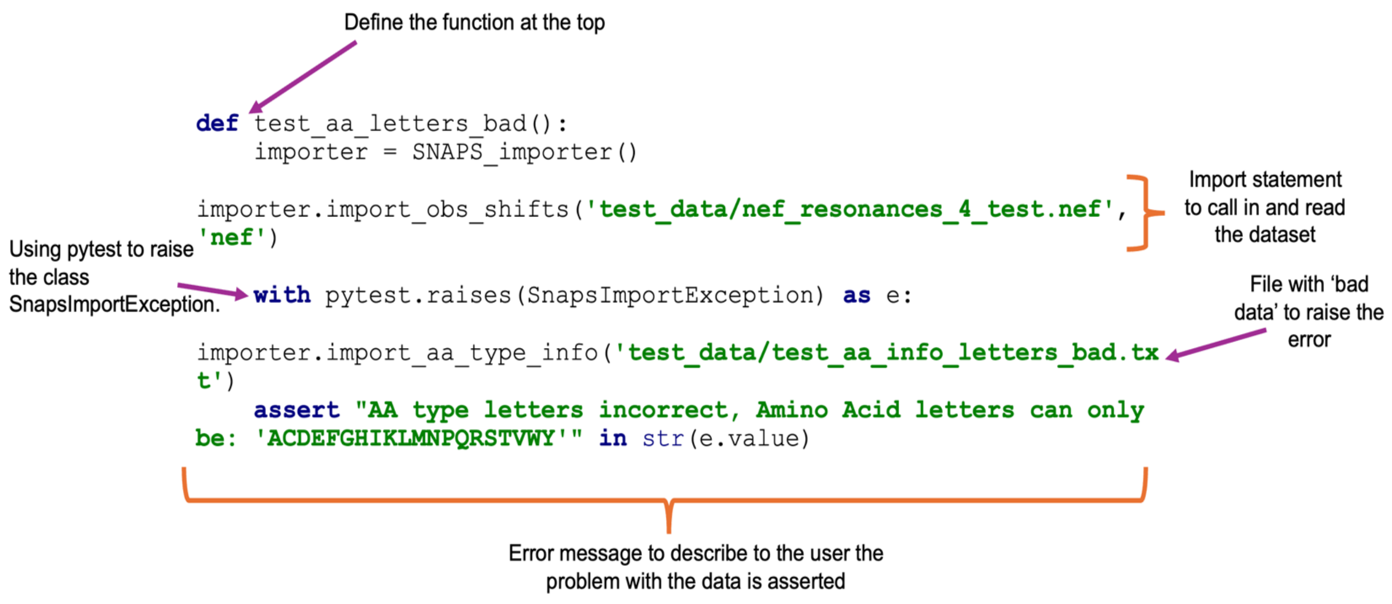


```
def check_aa_letters_correct_and_raise_if_bad(self,
aa_info_df):

    expected_1let_aa_set = set(POSSIBLE_1LET_AAS_STR)
    all_aa_1lets_ok =
[set(elem).issubset(expected_1let_aa_set) for elem in
aa_info_df['AA']]

    if not all(all_aa_1lets_ok):
        msg = "AA type letters incorrect, Amino Acid letters
can only be: 'ACDEFGHIKLMNPQRSTVWY'"
        raise SnapsImportException(msg)
```

Define the function at the top: checking the amino acid letters are correct and raising an exception if they're not.
Parameters required: Self and an amino acid info data frame
Defining a set of expected AA types that the program will accept as valid
If the dataset the user has provided doesn't match the expected set of AA types, then the program will raise SnapsImportException and provide the user with the above error message.

Figure 34- A block of code which demonstrates how a test script looks for checking correct amino acid letters are being implemented.

This is written in the test script. This script simply checks that the test is working correctly. This means that is supposed to fail- so in this instance, because it fails, the test passes! Now that we know that the test is definitely working, it can be applied to the main SNAPS program. This is slightly different here. In this case, if SnapsImportException is raised then this means the user has provided incorrect/ invalid information. This test script is specifically looking at the imports for SNAPS and so SnapsImportException is defined as a class at the


Define the function at the top
Import statement to call in and read the dataset
Using pytest to raise the class SnapsImportException.
File with 'bad data' to raise the error
Error message to describe to the user the problem with the data is asserted

```
def test_aa_letters_bad():
    importer = SNAPS_importer()

importer.import_obs_shifts('test_data/nef_resonances_4_test.nef',
'nef')
    with pytest.raises(SnapsImportException) as e:

importer.import_aa_type_info('test_data/test_aa_info_letters_bad.tx
t')
    assert "AA type letters incorrect, Amino Acid letters can only
be: 'ACDEFGHIKLMNPQRSTVWY'" in str(e.value)
```

Figure 35- A block of code that's embedded in the main script for SNAPS that tests that the information provided by the user is in fact correct/ valid. The test is validated from the previous figure.

top of the SNAPS_importer script. From the first table, J is an invalid amino acid letter, hence, an error would be raised via SnapsImportException.

2.5 NEF pipelines:

A software called NEF pipelines has also been written that allows the user to directly download chemical shift data from the BMRB and save them as NEF files or MARS files depending on the users preference. In turn, this then allows the user to access different spectra from the BMRB such as HSQC, triple resonance as well as assignment information. With NEF pipelines, the user can then plug in the spectra/ assignment information obtained from the BMRB into SNAPS for it to be assigned. This gives also gives the user the option of not having to run their own NMR experiments and can also assist in scientific contribution of assigning proteins for the BMRB.

2.6 Growth and expression of PctA:

In the lab ^{15}N labelled PctA and tyrosine unlabelled PctA was expressed and purified the protein PctA for NMR analysis. The PctA plasmid and restriction site maps were unavailable, but used from the Dr Tino Krell paper "correlation between signal input in PctA and PctB amino acid chemoreceptor of *Pseudomonas aeruginosa*" (91)

2.6.1 Transformation:

Using a pET28A plasmid with the PctA gene and BL21 [DE3] competent cells were employed (from Dr G. Thompson).

BL21 cells were thawed on ice and 20μl transferred into an Eppendorf followed by 2μl of the PctA plasmid. The mixture was then left on ice for a further 25 minutes. Following this, the mixture was heat shocked in a water bath at 42°C for 45 seconds and then put back on ice for 5 minutes. 250μl of sterile LB (table 3) was then added and then placed in a shaking incubator, 250rpm at 37°C for 60 minutes.

100μl was pipetted and spread onto agar plates with kanamycin taken from 50mg/ml stock solution. and left to incubate overnight. Several plates were made up and showed isolated single colonies with no signs of contamination.

Table 3- components of LB

LB	
Tryptone	2.5g
Yeast extract	1.25g
NaCl	2.5g
Kanamycin	250μl
Water	250ml

Table 4- Components for 250ml of LB Agar made for plates

LB agar	
Tryptone	2.5g
Yeast extract	1.25g
NaCl	2.5g
Agar	3.75g
Kanamycin	250μl
Water	250ml

2.6.2 Culture growth and expression:

Day 1:

2 starter cultures were prepared using 10ml of sterile LB and 10μl (50mg/ml stock) of kanamycin as well as a colony from the plate. These were left to incubate overnight at 250rpm and 37°C.

Day 2:

Once incubated overnight, the starter cultures were centrifuged for 15 minutes at 4000rpm, and the supernatant discarded. The. Cultures were subsequently resuspended in 10ml of sterile M9 media (detailed in tables below) and then added to the main 500ml flask (with M9 media) and afterwards placed in the incubator at 37°C at 200-220rpm and the OD measured at regular intervals to ensure the *E. coli* are growing at an expected rate. The M9 media contained ^{15}N ammonium chloride (NLM467-10- Cambridge Isotope Laboratories, obtained from Goss Scientific Instruments) to ensure that the nitrogen's in the protein are ^{15}N rather than the usual ^{14}N so that it will be visible with NMR experiments/ analysis. The M9 media was also made to pH 7.3, optimum for growth.

10X M9 salts stock:

Table 5- Components for 10X and 1X M9 salts.

10x M9 salts stock (500ml)	
$Na_2HPO_4.7H_2O$	64g
KH_2PO_4	15g
NaCl	2.5g
Water (MilliQ)	500ml
1x M9 salts	
10 X M9 salts stock	10ml
Water (MilliQ)	900ml

The components for the M9 media:

Table 6- Components of M9 media that were used to grow the *E.coli*

M9 Media	
1 x M9 salts	480ml
1M $MgSO_4$	1ml
1M $MgCl_2$	50μl
20% Glucose (w/v)	20ml
^{15}N ammonium chloride (25%)	5ml
Kanamycin	500μl
Thiamine	500μl

The initial growth goes through a lag phase before reaching an exponential growth phase. Once the OD reached 0.6, the flasks were inoculated with 0.5ml of 1M IPTG and left to incubate overnight at 25°C at 200rpm. It is important that IPTG is added because it is a molecular mimic of allolactose which triggers the transcription of the lac operon, in turn, this induces protein expression(92).

Day 3:

Following on from the overnight growth, the cells were then centrifuged at 10°C,4000rpm for 30 minutes. Again, the supernatant is discarded, and the pellet is resuspended in 10ml of 1x M9 salts in a 50ml falcon tube and then frozen.

2.6.3 PctA purification:

2.6.3.1 Lysis:

The cultures were thawed and then the falcon tube is added to a beaker containing a water and ice mixture. The culture is then placed in the sonicator (MSE Soniprep 150, power level 17-20 microns), 30 seconds on and 30 seconds off repeated for 10 minutes (still on ice) This breaks the cells open to release the protein.

The cells then need to be spun down to remove the debris from the sonication process. The solution was centrifuged for 45 minutes 12000g at 4°C and the pellet discarded.

2.6.3.2 Chromatography:

The sample then needs to be purified using nickel ion affinity chromatography, column volume (CV) 2ml. The steps used were as follows:

1. Run MilliQ and 20% ethanol to cleanse the column
2. equilibrate column with 10CV buffer A
3. load supernatant on to the column *
4. wash with 5CV buffer A *
5. wash with 5CV of buffer A containing 15% of buffer B *
6. wash with 30%B 10CV*
7. was with 60%B 10CV*
8. wash with 100%B 10CV*
9. Clean column again with 10CV MilliQ water and 10CV 20% ethanol

Key- *= Eluant kept and A280 recorded as well as 30μl sample taken for an SDS-PAGE gel.

Chromatography Buffer A		
Component	Concentration	Grams
Sodium phosphatase	20mM	1.20
NaCl	0.5M	14.61
Chromatography Buffer B		
Sodium phosphatase	20mM	1.20
NaCl	0.5	14.61
Imidazole	500mM	17
Both buffers made to pH 7.3		

Table 7- Recipe for chromatography buffers

2.6.3.3 Dialysis:

After chromatography, an SDS-PAGE gel was run to determine the fractions containing PctA and their purity (method 2.6.3.6). The nanodrop was used to determine protein concentrations (A_{280} absorbance readings) in the fractions that contained the PctA.

The fractions with the greatest absorbency of PctA protein were selected for dialysis. 4L of dialysis buffer was made up and two PctA samples were left to dialyse overnight. The point of dialysis is to remove any further imidazole present in the samples from the chromatography step as well as equilibrates the sample in the final NMR buffer.

Dialysis buffer		
Component	Concentration	Amount
KH_2PO_4	0.1M	53.2g
Na_2HPO_4	0.1M	12.0g
Water (MilliQ)	N/A	4L

Table 8- Components for dialysis buffer

2.6.3.4 Size exclusion chromatography (SEC) and concentrator:

Fractions were placed into a Regenerated cellulose concentrator (Merck, 10000mw cutoff, Amicon ultra-15 ultracel) and continuously centrifuged until a roughly 5ml volume was obtained. Duration of centrifugation varied, samples were spun at 4°C and 4000g. Following this, the sample was put in the SEC (AKTA FPLC)for further purification; a superdex 200 increase column was used and the sample was placed in NMR buffer This elutes other impurities as well as individual conformations of the protein (monomer, dimer, tetramer etc.) The buffer used for this was the same as the dialysis buffer.

2.6.3.5 Dialysis 2:

Following the SEC, samples were re-dialysed with the addition of glycerol to make more suitable for freezing as well as providing long term stability to the protein. 5.5% v/v D_2O was also added to allow for the sample to be used for NMR analysis as this acts as a lock solvent.

Dialysis buffer 2		
Component	Concentration	Amount
KH_2PO_4	20mM	9.60g
NaCl	50mM	11.68g
Alanine	3mM	1.07g
Glycerol	2%	80ml
Azide	0.3%	12ml

Table 9- Components used for second dialysis to ensure sample was pure prior to NMR analysis, 4L, pH 6.5.

4L of Dialysis buffer 2 was made, topped up with MilliQ water and made to pH 6.5.

2.6.3.6 Preparing SDS-PAGE gels:

Running SDS-PAGE tests the purity of the PctA. Making the gel requires two aspects detailed in the table below:

Table 10- Components needed to make SDS-PAGE gels

Separating Gel	
30% Acrylamide/ BIS	10ml
Separating gel buffer	9.4ml
10% SDS	250μl
50% Sucrose	4ml
Water (MilliQ)	800μl
TEMED	6.25μl
Ammonium Persulphate	625μl
Stacking Gel	
30% Acrylamide/ BIS	1.6ml
Stacking gel buffer	4.2ml
10% SDS	125μl
Water (MilliQ)	5.7ml
TEMED	5.0μl
Ammonium Persulphate	1ml

Separating Gel forms the bottom 4/5 of the chamber and the stacking gel is added to the top 1/5 with a comb inserted in the top to make the wells for the samples (from the chromatography). For each of the chromatography steps, a 20μl sample was taken - 5μl of 5x sample buffer (loading dye) was added and then heat shocked in a water bath for 10 mins at 70°C before being loaded into the gel.

The gel was then placed in a tank surrounded by MOPS buffer and the voltage set to 100v and left to run until the gel front reached the bottom.

2.6.4 Acquisition of NMR spectra:

^{1}H spectra and a ^{15}N HSQC TROSY.

Using a 600MHz Bruker Advance III NMR spectrometer equipped with a QCI-P triple resonance probe at 298K.

An HSQC spectrum size is around 25kDa and following this the best approach is TROSY(93).

2.7 Unlabelling experiments:

The same protocol for preparing samples was followed for the unlabelling experiment. The only difference comes after the overnight growth to OD 600 and induction where 1g/L of the unlabelling amino acid was added (as solid) prior to the addition of IPTG. In this case, tyrosine was used. This is the most important step of the unlabelling experiment as the tyrosine used was with ^{14}N so peaks corresponding to Tyrosine Amino Acids won't be visible.

3 RESULTS:

3.1 The NEF importer:

The script written to import NEF files has effectively been integrated into the SNAPS.py program so that NEF is now an accepted file-type for the program alongside CCPN, Sparky, MARS and xeasy.

The NEF importer improves SNAPS by making the program much more user-friendly by allowing the direct import of NEF files and formats. The NMR community lack conformity with their file types so others are allowed; but NEF is the easiest to understand and will hopefully be the most uniform file type used by the community (eventually).

NEF input support is fully integrated into SNAPS and accepted in the program. To support NEF shifts input, the following changes were made :

Different file types accepted into the script.

```
parser.add_argument("--shift_type",
                    choices=["snaps", "ccpn", "sparky", "mars",
                             "xeasy", "nmrpipe", "nef", "test"],
                    default="snaps",
                    help="The format of the observed shift file.")
```

The routine import_obs_shifts in SNAPS_importer.py was updated to import, NEF_importer.py and a call read_nef_shift_from_file_to_pandas_ was added:

```
from NEF_reader import read_nef_shifts_from_file_to_pandas

POSSIBLE_1LET_AAS_STR = "ACDEFGHIKLMNPQRSTVWY"


def import_obs_shifts(self, filename, filetype, SS_num=False):


        # Import from file
      if filetype=="snaps":
          obs = pd.read_table(filename)
      …

      elif filetype=='nef':

            obs = read_nef_shifts_from_file_to_pandas(filename)
```

The NEF shift importer itself in NEF_reader.py and consists of 11 routines. The script can be found in **appendix 1**. The PyNMRSTAR library is used to import data in the STAR file format and convert to NEF.

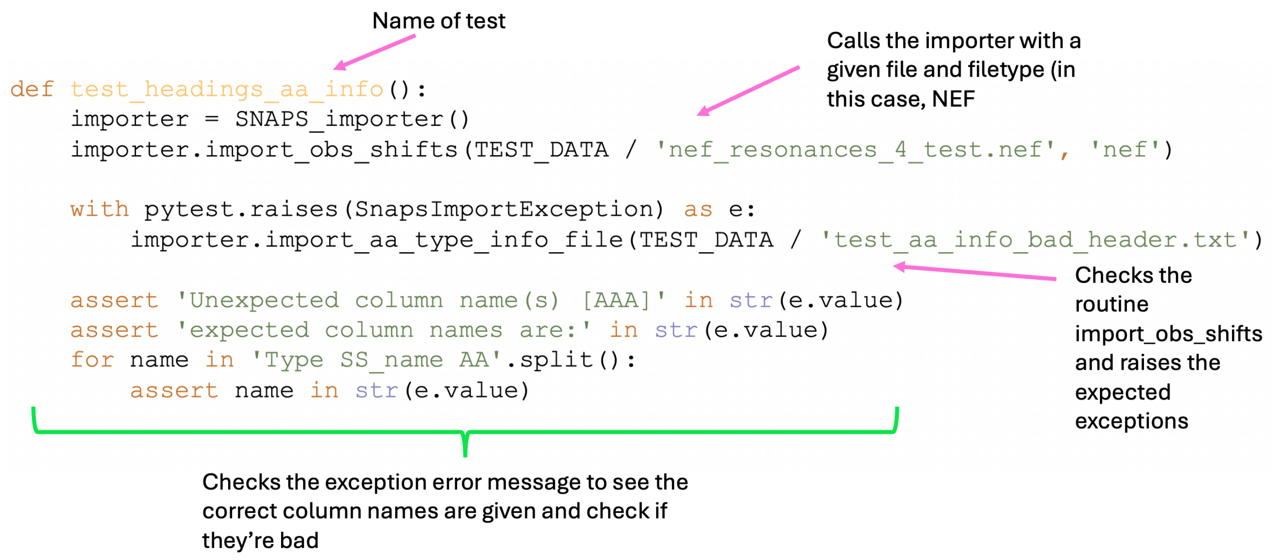
3.2 Writing of unit tests:

The writing of the unit tests is to prove just how fool-proof the dataset is. Sections of code are tested with given parameters to ensure that they run correctly and as expected. The unit tests written were to look at the import data, and more specifically the runtime errors that would be associated with them should any incorrect/ missing data is passed into the SNAPS program. Figure 36 is just an example of one of the 8 test scripts and how these are implemented to in the end give the user clear error messages (as a runtime error when the whole program is run). In turn, this then allows the user to be able to easily fix the error as well as the fact that bad/ missing data isn't allowed to pass through the program.

In the case of figure 36, an incorrect heading name is given to the input table. As a result, it means that SNAPS cannot understand it and therefore not be able to grab the necessary data to run the program. The actual column headings that are required are SS_name, AA and type. If these are wrong then SNAPS cannot run. Table 11 Shows an ideal input format. In the case of the test below, the input file has been changed so it looks like this: (AAA is not a valid column name).

Table 11- An input table with an invalid column name to raise the SNAPSImportException.

SS_name	**AAA**	Type
245Met	M	IN
01Dev	J	EX



Name of test

Calls the importer with a given file and filetype (in this case, NEF

```
def test_headings_aa_info():
    importer = SNAPS_importer()
    importer.import_obs_shifts(TEST_DATA / 'nef_resonances_4_test.nef', 'nef')

    with pytest.raises(SnapsImportException) as e:
        importer.import_aa_type_info_file(TEST_DATA / 'test_aa_info_bad_header.txt')

    assert 'Unexpected column name(s) [AAA]' in str(e.value)
    assert 'expected column names are:' in str(e.value)
    for name in 'Type SS_name AA'.split():
        assert name in str(e.value)
```

Checks the routine import_obs_shifts and raises the expected exceptions

Checks the exception error message to see the correct column names are given and check if they're bad

Figure 36- A Pytest written to check if input is bad and using assert checks an exception is raised with the correct error message. If the test is passed then it can be implemented into the program for the user to use.

3.3 The Hungarian Cost Matrix to include amino acid typing information:

The central data structure of SNAPS is the Hungarian cost Matrix. This acts to assign the predicted chemical shifts from SHIFTX2/SPARTA+ with the experimental data in the most 'cost effective' or consistent way. A full explanation of how the Hungarian Matrix acts to solve the assignment problem is found in figure 23.

To test the unlabelleing code, a 10-residue segment of the protein P3a_L273R was selected. The data from the observed chemical shifts and data from the predicted shift were prepared to see whether-or-not the matrix would match the pairs accordingly and SNAPS was modified to allow access to the raw Hungarian cost Matrix . The figure for the structure of this protein and the 10-residue segment is shown as figure 28.

Table 12- The 10-residue section of P3a_L273R used to initially test the matrix and unlabelling.

1	2	3	4	5	6	7	8	9	10
TYR	ILE	GLN	LEU	LEU	SER	GLU	ILE	ALA	LYS

This 10-residue section was selected because it is a structured part of the protein (an alpha helix) and is small enough to demonstrate a relatively straightforward assignment and analysis. This example is a small demonstration just to present how or whether the matrix works as predicted to show a solution to the assignment problem as expected.

Table 13- The input triple resonance data (chemical shift values) for SNAPS to assign the 10-residue segment of protein. Usually this would be provided as a NEF file.

OBS SHIFTS (NEF)				PRED SHIFTS (SHIFTX2)				
INDEX	RES NAME	ATOM NAME	CHEM SHIFT	INDEX	RES NAME	ATOM NAME	CHEM SHIFT	DIFFERENCES:
241	Y	C	177.245	33	Y	C	178.1953	-0.9503
242	I	C	179.489	34	I	C	178.6216	0.8674
243	Q	C	179.03	35	Q	C	178.2565	0.7735
244	L	C	179.424	36	L	C	179.9266	-0.5026
245	L	C	177.972	37	L	C	178.5337	-0.5617
246	S	C	177.101	38	S	C	177.1401	-0.0391
247	E	C	179.523	39	E	C	179.7234	-0.2004
248	I	C	177.569	40	I	C	177.4008	0.1682
249	A	C	179.917	41	A	C	178.6942	1.2228
250	K	C	179.208	42	K	C	178.6432	0.5648
241	Y	CA	61.138	33	Y	CA	61.5209	-0.3829
242	I	CA	65.162	34	I	CA	65.587	-0.425
243	Q	CA	58.581	35	Q	CA	58.3392	0.2418
244	L	CA	58.229	36	L	CA	58.1711	0.0579
245	L	CA	58.085	37	L	CA	57.9662	0.1188
246	S	CA	61.766	38	S	CA	61.8295	-0.0635
247	E	CA	59.293	39	E	CA	59.1952	0.0978
248	I	CA	63.464	40	I	CA	65.3961	-1.9321
249	A	CA	54.833	41	A	CA	54.6011	0.2319
250	K	CA	58.773	42	K	CA	59.1886	-0.4156
241	Y	CB	38.558	33	Y	CB	37.406	1.152
242	I	CB	37.949	34	I	CB	37.8132	0.1358
243	Q	CB	28.129	35	Q	CB	28.4014	-0.2724
244	L	CB	42.004	36	L	CB	41.5672	0.4368
245	L	CB	40.936	37	L	CB	41.4213	-0.4853
246	S	CB	62.695	38	S	CB	62.7807	-0.0857
247	E	CB	29.274	39	E	CB	29.4667	-0.1927

248	I	CB	37.11	40	I	CB	37.9554	-0.8454
249	A	CB	18.878	41	A	CB	17.841	1.037
250	K	CB	32.314	42	K	CB	32.1502	0.1638
241	Y	H	7.637	33	Y	H	7.8116	-0.1746
242	I	H	8.041	34	I	H	8.0554	-0.0144
243	Q	H	7.864	35	Q	H	8.1005	-0.2365
244	L	H	8.337	36	L	H	8.41	-0.073
245	L	H	8.566	37	L	H	8.5666	-0.0006
246	S	H	7.905	38	S	H	7.9916	-0.0866
247	E	H	7.889	39	E	H	7.9075	-0.0185
248	I	H	8.033	40	I	H	7.9262	0.1068
249	A	H	8.438	41	A	H	8.4586	-0.0206
250	K	H	7.348	42	K	H	7.8264	-0.4784
241	Y	HA	4.117	33	Y	HA	4.2422	-0.1252
242	I	HA	3.619	34	I	HA	3.752	-0.133
243	Q	HA	4.029	35	Q	HA	4.1145	-0.0855
244	L	HA	4.139	36	L	HA	3.956	0.183
245	L	HA	4.204	37	L	HA	4.0435	0.1605
246	S	HA	4.064	38	S	HA	4.0693	-0.0053
247	E	HA	4.067	39	E	HA	4.029	0.038
248	I	HA	3.808	40	I	HA	3.6669	0.1411
249	A	HA	2.051	41	A	HA	2.3532	-0.3022
250	K	HA	3.951	42	K	HA	3.8864	0.0646
241	Y	N	122.983	33	Y	N	124.8689	-1.8859
242	I	N	118.682	34	I	N	118.346	0.336
243	Q	N	121.655	35	Q	N	119.6493	2.0057
244	L	N	120.86	36	L	N	120.02	0.84
245	L	N	119.145	37	L	N	119.45	-0.305
246	S	N	114.314	38	S	N	115.2441	-0.9301
247	E	N	122.251	39	E	N	121.4367	0.8143
248	I	N	122.227	40	I	N	121.2715	0.9555
249	A	N	123.367	41	A	N	122.4152	0.9518
250	K	N	116.377	42	K	N	116.3273	0.0497

Table 14- The output 10x10 matrix for showing the best assignments for this 10-residue section of protein. The Green highlighted regions show the best matched solutions for each of the 10 residues.

Res_name	33Y	34I	35Q	36L	37L	38S	39E	40I	41A	42K
241Y	-5.569053	-21.487264	-59.748468	-23.486173	-21.238368	-291.97252	-48.159483	-16.45902	-268.59397	-31.891097
242I	-20.060049	-4.83341	-75.247407	-43.10468	-43.391379	-307.82373	-60.215889	-6.734772	-287.82338	-40.416968
243Q	-51.070505	-78.575118	-5.091785	-95.078503	-92.365754	-585.23819	-5.260603	-77.162114	-96.977035	-14.83357
244L	-26.61955	-50.192168	-96.393888	-4.697919	-5.172901	-219.14093	-81.806168	-47.625966	-313.02787	-56.152572
245L	-23.812334	-44.927075	-80.935892	-7.309079	-4.708396	-239.75557	-71.142582	-42.591081	-291.3394	-44.912928
246S	-321.76504	-312.94988	-573.94218	-226.80055	-228.57218	-4.032074	-538.921	-308.89556	-1003.9934	-453.30649
247E	-39.480306	-64.273689	-6.615158	-81.854862	-80.112638	-547.57933	-4.10792	-63.223786	-114.71548	-11.971707
248I	-8.842661	-9.341097	-55.251977	-36.717822	-34.501144	-323.99142	-43.896986	-6.741095	-255.64816	-28.650465
249A	-249.86586	-289.87815	-105.279	-282.20196	-282.45566	-972.08815	-114.39883	-285.97218	-6.560792	-140.40588
250K	-27.811536	-45.770107	-13.882974	-51.485557	-49.789161	-454.24016	-10.133211	-46.053763	-144.2071	-4.901342

From the data (table 14), it is clear to see that the best matches are shown in green. The data is clearly correlated as the lowest numbers are displayed in green which are all across the diagonal, so the lowest possible cost solution is given. Table 13 shows the full input data table. The differences between the chemical shift columns have been applied and there is no greater difference here than 2.0057. This implies that there was great success in the assignment.

You can also see that the numbers across the diagonal are always the smallest number in that column and row. This is because the Hungarian matrix works to find the lowest cost solution. This is only a 10x10 matrix and so it is much simpler to solve. However, the Hungarian algorithms act to rapidly solve the assignment problem, even on much larger data sets, with much more data- such as those for large proteins. It can be inferred from this example alone that the algorithm behaves in the way that it is expected to in order to best align the data, regardless of the size of the dataset.

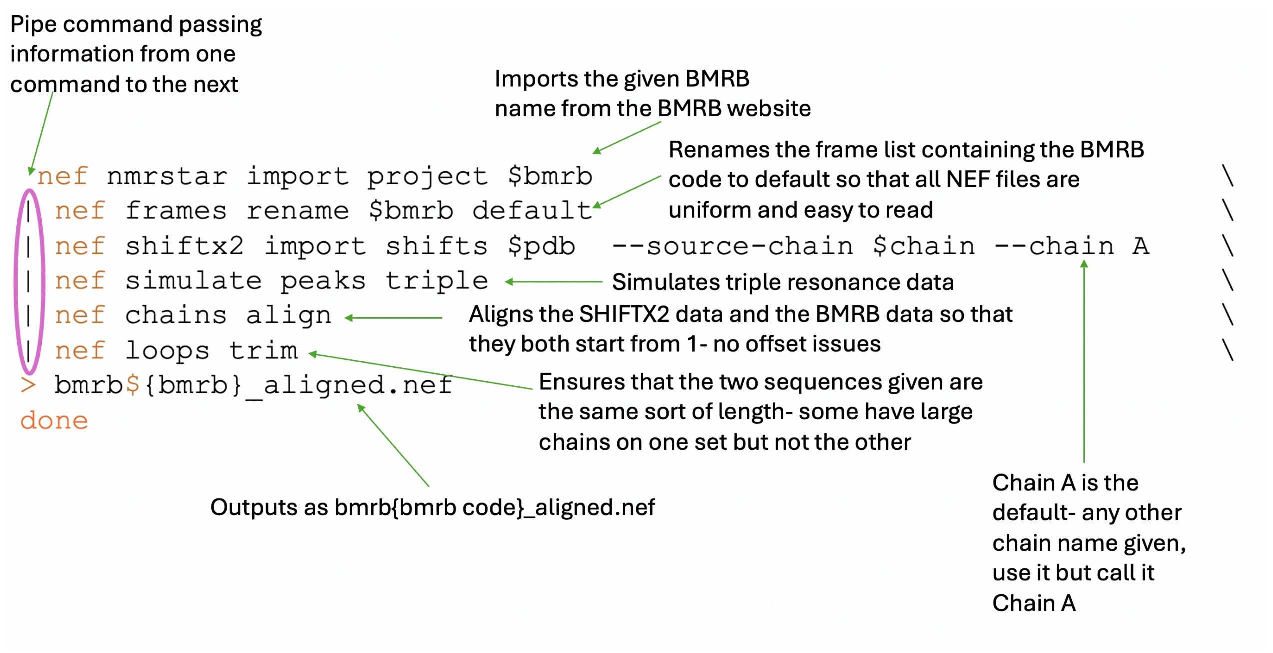
Another important aspect about this algorithm is time complexity. This is a type of computational complexity that describes the time required to execute an algorithm. If this is very long, then it isn't the best solution for working on large test sets of data- otherwise the user will be waiting a very long time for their data to be ready! (94) On the whole, the Hungarian algorithm has a worst-case run-time complexity of O(n^3), which means it's pretty good at producing rapid solutions and won't leave the user waiting for unnecessary extended periods.

3.4 To what degree does SNAPS accurately assign proteins?

A list of 145 proteins from the BMRB, more specifically those that were used in producing SHIFTX2 were selected to build a robust dataset to test the accuracy of SNAPS to assign proteins. These were selected because they display proteins with a wide range of sizes and different dynamics. This is largely important because this can be used to best showcase how accurately and effectively SNAPS can work on a myriad of different proteins. As well as varying in structure and dynamics, the proteins also vary in length (the smallest being 56 residues and the largest 872). The program can only be considered a success if it is able to assign a range of different proteins (various sizes, structural composition etc.,)

The list of proteins in the test set were from SHIFTX2 and had to be taken from supplementary information on their website and then placed in text file. This is so that it could be used by BASH scripts to import the data directly from the BMRB and also from SHIFTX2- so that this may then be run in SNAPS. NEF pipelines allow you to use a myriad of different tools to import from the BMRB as well as SHIFTX2 and produce AlphaFold models! Everything is run in pipelines with the “|” (pipe) symbol where the output of one pipe function is passed along to the next. This is a hugely powerful tool for the import of the BMRB and SHIFTX2 datasets.

Here is an example of how NEF pipelines can work to format imported BMRB files and SHIFTX2 shifts:



```
 nef nmrstar import project $bmrb                                         \
 | nef frames rename $bmrb default                                        \
 | nef shiftx2 import shifts $pdb  --source-chain $chain --chain A        \
 | nef simulate peaks triple                                              \
 | nef chains align                                                       \
 | nef loops trim                                                         \
> bmrb${bmrb}_aligned.nef
done
```

Pipe command passing information from one command to the next

Imports the given BMRB name from the BMRB website

Renames the frame list containing the BMRB code to default so that all NEF files are uniform and easy to read

Simulates triple resonance data

Aligns the SHIFTX2 data and the BMRB data so that they both start from 1- no offset issues

Ensures that the two sequences given are the same sort of length- some have large chains on one set but not the other

Outputs as bmrb{bmrb code}_aligned.nef

Chain A is the default- any other chain name given, use it but call it Chain A

Figure 37- An example of the original NEF pipelines import script and a breakdown of what the different functions can do.

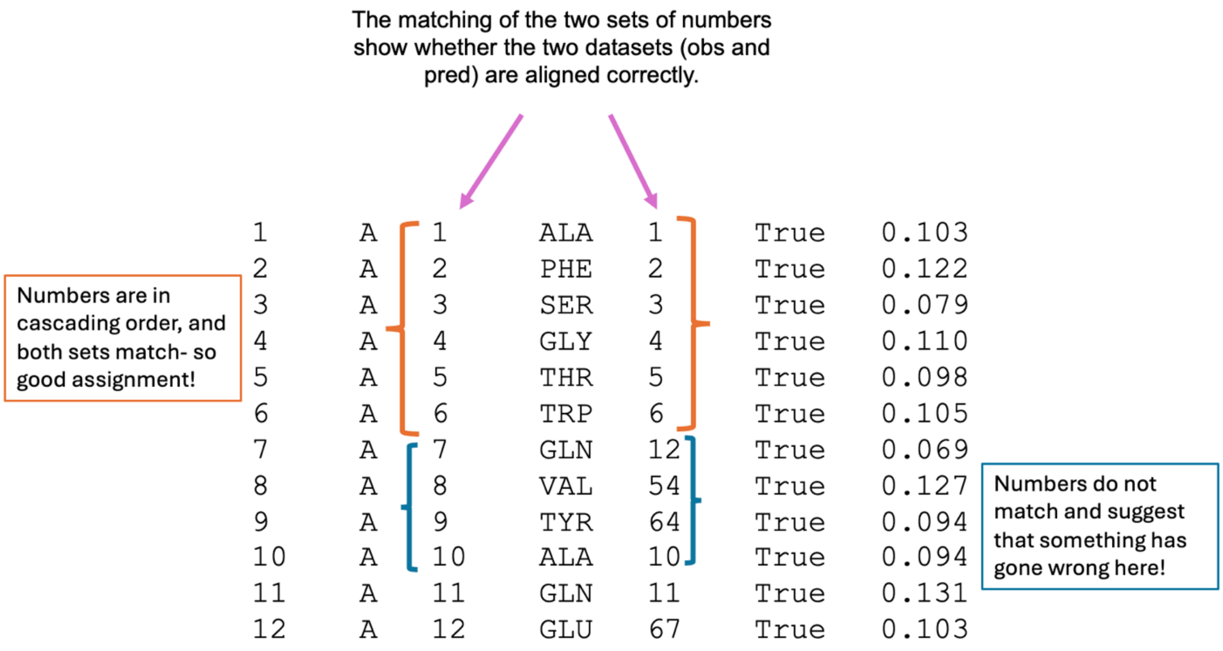
The import script underwent a number of changes to ensure all of the data was imported. This was to include corrections of a number of errors found in the NEF pipelines tools.

As each of the proteins has both a BMRB shift list and assignment and a set of coordinates from which SHIFTX2 can predict shifts it is possible to compare

The proteins listed are in accordance with their BMRB code and their PDB code. Each of them already has 'known' assignments and so are listed on the BMRB. This is particularly useful as it means that the proteins can be assigned using SNAPS. Then a comparison can be drawn to determine just how effective SNAPS is at producing assignments for different proteins. The table of proteins used can be found in **appendix 2**. It should be noted that some of the proteins in the SHIFTX2 test set are not in the BMRB and so can't be converted, as well as some listed have more than one PDB code, these are noted in **appendix 3**.

A spearman's rank correlation coefficient as well as t testing might have been viewed as the optimal method to test significance and validity. However, these values can't be obtained without a mean and standard deviation for the datasets. Instead, manual analysis of the data was used as an alternative. A simple score of the number of correctly assigned spin systems for each protein was used. This analysis was greatly helped by arranging for the numbering of assigned and unassigned spin systems to match so that the nef output table could be easily analysed (see figure 37)

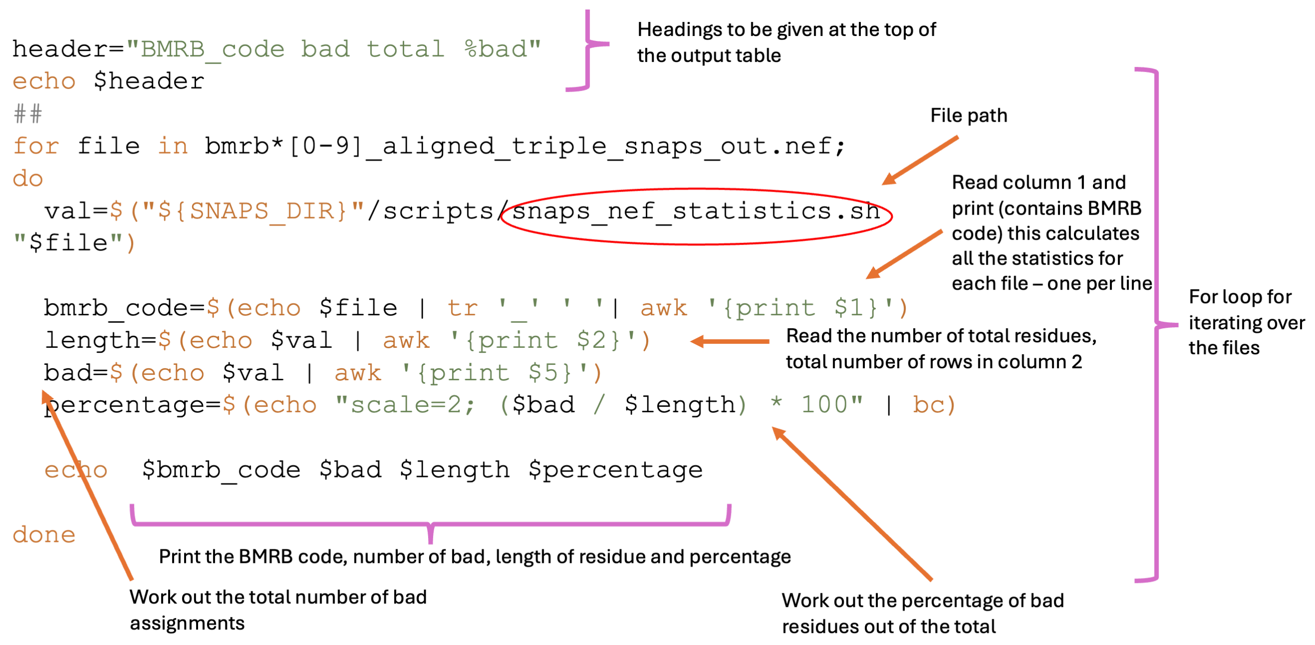
Now that SNAPS was run on the 145 proteins in the test set, the best way to assess to what degree does SNAPS work to better assign proteins is to have a look at the output files and see how many residues have been correctly assigned.


The matching of the two sets of numbers show whether the two datasets (obs and pred) are aligned correctly.
Numbers are in cascading order, and both sets match- so good assignment!
Numbers do not match and suggest that something has gone wrong here!

```
1     A   1    ALA   1    True   0.103
2     A   2    PHE   2    True   0.122
3     A   3    SER   3    True   0.079
4     A   4    GLY   4    True   0.110
5     A   5    THR   5    True   0.098
6     A   6    TRP   6    True   0.105
7     A   7    GLN   12   True   0.069
8     A   8    VAL   54   True   0.127
9     A   9    TYR   64   True   0.094
10    A   10   ALA   10   True   0.094
11    A   11   GLN   11   True   0.131
12    A   12   GLU   67   True   0.103
```

Figure 38- An example of a section of SNAPS NEF file assignment with some good assignment as well as some incorrect assignments. The region highlighted in blue shows that SNAPS can't predict 100% of these proteins assignments as there are clear inconsistencies with the two datasets here.

Once all 142 proteins in the test set have been analysed with SNAPS, a shell script was written to print out the total number of residues in the protein as well as the number of mismatches within the NEF file to determine just how many ‘bad’ assignments there are in respect to the size of the protein.



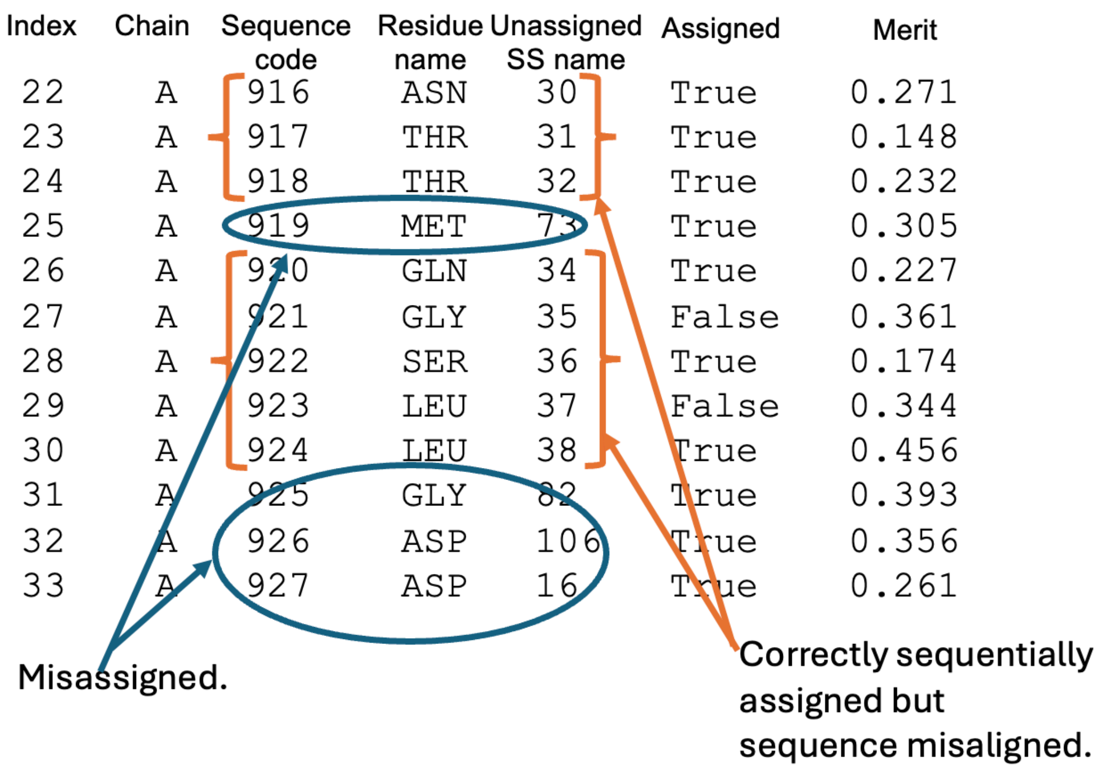
```
header="BMRB_code bad total %bad"
echo $header
##
for file in bmrb*[0-9]_aligned_triple_snaps_out.nef;
do
  val=$("${SNAPS_DIR}"/scripts/snaps_nef_statistics.sh
"$file")

  bmrb_code=$(echo $file | tr '_' ' '| awk '{print $1}')
  length=$(echo $val | awk '{print $2}')
  bad=$(echo $val | awk '{print $5}')
  percentage=$(echo "scale=2; ($bad / $length) * 100" | bc)

  echo  $bmrb_code $bad $length $percentage

done
```

Headings to be given at the top of the output table
File path
Read column 1 and print (contains BMRB code) this calculates all the statistics for each file – one per line
For loop for iterating over the files
Read the number of total residues, total number of rows in column 2
Print the BMRB code, number of bad, length of residue and percentage
Work out the total number of bad assignments
Work out the percentage of bad residues out of the total

It should be noted that “snaps_nef_statistics.sh” is a shell script used to work out the total number of bad assignments out of all the residues and give the value as a percentage.

Figure 39- The script used to analyse the snaps_out.nef files to determine how many 'bad' the assignments there are for each protein.

In order to effectively analyse the data, some issues needed to be overcome. The first of these was that a number of proteins the sequence in the PDB file did not match that in the BMRB file. As a result, it would seem from this test alone that certain proteins were left 100% unassigned; in fact, this is not the case. It would seem that SHIFTX2 doesn’t always start the sequence with 1, unlike the observed (obs) dataset. This is an issue that needed to be overcome. The results from this are in **appendix 4.** Figure 40 shows an example of how the output files looks when the offset is bad.



Index	Chain	Sequence code	Residue name	Unassigned SS name	Assigned	Merit
22	A	916	ASN	30	True	0.271
23	A	917	THR	31	True	0.148
24	A	918	THR	32	True	0.232
25	A	919	MET	73	True	0.305
26	A	920	GLN	34	True	0.227
27	A	921	GLY	35	False	0.361
28	A	922	SER	36	True	0.174
29	A	923	LEU	37	False	0.344
30	A	924	LEU	38	True	0.456
31	A	925	GLY	82	True	0.393
32	A	926	ASP	106	True	0.356
33	A	927	ASP	16	True	0.261

Misassigned.
Correctly sequentially assigned but sequence misaligned.

Figure 40- A section of a NEF file where the numbers seem to be very misaligned and so doesn't appear assigned when the analysis tool is run. The numbers highlighted in orange on each side seem to go up in a cascading order with a consistent offset between the two sets of data, so this would imply that SNAPS has worked to correctly assign some of the protein.

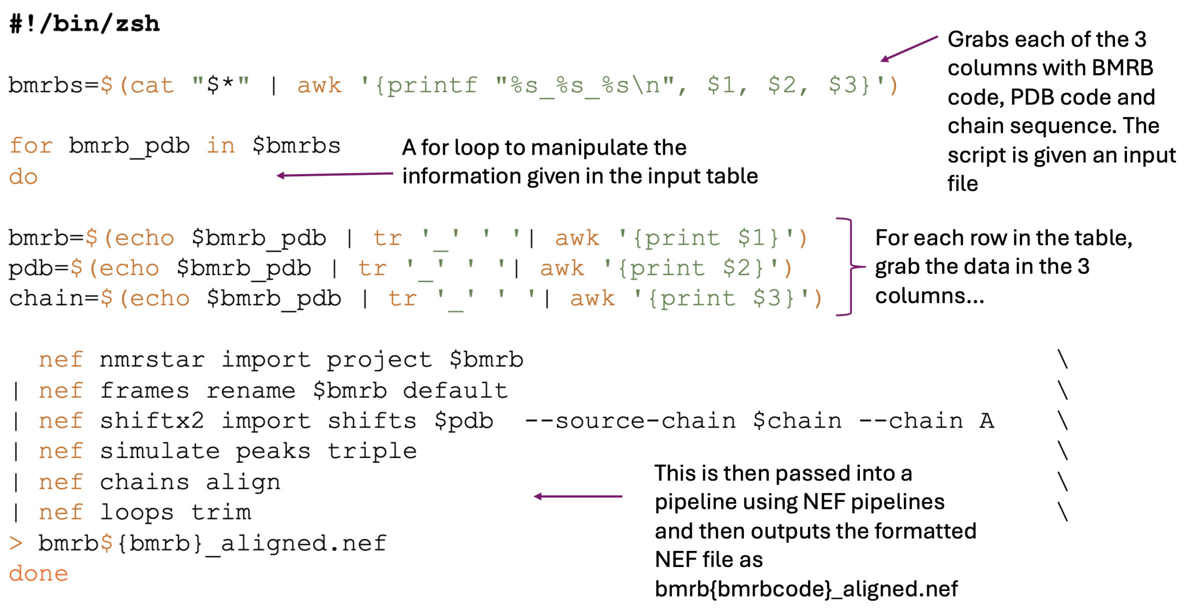
Despite this, the simple shell script was a powerful tool to analyse how well SNAPS assignments were working. As an example, the 124-residue segmented bovine ribonuclease (PDB 3RN3, BMRB 4031) was assigned by SNAPS with a 96.77% accuracy! Only 4/124 residues were badly assigned! This is a good example, but it's important to assess multiple proteins. For example, SNAPS may only be good with dealing with small proteins/ peptide chains. This would be because when proteins are smaller, there are less combinations of amino acids and so less room for error (in comparison to larger proteins). Another theory could be that SNAPS assigns better based on dynamics. Proteins with large regions of random coil can be much more difficult to assign due to it being a more fluid structure. This would a much harder to assign compared to an amino acid found in a beta sheet or alpha helix. This is because of hydrogen bonding of the molecule. Beta sheets and Alpha helices have hydrogen bonding which makes it more stable.

To overcome issues with alignments, the script would need to be amended to take this issue into account. With the use of NEF pipelines, the script to import the data from the BMRB is very simple as well as driving consistency. There was an option to align the datasets. This essentially means that if the dataset comes from SHIFTX2 and starts with for example, 75Ala, 76Met, 77His and the observed dataset (in this case from the BMRB) starts with 1Ala, 2Met, 3His then the SHIFTX2 sequence will be changed to start from 1 and therefore fix the offset issue. In turn, this makes the data much more easy and quicker to analyse. As a result of this, the first run without the align command had 52/114 or 45% of items in the test set were marked as "99-100% misassigned" meaning that when the analysis tool is run, they appear completely misassigned. When the align command was added this changed to 8/109 7% were between "98-100% misassigned" so shows a dramatic change to how the data is perceived.

The next task was to work on the remaining 7% that were still presenting as poorly assigned. Upon closer inspection and looking at the different inputs (being the SHIFTX2 and the BMRB data) it was evident that the wrong chains were being selected for certain proteins. Some proteins in the PDB have more than one chain, this needs to be specifically given to the input file script otherwise it takes from chain A rather than chain B, C, D… etc. Thankfully, in most cases it seemed that chain A was the one that needed to be selected, but there were a few instances where this wasn't true. As a result, different chains were being read and analysed which have different sequences or are completely different in length! Adding this change to the script resulted in the number of "98-100% misassigned" to go from 8/109 to 3/109.

It may be noticed from the previous data mentioned but run 1 had a total of 114 residues and this drops to 109 on the second run. Upon further analysis of comparing the differences between the first two runs there were 5 proteins that were missing in the first run but present in the second as well as 10 proteins that were listed in the first on but not in the second. This is most likely due to the addition of the align command, and potential bugs within NEF pipelines itself. However, further investigation needed to be undertaken to understand as best as possible why these residues cease to produce any results and can no longer be run within SNAPS.

There were still some issues regarding alignment again. When the NEF files were saved, the SHIFTX2 and BMRB files did seem to match quite well but when run in SNAPS, the



```
#!/bin/zsh

bmrbs=$(cat "$*" | awk '{printf "%s_%s_%s\n", $1, $2, $3}')

for bmrb_pdb in $bmrbs
do

bmrb=$(echo $bmrb_pdb | tr '_' ' '| awk '{print $1}')
pdb=$(echo $bmrb_pdb | tr '_' ' '| awk '{print $2}')
chain=$(echo $bmrb_pdb | tr '_' ' '| awk '{print $3}')

  nef nmrstar import project $bmrb                                    \
| nef frames rename $bmrb default                                     \
| nef shiftx2 import shifts $pdb  --source-chain $chain --chain A     \
| nef simulate peaks triple                                           \
| nef chains align                                                    \
| nef loops trim                                                      \
> bmrb${bmrb}_aligned.nef
done
```

Grabs each of the 3 columns with BMRB code, PDB code and chain sequence. The script is given an input file

A for loop to manipulate the information given in the input table

For each row in the table, grab the data in the 3 columns...

This is then passed into a pipeline using NEF pipelines and then outputs the formatted NEF file as bmrb{bmrbcode}_aligned.nef

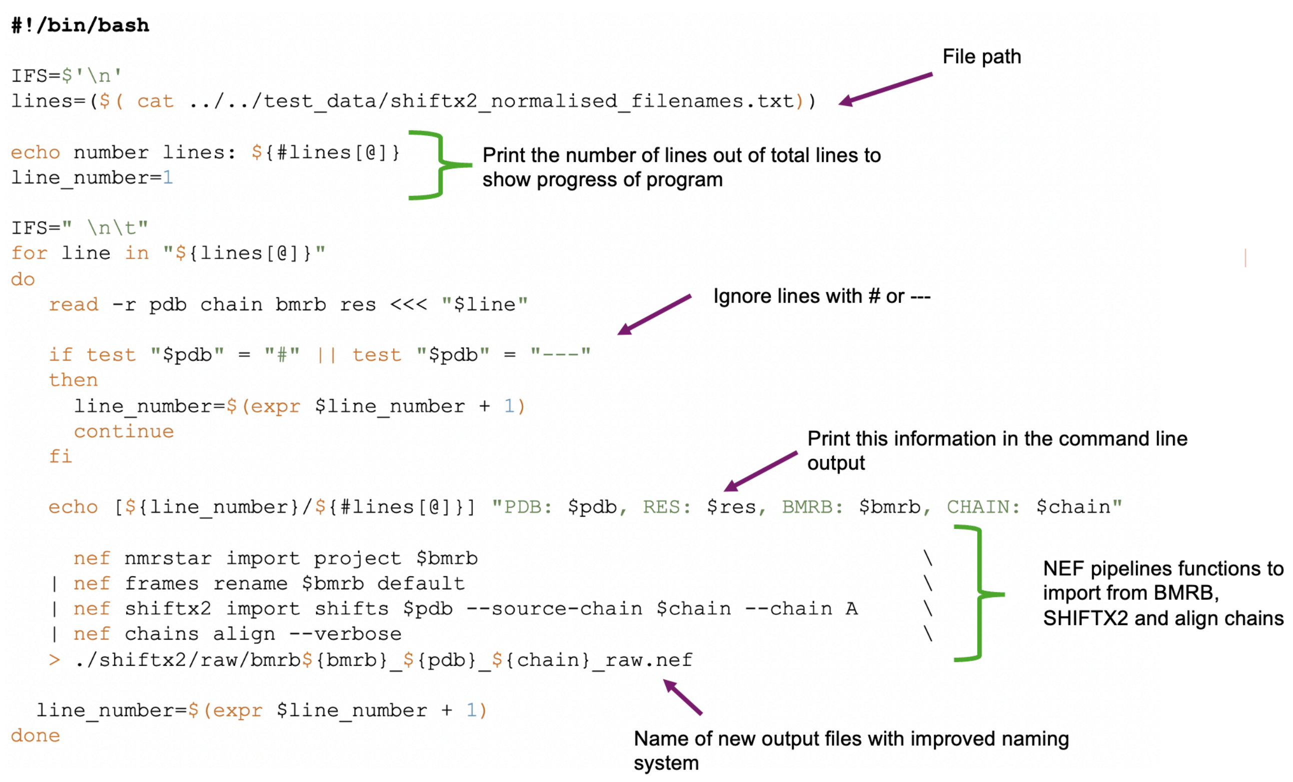
Figure 41- The original shell script used to import data from SHIFTX2 and the BMRB to produce suitable NEF input files to put into SNAPS. It is broken down to grab the appropriate columns from the input file and apply these to the NEF pipelines for formatting and eventually having the correct format to use in SNAPS.

It was discovered that some BMRB codes had more than one 1PDB match. This meant that in some cases when the same BMRB code was used more than once with different PDB codes, it was instantly writing over the previous file rather than saving it as a new one. This called for a new filing method where files were saved with all of the relevant information to prevent this writing over error to happen again. This accounted for around 10 missing files from the original list of 145.

It should be noted that some BMRB codes couldn't be used because they weren't V3 BMRB files (they were V2) and so couldn't be read. One protein also contained a non-standard residue which also meant this wasn't compatible with SHIFTX2 and so was ignored. Some files required multiple BMRB files. For some files were only available in SPARTA+ and so were ignored also.

Another issue encountered with this script was that when there were updates to NEF pipelines or different data needed to be passed through it, SHIFTX2 would often crash (for reasons unknown) so out of the 145 sets of data that needed to be run through this would randomly meet errors within SHIFTX2 and crash. It also didn't produce sufficient output when running. If the BMRB code had passed successfully the script would output "bmrb{bmrb_code} 0" or if unsuccessful "bmrb{bmrb_code} 1" as well as the stack trace. But in some cases, even where it seemed to be a successful download from the BMRB and SHIFTX2, there were errors with running them in SNAPS or the file appeared empty altogether.

An improved import script was therefore written with many similarities to the first but with changes to make it more robust and of a higher efficiency for when dealing with larger volumes of data, this is shown in figure 42.



```
#!/bin/bash

IFS=$'\n'
lines=($( cat ../../test_data/shiftx2_normalised_filenames.txt))

echo number lines: ${#lines[@]}
line_number=1

IFS=" \n\t"
for line in "${lines[@]}"
do
   read -r pdb chain bmrb res <<< "$line"

   if test "$pdb" = "#" || test "$pdb" = "---"
   then
     line_number=$(expr $line_number + 1)
     continue
   fi

   echo [${line_number}/${#lines[@]}] "PDB: $pdb, RES: $res, BMRB: $bmrb, CHAIN: $chain"

     nef nmrstar import project $bmrb                                  \
   | nef frames rename $bmrb default                                   \
   | nef shiftx2 import shifts $pdb --source-chain $chain --chain A    \
   | nef chains align --verbose                                        \
   > ./shiftx2/raw/bmrb${bmrb}_${pdb}_${chain}_raw.nef

  line_number=$(expr $line_number + 1)
done
```

File path
Print the number of lines out of total lines to show progress of program
Ignore lines with # or ---
Print this information in the command line output
NEF pipelines functions to import from BMRB, SHIFTX2 and align chains
Name of new output files with improved naming system

Figure 42- An improved version of the import script where each protein dataset imported from the BMRB/ SHIFTX2 is shown in greater detail in the shell to show which protein out of how many is being imported with each step. There is also an improved naming system for the files- now saved by PDB and chain code as well as BMRB entry number. Simulating triple resonance and unlabelling is not applied at this stage.

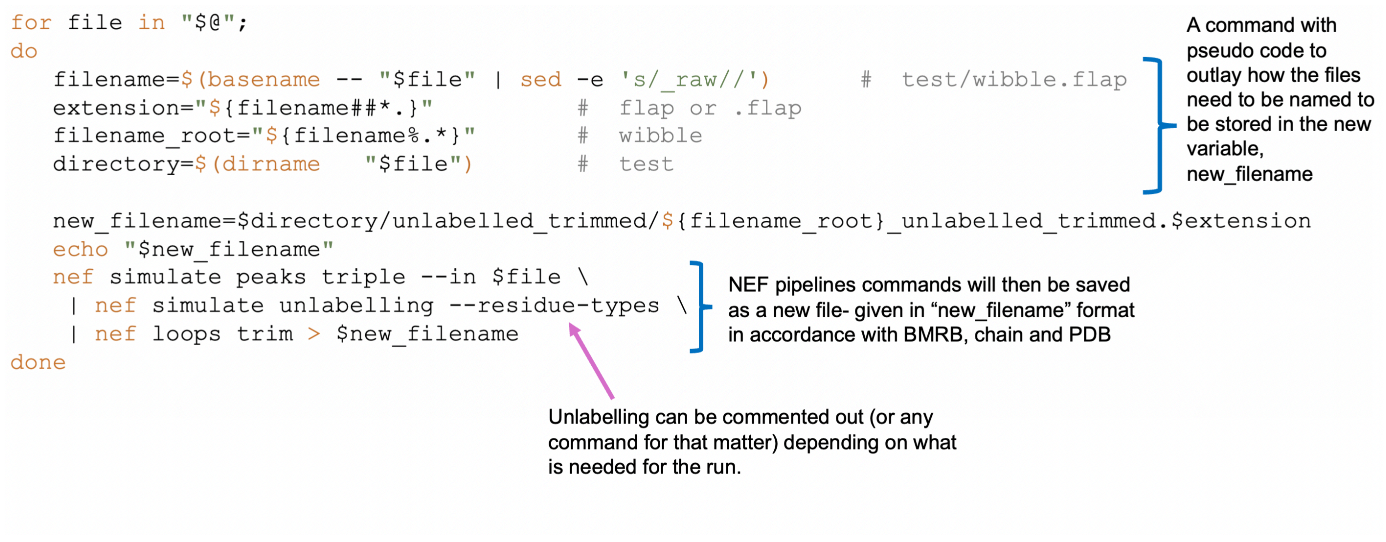
The main idea for this improved import script was to initially save everything with the format:

"bmrb{bmrb_code}_{PDB}_{chain}_raw.nef" doing the minimum to read the BMRB and SHIFTX2

This naming change is also vastly important due to the errors mentioned above where some BMRB codes had more than one corresponding PDB code or chain that was instantly being written over by the most recent run.

The use of the 'raw' dataset also meant that there was a raw unedited dataset which contained the BMRB data as well as that of SHIFTX2. As a result, SHIFTX2 didn't have to be re-run multiple times over (because doing this was very time consuming and sometimes the SHIFTX2 server wasn't working very well). The 'raw' data provides a better jumping off point for manipulating the data further with features such as align or trim or having to do re-runs because of errors without having to wait for extended periods of time for SHIFTX2 to run; so, much simpler!

The ‘raw’ dataset was then used by another script which included nef pipelines functions align and trim as well as simulating the triple resonance data- for later analysis and adding simulated unlabelling. This is shown in figure 43.



```
for file in "$@";
do
   filename=$(basename -- "$file" | sed -e 's/_raw//')       #  test/wibble.flap
   extension="${filename##*.}"           #  flap or .flap
   filename_root="${filename%.*}"        #  wibble
   directory=$(dirname   "$file")        #  test

   new_filename=$directory/unlabelled_trimmed/${filename_root}_unlabelled_trimmed.$extension
   echo "$new_filename"
   nef simulate peaks triple --in $file \
    | nef simulate unlabelling --residue-types \
    | nef loops trim > $new_filename
done
```

A command with pseudo code to outlay how the files need to be named to be stored in the new variable, new_filename

NEF pipelines commands will then be saved as a new file- given in “new_filename” format in accordance with BMRB, chain and PDB

Unlabelling can be commented out (or any command for that matter) depending on what is needed for the run.

Figure 43- The new import script used to manipulate the "raw" dataset. This was used to apply additional NEF pipelines commands such as align, trim etc., and save it as a suitable file to be used as an input into SNAPS.

Data could then be saved as “bmrb{bmrb_code}_{PDB}_{chain}_triple.nef” or for unlabelled data“bmrb{bmrb_code}_{PDB}_{chain}_unlabelled_triple.nef” and then is passed into the running SNAPS script to produce:

“bmrb{bmrb_code}_{PDB}_{chain}_triple_snaps_out.nef”

The script for running the data in SNAPS is fairly simple and straightforward as long as the correct parameters are given. As already discussed, the Hungarian algorithm is quick to provide output- even when 134 sets of data are being tested back-to-back (10-20 minutes total)

Clear and concise file naming is very important for identifying the correct data and ensuring that it is all kept accordingly. Here is a final summary of each run in excel for processing the SHIFTX2 data set. This table shows the control dataset. For this dataset, the command '*NEF simulate unlabelling – residue-types'* wasn't used:

Run Number	Total number of proteins in that run	Total % misassigned	Run comments
1	97	18.6	First "initial run"
2	99	18.3	Adding alignment data
3	109	16.6	Improvements to align
4	131	15.8	Improvements with imports
5	134	14.5	Adding the trim function
6	140	14.3	Improvements to align and SNAPS updates

Table 15- The amendments made for each run and how successful each run was- highest score being 0% misassigned.

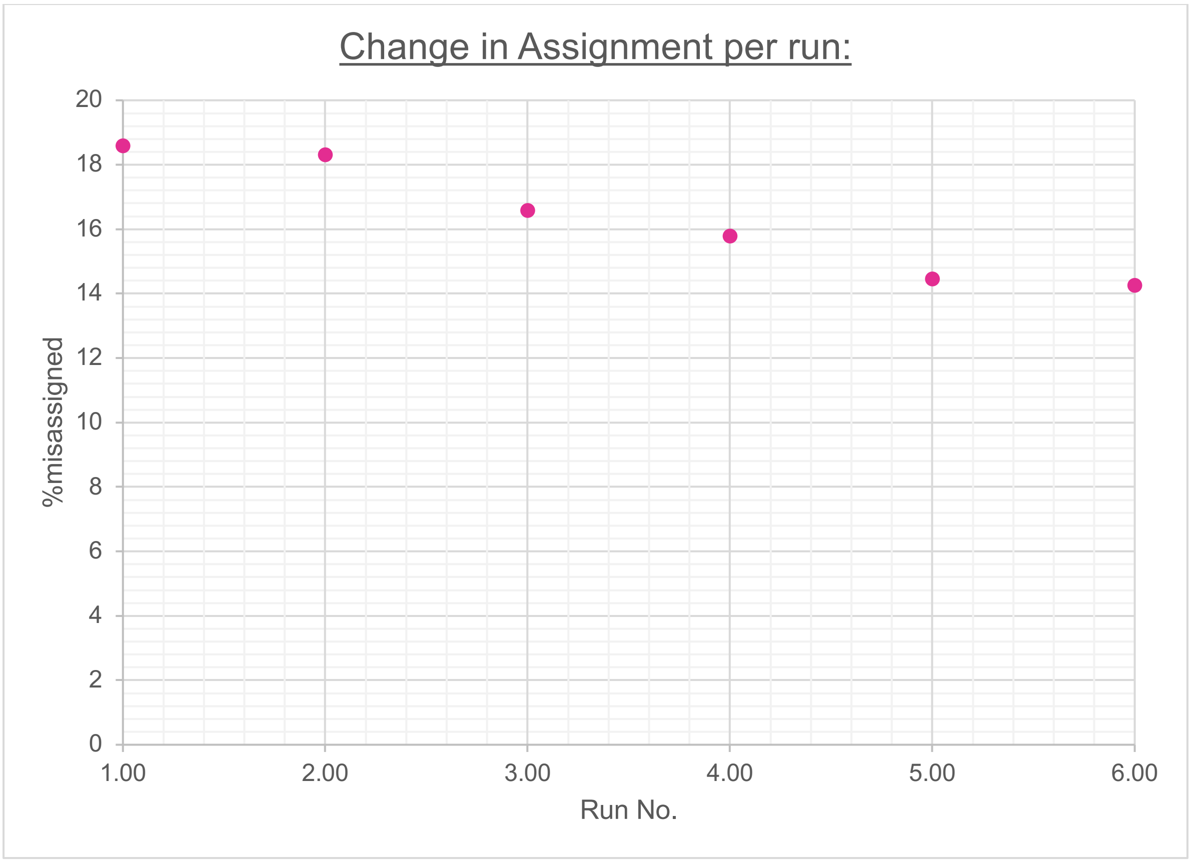


Figure 44- A graph to show the gradual decrease in %misassigned for each run. As you can see from table 12, each change allowed for gradual decrease in %misassigned.

The final run of SNAPS bought the dataset to only 14%misassigned– so 86% of all of the dataset successfully assigned. It is believed the remaining errors are due to inaccuracies in SNAPS/ the shifts in SHIFTX2 in the underlying data. Final run results in **Appendix 5.**

3.5 To what degree does SNAPS work with unlabelling data?

In order to determine how effective the unlabelling experiment is, all tests listed in the last section were performed on both unlabelled and non-unlabelled or, 'control' proteins in parallel. This is important because the addition of data from unlabelling should hopefully show improvements to the assignment of the poorer data sets. As mentioned previously, the unlabelling should tell SNAPS categorically where some of the assignments of the amino acids are. So as a result, it would be expected that there is a better standard/ higher percentage of correctly assigned residues here because SNAPS has a reduced number of matches to choose from.

Run Number	Total number of proteins in that run (with unlabelling command)	Total %misassigned	Run Comments
1	114	54.5	First “initial run” with unlabelling command
2	109	23.6	Adding Alignment data
3	109	20.3	Chain fixes
4	112	20.0	Finding missing data- running SHIFTX2 manually
5	113	19.9	Finding missing data- running SHIFTX2 manually- attempt 2
6	131	15.7	Improvements to align and import script
7	134	14.5	Addition of trim function
8	140	12.9	Chain and PDB information in file name to avoid duplicates being written over

Table 16- Each unlabelled run and the %misassigned, the %misassigned seems to decrease with each given run but there are fluctuations in the total number of proteins that passed each run also.

Unlabelling was simulated in NEF pipelines with the command “nef simulate unlabelling”. This simulates a clean unlabelling of CYS, GLN, ARG, ASN, MET.

With the first attempts of implementing the unlabelling command, the results were identical to the data set where the unlabelling command wasn't being called. This could be because of the following:

- The first idea is that the unlabelling doesn't provide any difference to the SNAPS assignment algorithm. The same outcome is reached with and without the unlabelling data.
- The second theory is that something wasn't working within SNAPS – so when the script is being run, the unlabelling data cannot be utilised properly, thus it appears to not present any difference.

The SNAPS program needed to be re-examined. Something was clearly missing as the unlabelling data provided hasn't been utilised whatsoever. A few amendments had to be made to the program in order to ensure that the correct things were being read by SNAPS for the unlabelling data. The problem was that SNAPS needs the unlabelling data in the format {residue number}_{residue} e.g., 123Asp. The unlabelling data from NEF piplelines was giving this in the wrong format- which wasn't compatible with SNAPS. A mini script was written to amend the files using awk:

```
BEGIN {in_unlabelling=0}

/save_nefpls_residue_types_/ {in_labelling=1}

in_labelling && /stop_/ {in_labelling=0}

in_labelling && NF==5 {printf "    %2i    %s   %s%s    %s    %s\n",
$1, $2, $3,$4, $4, $5}

in_labelling && NF != 5 {print $0}

!in_labelling {print $0}
```

Essentially, this combines the columns in the unlabelling section of the NEF file only.

The other issue was that in the script where the data was run in SNAPS, the command where it tells the program that there is unlabelling data present, just simply wasn't being used! The script needed to have a command to tell SNAPS to use the unlabelling data. This was executed by including the argument `"-- aa-restrains $file"` this then tells the program where the unlabelling data is and to use it.

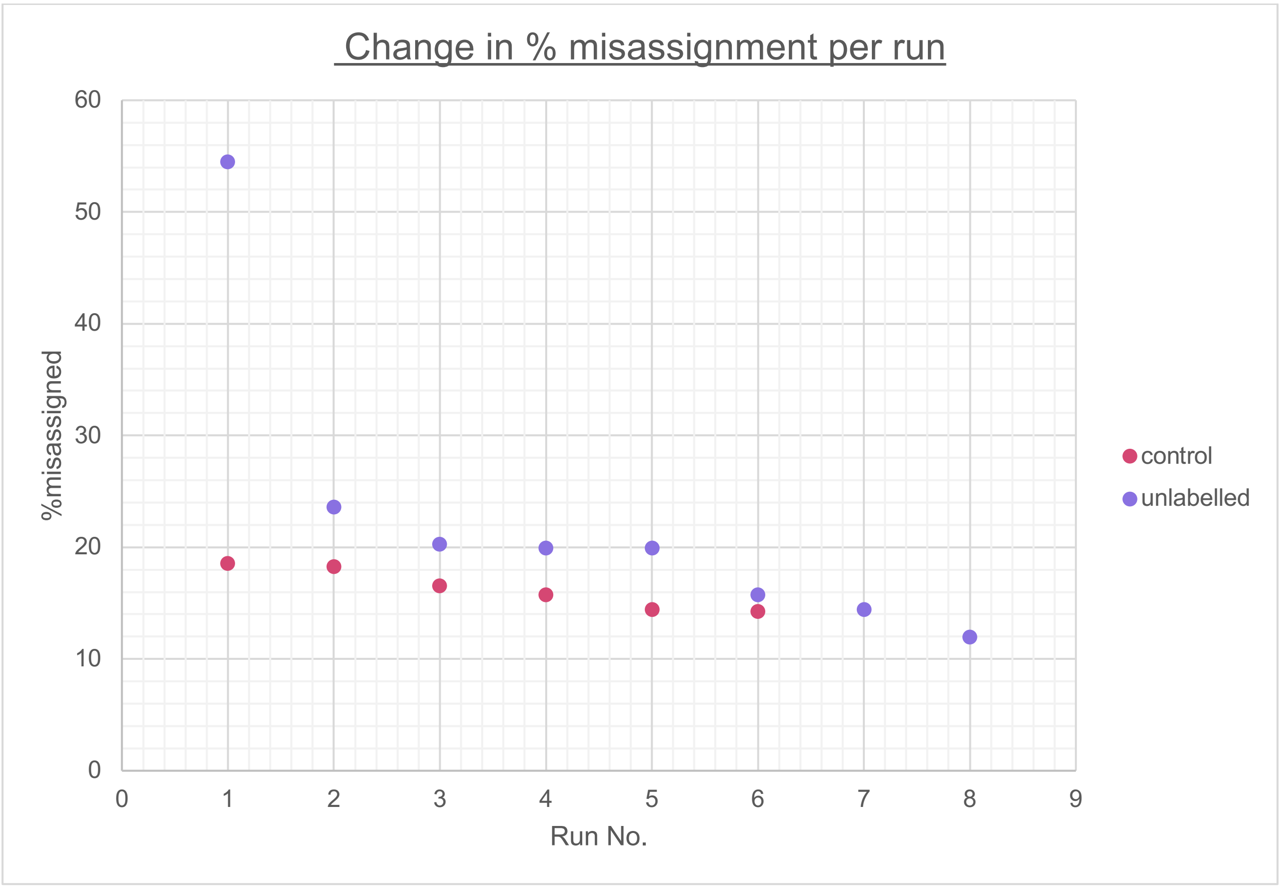


Figure 45- A graph to show how both unlabelled and control datasets both decreased in %misassigned through each run. Unlabelled had a lower misassigned value and so could argue as more successful!

From the final unlabelling run, there is a %misassigned of an average of 12%, this is 2% better than the control run and so suggests that the unlabelling is being correctly used to assign residues. Although 2% isn't a large improvement, this is only shown as an average out of the 140 proteins that are present in both the control and unlabelled datasets respectively. The table below looks at the 5 worst assigned proteins in the whole dataset. In the control data, the worst misassignment value is 68%, with unlabelling, this improves by 10%. All of the worst datasets improve somewhat with the addition of unlabelling data.

	% Misassigned		
BMRB/PDB code	**Controlled**	**Unlabelled**	**%Change**
15722	47	41	6
4061	68	58	10
4174	56	40	16
4127	49	36	13

Table 17- The highest misassigned residues for unlabelled and control and the % difference between the two.

From the 140 that ran, 64 showed improvement when unlabelling was implemented. The residues that demonstrated the greatest improvement when unlabelling was implemented were:

Unlabelling also made some datasets worse:

	% Misassigned		
BMRB/PDB code	**Controlled**	**Unlabelled**	**%Change**
7086	17	20	-3
4299	29	33	-4
2371	8	12	-4
5508	21	26	-5

Table 18- Cases where unlabelling made the %misassigned worse and the % differences between the control and unlabelled set.

Out of the 140 that ran, in 26 cases the assignment became worse with the addition of the unlabelling data. In 22/26 the difference was only 1-2%. This could be because forcing one residue to be something causes further mismatch down the line that the program cannot comprehend. This was unexpected and needs further investigation.

The remaining 50 proteins showed no difference in %misassigned when unlabelling was executed. Perhaps by simulating unlabelling for different residues this may see a change. For the purpose of this experiment, unlabelling was only shown with amino acids that don’t scramble. The dataset is also relatively ‘perfect’ because it was used to generate SHIFTX2

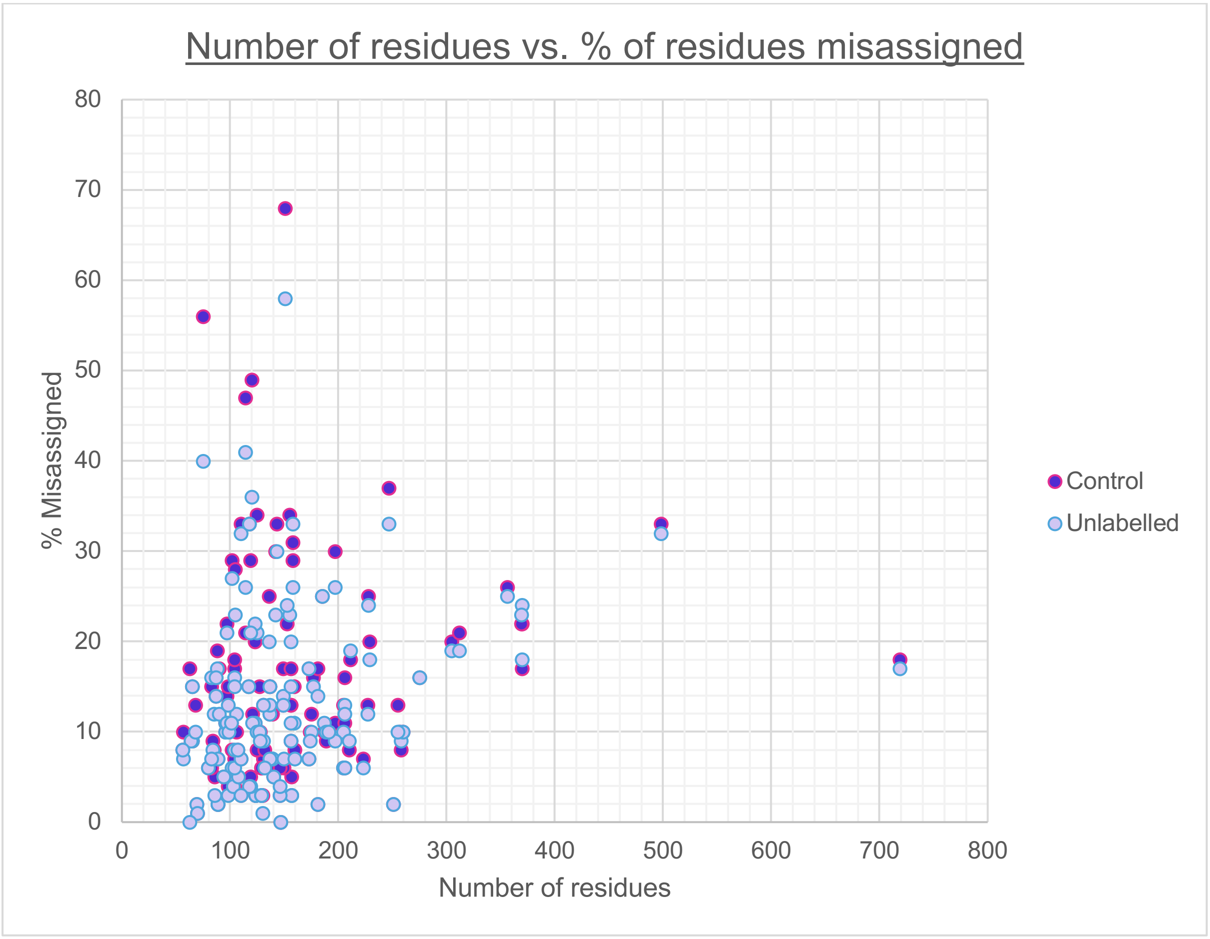


Figure 46- Number of residues vs. % Misassigned. From this graph it is clear that in a lot of cases the control has a higher % misassigned in comparison to the unlabelled.

Unlabelling would be expected to work best with data that has poor CB and CB-1 coverage most of the SHIFTX2 data has good coverage and SNAPS currently doesn’t support data with poor CB coverage (missing CB-1 shifts/ no CB shifts). These will be required for further work. Specifically, analysis of simulated data with missing CB/CB-1 shifts, support for the used correlation data in SNAPS and support for spin systems with missing CB/CB-1 shifts in SNAPS. Figure 46 shows the difference in %misassigned for both control and unlabelled samples by also comparing to the size of the protein- with the control having a higher %misassigned in some cases.

Overall, improvements seen with the least well-assigned data sets are encouraging and require future work. Unlabelled data output tables are in **Appendix 6 and 7**

3.6 How does this link to AlphaFold and the protein folding problem?

SNAPS requires a structure from which to predict chemical shifts to provide assignment, which has, until recently limited it to assignment of proteins with known structures. However, the development of AlphaFold opens up a new approach. This is because an AlphaFold structure can be used to predict a structure and chemical shifts which can then be validated and assigned with SNAPS.

The NEF pipelines tool “SHIFTX2 import shifts” was modified to allow an AlphaFold structure to be derived from a PDB code if the structure is available on the EBI AlphaFold structure server by lookup of the UniProt ID*[2] equivalent to the PDB code. (This process is illustrated in the flow diagram below).

*[1].expasy.org/resources/uniprotkb-swiss-prot

*[2] alphafold.ebi.ac.uk

It should be noted that this approach currently excludes some proteins from the SHIFTX2 as they are not present in the EBI dataset (reasons why explained later).

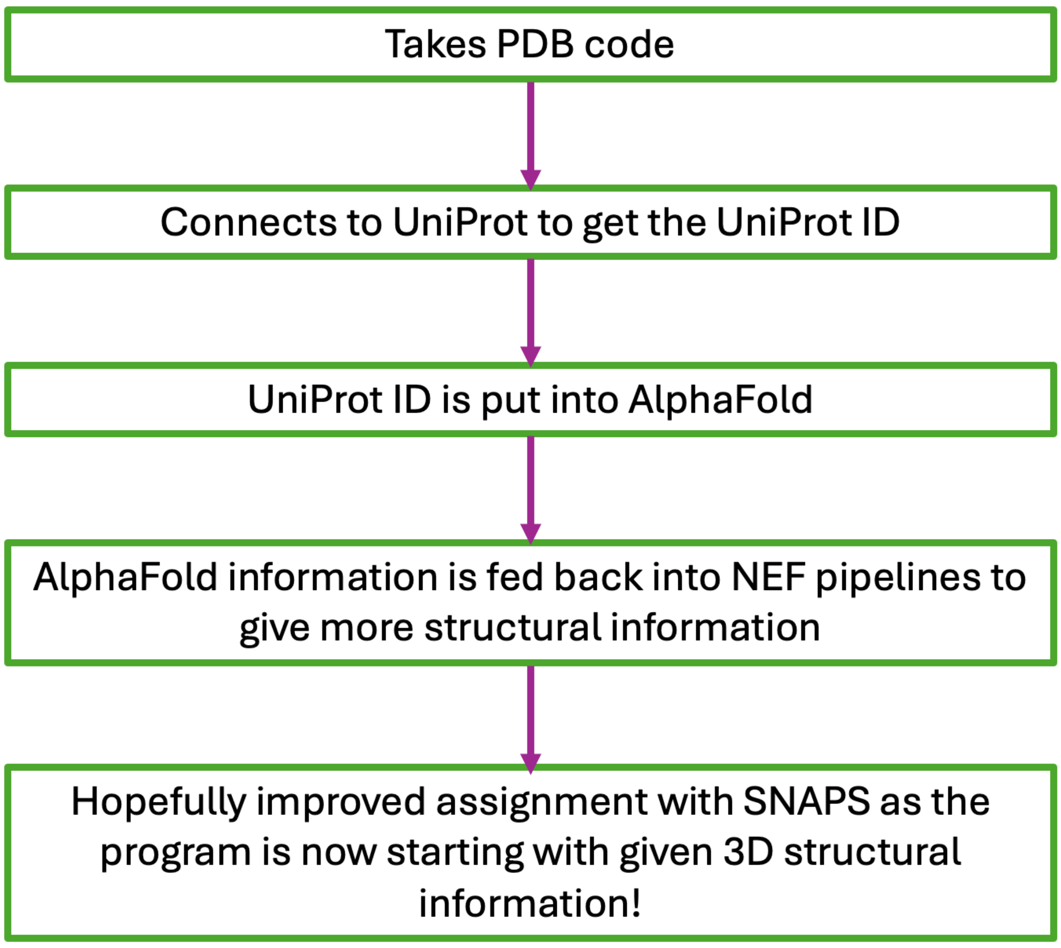

Takes PDB code
Connects to UniProt to get the UniProt ID
UniProt ID is put into AlphaFold
AlphaFold information is fed back into NEF pipelines to give more structural information
Hopefully improved assignment with SNAPS as the program is now starting with given 3D structural information!

Figure 47- A flow diagram of how NEF pipelines is compatible with AlphaFold and how this can be applied to SNAPS

Tests were also performed on this dataset- the same as the controlled and unlabelled to see the significance of applying AlphaFold information to this software.

AlphaFold is a relatively new program and so not all the UniProt sequences are readily available from AlphaFold. Sequences alone can be run but this will be something that will be implemented in the future so that any given protein or sequence can be applied to SNAPS and be assigned. Given that SNAPS can run quickly, this meant that again, lots of proteins could be analysed with the AlphaFold command.

Not all sequences are currently covered in AlphaFold for the some of the following reasons:

- It contains non-standard amino acids e.g., X
- It has been added or modified by UniProt in a more recent release
- It is a viral protein, these are currently excluded
- It is not in the “one sequence per gene” reference proteome FASTA file(95)

With the testing of the AlphaFold data, results were yielded using the same methods as previous. The only difference is that the shift is based on data is based on an AlphaFold structure fed to SHIFTSX2 rather than the PDB structure. The method for how this is used in NEF pipelines is shown in figure 47 above. In the import script the command:

```
| nef shiftx2 import shifts $pdb --source-chain $chain --chain A  --alphafold
```

Some proteins from the list were unable to be used because of the reasons mentioned above so a databank of 120 non-unlabelled and 115 unlabelled proteins were obtained from this test.

Although there are 5 less results for the unlabelled, this set was 5% better than the control dataset. This here clearly shows that the unlabelling function is correctly being implemented. Both datasets also show to be of high quality with the fact that the bulk of the dataset only has 20%-40% of the protein misassigned (shown as % misassigned).

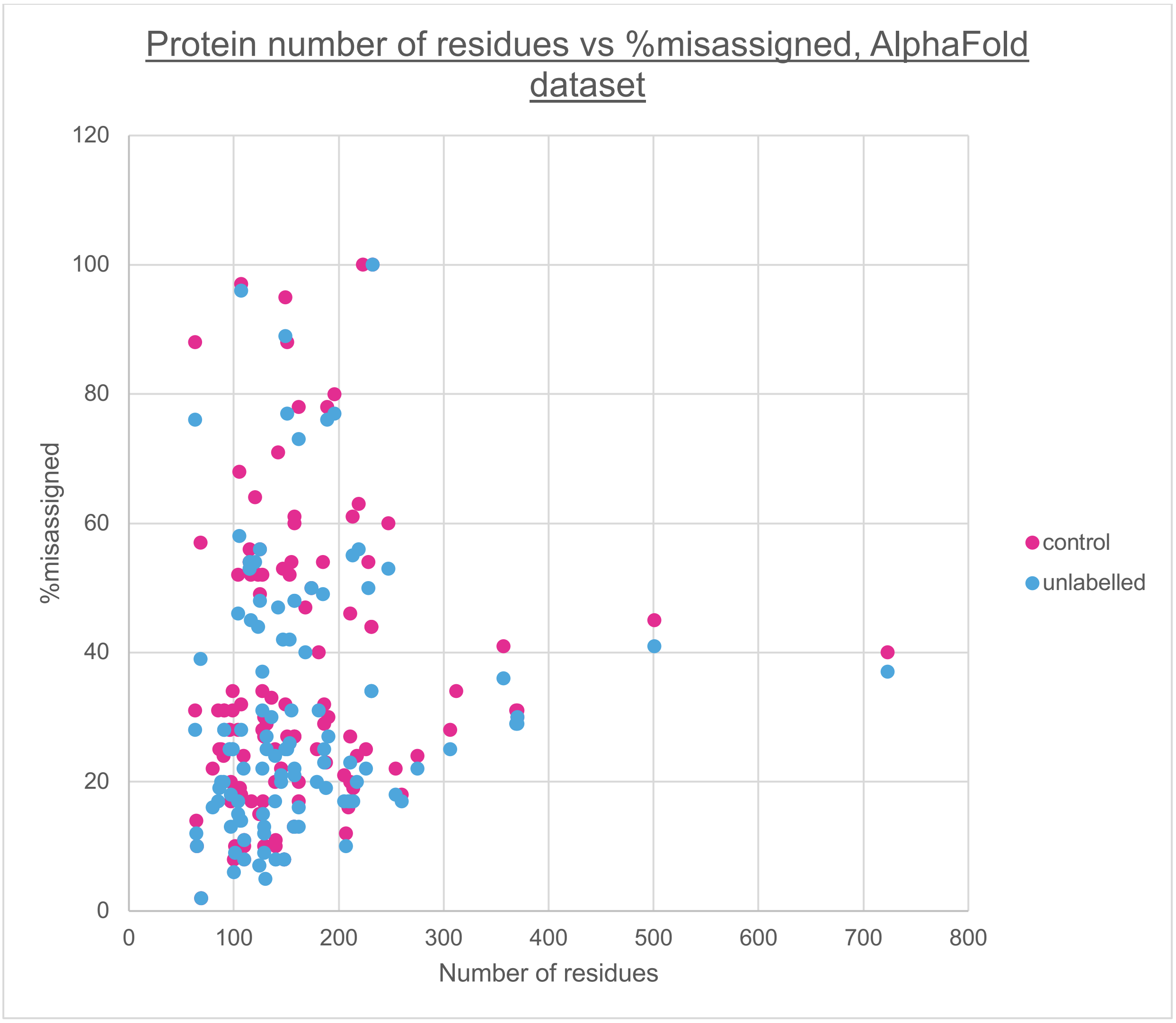


Figure 48- AlphaFold dataset of total residue number (index) vs. %misassigned. Most proteins fall somewhere in 20-40%of misassigned but very few are also longer than 300 residues. There is a weak positive correlation that suggests the larger the protein is, the worse the assignment. It is also noticeable that the control dataset often has a higher %bad than the unlabelled.

Another way to evaluate the success of AlphaFold integration is to look at the data compared to that without AlphaFold and instead use the traditional method from the PDB. From looking at this data, there were only 4 cases where AlphaFold outperformed the original test set:

Table 19- Cases where the AlphaFold dataset was better than the original PDB derived dataset.

BMRB Code	PDB	Chain	AlphaFold % misassigned	PDB % misassigned	% Difference
5485	1H7M	A	8	14	6
5485	1W41	A	10	11	1
6776	1UJ8	A	10	15	5
7086	2FCL	A	13	17	4

Although this may look okay on the surface, the AlphaFold test set didn't perform particularly well. Only 120 ran in comparison to 140 on the PDB dataset, some of this was because of some of the proteins not being compatible with AlphaFold for the reasons mentioned above. Without taking that into account, the AlphaFold data was 21% worse than the original dataset with an average %misassignment value of 35.4% compared to the control PDB samples of 14.3%. 28/120 proteins had a %misassigned greater than or equal to 50%, with some still being 100% misassigned altogether.

In addition, there were 6 cases where %misassigned was more than 70% worse with the implementation of AlphaFold:

Table 20- Cases where AlphaFold usage made the %misassigned significantly worse.

BMRB Code	PDB Code	Chain	AlphaFold % Misassigned	PDB % Misassigned	% Difference
4084	1L3K	A	80	9	71
4162	1EPF	C	78	4	74
4968	1F5R	I	100	10	90
5097	1F94	A	88	17	71
5359	1YKT	B	100	8	92
6720	3FAP	B	97	5	92

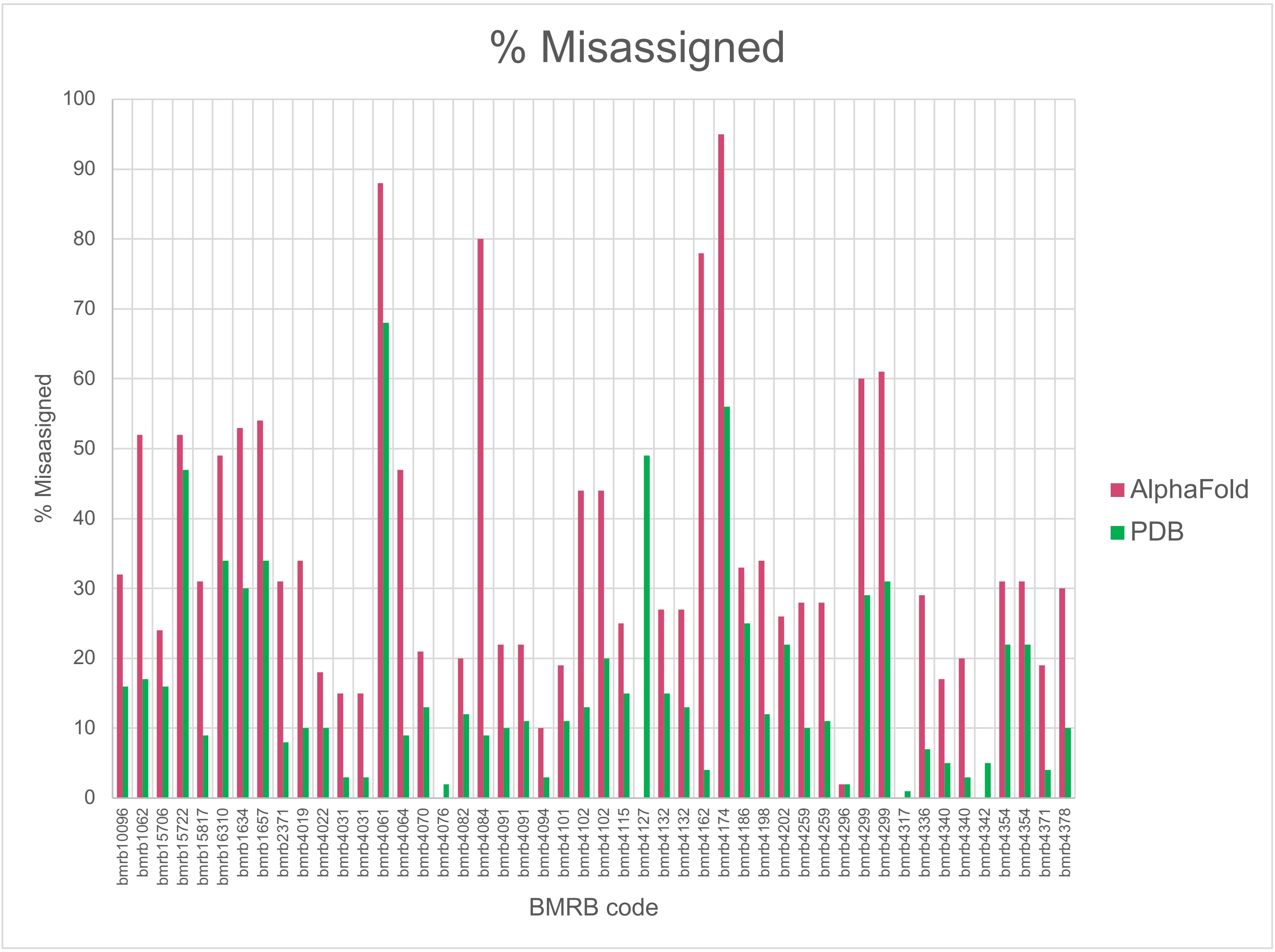


Figure 49- A comparison of % misassigned for AlphaFold data and PDB derived data (first 50 data points). Overall, the % misassigned is higher in the AlphaFold data. Full table, Appendix 8

The cause for the AlphaFold dataset being worse of isn't completely clear. Upon inspection of the output files for the worst outcomes, files either seemed to show terrible assignment, no sort of logical matching or a lack of spin system data- that is replaced as a dummy spin system.

The reasons for the AlphaFold dataset being worse than the PDB are unclear. It could potentially lie with the NEF pipelines import script or the AlphaFold command not being properly implemented in SNAPS.

3.7 *Unlabelling with the protein PctA:*

An unlabelling experiment of PctA was performed to generate an experimental dataset to test with SNAPS. The unlabelling method was used with tyrosine to later adapt into SNAPS so it is able to deal with unlabelling data. Essentially, when comparing the unlabelled to the control sample, one can easily infer which resonances correspond to residues that are unlabelled and so can be assigned. The idea is that this can be applied to SNAPS so that some of the protein is already assigned when it is inserted into the program.

3.7.1 *Production of unlabelled PctA:*

The transformation and initial plate growing process produced substantial isolated colonies that could easily be selected for initial overnight growth. This is clear to see from figure 50.

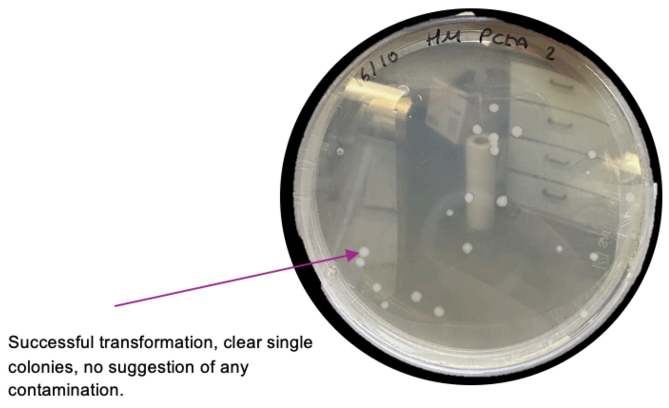

6/10 HM PctA 2
Successful transformation, clear single colonies, no suggestion of any contamination.

Figure 50- An agar gel plate with successful overnight growth after transformation process. The plate shows isolated, single colonies with no immediate signs of contamination.

From the following growth and expression performed, PctA was quantified in high yields, 0.52mM. The protein was eluted using a batch elution by the addition of a buffer containing increasing concentrations of imidazole (see method 2.6.3.1) . From the elution profile figure 51, it is clear that most of the bound protein was found in the 60%B aliquot for both the tyrosine and the control samples. The elution profile shows a higher quantity of protein obtained from the tyrosine sample as the 60%B column has the highest A280 value. The

60B fraction for the control sample is also high but there is still evidence of protein present in the 30B and 100B fractions.

This is further supported by the SDS-PAGE gel was stained with Coomassie blue, figure 52 where it is clear that for the control sample the last 3 fractions contain some protein- but most of the protein is in the 60%B fraction and the molecular weight matches that of PctA (29kDa). It is also clear that for both samples the protein purity is high- there is little suggestion of any contamination as the gel shows one clear, distinguished band. The band in the 100%B well of the tyrosine sample was likely because the well was overfilled on the 60%B column and spilled over a bit into the 2nd lane and isn't representative of protein being present.

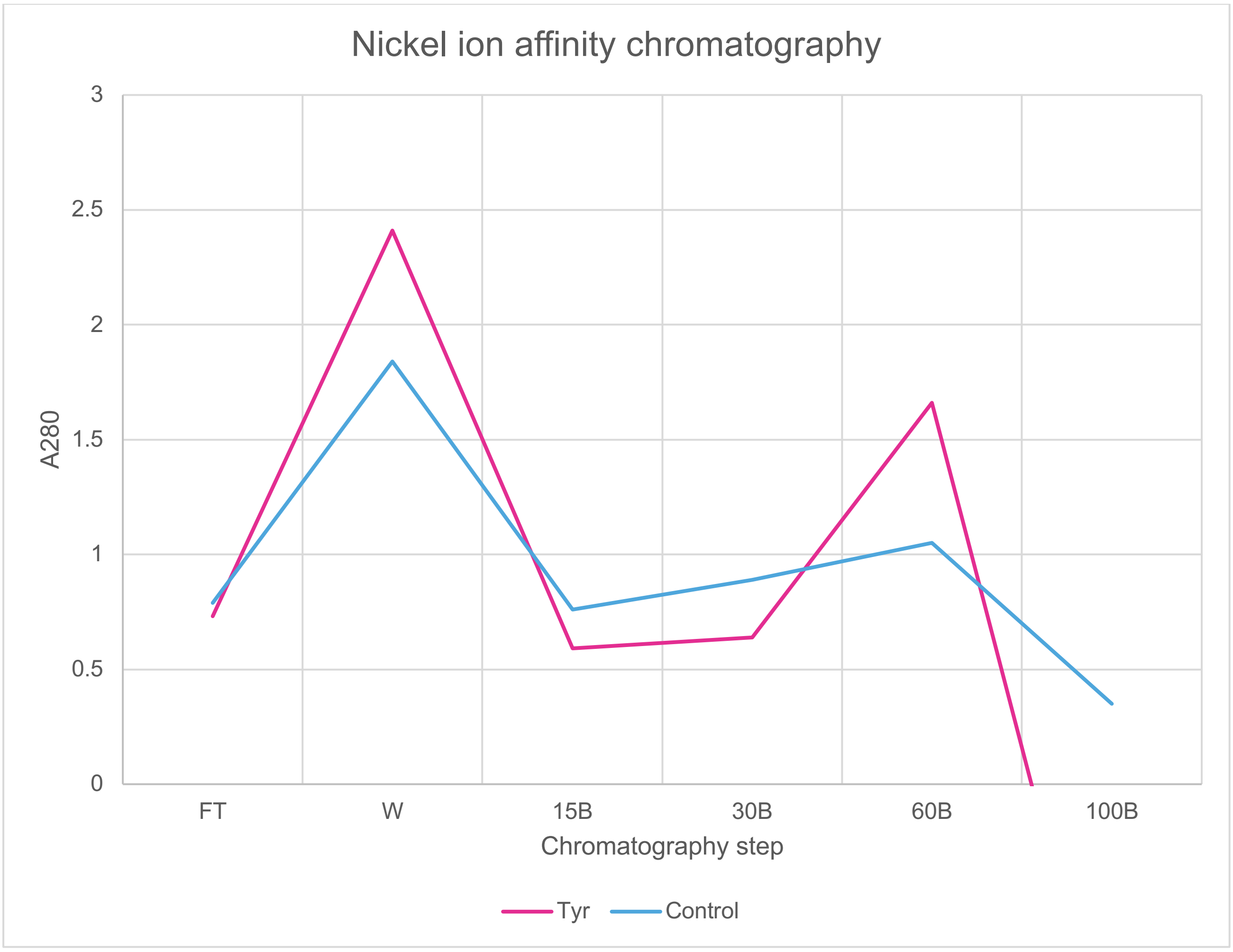


Figure 51- The nickel-ion affinity chromatography of PctA for the control sample and the tyrosine unlabelled sample. The X-axis displays the different nickel-ion affinity chromatography fractions. (FT= Flow through, W= wash, 15-100B as % buffer B in sample)

Even though the gel didn't show any impurities being present in the sample, the tyrosine 60B fraction was further purified by size exclusion chromatography to obtain a pure and homogenous sample of PctA.

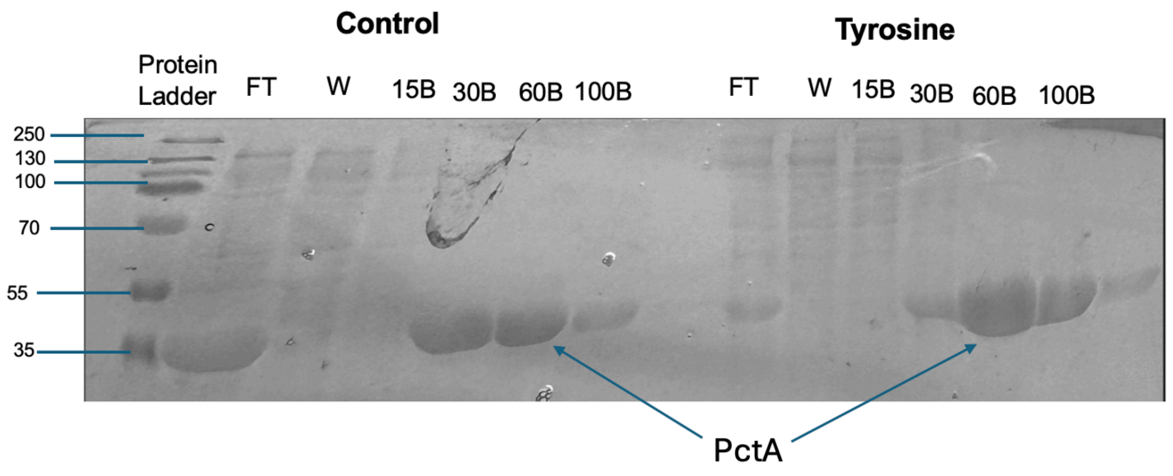


Figure 52- An SDS-PAGE gel of the nickel-ion affinity chromatography- clear bands indicate no contaminants. PctA has a molecular weight of 29kDa

Both samples were concentrated to give a 2-5ml sample down from the initial 10ml. This concentration step is very important as a high protein concentration is needed for NMR analysis due to its low sensitivity.

Once and appropriate volume was reached both fractions were dialysed for NMR dialysis buffer in methods 2.6.3.5 . The final protein concentrations were: 0.52mM of tyrosine unlabelled protein and 0.34mMof the control sample. The measuring of this was taken from the nanodrop using the theoretical extinction coefficient derived from the protein sequence.

3.7.2 PctA NMR data:

2D ^{1}H- ^{15}N/ Transverse Relaxation Optimised Spectra (TROSY) spectra were acquired on a Bruker Advance III spectrometer at 600 MHz at 298K using a QCI-P cryoprobe. Data was acquired with 2048 points (18.04ppm) in the direct F2 dimension (^{1}H) and 256 points (35.00ppm) in F1 (^{15}N) indirect dimension. There was a total of 144 scans per increment (*api* state) and 128 (control) 120 (unlabelled) scans per increment (*holo* state), the relaxation delay time (D1) in the experiments was 1.4 seconds and 32 dummy scans were used for both experiments. Data was processing using Bruker Topspin 3.2 with a gaussian window (GM) function in both dimensions with LB=-25 and GB= 0.2. Zero filling was applied in both direction and linear prediction in the indirect dimension.

The PctA data was analysed using CCPN assign. Both the control and Tyrosine unlabelled samples gave very clear ^{1}H-^{1}HN TROSY spectra. These spectra can be seen in the figures below.

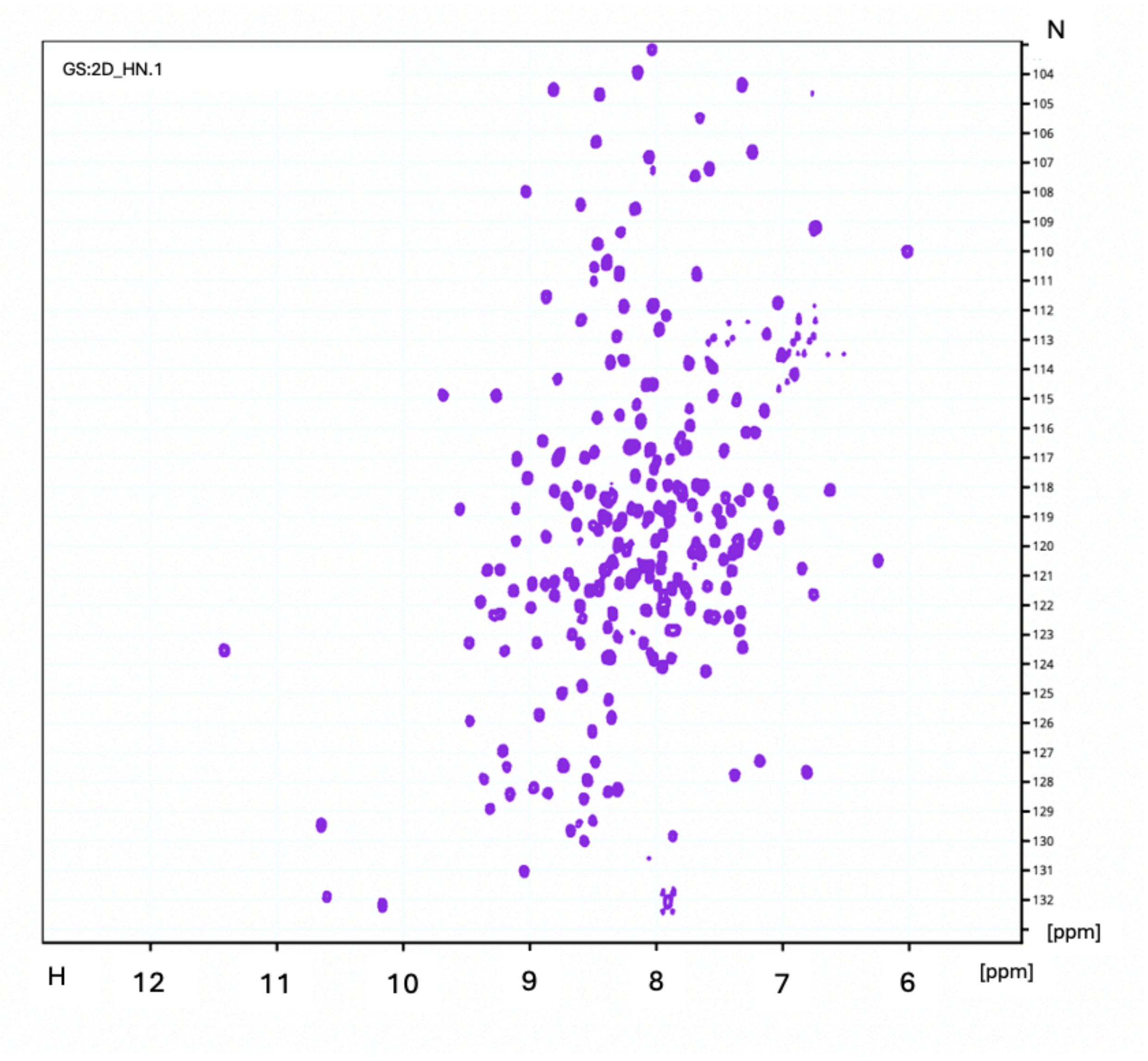


Figure 53- A ^{1}H ^{15}N TROSY spectrum of the control PctA sample. Anything below 5ppm can be regarded as noise.

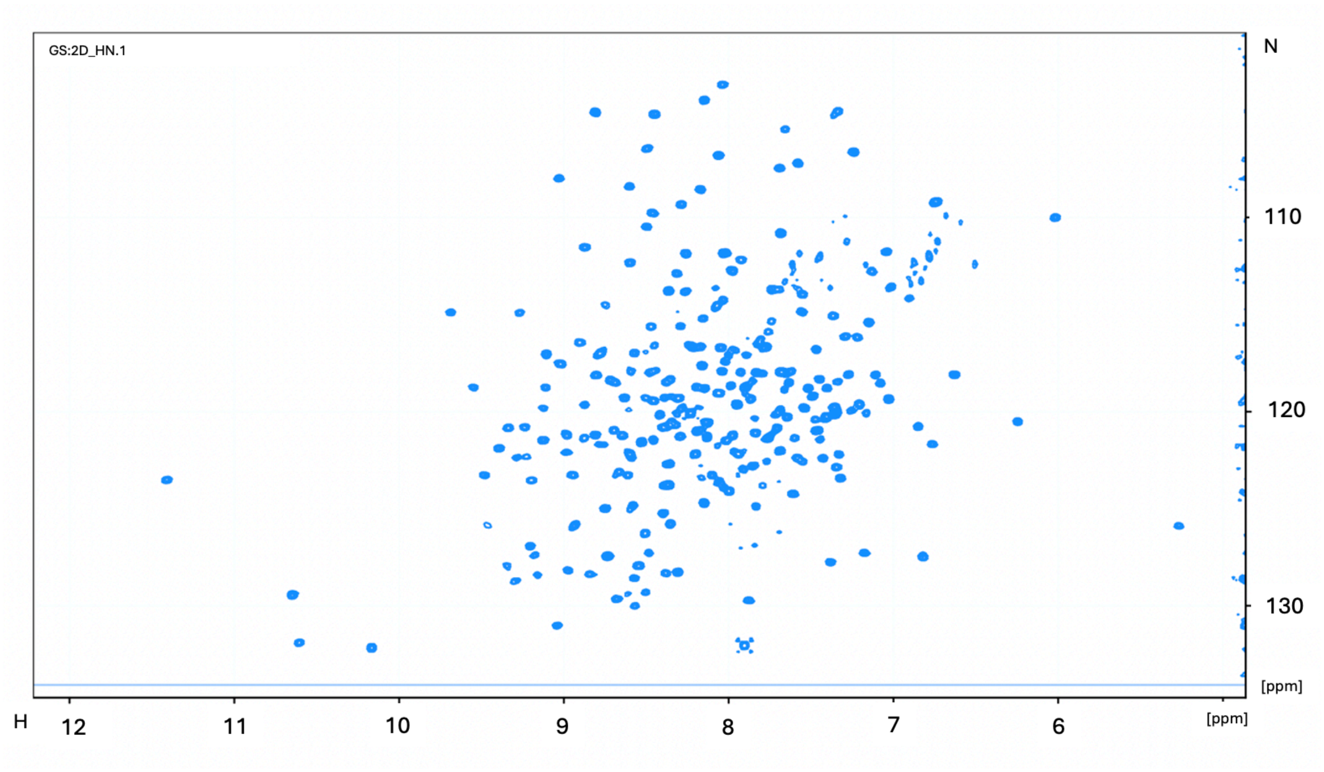


Figure 54- ^{1}H ^{15}N TROSY spectrum for PctA Tyrosine unlabelled sample.

There is no amino acid unlabelling in figure 53. A good quality spectrum can be identified with strong isolated peaks and little evidence of contamination/ degradation.

Unlabelling was a success for Tyrosine and can be seen in figure 54. Although it may not be easy to tell from the unlabelling alone, the tyrosine unlabelling experiment can easily be seen as a success when the spectra of the control and unlabelled are overlayed. From figure 55 there is no overlap of spectra on spin system @140 (highlighted in green). The absence of a blue peak here means that this residue cannot be seen in the unlabelled spectra, and so, is clearly unlabelled.

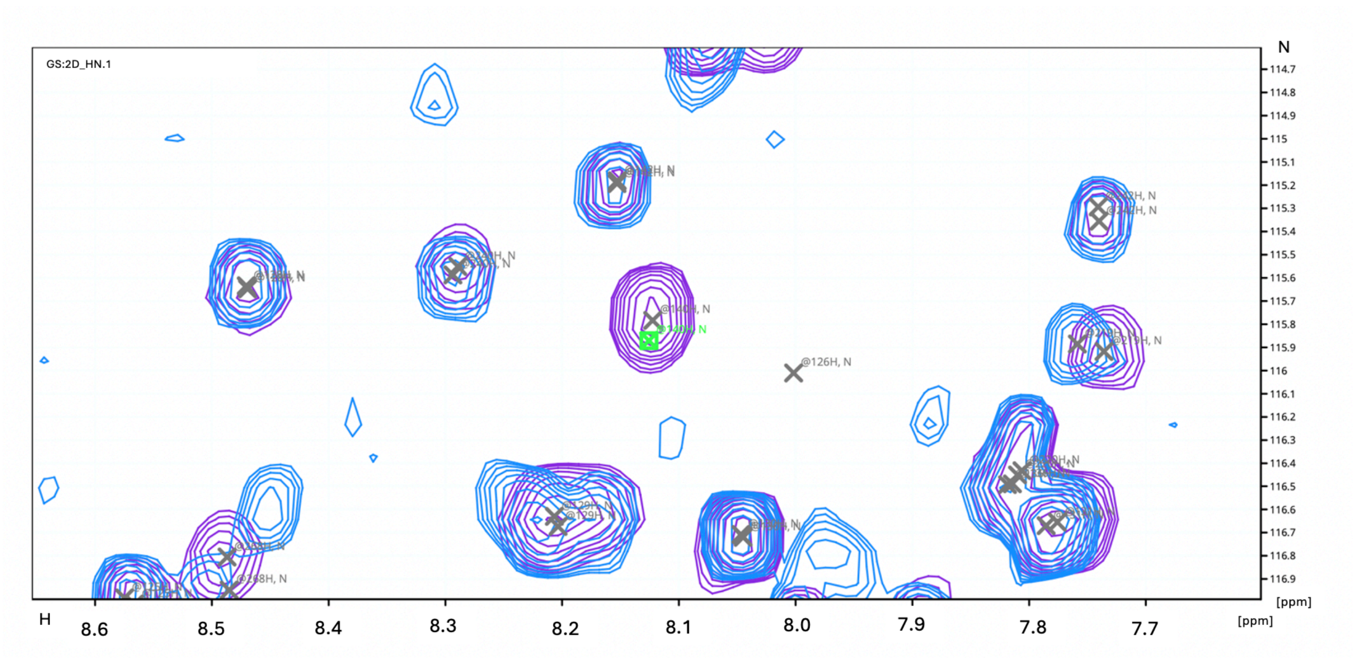


Figure 55- An example of unlabelling success in the Tyrosine spectra. The control (purple) peak is present for Spin system 140, but the blue has disappeared.

The next step was to assess how well SNAPS could work to assign the protein. The sequence for the expressed ligand binding domain is as follows with 10 Tyrosine residues which are highlighted in bold Pink:

GSSHHHHHHSSGLVPRGSHMND**Y**LQRNAIREDLES**Y**LREMGDVTSSNIQNWLGGRLLLVEQTAQTLAR DHSPETVSALLEQPALTSTFSFT**Y**LGQQDGVFTMRPDSPMPAG**Y**DPRSRPW**Y**KDAVAAGGLTLTEP**Y**V DAATQELIITAATPVKAAGNTLGVVGGDLSLKTLVQIINSLDFSGMG**Y**AFLVSGDGKILVHPDKEQVM KTLSEV**Y**PQNTPKIATGFSEAELHGHTRILAFTPIKGLPSVTW**Y**LALSIDKDKA**Y**AMLSKFRVSA

The determination of which spin systems are unlabelled and which are not can be unclear and so more difficult in certain areas of the spectrum. The spectra are sometimes showing shifts possibly/ probably because of protease degradation- even though co-dialysis was used, peaks from these still don't perfectly overlap. Because the two samples were made from two separate batches, the same spin systems don't necessarily show up with identical peak positions on the spectra. As a result, the protein can shift slightly on the spectrum. This can be seen in the figure 56 below which shows most of peaks clearly overlapping as well as some areas which show that some of the peaks have. At a first glance, it may look like spin system 121 (circled in red) is unlabelled because of the absence of the blue peak on top of it. In fact, the peak has just shifted to the right. The status of peak 90 (circled in black) is even harder to determine because of the movement of peaks around it.

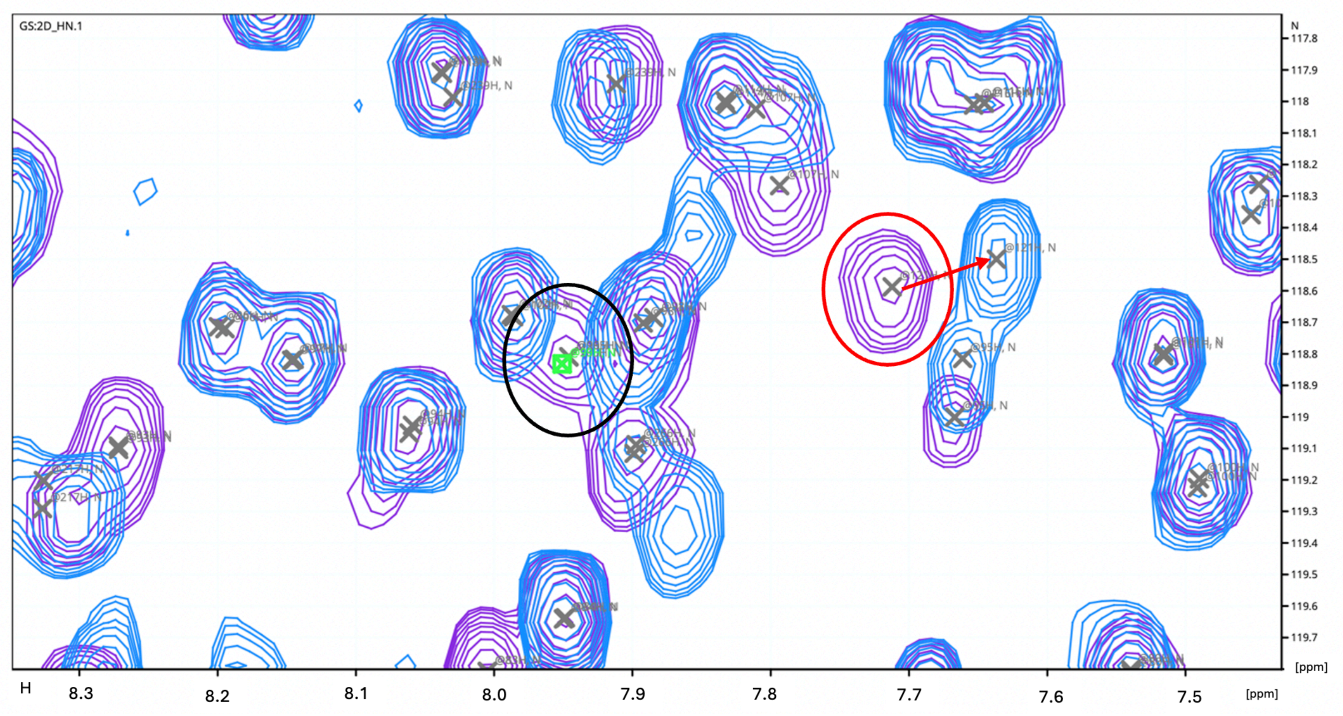


Figure 56- A section of overlayed HSQC spectra for tyrosine unlabelled and control samples. Here the peaks don't align perfectly and show shifting between the two sets of data. This can make it hard to determine whether or not unlabelling has taken place here.

To analyse spectra best efforts were made to assign spin systems to peaks matching previously measured triple resonance data. Then the unlabelled is labelled peak ratios were displayed and analysed using 2 graphs. The first showed peak ratios of unlabelled/ control in spin system order. The second shows the same date ordered from the high ratios to the low ratios in order. The form of the graph should the approximately a step function with an upper plateau of 1 and a lower of plateau at 0 in the presence of noise it will be distorted to give a raised area of high ratios and a roll off at low values. This can be displayed in the figure 57:

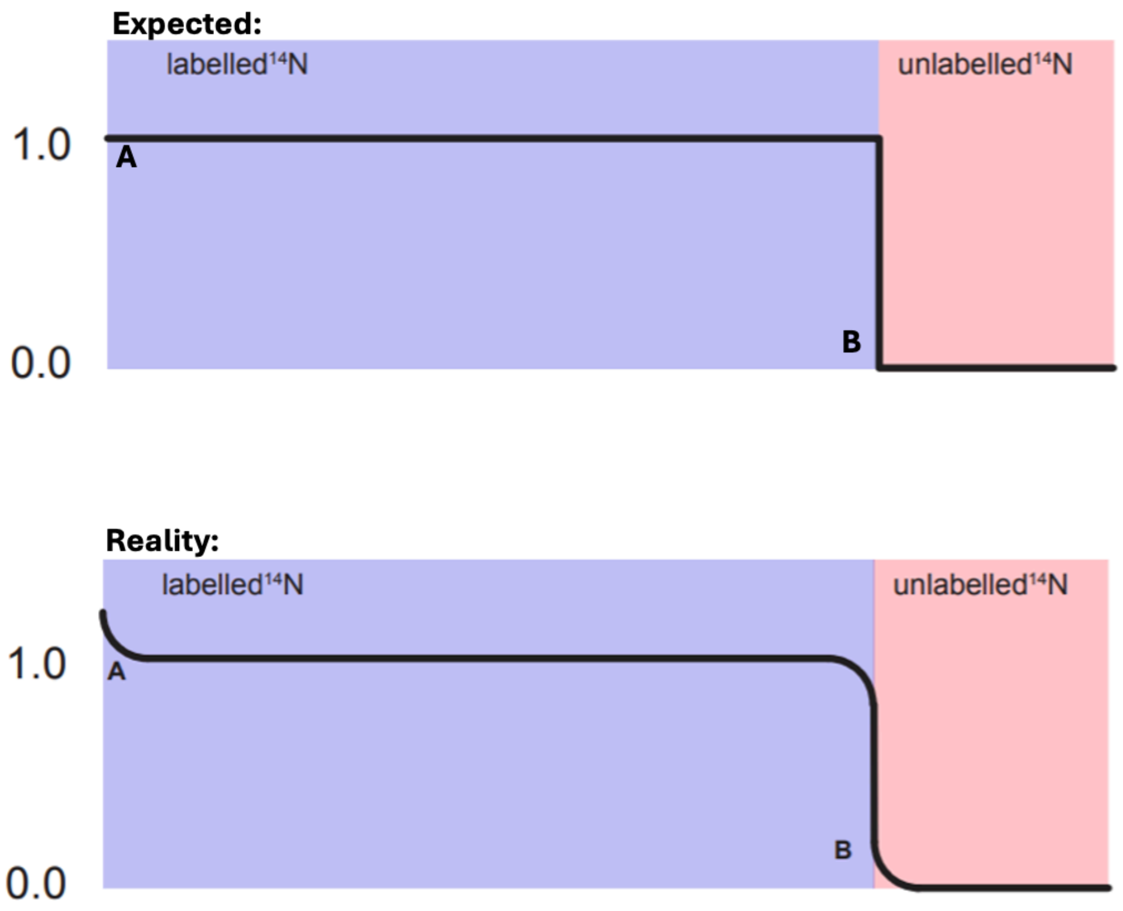


Figure 57- A graphic of how errors appear in relative peak ratios between a control and ^{14}N amino acid selective unlabelled spectrum. A. Expected relative sorted peak intensities in an ideal system. B. Reality, due to noise, there isn't a prefect step function (as shown in the expected), but a sloped with curves at each end. The labelled positions (A and B) should have a value of 1 labelled and 0 unlabelled (hence the formation of a step function). The curvature at A and B is caused by errors in the peak intensities due to noise on weak peaks with random noise in either the divisor or the exponent causing the ratio to deviate from 1.

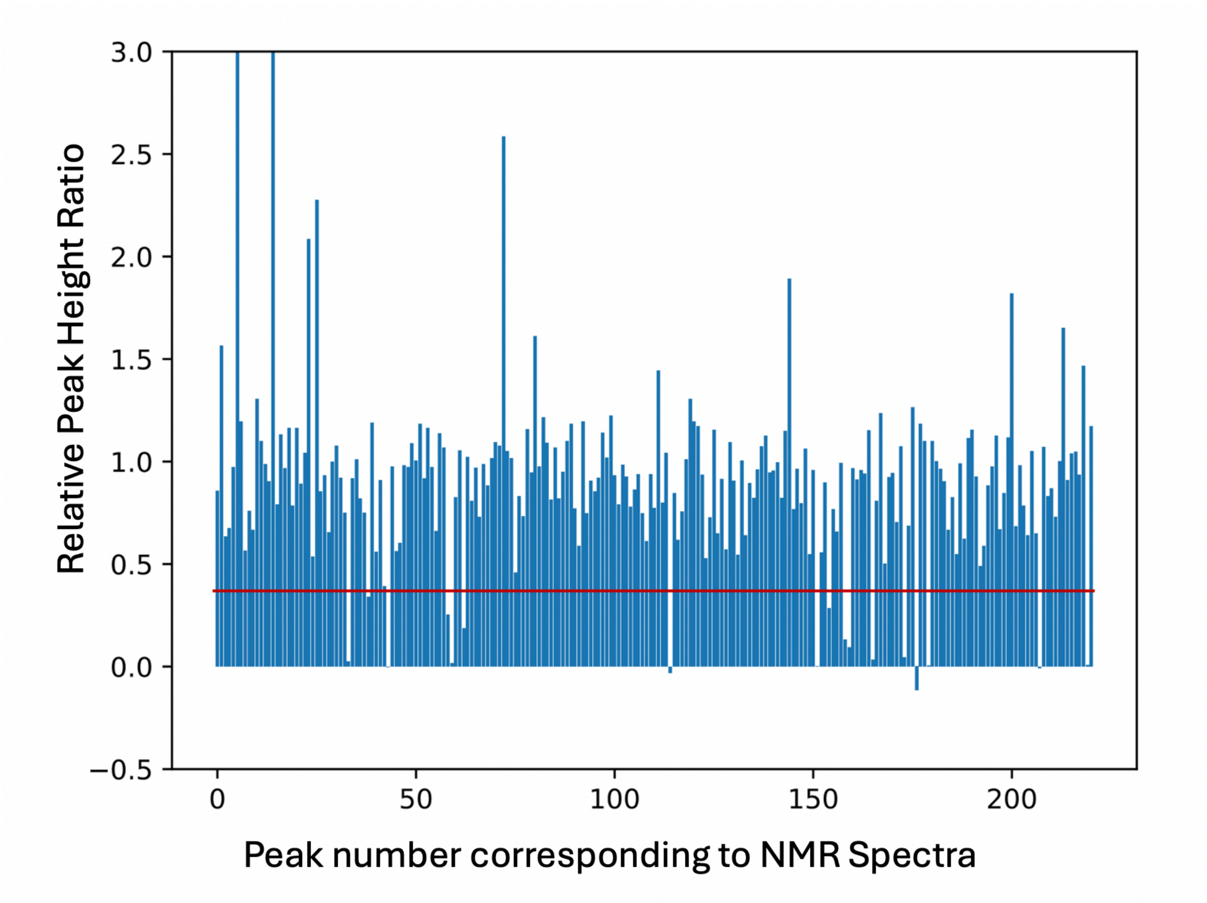


Figure 58- Tyrosine unlabelling shown by peak height ratio vs. peak number on NMR spectrum. Areas below the red line that show no peak/ very small peak indicate that tyrosine has been unlabelled.

From this analysis of this data. Given the sequence only had 10 Tyrosine residues, it appears that peak numbers 66, 68, 69, 83, 93, 99, 140, 189, 191, 214, 263, 264 and 286 appear to be the most likely unlabelled spin systems. It was hard to tell whether 46 or 51 were unlabelled also. The relative peak heights are displayed in order of residue number (index)in figure 58 and in descending order in figure 59, the unlabelled residues have little-to-no peak present.

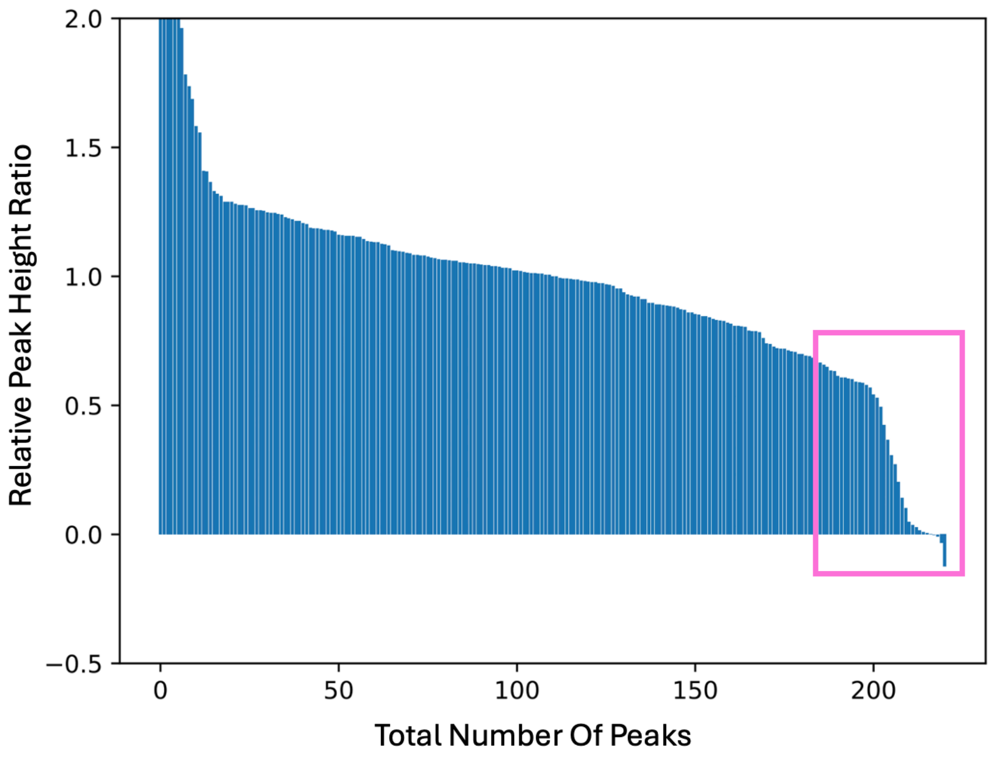


Figure 59- Relative peak height ratios for Tyrosine unlabelled spectrum compared to the control in descending order. The area in the pink box indicates the gradual decrease in peak height ratios.

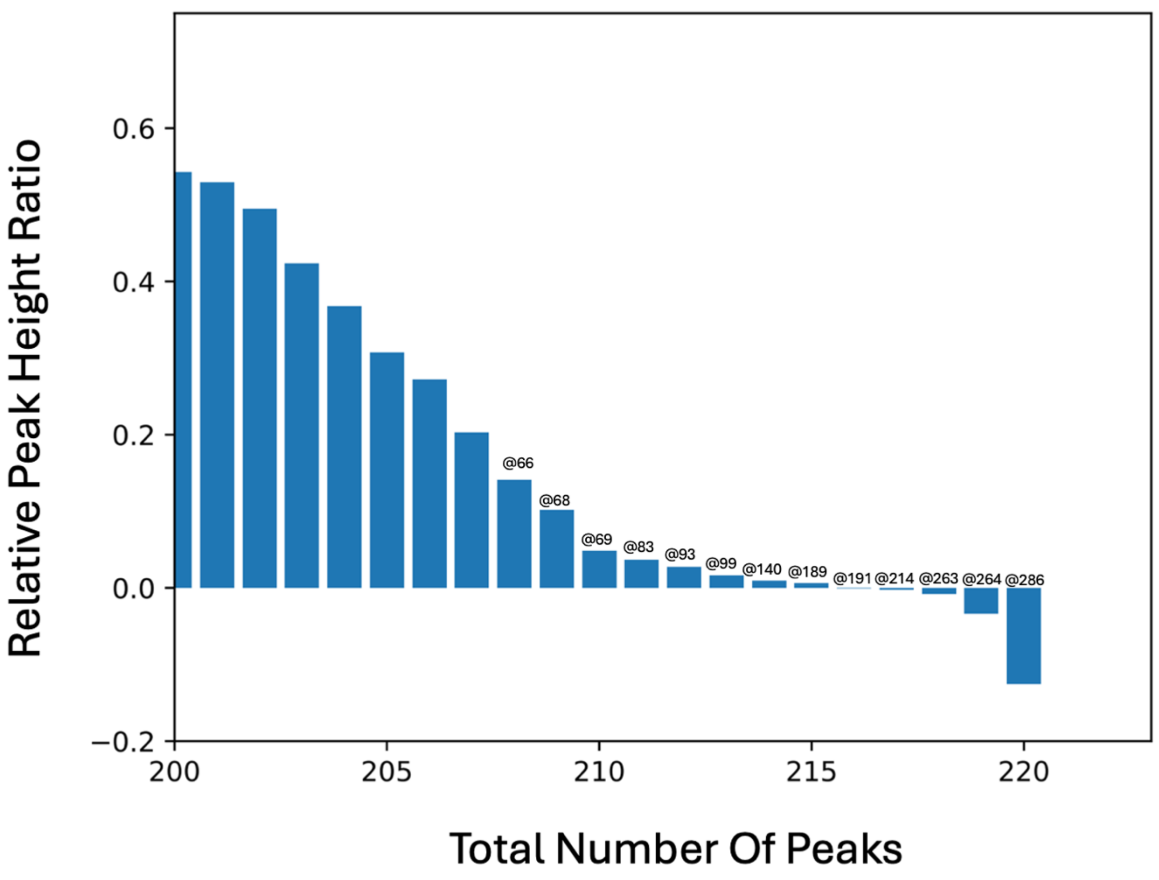


Figure 60- Tyrosine unlabelled PctA relative peak height ratios in descending order. The pink box from figure 59 zoomed in to show the decrease in peak height ratios for the spin systems.

From this graph, it appears that there are definitely 10 unlabelled residues- the same number of unlabelled residues expected. From the previous graph (Figure 59) it is less clear on how many are exactly unlabelled due to varying intensities. It can be said that Tyrosine was definitely unlabelled as well as some scrambling with phenylalanine.

Triple resonance data was processed from this experiment and saved in the NEF file format and predicted shifts were simulated with SHIFTX2. This was able to run in SNAPS and so further demonstrates the success of the NEF importer. The file ran with no issues. However, due to issues unknown, PctA cannot be assigned very well:

Control: 150 Unreliable, 44 Undefined, 18 Low, 10 Medium, 4 High

Unlabelling: (slightly better) 147 Unreliable, 46 Undefined, 19 Low, 11 Medium, 3 High

The reasons for this being poor isn't immediately clear, the first thought was that it could potentially be due to errors in peak referencing. From data of the previously assigned P3a_L273R (by Dr Alex Heyam) the shift referencing with the same process is perfect (from shift cloud data). This is also the same compared to PctA. This leaves crystal structure, and PctA displays similar structure to other cache proteins. This leaves the degree of missing data as a potential cause. The PctA data used in SNAPS had approximately 30% missing.

4 CONCLUSIONS AND FUTURE WORK:

Overall, it can be deduced that the Hungarian algorithm works well to tackle the assignment problem. From the SNAPS results can correctly assign (on average) between 86-88%- depending on whether or not unlabelling command is implemented. The unlabelling command did generate a lower % misassigned score (as shown in figure 46) when tested with known PDB structural data.

However, the assignment process was less effective for AlphaFold Figures 48 and 49. The reason for this remains unclear but may be due to issues with AlphaFold being relatively new. Alternatively, the internal code for the application of AlphaFold in NEF pipelines and SNAPS may not be calling AlphaFold properly, as a result of this, information from AlphaFold may not be being utilised correctly.

Furthermore, the data derived from the test set supplied on the SHIFTX2 website is relatively well assigned and is more representative of a perfect dataset as it is published. This doesn't assess how well SNAPS and the Hungarian algorithm would work with a less well-polished input of observed chemical shift data.

The NEF importer allows for seamless usage of NEF files for input as demonstrated by the SHIFTX2/ BMRB test set as well as the usage of the PctA triple resonance data.

The Unit tests written allowed for a fool-proof input dataset. For future use, the user will be told exactly how to provide the input data and any problems that there may be with what is supplied to SNAPS- incorrect amino acid name etc.

As for the lab unlabelling experiment, from the NMR data supplied, it is clear that the PctA produced was tyrosine unlabelled with evidence of some phenylalanine scrambling. The files did successfully run in SNAPS (observed triple resonance and predicted from SHIFTX2). The success of unlabelleing couldn't properly be evaluated due to correlation data not being taken into account (correlation data should be implemented for future work) but shows great promise for further development of this project.

Future work for the project could include looking at fixes for the AlphaFold command and testing real observed NMR data on the program rather than what is provided from the BMRB. In addition to this, correlation data could also be implemented to give the program more information for when assigning chemical shifts to the sequence. By using correlation data this would allow SNAPS to have more information about the positioning of the chemical shift data in reference to other chemical shift values on the spectrum and then use this information to aid assignment. In turn, it should facilitate the assignment process- to put simply, it gives assignment relative to the chemical shift values it's near to in space. This could also give a better indication of to what degree is the protein is assigned (if the assignment up until this point is unknown), as a high correlation would imply that the protein has been more successfully assigned. In addition, the program should also be made compatible with ARTINA for 3D structures information as an input option.

4.1.1 Bug fixes since completion:

After the submission of this thesis, work continued on investigating the underlying cause for the AlphaFold dataset being worse than the PDB dataset by master's student, Cian Bartholomew. There were several issues as follows:

- Internal errors within SNAPS and statistics calculations
- Trimming errors within NEF pipelines causing misalignment of residues being run through SNAPS

Running of the latest developed code provides an average of 20.6% misassigned compared to previously being 35.4% misassigned (14.8% improvement!). It can be expected that the AlphaFold dataset is slightly worse off than the PDB one because AlphaFold generates predictions.

5 APPENDIX:

1. NEF_reader.py script:

```
import logging
from pathlib import Path
from typing import List
import sys
import pynmrstar
import pandas as pd

_CHEMICAL_SHIFT_LIST_FRAME = 'nef_chemical_shift_list'  # tag for the
NEF chemical shift list frame
_CHEMICAL_SHIFT_LOOP = 'nef_chemical_shift'  # tag for the NEF chemical
shift list loop
_CHAIN_CODE = 'chain_code'  # NEF loop heading for chain code
_SEQUENCE_CODE = 'sequence_code'  # NEF loop heading for sequence
_ATOM_NAME = 'atom_name'  # NEF loop heading for atom name
_RESIDUE_NAME = 'residue_name'  # NEF loop heading for residue name
_SHIFT_VALUE = 'value'  # NEF loop heading for  the chemical shift list
value

_OFFSET_SLICE = slice(-2, None)  # equivalent to value[-2:]
_RESIDUE_CODE_SLICE = slice(0, -2)  # equivalent to value[:-2]
_PREVIOUS_RESIDUE_FLAG = '_m'  # the flag SNAPS uses to indicate the
correlated shift from the previous residue
_PREVIOUS_RESIDUE_OFFSET = '-1'  # the offset to the previous residue as
used by CCPN and NEF

_DEFAULT_SHIFT_LIST = 'default'

TRANSLATIONS_3_1_PROTEIN = {
    "ALA": "A",
    "ARG": "R",
    "ASN": "N",
    "ASP": "D",
    "CYS": "C",
    "GLU": "E",
    "GLN": "Q",
    "GLY": "G",
    "HIS": "H",
    "ILE": "I",
    "LEU": "L",
    "LYS": "K",
    "MET": "M",
    "PHE": "F",
    "PRO": "P",
    "SER": "S",
    "THR": "T",
    "TRP": "W",
    "TYR": "Y",
    "VAL": "V",
}
TRANSLATIONS_1_3_PROTEIN = {
    value: key for (key, value) in TRANSLATIONS_3_1_PROTEIN.items()
}

# ignore pynmrstar warning about loops with no data
logging.getLogger('pynmrstar').addFilter(lambda record: "Loop with no
data" not in record.msg)
```

```
def read_nef_shifts_from_file(file_name: Path, shift_list_name: str =
_DEFAULT_SHIFT_LIST) -> List[List[str]]:
    """
    read nef shifts from a file given by filename and return a table of
shifts with headings sequence_code, atom_name
    and shifts as a list of lists. note shifts from the previous residue
are indicated by appending '_m' to the atom
    name as used by SNAPS

    :param file_name: file name
    :param shift_list_name: name of the shift list frame defaults to
"default"
    :return: list of lists with each sub list being a row with headings
sequence code, atom name and shift
    """

    print(file_name)
    with open(file_name, 'r') as file_handle:
        result_shifts = read_nef_shifts(file_handle, shift_list_name)

    print(results_shifts)
    return result_shifts


def read_nef_shifts(file_handle, shift_list_name=_DEFAULT_SHIFT_LIST,
chain='A'):
    """

    :param file_handle:
    :param shift_list_name:
    :return:
    """
    file_name = file_handle.name


    entry = pynmrstar.Entry.from_string(file_handle.read())

    first_shift_frame = _read_named_shift_frame_or_error(entry,
shift_list_name, file_name)

    chem_shift_loop = _frame_to_shift_loop_or_error(file_handle.name,
first_shift_frame)

    #TODO should we be dealing with chains here
    chain_index = _read_heading_index_or_error(first_shift_frame,
_CHAIN_CODE, chem_shift_loop, file_name)
    sequence_index = _read_heading_index_or_error(first_shift_frame,
_SEQUENCE_CODE, chem_shift_loop, file_name)
    atom_index = _read_heading_index_or_error(first_shift_frame,
_ATOM_NAME, chem_shift_loop, file_name)
    chem_shift_index = _read_heading_index_or_error(first_shift_frame,
_SHIFT_VALUE, chem_shift_loop, file_name)
    residue_name_index = _read_heading_index_or_error(first_shift_frame,
_RESIDUE_NAME, chem_shift_loop, file_name)

    output = []
    for row_index, row in enumerate(chem_shift_loop.data):

        chemical_shift =
_read_shift_from_row_or_error(row[chem_shift_index], row_index,
shift_list_name, row,
                                                         file_name)
        sequence_code = row[sequence_index]
        atom_name = row[atom_index]
        residue_name = row[residue_name_index]

        if sequence_code[_OFFSET_SLICE] == _PREVIOUS_RESIDUE_OFFSET:
            sequence_code = sequence_code[:-2]
            atom_name = atom_name + _PREVIOUS_RESIDUE_FLAG

        output_row = [sequence_code, atom_name, chemical_shift,
residue_name]

        output.append(output_row)

    return output


def read_nef_obs_shifts_from_file_to_pandas(raw_file_name, chain):

    output = _raw_read_shifts_to_pandas(chain, raw_file_name)

    output = output.rename(columns={'sequence_code': 'SS_name',
'atom_name': 'Atom_type', 'value': 'Shift'})

    output['SS_name'] = output['SS_name'].astype(str) +
output['residue_name']
    #TODO: why is this title case
    output['SS_name'] = output['SS_name'].str.title()
    del output['residue_name']

    return output

def read_nef_pred_shifts_from_file_to_pandas(raw_file_name, chain):

    output = _raw_read_shifts_to_pandas(chain, raw_file_name)

    output = output.rename(columns={'sequence_code': 'Res_N',
'atom_name': 'Atom_type', 'value': 'Shift', 'residue_name': 'Res_type'})

    output = output.replace({'Res_type':TRANSLATIONS_3_1_PROTEIN})

    return output



def _raw_read_shifts_to_pandas(chain, raw_file_name):
    print(raw_file_name)
    if not Path(raw_file_name).exists():
        file_name, shift_list_name =
_split_path_and_frame(raw_file_name)
    else:
        file_name = raw_file_name
        shift_list_name = _DEFAULT_SHIFT_LIST

    with open(file_name, 'r') as file_handle:
        output = read_nef_shifts_to_pandas(file_handle, shift_list_name,
chain)

    return output
```

```
def _split_path_and_frame(file_name):
    if ':' in file_name:
        path_fields = file_name.split(':')
        shift_list_name = path_fields[-1].strip()
        if not shift_list_name:
            shift_list_name = _DEFAULT_SHIFT_LIST

        file_name = ':'.join(path_fields[:-1]).strip()
    else:
        shift_list_name = _DEFAULT_SHIFT_LIST
    return file_name, shift_list_name


def read_nef_shifts_to_pandas(file_handle,
shift_list_name=_DEFAULT_SHIFT_LIST, chain="A"):
    snaps_shifts = read_nef_shifts(file_handle, shift_list_name, chain)

    pandas_columns = [_SEQUENCE_CODE, _ATOM_NAME, _SHIFT_VALUE,
_RESIDUE_NAME]
    return pd.DataFrame(snaps_shifts, columns=pandas_columns)


def _read_shift_from_row_or_error(shift_string, row_index, row,
loop_name, file_name):
    try:
        chemical_shift = float(shift_string)
    except ValueError:
        row = ' '.join(row)
        msg = f"Error: can't convert '{shift_string}' to float in row
number {row_index} [{row}] " + \
              f"of loop {loop_name} in {file_name}"
        raise Exception(msg)

    return chemical_shift


def _frame_to_shift_loop_or_error(file_name, first_shift_frame):
    try:
        chem_shift_loop =
first_shift_frame.get_loop(_CHEMICAL_SHIFT_LOOP)
    except KeyError:
        raise Exception(f'ERROR: there are no chemical shift list loops
in {file_name}')
    return chem_shift_loop


def _read_named_shift_frame_or_error(entry, frame_name_selector,
file_name):
    """

    :param entry: pynmrstar.Entry.from_file(file_handle)
    :param frame_name_selector:first_shift_frame
    :param file_name:file name
    :return error if there is no saveframe that reads 'chemical shift
list' in the file. Otherwise, return a list of the
    chemical shifts from the file
    """
    chem_shift_saveframes =
entry.get_saveframes_by_category(_CHEMICAL_SHIFT_LIST_FRAME)
    if len(chem_shift_saveframes) < 0:
        raise Exception(f'ERROR: there are no chemical shift list frames
in {file_name}')

    result_shifts = None
    for frame in chem_shift_saveframes:
        frame_name = frame.name[len(frame.category):].lstrip('_')
        if frame_name == frame_name_selector:
            result_shifts = frame
            break

    if result_shifts is None:
        raise Exception(f'ERROR: there are no chemical shift list frame
called {frame_name_selector} in {file_name}')

    return result_shifts


def _read_heading_index_or_error(save_frame, heading, loop, file_name):
    """

    :param save_frame: _CHEMICAL_SHIFT_LIST_FRAME
    :param heading: _SEQUENCE CODE
    :param loop: _CHEMICAL_SHIFT_LOOP
    :param file_name: file name
    :return: error if there is no sequence index/ heading  for {sequence
code} in file name. Otherwise, return sequence
    index for headings in the saveframe
    """
    sequence_index = loop.tag_index(heading)
    if sequence_index is None:
        msg = f"ERROR: couldn't find a sequence heading {_SEQUENCE_CODE}
in save frame {save_frame.name} in {file_name}"
        raise Exception(msg)
    return sequence_index


ERROR = 1

if __name__ == '__main__':
    num_args = len(sys.argv)
    required_args = 'a nef file name and an optional chain_code
[default=A]'
    if num_args > 3:
        print(f'Error: arguments are {required_args}', file=sys.stderr)
        print('exiting...', file=sys.stderr)

        sys.exit(ERROR)
    elif num_args < 2:
        print(f'Error: I need {required_args}', file=sys.stderr)
        print('exiting...', file=sys.stderr)
        sys.exit(ERROR)

    file_path = Path(sys.argv[1])
    if num_args == 3:
        chain_code = sys.argv[2]
    else:
        chain_code = 'A'
    result = read_nef_obs_shifts_from_file_to_pandas(file_path,
chain_code)
    print(result)
```

2. **List of residues taken from SHIFTX2 website containing PDB codes (left) with corresponding BMRB codes**

5226	2BKYA	4342	1EKG	16310	1TVQA
4102	1A7T	10096	1IWMA	4421	1GXQ
4094	2B8X	7264	1EXP_	4336	1HPCB
4566	1AYF	5241	1NW2H	4132	1U9B
6019	2BKYB	15817	1GZIA	5712	1IPB
6939	2F1YA	5843	1Q4R	5898	1UOH
4162	1EPFC	6074	1H70	5206	1MHOA
4354	1LAXA	6754	1QAV	5921	1VC1
4132	2GROA	4202	1HL5J	4296	1MJC
4102	1ZNB	5666	1RMMA	4340	1YP7A
6876	2NNRA	4955	1IIBB	4986	1ZJLA
5508	2GABB	5761	1IU1	5756	1N0S
7264	1FH9A	4084	1UP1_	4174	2BE6A
6760	3FAPB	6283	1VFQA	4562	3LZTA
4077	2FKEA	4717	1Y2GB	5097	1F94
4317	1AILA	6122	1SMXB	4101	1CEXA
6266	1FE4B	5355	1ZW9A	5623	1MN8D
5081	1B2VA	5456	2ADFA	4061	1A6KA
5058	1GNU	4259	1UCKB	5211	1D4TA
4964	1BRIB	4840	2CDNA	5352	1H4GA
4340	1JV4	4553	1XUO	4084	1L3K
4198	1EZ3	6980	2D58	6375	1U07
6776	1UJ8A	6416	1ZLQ	4259	1RGEB
6090	1FF3A	6932	1KBL	5194	1TJM
4401	1TOPA	4076	1OSPO	6542	1M15A
6786	1FF3C	5471	1P7TA	7216	1NEY
6231	1UV0A	5299	2VFX	7086	2FCLA
4354	1FQA	4070	1ZE3C	4031	1RUV
4425	1BDO	5485	1H7M	6494	1DBF
4567	1FZY	6779	1ZE3D	5571	1F4P
4438	1C44	5355	1AM1	4019	1PLC
4378	1G8I	6332	1N2DA	5220	1I1JB
4186	1CBS	15706	1SCJA	5792	1EW4
5352	1H4H	1634	1CDLA	4909	1KJLA
2371	1CM2A	6122	1SN8B	5275	1KQR
4797	1IAZA	4082	1FIL	6575	2CIAA

5244	2END	4766	1DDW	4968	1F5RI
4031	3RN3	5142	1LZ1	4299	1G4CB
5182	1M5E	4472	1U8TA	6332	1M45A
1657	2RN2	4562	1VDQA	4127	2ITLA
6136	8ABP	4857	2A0B	4115	1EMV
4717	1F46A	4091	1BFG	6503	1F2F
4064	1HFCA	4077	1BKFA	6184	1J54
4299	1HKA	4022	1HCBA	4371	1ONC
7356	1ICMA	6075	1JL3B	5485	1W41A
5799	1J97	6923	2AWG	5359	1YKTB
1062	1L2HA	6922	2D3D	4974	1YPCI
5182	2GFEB	7086	2EWRA		
15722	1NQDA	4091	4FGFA		

3. **Amended list of BMRB codes/ PDB codes and which of these didn't run:**

```
#    pdb    chain    bmrb                        res
---  -----  -------  ------------------------  -----
# page 1 col 1
     3LZT   A        4562                       0.93
     1F94   A        5097                       0.97
     1CEX   A        4101                       1.00
     1MN8   D        5623                       1.00
#    5PTI   A        bpti                       1.00
#    1CNR   A        6455,6504                  1.05
     1A6K   A        4061                       1.10
     1D4T   A        5211                       1.10
     1H4G   A        5352                       1.10
     1L3K   A        4084                       1.10
     1U07   A        6375                       1.13
     1RGE   B        4259                       1.15
     1TJM   A        5194                       1.18
     1M15   A        6542                       1.20
     1NEY   A        7216                       1.20
     2FCL   A        7086                       1.20
     1RUV   A        4031                       1.25
     1DBF   A        6494                       1.30
     1F4P   A        5571                       1.30
     1PLC   A        4019                       1.33
     1I1J   B        5220                       1.39
     1EW4   A        5792                       1.40
#    1IWT   A        5130,5142                  1.40
     1KJL   A        4909                       1.40
     1KQR   A        5275                       1.40
     2CIA   A        6575                       1.45
     2END   A        5244                       1.45
#    2F3Y   A        5286,6541,15650,15852      1.45
     3RN3   A        4031                       1.45
     1M5E   A        5182                       1.46
     2RN2   A        1657                       1.48
     8ABP   A        6136                       1.49
#    1A2P   A        975,4964                   1.50
```

```
     1F46   A        4717                        1.50
     1HFC   A        4064                        1.50
     1HKA   A        4299                        1.50
     1ICM   A        7356                        1.50
     1J97   A        5799                        1.50
#    page 1 col 2
     1LZ1   A        5142                        1.50
     1U8T   A        4472                        1.50
     1VDQ   A        4562                        1.50
#    3EZM   A        cvn                         1.50
     1L2H   A        1062                        1.54
     2GFE   B        5182                        1.54
     2A0B   A        4857                        1.57
     1BFG   A        4091                        1.60
# 4077 only available in v2 format
#    1BKF   A        4077                        1.60
     1HCB   A        4022                        1.60
     1JL3   B        6075                        1.60
#    1Q5P   A        maxacal                     1.60
#    1QRX   A        alpha_LP                    1.60
     2AWG   A        6923                        1.60
     2D3D   A        6922                        1.60
     2EWR   A        7086                        1.60
     4FGF   A        4091                        1.60
#    4ICB   A        390,6699                    1.60
#    2CPL   A        cyclophilin                 1.63
     1F5R   I        4968                        1.65
     1G4C   B        4299                        1.65
     1M45   A        6332                        1.65
     1NQD   A        15722                       1.65
#    1SNC   A        snase,4052                  1.65
     2ITL   A        4127                        1.65
#    2TRX   A        62,1812                     1.68
#    1CLL   A        547,6023                    1.70
     1DDW   A        4766                        1.70
     1EMV   A        4115                        1.70
#    1ERT   A        thioredoxin_red             1.70
     1F2F   A        6503                        1.70
     1J54   A        6184                        1.70
#    1MXE   A        547,1634                    1.70
     1ONC   A        4371                        1.70
#    1UB4   B        6828,6833                   1.70
     1W41   A        5485                        1.70
     1YKT   B        5359                        1.70
     1YPC   I        4974                        1.70
# page 2 col 1
     2B8X   A        4094                        1.70
     2BKY   A        5226                        1.70
     2BKY   B        6019                        1.70
     2F1Y   A        6939                        1.70
#    2GBT   C        15712,15713                 1.70
     2GRO   A        4132                        1.70
     2NNR   A        6876                        1.70
     1FH9   A        7264                        1.72
# 4077 only available in v2 format
#    2FKE   A        4077                        1.72
     1FE4   B        6266                        1.75
     1GNU   A        5058                        1.75
     1JV4   A        4340                        1.75
     1UJ8   A        6776                        1.75
     1TOP   A        4401                        1.78
     1UV0   A        6231                    1.78
     1BDO   A        4425                    1.80
     1C44   A        4438                    1.80
     1CBS   A        4186                    1.80
     1CM2   A        2371                    1.80
     1EKG   A        4342                    1.80
#    1EXP   _        7264                    1.80
     1GZI   A        15817                   1.80
     1H70   A        6074                    1.80
     1HL5   J        4202                    1.80
# 4955 contains a non standard residue
#    1IIB   B        4955                    1.80
     1IU1   A        5761                    1.80
#    1NCX   A        5071,5738               1.80
#    1RX2   _        4554,5740               1.80
     1SMX   B        6122                    1.80
#    1UBQ   A        5387,ubiquitin          1.80
     1UCK   B        4259                    1.80
     1XUO   A        4553                    1.80
     1ZLQ   A        6416                    1.80
#    2RNT   A        Ribonouclease_T1        1.80
#    1BCX   A        4704,4705               1.81
#    1IV7   A        4638,5529               1.82
     1ZE3   C        4070                    1.84
     1ZE3   D        6779                    1.84
# page 2 col 2
     1A7T   A        4102                    1.85
     1AYF   A        4566                    1.85
#    1EB0   A        5484,5826               1.85
     1EPF   C        4162                    1.85
     1LAX   A        4354                    1.85
     1ZNB   A        4102                    1.85
     2GAB   B        5508                    1.85
     3FAP   B        6760                    1.85
     1AIL   A        4317                    1.90
     1B2V   A        5081                    1.90
     1BRI   B        4964                    1.90
     1EZ3   A        4198                    1.90
     1FF3   A        6090                    1.90
     1FF3   C        6786                    1.90
     1FQA   A        4354                    1.90
     1FZY   A        4567                    1.90
     1G8I   A        4378                    1.90
     1H4H   A        5352                    1.90
     1IAZ   A        4797                    1.90
     1IWM   A        10096                   1.90
     1NW2   H        5241                    1.90
     1Q4R   A        5843                    1.90
     1QAV   A        6754                    1.90
     1RMM   A        5666                    1.90
#    1RSY   A        4039,4041,4167          1.90
#    1UP1   _        4084                    1.90
     1VFQ   A        6283                    1.90
     1Y2G   B        4717                    1.90
     1ZW9   A        5355                    1.90
     2ADF   A        5456                    1.90
     2CDN   A        4840                    1.90
     2D58   A        6980                    1.90
     1KBL   A        6932                    1.94
     1OSP   O        4076                    1.95
     1P7T   A        5471                    1.95
     2VFX    A         5299                       1.95
     1H7M    A         5485                       1.96
#    1A30    A         HIV_protease_(HIV)         2.00
# page 3 col 1
     1AM1    A         5355                       2.00
#    1C7F    B         5540,5571                  2.00
     1CDL    A         1634                       2.00
     1FIL    A         4082                       2.00
     1GXQ    A         4421                       2.00
     1HPC    B         4336                       2.00
     1IPB    A         5712                       2.00
     1MHO    A         5206                       2.00
     1MJC    A         4296                       2.00
#    1MKA    A         dehydrase                  2.00
     1N0S    A         5756                       2.00
# page 3 col 2
     1N2D    A         6332                       2.00
     1SCJ    A         15706                      2.00
     1SN8    B         6122                       2.00
     1TVQ    A         16310                      2.00
#    1TW4    B         15084,15854                2.00
     1U9B    A         4132                       2.00
     1UOH    A         5898                       2.00
     1VC1    A         5921                       2.00
     1YP7    A         4340                       2.00
     1ZJL    A         4986                       2.00
     2BE6    A         4174                       2.00
```
** It should be noted that any lines with a # at the start were not run.

4. **First non-unlabelled run results from PDB data:**

BMRB_code	bad	total	%bad	%good
bmrb10096	30	177	16	84
bmrb1062	26	149	17	83
bmrb15706	41	275	14	86
bmrb15722	51	114	44	56
bmrb15817	6	65	9	91
bmrb16310	42	125	33	67
bmrb1634	39	142	27	73
bmrb1657	49	155	31	69
bmrb2371	7	85	8	92
bmrb4019	10	99	10	90
bmrb4022	27	257	10	90
bmrb4061	101	151	66	34
bmrb4064	15	156	9	91
bmrb4070	27	205	13	87
bmrb4082	15	139	10	90
bmrb4084	101	174	58	42
bmrb4094	5	129	3	97
bmrb4101	21	197	10	90

bmrb4102	46	229	20	80
bmrb4162	105	189	55	45
bmrb4174	117	144	81	19
bmrb4186	35	136	25	75
bmrb4198	20	124	16	84
bmrb4202	32	153	20	80
bmrb4296	2	69	2	98
bmrb4317	1	70	1	99
bmrb4336	17	131	12	88
bmrb4342	6	119	5	95
bmrb4371	6	103	5	95
bmrb4378	18	187	9	91
bmrb4401	104	162	64	36
bmrb4421	18	104	17	83
bmrb4425	5	80	6	94
bmrb4438	25	123	20	80
bmrb4472	11	125	8	92
bmrb4553	5	181	2	98
bmrb4562	8	129	6	94
bmrb4566	30	102	29	71
bmrb4567	20	149	13	87
bmrb4717	9	139	6	94
bmrb4766	33	110	30	70
bmrb4797	22	175	12	88
bmrb4857	5	118	4	96
bmrb4909	23	137	16	84
bmrb4964	10	108	9	91
bmrb4968	6	57	10	90
bmrb4974	6	64	9	91
bmrb4986	68	370	18	82
bmrb5058	18	117	15	85
bmrb5081	13	173	7	93
bmrb5097	10	63	15	85
bmrb5142	4	130	3	97
bmrb5206	17	88	19	81
bmrb5220	18	106	16	84
bmrb5226	2	89	2	98
bmrb5241	42	105	40	60
bmrb5244	6	137	4	96
bmrb5352	12	205	5	95
bmrb5355	28	213	13	87
bmrb5456	17	189	8	92
bmrb5471	127	719	17	83

bmrb5485	15	100	15	85
bmrb5508	32	114	28	72
bmrb5571	7	146	4	96
bmrb5623	30	95	31	69
bmrb5666	57	228	25	75
bmrb5712	22	191	11	89
bmrb5756	30	173	17	83
bmrb5761	34	120	28	72
bmrb5792	106	106	100	0
bmrb5799	15	210	7	93
bmrb5843	9	102	8	92
bmrb5898	17	223	7	93
bmrb5921	4	110	3	97
bmrb6019	3	89	3	97
bmrb6074	34	255	13	87
bmrb6075	20	137	14	86
bmrb6090	34	211	16	84
bmrb6122	7	87	8	92
bmrb6136	62	305	20	80
bmrb6184	17	174	9	91
bmrb6266	20	68	29	71
bmrb6283	10	132	7	93
bmrb6375	14	90	15	85
bmrb6416	159	498	31	69
bmrb6494	20	127	15	85
bmrb6503	5	104	4	96
bmrb6542	79	356	22	78
bmrb6575	15	98	15	85
bmrb6786	76	211	36	64
bmrb6876	8	110	7	93
bmrb6922	5	83	6	94
bmrb6923	118	118	100	0
bmrb6980	7	107	6	94
bmrb7086	23	156	14	86
bmrb7216	86	247	34	66
bmrb7356	18	131	13	87

5. Last non-unlabelled (control run) – 6th (PDB)

control

BMRB_code	PDB	chain	bad	total	%bad

bmrb10096	1IWM	A	30	177	16
bmrb1062	1L2H	A	26	149	17
bmrb15706	1SCJ	A	44	275	16
bmrb15722	1NQD	A	54	114	47
bmrb15817	1GZI	A	6	65	9
bmrb16310	1TVQ	A	43	125	34
bmrb1634	1CDL	A	43	142	30
bmrb1657	2RN2	A	54	155	34
bmrb2371	1CM2	A	7	85	8
bmrb4019	1PLC	A	10	99	10
bmrb4022	1HCB	A	28	257	10
bmrb4031	1RUV	A	4	124	3
bmrb4031	3RN3	A	4	124	3
bmrb4061	1A6K	A	104	151	68
bmrb4064	1HFC	A	15	156	9
bmrb4070	1ZE3	C	27	205	13
bmrb4076	1OSP	O	7	251	2
bmrb4082	1FIL	A	17	139	12
bmrb4084	1L3K	A	8	84	9
bmrb4091	1BFG	A	13	126	10
bmrb4091	4FGF	A	14	123	11
bmrb4094	2B8X	A	5	129	3
bmrb4101	1CEX	A	22	197	11
bmrb4102	1A7T	A	31	227	13
bmrb4102	1ZNB	A	48	229	20
bmrb4115	1EMV	A	13	83	15
bmrb4127	2ITL	A	59	120	49
bmrb4132	1U9B	A	25	159	15
bmrb4132	2GRO	A	21	156	13
bmrb4162	1EPF	C	4	98	4
bmrb4174	2BE6	A	42	75	56
bmrb4186	1CBS	A	35	136	25
bmrb4198	1EZ3	A	15	121	12
bmrb4202	1HL5	J	35	153	22
bmrb4259	1RGE	B	10	96	10
bmrb4259	1UCK	B	11	96	11
bmrb4296	1MJC	A	2	69	2
bmrb4299	1G4C	B	47	158	29
bmrb4299	1HKA	A	49	158	31
bmrb4317	1AIL	A	1	70	1
bmrb4336	1HPC	B	10	131	7
bmrb4340	1JV4	A	8	157	5
bmrb4340	1YP7	A	6	157	3

bmrb4342	1EKG	A	6	119	5
bmrb4354	1FQA	A	85	370	22
bmrb4354	1LAX	A	84	369	22
bmrb4371	1ONC	A	5	103	4
bmrb4378	1G8I	A	20	187	10
bmrb4401	1TOP	A	16	90	17
bmrb4421	1GXQ	A	18	104	17
bmrb4425	1BDO	A	5	80	6
bmrb4438	1C44	A	25	123	20
bmrb4472	1U8T	A	11	125	8
bmrb4553	1XUO	A	5	181	2
bmrb4562	1VDQ	A	8	129	6
bmrb4562	3LZT	A	9	129	6
bmrb4566	1AYF	A	30	102	29
bmrb4567	1FZY	A	20	149	13
bmrb4717	1F46	A	9	139	6
bmrb4717	1Y2G	B	7	140	5
bmrb4766	1DDW	A	37	110	33
bmrb4797	1IAZ	A	22	175	12
bmrb4840	2CDN	A	31	181	17
bmrb4857	2A0B	A	5	118	4
bmrb4909	1KJL	A	21	137	15
bmrb4964	1BRI	B	6	108	5
bmrb4968	1F5R	I	6	57	10
bmrb4974	1YPC	I	6	64	9
bmrb4986	1ZJL	A	65	370	17
bmrb5058	1GNU	A	18	117	15
bmrb5081	1B2V	A	13	173	7
bmrb5097	1F94	A	11	63	17
bmrb5142	1LZ1	A	4	130	3
bmrb5182	1M5E	A	22	258	8
bmrb5182	2GFE	B	28	260	10
bmrb5194	1TJM	A	9	150	6
bmrb5206	1MHO	A	17	88	19
bmrb5211	1D4T	A	19	104	18
bmrb5220	1I1J	B	8	104	7
bmrb5226	2BKY	A	2	89	2
bmrb5241	1NW2	H	30	105	28
bmrb5244	2END	A	10	137	7
bmrb5275	1KQR	A	14	160	8
bmrb5299	2VFX	A	61	197	30
bmrb5352	1H4G	A	14	205	6
bmrb5352	1H4H	A	13	206	6

bmrb5355	1AM1	A	33	206	16
bmrb5355	1ZW9	A	24	206	11
bmrb5359	1YKT	B	5	56	8
bmrb5456	2ADF	A	18	189	9
bmrb5471	1P7T	A	133	719	18
bmrb5485	1H7M	A	14	97	14
bmrb5485	1W41	A	11	99	11
bmrb5508	2GAB	B	25	114	21
bmrb5571	1F4P	A	5	146	3
bmrb5623	1MN8	D	22	97	22
bmrb5666	1RMM	A	58	228	25
bmrb5712	1IPB	A	21	191	10
bmrb5756	1N0S	A	30	173	17
bmrb5761	1IU1	A	35	119	29
bmrb5792	1EW4	A	11	106	10
bmrb5799	1J97	A	18	210	8
bmrb5843	1Q4R	A	9	102	8
bmrb5898	1UOH	A	17	223	7
bmrb5921	1VC1	A	4	110	3
bmrb6019	2BKY	B	2	89	2
bmrb6074	1H70	A	34	255	13
bmrb6075	1JL3	B	21	137	15
bmrb6090	1FF3	A	40	211	18
bmrb6122	1SMX	B	13	87	14
bmrb6122	1SN8	B	14	87	16
bmrb6136	8ABP	A	62	305	20
bmrb6184	1J54	A	18	174	10
bmrb6231	1UV0	A	17	137	12
bmrb6266	1FE4	B	9	68	13
bmrb6283	1VFQ	A	11	132	8
bmrb6332	1M45	A	9	146	6
bmrb6332	1N2D	A	1	147	0
bmrb6375	1U07	A	7	89	7
bmrb6416	1ZLQ	A	165	498	33
bmrb6494	1DBF	A	20	127	15
bmrb6503	1F2F	A	5	104	4
bmrb6542	1M15	A	94	356	26
bmrb6575	2CIA	A	15	98	15
bmrb6754	1QAV	A	5	86	5
bmrb6760	3FAP	B	5	94	5
bmrb6776	1UJ8	A	10	65	15
bmrb6779	1ZE3	D	12	101	11
bmrb6786	1FF3	C	48	185	25

bmrb6876	2NNR	A	8	110	7
bmrb6922	2D3D	A	5	83	6
bmrb6923	2AWG	A	40	118	33
bmrb6932	1KBL	A	13	128	10
bmrb6939	2F1Y	A	48	143	33
bmrb6980	2D58	A	9	107	8
bmrb7086	2EWR	A	21	156	13
bmrb7086	2FCL	A	27	156	17
bmrb7216	1NEY	A	93	247	37
bmrb7264	1FH9	A	66	312	21
bmrb7356	1ICM	A	18	131	13

6. **First unlabelled run results (PDB):**

BMRB_code	bad	total	%bad
bmrb10096	177	177	100
bmrb1062	149	149	100
bmrb15706	41	275	14
bmrb15722	114	114	100
bmrb15817	65	65	100
bmrb16310	42	125	33
bmrb1634	39	142	27
bmrb1657	49	155	31
bmrb2371	7	85	8
bmrb4019	10	99	10
bmrb4022	27	257	10
bmrb4031	4	124	3
bmrb4061	151	151	100
bmrb4064	156	156	100
bmrb4070	27	205	13
bmrb4076	214	214	100
bmrb4082	15	139	10
bmrb4084	101	174	58
bmrb4091	123	123	100
bmrb4094	129	129	100
bmrb4101	21	197	10
bmrb4132	159	159	100
bmrb4174	144	144	100
bmrb4186	35	136	25
bmrb4198	124	124	100
bmrb4202	32	153	20

bmrb4259	15	96	15
bmrb4296	2	69	2
bmrb4299	57	158	36
bmrb4317	1	70	1
bmrb4336	17	131	12
bmrb4340	6	157	3
bmrb4342	119	119	100
bmrb4354	83	370	22
bmrb4371	103	103	100
bmrb4378	18	187	9
bmrb4401	104	162	64
bmrb4421	104	104	100
bmrb4425	80	80	100
bmrb4438	25	123	20
bmrb4472	11	125	8
bmrb4553	180	181	99
bmrb4562	8	129	6
bmrb4567	148	149	99
bmrb4717	9	139	6
bmrb4766	109	110	99
bmrb4797	22	175	12
bmrb4857	118	118	100
bmrb4909	137	137	100
bmrb4964	10	108	9
bmrb4968	232	232	100
bmrb4974	64	64	100
bmrb4986	68	370	18
bmrb5058	117	117	100
bmrb5081	172	173	99
bmrb5097	10	63	15
bmrb5142	4	130	3
bmrb5182	34	260	13
bmrb5194	157	157	100
bmrb5206	87	88	98
bmrb5220	106	106	100
bmrb5226	2	89	2
bmrb5241	42	105	40
bmrb5244	6	137	4
bmrb5275	160	160	100
bmrb5299	206	206	100
bmrb5352	12	205	5
bmrb5355	213	213	100
bmrb5359	230	230	100

bmrb5456	189	189	100
bmrb5471	127	719	17
bmrb5485	15	100	15
bmrb5508	114	114	100
bmrb5571	7	146	4
bmrb5623	30	95	31
bmrb5666	57	228	25
bmrb5712	22	191	11
bmrb5756	30	173	17
bmrb5761	120	120	100
bmrb5792	106	106	100
bmrb5799	15	210	7
bmrb5843	9	102	8
bmrb5898	17	223	7
bmrb5921	4	110	3
bmrb6074	255	255	100
bmrb6075	20	137	14
bmrb6090	34	211	16
bmrb6122	87	87	100
bmrb6136	62	305	20
bmrb6184	17	174	9
bmrb6231	140	140	100
bmrb6266	20	68	29
bmrb6283	132	132	100
bmrb6332	8	146	5
bmrb6375	90	90	100
bmrb6416	159	498	31
bmrb6494	20	127	15
bmrb6503	104	104	100
bmrb6542	79	356	22
bmrb6575	98	98	100
bmrb6754	90	90	100
bmrb6760	105	107	98
bmrb6776	73	73	100
bmrb6779	205	205	100
bmrb6786	76	211	36
bmrb6876	110	110	100
bmrb6922	83	83	100
bmrb6923	118	118	100
bmrb6932	872	872	100
bmrb6980	107	107	100
bmrb7086	23	156	14
bmrb7216	86	247	34

bmrb7264	63	312	20
bmrb7356	18	131	13

7. Final unlabelled run results (8th)

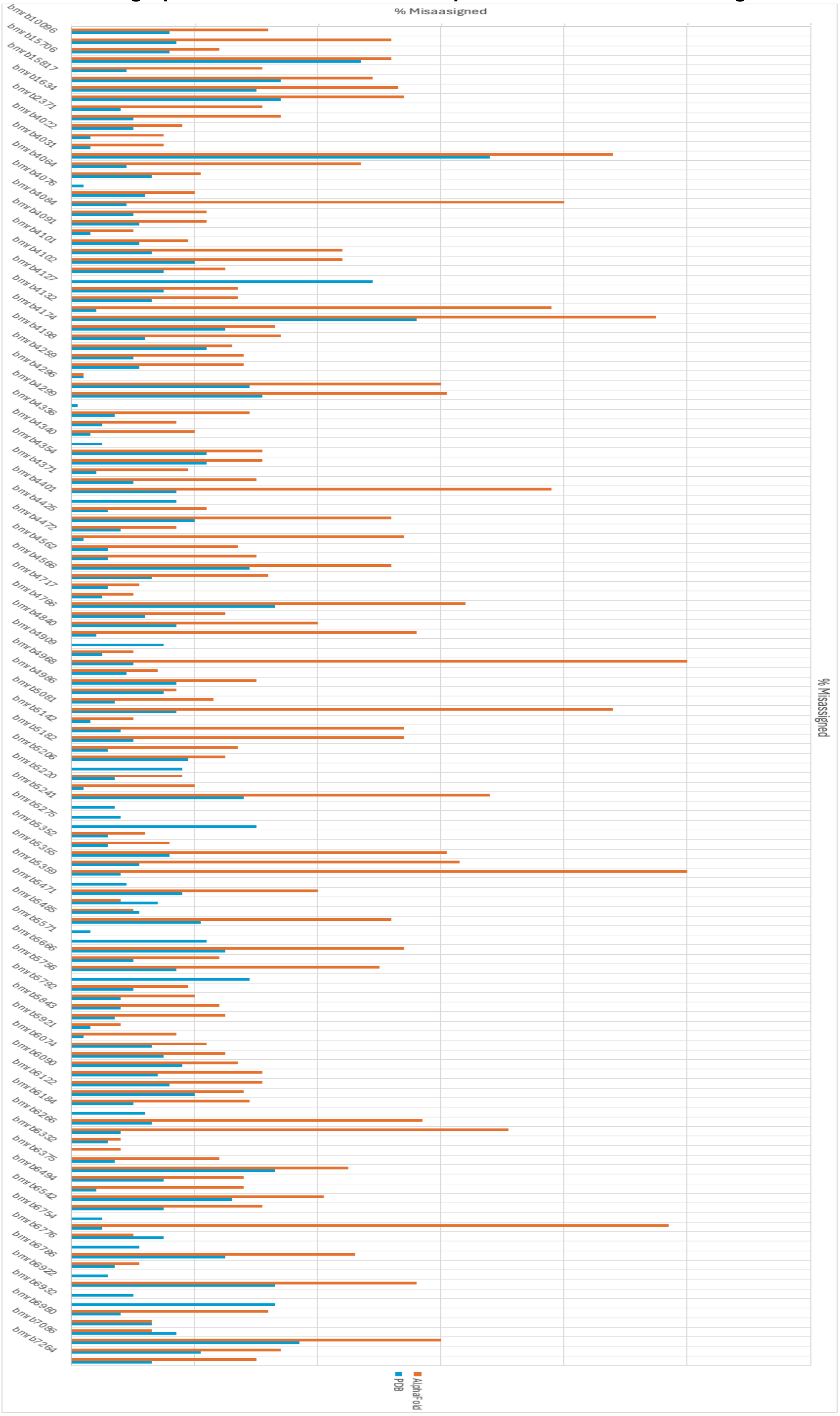
BMRB_code	PDB	chain	bad	total	%bad
bmrb10096	1IWM	A	27	177	15
bmrb1062	1L2H	A	22	149	14
bmrb15706	1SCJ	A	44	275	16
bmrb15722	1NQD	A	47	114	41
bmrb15817	1GZI	A	6	65	9
bmrb16310	1TVQ	A	27	125	21
bmrb1634	1CDL	A	33	142	23
bmrb1657	2RN2	A	37	155	23
bmrb2371	1CM2	A	11	85	12
bmrb4019	1PLC	A	10	99	10
bmrb4022	1HCB	A	28	257	10
bmrb4031	1RUV	A	4	124	3
bmrb4031	3RN3	A	4	124	3
bmrb4061	1A6K	A	89	151	58
bmrb4064	1HFC	A	15	156	9
bmrb4070	1ZE3	C	22	205	10
bmrb4076	1OSP	O	6	251	2
bmrb4082	1FIL	A	11	139	7
bmrb4084	1L3K	A	7	84	8
bmrb4091	1BFG	A	13	126	10
bmrb4091	4FGF	A	14	123	11
bmrb4094	2B8X	A	5	129	3
bmrb4101	1CEX	A	18	197	9
bmrb4102	1A7T	A	28	227	12
bmrb4102	1ZNB	A	42	229	18
bmrb4115	1EMV	A	14	83	16
bmrb4127	2ITL	A	44	120	36
bmrb4132	1U9B	A	19	159	11
bmrb4132	2GRO	A	18	156	11
bmrb4162	1EPF	C	3	98	3
bmrb4174	2BE6	A	30	75	40
bmrb4186	1CBS	A	28	136	20
bmrb4198	1EZ3	A	14	121	11
bmrb4202	1HL5	J	37	153	24
bmrb4259	1RGE	B	10	96	10

bmrb4259	1UCK	B	11	96	11
bmrb4296	1MJC	A	2	69	2
bmrb4299	1G4C	B	53	158	33
bmrb4299	1HKA	A	42	158	26
bmrb4317	1AIL	A	1	70	1
bmrb4336	1HPC	B	12	131	9
bmrb4340	1JV4	A	6	157	3
bmrb4340	1YP7	A	5	157	3
bmrb4342	1EKG	A	5	119	4
bmrb4354	1FQA	A	89	370	24
bmrb4354	1LAX	A	85	369	23
bmrb4371	1ONC	A	5	103	4
bmrb4378	1G8I	A	21	187	11
bmrb4401	1TOP	A	11	90	12
bmrb4421	1GXQ	A	17	104	16
bmrb4425	1BDO	A	5	80	6
bmrb4438	1C44	A	28	123	22
bmrb4472	1U8T	A	13	125	10
bmrb4553	1XUO	A	5	181	2
bmrb4562	1VDQ	A	4	129	3
bmrb4562	3LZT	A	4	129	3
bmrb4566	1AYF	A	28	102	27
bmrb4567	1FZY	A	20	149	13
bmrb4717	1F46	A	10	139	7
bmrb4717	1Y2G	B	7	140	5
bmrb4766	1DDW	A	36	110	32
bmrb4797	1IAZ	A	19	175	10
bmrb4840	2CDN	A	26	181	14
bmrb4857	2A0B	A	5	118	4
bmrb4909	1KJL	A	21	137	15
bmrb4964	1BRI	B	6	108	5
bmrb4968	1F5R	I	4	57	7
bmrb4974	1YPC	I	6	64	9
bmrb4986	1ZJL	A	70	370	18
bmrb5058	1GNU	A	18	117	15
bmrb5081	1B2V	A	13	173	7
bmrb5097	1F94	A	0	63	0
bmrb5142	1LZ1	A	2	130	1
bmrb5182	1M5E	A	24	258	9
bmrb5182	2GFE	B	28	260	10
bmrb5194	1TJM	A	11	150	7
bmrb5206	1MHO	A	15	88	17
bmrb5211	1D4T	A	16	104	15

bmrb5220	1I1J	B	9	104	8
bmrb5226	2BKY	A	2	89	2
bmrb5241	1NW2	H	25	105	23
bmrb5244	2END	A	10	137	7
bmrb5275	1KQR	A	12	160	7
bmrb5299	2VFX	A	53	197	26
bmrb5352	1H4G	A	13	205	6
bmrb5352	1H4H	A	14	206	6
bmrb5355	1AM1	A	28	206	13
bmrb5355	1ZW9	A	26	206	12
bmrb5359	1YKT	B	5	56	8
bmrb5456	2ADF	A	20	189	10
bmrb5471	1P7T	A	123	719	17
bmrb5485	1H7M	A	11	97	11
bmrb5485	1W41	A	10	99	10
bmrb5508	2GAB	B	30	114	26
bmrb5571	1F4P	A	5	146	3
bmrb5623	1MN8	D	21	97	21
bmrb5666	1RMM	A	55	228	24
bmrb5712	1IPB	A	21	191	10
bmrb5756	1N0S	A	31	173	17
bmrb5761	1IU1	A	25	119	21
bmrb5792	1EW4	A	13	106	12
bmrb5799	1J97	A	19	210	9
bmrb5843	1Q4R	A	7	102	6
bmrb5898	1UOH	A	14	223	6
bmrb5921	1VC1	A	4	110	3
bmrb6019	2BKY	B	2	89	2
bmrb6074	1H70	A	27	255	10
bmrb6075	1JL3	B	17	137	12
bmrb6090	1FF3	A	42	211	19
bmrb6122	1SMX	B	13	87	14
bmrb6122	1SN8	B	14	87	16
bmrb6136	8ABP	A	60	305	19
bmrb6184	1J54	A	17	174	9
bmrb6231	1UV0	A	19	137	13
bmrb6266	1FE4	B	7	68	10
bmrb6283	1VFQ	A	8	132	6
bmrb6332	1M45	A	7	146	4
bmrb6332	1N2D	A	1	147	0
bmrb6375	1U07	A	7	89	7
bmrb6416	1ZLQ	A	162	498	32
bmrb6494	1DBF	A	13	127	10

bmrb6503	1F2F	A	7	104	6
bmrb6542	1M15	A	91	356	25
bmrb6575	2CIA	A	13	98	13
bmrb6754	1QAV	A	3	86	3
bmrb6760	3FAP	B	5	94	5
bmrb6776	1UJ8	A	10	65	15
bmrb6779	1ZE3	D	12	101	11
bmrb6786	1FF3	C	47	185	25
bmrb6876	2NNR	A	8	110	7
bmrb6922	2D3D	A	6	83	7
bmrb6923	2AWG	A	39	118	33
bmrb6932	1KBL	A	12	128	9
bmrb6939	2F1Y	A	43	143	30
bmrb6980	2D58	A	9	107	8
bmrb7086	2EWR	A	24	156	15
bmrb7086	2FCL	A	32	156	20
bmrb7216	1NEY	A	82	247	33
bmrb7264	1FH9	A	60	312	19
bmrb7356	1ICM	A	18	131	13

8. Full graph of differences between AlphaFold and PDB %misassigned



6 BIBLIOGRPHY:

1. Dill KA, Ozkan SB, Shell MS, Weikl TR. The Protein Folding Problem. Annu Rev Biophys [Internet]. 2008 Jun 6 [cited 2024 Jan 15];37:289. Available from: /pmc/articles/PMC2443096/

2. Breda A, Valadares NF, Souza ON de, Garratt RC. Protein Structure, Modelling and Applications. 2007 Sep 14 [cited 2025 Feb 10]; Available from: https://www.ncbi.nlm.nih.gov/books/NBK6824/

3. Jumper J, Evans R, Pritzel A, Green T, Figurnov M, Ronneberger O, et al. Highly accurate protein structure prediction with AlphaFold. Nature 2021 596:7873 [Internet]. 2021 Jul 15 [cited 2024 Jun 10];596(7873):583–9. Available from: https://www.nature.com/articles/s41586-021-03819-2

4. Baek M, DiMaio F, Anishchenko I, Dauparas J, Ovchinnikov S, Lee GR, et al. Accurate prediction of protein structures and interactions using a three-track neural network. Science (1979) [Internet]. 2021 Aug 20 [cited 2025 Feb 10];373(6557):871–6. Available from: https://www.science.org/doi/10.1126/science.abj8754

5. Yang Z, Zeng X, Zhao Y, Chen R. AlphaFold2 and its applications in the fields of biology and medicine. Signal Transduction and Targeted Therapy 2023 8:1 [Internet]. 2023 Mar 14 [cited 2024 Jan 15];8(1):1–14. Available from: https://www.nature.com/articles/s41392-023-01381-z

6. Scopus - Citation Overview [Internet]. [cited 2025 Mar 5]. Available from: https://www-scopus-com.chain.kent.ac.uk/pages/citationOverview?key=d9d3c278-8104-4da9-a083-05d40ebc37a8&origin=resultslist

7. Kryshtafovych A, Schwede T, Topf M, Fidelis K, Moult J. Critical assessment of methods of protein structure prediction (CASP)-Round XIV. Proteins [Internet]. 2021 Dec 1 [cited 2024 Jan 15];89(12):1607–17. Available from: https://pubmed.ncbi.nlm.nih.gov/34533838/

8. Tejero R, Huang YJ, Ramelot TA, Montelione GT. AlphaFold Models of Small Proteins Rival the Accuracy of Solution NMR Structures. Front Mol Biosci. 2022 Jun 13;9:877000.

9. Strengths and limitations of AlphaFold2 | AlphaFold [Internet]. [cited 2025 Mar 3]. Available from: https://www.ebi.ac.uk/training/online/courses/alphafold/an-introductory-guide-to-its-strengths-and-limitations/strengths-and-limitations-of-alphafold/

10. Laurents D V. AlphaFold 2 and NMR Spectroscopy: Partners to Understand Protein Structure, Dynamics and Function. Front Mol Biosci. 2022 May 17;9:906437.

11. Example of AlphaFold structure. AlphaFold model of Mid1-interacting... | Download Scientific Diagram [Internet]. [cited 2024 Aug 1]. Available from: https://www.researchgate.net/figure/Example-of-AlphaFold-structure-AlphaFold-model-of-Mid1-interacting-protein-1-downloaded_fig1_358754786

12. ANFINSEN CB, HABER E. Studies on the Reduction and Re-formation of Protein Disulfide Bonds. Journal of Biological Chemistry. 1961 May 1;236(5):1361–3.

13. Gambardella G, Notari S, Cavaterra D, Iavarone F, Castagnola M, Bocedi A, et al. The Anfinsen Dogma: Intriguing Details Sixty-Five Years Later. Int J Mol Sci [Internet]. 2022 Jul 1 [cited 2025 Feb 10];23(14):7759. Available from: https://pmc.ncbi.nlm.nih.gov/articles/PMC9318638/

14. Protein folding guided by funnel-shaped energy landscape (see Ref. 20). | Download Scientific Diagram [Internet]. [cited 2025 Mar 5]. Available from: https://www.researchgate.net/figure/Protein-folding-guided-by-funnel-shaped-energy-landscape-see-Ref-20_fig2_321731034

15. Yang Z, Zeng X, Zhao Y, Chen R. AlphaFold2 and its applications in the fields of biology and medicine. Signal Transduction and Targeted Therapy 2023 8:1 [Internet]. 2023 Mar 14 [cited 2024 Jun 25];8(1):1–14. Available from: https://www.nature.com/articles/s41392-023-01381-z

16. Rathore I, Mishra V, Bhaumik P. Advancements in macromolecular crystallography: from past to present. Emerg Top Life Sci [Internet]. 2021 May 14 [cited 2024 Oct 7];5(1):127–49. Available from: /emergtoplifesci/article/5/1/127/228615/Advancements-in-macromolecular-crystallography

17. Thomas JM. The birth of X-ray crystallography. Nature 2012 491:7423 [Internet]. 2012 Nov 7 [cited 2025 Feb 10];491(7423):186–7. Available from: https://www.nature.com/articles/491186a

18. Smyth MS, Martin JHJ. x Ray crystallography. Molecular Pathology [Internet]. 2000 [cited 2024 Jan 30];53(1):8. Available from: /pmc/articles/PMC1186895/

19. What is Protein X-Ray Crystallography? | John Innes Centre [Internet]. [cited 2024 Jan 30]. Available from: https://www.jic.ac.uk/blog/what-is-protein-x-ray-crystallography/

20. Immuto Scientific [Internet]. [cited 2024 Jan 30]. Available from: https://www.immutoscientific.com/blog/x-ray-crystallography

21. Protein Structure Analysis | Protein Electron Microscopy | Thermo Fisher Scientific - UK [Internet]. [cited 2024 Jan 30]. Available from: https://www.thermofisher.com/uk/en/home/electron-microscopy/life-sciences/protein-analysis.html

22. Callaway E. Revolutionary cryo-EM is taking over structural biology. Nature. 2020 Feb 1;578(7794):201.

23. Glaeser RM, Hall RJ. Reaching the Information Limit in Cryo-EM of Biological Macromolecules: Experimental Aspects.

24. Benjin X, Ling L. Developments, applications, and prospects of cryo-electron microscopy. 2019 [cited 2024 Jan 30]; Available from: https://onlinelibrary.wiley.com/doi/10.1002/pro.3805

25. Singh MK, Singh A. Nuclear magnetic resonance spectroscopy. Characterization of Polymers and Fibres [Internet]. 2022 [cited 2024 Feb 5];321–39. Available from: https://linkinghub.elsevier.com/retrieve/pii/B9780128239865000117

26. CHAPTER-3 [Internet]. [cited 2025 Feb 10]. Available from: https://www.cis.rit.edu/htbooks/nmr/chap-3/chap-3.htm

27. How NMR Works | NMR 101 | Spectroscopy | Bruker | Bruker [Internet]. [cited 2024 Feb 5]. Available from: https://www.bruker.com/en/resources/library/application-notes-mr/nmr-101.html

28. X-Ray Crystallography vs. NMR Spectroscopy [Internet]. [cited 2024 Feb 5]. Available from: https://www.news-medical.net/life-sciences/X-Ray-Crystallography-vs-NMR-Spectroscopy.aspx

29. Nuclear Magnetic Resonance | [Internet]. [cited 2024 Sep 2]. Available from: https://chembam.com/definitions/nmr/

30. NMR Spectroscopy Principles, Interpreting an NMR Spectrum and Common Problems | Technology Networks [Internet]. [cited 2024 Sep 2]. Available from: https://www.technologynetworks.com/analysis/articles/nmr-spectroscopy-principles-interpreting-an-nmr-spectrum-and-common-problems-355891

31. Mazumder A, Dubey DK. Nuclear Magnetic Resonance (NMR) Spectroscopy. Reference Module in Chemistry, Molecular Sciences and Chemical Engineering. 2013;

32. Doucleff M, Hatcher-Skeers M, Crane NJ. Pocket guide to biomolecular NMR. Pocket Guide to Biomolecular NMR. Springer; 2011. 1–159 p.

33. Pauli R, Wilson M. The Basic Principles of Magnetic Resonance Imaging. Encyclopedia of Behavioral Neuroscience: Volumes 1-3, Second edition. 2021 Jan 1;1–3:V1-105-V1-113.

34. V J-Coupling.

35. NMR J-couplings [Internet]. [cited 2024 Feb 12]. Available from: https://www.tcm.phy.cam.ac.uk/castep/documentation/WebHelp/content/modules/castep/thcastepjcoupling.htm

36. Saalwächter K, Spiess HW. 2.07 - Solid-State NMR of Polymers. Polymer Science: a Comprehensive Reference: Volume 1-10. 2012 Jan 1;1–10:185–219.

37. Assignment Theory | Protein NMR [Internet]. [cited 2024 Feb 12]. Available from: https://protein-nmr.org.uk/solution-nmr/assignment-theory/

38. Bishop AC, Torres-Montalvo G, Kotaru S, Mimun K, Wand AJ. Robust automated backbone triple resonance NMR assignments of proteins using Bayesian-based simulated annealing. Nature Communications 2023 14:1 [Internet]. 2023 Mar 21 [cited

2024 Feb 5];14(1):1–15. Available from: https://www.nature.com/articles/s41467-023-37219-z

39. Figure S4. Alignment of 2D 1 H-15 N HSQC spectra of DKP-proinsulin and... | Download Scientific Diagram [Internet]. [cited 2024 Nov 5]. Available from: https://www.researchgate.net/figure/Figure-S4-Alignment-of-2D-1-H-15-N-HSQC-spectra-of-DKP-proinsulin-and-DKP-insulin-at-pH_fig4_41165982

40. Analysis of NMR Spectra [Internet]. [cited 2024 Feb 12]. Available from: https://www.cryst.bbk.ac.uk/PPS2/projects/schirra/html/assign.htm

41. AnalysisAssign - Software for analysis of biological NMR data - CCPN [Internet]. [cited 2024 Sep 2]. Available from: https://ccpn.ac.uk/software/analysisassign/

42. Hu Y, Cheng K, He L, Zhang X, Jiang B, Jiang L, et al. NMR-Based Methods for Protein Analysis. Anal Chem [Internet]. 2021 Feb 2 [cited 2025 Apr 29];93(4):1866–79. Available from: /doi/pdf/10.1021/acs.analchem.0c03830

43. Tugarinov V, Muhandiram R, Ayed A, Kay LE. Four-dimensional NMR spectroscopy of a 723-residue protein: Chemical shift assignments and secondary structure of malate synthase G. J Am Chem Soc [Internet]. 2002 Aug 28 [cited 2025 Apr 29];124(34):10025–35. Available from: /doi/pdf/10.1021/ja0205636

44. Li GW, Liu H, Qiu F, Wang XJ, Lei XX. Residual Dipolar Couplings in Structure Determination of Natural Products. Nat Prod Bioprospect [Internet]. 2018 Aug 1 [cited 2024 Feb 13];8(4):279–95. Available from: https://link.springer.com/article/10.1007/s13659-018-0174-x

45. Saibo N V., Maiti S, Acharya B, De S. Analysis of structure and dynamics of intrinsically disordered regions in proteins using solution NMR methods. Advances in Protein Molecular and Structural Biology Methods. 2022 Jan 1;535–50.

46. Strickland M, Tjandra N, Strickland M, Tjandra • N. Residual Dipolar Coupling for Conformational and Dynamic Studies. Modern Magnetic Resonance [Internet]. 2017 [cited 2024 Sep 2];1–16. Available from: https://link.springer.com/referenceworkentry/10.1007/978-3-319-28275-6_86-1

47. Non-Uniform Sampling (NUS) | Bruker [Internet]. [cited 2024 Feb 26]. Available from: https://www.bruker.com/en/products-and-solutions/mr/nmr-software/topspin/nus.html

48. Hyberts SG, Arthanari H, Wagner G. Applications of non-uniform sampling and processing. Top Curr Chem [Internet]. 2012 [cited 2024 Feb 26];316:125. Available from: /pmc/articles/PMC3292636/

49. Rowlinson B, Crublet E, Kerfah R, Plevin MJ. Specific isotopic labelling and reverse labelling for protein NMR spectroscopy: using metabolic precursors in sample preparation. Biochem Soc Trans [Internet]. 2022 Dec 12 [cited 2024 Feb 26];50(6):1555. Available from: /pmc/articles/PMC9788560/

50. Fig. S1 2D 15 N, 1 H-HSQC spectrum of 15 N-labeled ubiquitin with the... | Download Scientific Diagram [Internet]. [cited 2024 Oct 7]. Available from: https://www.researchgate.net/figure/Fig-S1-2D-15-N-1-H-HSQC-spectrum-of-15-N-labeled-ubiquitin-with-the-same-sample_fig4_51825588

51. Krishnarjuna B, Jaipuria G, Thakur A, D'Silva P, Atreya HS. Amino acid selective unlabeling for sequence specific resonance assignments in proteins. J Biomol NMR [Internet]. 2010 Jan [cited 2025 Feb 26];49(1):39. Available from: https://pmc.ncbi.nlm.nih.gov/articles/PMC3020294/

52. Rodger A, Marshall D. Beginners guide to circular dichroism. Biochem (Lond) [Internet]. 2021 Apr 12 [cited 2024 Aug 19];43(2):58–64. Available from: /biochemist/article/43/2/58/228163/Beginners-guide-to-circular-dichroism

53. Far UV Circular Dichroism Spectroscopy Analysis - Creative Proteomics [Internet]. [cited 2024 Aug 19]. Available from: https://www.creative-proteomics.com/pronalyse/circular-dichroism-spectroscopy-far-uv.html

54. Standard CD spectra redrawn from Corrêa et al.[5] Each of the three... | Download Scientific Diagram [Internet]. [cited 2024 Aug 27]. Available from: https://www.researchgate.net/figure/Standard-CD-spectra-redrawn-from-Correa-et-al5-Each-of-the-three-basic-secondary_fig1_266950207

55. » What is the energy of a hydrogen bond? [Internet]. [cited 2024 Aug 27]. Available from: https://book.bionumbers.org/what-is-the-energy-of-a-hydrogen-bond/

56. Pandas Pros and Cons Compared [Internet]. [cited 2024 Sep 13]. Available from: https://www.altexsoft.com/blog/pandas-library/

57. The Assignment Problem - HungarianAlgorithm.com [Internet]. [cited 2024 Feb 6]. Available from: https://www.hungarianalgorithm.com/assignmentproblem.php

58. Hungarian Maximum Matching Algorithm | Brilliant Math & Science Wiki [Internet]. [cited 2024 Feb 5]. Available from: https://brilliant.org/wiki/hungarian-matching/

59. Hungarian Maximum Matching Algorithm | Brilliant Math & Science Wiki [Internet]. [cited 2024 Oct 8]. Available from: https://brilliant.org/wiki/hungarian-matching/

60. Han B, Liu Y, Ginzinger SW, Wishart DS. SHIFTX2: significantly improved protein chemical shift prediction. J Biomol NMR [Internet]. 2011 May [cited 2024 Mar 19];50(1):43. Available from: /pmc/articles/PMC3085061/

61. Shen Y, Bax A. SPARTA+: a modest improvement in empirical NMR chemical shift prediction by means of an artificial neural network. J Biomol NMR [Internet]. 2010 Sep [cited 2024 Mar 19];48(1):13. Available from: /pmc/articles/PMC2935510/

62. Ptaszek AL, Li J, Konrat R, Platzer G, Head-Gordon T. UCBShift 2.0: Bridging the Gap from Backbone to Side Chain Protein Chemical Shift Prediction for Protein Structures. J Am Chem Soc [Internet]. 2024 Nov 20 [cited 2025 Feb 10];146(46):31733–45. Available from: https://pubs.acs.org/doi/full/10.1021/jacs.4c10474

63. Jung YS, Zweckstetter M. Mars - Robust automatic backbone assignment of proteins. J Biomol NMR [Internet]. 2004 Sep [cited 2024 Feb 5];30(1):11–23. Available from: https://link.springer.com/article/10.1023/B:JNMR.0000042954.99056.ad

64. MARS - NMR Wiki [Internet]. [cited 2024 Feb 5]. Available from: http://nmrwiki.org/wiki/index.php?title=MARS

65. Shen Y, Delaglio F, Cornilescu G, Bax A. TALOS+: A hybrid method for predicting protein backbone torsion angles from NMR chemical shifts. J Biomol NMR [Internet]. 2009 [cited 2025 Feb 10];44(4):213. Available from: https://pmc.ncbi.nlm.nih.gov/articles/PMC2726990/

66. Macraild CA, Norton RS. RASP: rapid and robust backbone chemical shift assignments from protein structure.

67. NMRtist [Internet]. [cited 2025 Mar 3]. Available from: https://nmrtist.org/

68. Klukowski P, Riek R, Güntert P. Rapid protein assignments and structures from raw NMR spectra with the deep learning technique ARTINA. Nature Communications 2022 13:1 [Internet]. 2022 Oct 18 [cited 2025 Mar 3];13(1):1–12. Available from: https://www.nature.com/articles/s41467-022-33879-5

69. Pseudomonas Infection: Causes, Symptoms & Treatment [Internet]. [cited 2024 Feb 20]. Available from: https://my.clevelandclinic.org/health/diseases/25164-pseudomonas-infection

70. RCSB PDB - 6FU4: Ligand binding domain (LBD) of the p. aeruginosa histamine receptor TlpQ [Internet]. [cited 2024 Oct 8]. Available from: https://www.rcsb.org/structure/6fu4

71. pctA - Methyl-accepting chemotaxis protein PctA - Pseudomonas aeruginosa (strain ATCC 15692 / DSM 22644 / CIP 104116 / JCM 14847 / LMG 12228 / 1C / PRS 101 / PAO1) | UniProtKB | UniProt [Internet]. [cited 2024 Feb 20]. Available from: https://www.uniprot.org/uniprotkb/G3XD24/entry

72. BMRB Entry 27143 [Internet]. [cited 2024 Mar 18]. Available from: https://bmrb.io/data_library/summary/index.php?bmrbId=27143

73. Heyam A, Coupland CE, Dégut C, Haley RA, Baxter NJ, Jakob L, et al. Conserved asymmetry underpins homodimerization of Dicer-associated double-stranded RNA-binding proteins. Nucleic Acids Res [Internet]. 2017 Dec 12 [cited 2024 Mar 18];45(21):12577. Available from: /pmc/articles/PMC5716075/

74. Variables & Data Types - Python - Salford PsyTech Home →⌂ - Salford PsyTech Home →⌂ [Internet]. [cited 2024 Jul 23]. Available from: https://hub.salford.ac.uk/psytech/python/variables-data-types-python/

75. Python While Loop - GeeksforGeeks [Internet]. [cited 2024 Aug 28]. Available from: https://www.geeksforgeeks.org/python-while-loop/

76. Loops in Python - For, While and Nested Loops - GeeksforGeeks [Internet]. [cited 2024 May 14]. Available from: https://www.geeksforgeeks.org/loops-in-python/

77. Python For Loops [Internet]. [cited 2024 Aug 28]. Available from: https://www.w3schools.com/python/python_for_loops.asp

78. Python - Decision Making [Internet]. [cited 2024 May 14]. Available from: https://www.tutorialspoint.com/python/python_decision_making.htm

79. Python if, if...else Statement (With Examples) [Internet]. [cited 2024 Sep 18]. Available from: https://www.programiz.com/python-programming/if-elif-else

80. 9. Classes — Python 3.12.3 documentation [Internet]. [cited 2024 May 20]. Available from: https://docs.python.org/3/tutorial/classes.html

81. Python Classes [Internet]. [cited 2024 May 20]. Available from: https://www.w3schools.com/python/python_classes.asp

82. Python Program to Print Stack Trace - Scaler Topics [Internet]. [cited 2024 May 20]. Available from: https://www.scaler.com/topics/python-print-stack-trace/

83. Understanding Python error messages and stack traces | AppSignal APM [Internet]. [cited 2024 Jul 22]. Available from: https://www.appsignal.com/learning-center/understanding-python-error-messages

84. How To Fix Valueerror Exceptions In Python - GeeksforGeeks [Internet]. [cited 2024 Jul 23]. Available from: https://www.geeksforgeeks.org/how-to-fix-valueerror-exceptions-in-python/

85. Welford S, Cieplak T, Heinen M, Roth Chemplex GmbH B, Beilstein M, Hall SR.) Data Structure of the Beilstein Database, internal documentation, Linearly. In Software Developments in Chemistry J. Second International Meeting on Chemical Structures, Nordwijkhout [Internet]. 1991 [cited 2024 Oct 8];31(1):111–5. Available from: https://pubs.acs.org/sharingguidelines

86. NMR Exchange Format - a single, universal file format - CCPN [Internet]. [cited 2024 Mar 4]. Available from: https://ccpn.ac.uk/outreach/nmr-exchange-format/

87. Gutmanas A, Adams PD, Bardiaux B, Berman HM, Case DA, Fogh RH, et al. NMR Exchange Format: A unified and open standard for representation of NMR restraint

data. Nat Struct Mol Biol [Internet]. 2015 Jun 3 [cited 2024 Mar 4];22(6):433–4. Available from: https://ccpn.ac.uk/outreach/nmr-exchange-format/

88. NMR Exchange Format (NEF) [Internet]. [cited 2024 Mar 11]. Available from: https://www.ccpn.ac.uk/manual/v3/NEF.html

89. f-strings in Python - GeeksforGeeks [Internet]. [cited 2024 Mar 11]. Available from: https://www.geeksforgeeks.org/formatted-string-literals-f-strings-python/

90. Effective Python Testing With Pytest – Real Python [Internet]. [cited 2024 Mar 19]. Available from: https://realpython.com/pytest-python-testing/

91. Reyes-Darias JA, Yang Y, Sourjik V, Krell T. Correlation between signal input and output in PctA and PctB amino acid chemoreceptor of Pseudomonas aeruginosa. Mol Microbiol [Internet]. 2015 May 1 [cited 2025 Apr 28];96(3):513–25. Available from: https://onlinelibrary.wiley.com/doi/full/10.1111/mmi.12953

92. Briand L, Marcion G, Kriznik A, Heydel JM, Artur Y, Garrido C, et al. A self-inducible heterologous protein expression system in Escherichia coli OPEN. Nature Publishing Group [Internet]. 2016 [cited 2024 Mar 4]; Available from: www.nature.com/scientificreports/

93. Puthenveetil R, Vinogradova O. Solution NMR: A powerful tool for structural and functional studies of membrane proteins in reconstituted environments. J Biol Chem [Internet]. 2019 Nov 1 [cited 2025 Feb 27];294(44):15914. Available from: https://pmc.ncbi.nlm.nih.gov/articles/PMC6827292/

94. Time and Space complexity in Data Structure - Ultimate Guide [Internet]. [cited 2024 Aug 28]. Available from: https://www.simplilearn.com/tutorials/data-structure-tutorial/time-and-space-complexity

95. AlphaFold Protein Structure Database [Internet]. [cited 2024 Sep 3]. Available from: https://alphafold.ebi.ac.uk/faq