A Little Book of Bioinformatics

Leighton Pritchard

2023-01-01

Table of contents

Preface to *A Little Book of Bioinformatics*

Welcome to *A Little Book of Bioinformatics*. This is an online book, under continual development, which I am building as and when topics come to mind or prominence.

My goal is that this online book will come to be a fairly transparent and honest reference for students in bioinformatics, and maybe for some researchers, too.

I would be very grateful for feedback by email or through the GitHub repository Issues page

1 Introduction

This is a book created from markdown and executable code.

See Knuth (1984) for additional discussion of literate programming.

```
1 + 1
```

```
[1] 2
```

2 Summary

In summary, this book has no content whatsoever.

```
1 + 1
```

```
[1] 2
```

Part I

Representing Biological Data

This section of the book will introduce the computational representation of biological data.

Part II

Biological Databases

This section of the book will introduce biological databases.

Part III

Pairwise Sequence Alignment

This section of the book will introduce the concepts, algorithms, and application of pairwise sequence alignment.

Part IV

Multiple Sequence Alignment

This section of the book will introduce the concepts, algorithms, and application of multiple sequence alignment.

MAFFT

MAFFT is a very fast progressive alignment method, suitable for aligning large sequence sets, and tuneable for a range of input circumstances.

MAFFT stands for Multiple Alignment using Fast Fourier Transform, and was first described in 2002 (Katoh et al. 2002). Since then it has received multiple enhancements and extensions to improve scalability to larger sequence sets, and integration of other data such as protein structures (Katoh and Toh 2008a; Katoh and Standley 2013; Rozewicki et al. 2019; Katoh, Rozewicki, and Yamada 2019).

Where can I download MAFFT?

- https://MAFFT.cbrc.jp/alignment/software/

Where can I use MAFFT in a browser?

- https://mafft.cbrc.jp/alignment/server/
- https://www.ebi.ac.uk/Tools/msa/MAFFT/
- https://usegalaxy.eu/

How does MAFFT work?

Multiple workflows, tuned for different circumstances

MAFFT can be used as a "black box" alignment tool, but it offers several ways to combine its components. The various combinations can be chosen to suit your alignment task or available hardware, and fall into three main categories:

`FFT-NS1`, `FFT-NS-2`: progressive methods for large sequence sets

These approaches are simple progressive alignment workflows that take advantage of 6-tuple scoring to speed up initial distance matrix construction, and FFTs to speed up sequence and sequence group alignment.

1. A guide tree is constructed from 6-tuple similarity scores
2. A progressive alignment is performed on the guide tree (and the output alignment returned - `FFT-NS-1`)
3. The guide tree is reconstructed from this alignment (`FFT-NS-2`)
4. The sequences are realigned against the rebuilt guide tree (and the output alignment returned - `FFT-NS-2`)

`FFT-NS-2` is the default `MAFFT` algorithm, chosen to balance speed and accuracy (Katoh and Toh 2008a).

`FFT-NS-i`: progressive method for more accurate alignments of smaller sequence sets

This approach resembles that for `FFT-NS-2`, but iterates over guide tree reconstruction and sequence realignment until either no improvement can be detected in the alignment (by *Weighted Sum of Pairs*, WSP), or a maximum number of iterations is reached.

1. A guide tree is constructed from 6-tuple similarity scores
2. A progressive alignment is performed on the guide tree
3. The guide tree is reconstructed from this alignment
4. The sequences are realigned against the rebuilt guide tree
5. Steps 3 and 4 are repeated until there is no improvement in the alignment's WSP score, or a maximum number of iterations is reached

`L-INS-i`, `E-INS-i`, and `G-INS-i`: consistency scores for difficult alignment cases

These approaches resemble `FT-NS-i` in that they iterate over guide tree construction and realignment, but differ in that they carry out all-vs-all pairwise alignment initially, rather than the 6-tuple approximation, and employ a `COFFEE`-like *consistency score* alongside the WSP score, and different alignment algorithms during pairwise alignment.

- `E-INS-i` uses a global Needleman-Wunsch approach to pairwise alignment, and is suited for alignments between sequences that may have several alignable domains and a high proportion of non-alignable insertions.
- `L-INS-i` uses a local Smith-Waterman alignment to improve alignments where there is a single alignable domain, with non-alignable flanking sequence.
- `G-INS-i` uses a global Needleman-Wunsch alignment approach, and is suited to situations where it can be assumed that the entire sequence can be aligned for all inputs.

Pairwise sequence, or sequence group alignment

Fast Fourier Transform speed-up

MAFFT improves on the speed of simpler progressive alignment approaches (Feng and Doolittle 1987) by using efficient algorithms to target slow steps in the process. The main advance, which gives MAFFT its name, was to use the fast Fourier Transform (FFT) - an algorithm that converts a signal from one quantitative *domain*, such as time, to a representation in the *frequency domain* - to speed up alignment of protein sequences.

A key insight was to note that the frequency of amino acid substitutions is often determined by whether the substituting residue conserves important physicochemical properties (Katoh et al. 2002). Conservation of such properties is a feature of *neutral* or *near-neutral* substitutions. In the original paper, each amino acid (a) was assigned to a vector of two values, one representing the amino acid residue's volume ($v(a)$), and one representing its polarity ($p(a)$), normalised to zero mean and unit variance over all amino acids.

By converting the amino acid sequences to be aligned to a sequence of these vectors, MAFFT calculates the *correlation* between sequences in terms of their polarity and volume components. In doing this, MAFFT considers a *positional lag* of k sites in the sequence - a parameter controlling the size of sequence region being considered, and the overall correlation is reported for this value, as $c(k)$. By using a fast Fourier Transform for this operation, the CPU time for a set of N input sequences is reduced from $O(N^2)$ to $O(NlogN)$.

This concept can be extended from an alignment between two sequences to an alignment between two already-aligned groups of sequences, by replacing the vector of volume (or polarity) values for a single sequence with a vector that is the linear combination of the volume components for the aligned group (Katoh et al. 2002).

Finding homologous sequence segments

If a pair of sequences has homologous regions - conserving polarity and volume - then there will be a corresponding peak in the graph of $c(k)$. This identifies the *positional lag* of the match, but not its location. So, to determine the actual location of the match, a *sliding window analysis* (window size 30) is carried out to identify local homologies for the 20 highest peaks in $c(k)$. A score threshold of 0.7 per site is applied, and any window with a higher value is considered a *homologous segment*. Successive segments identified as *homologous segments* are combined, to a maximum of 150 sites per homolgous segment. Segments longer than this are divided into segments of maximum length 150 (Katoh et al. 2002).

Aligning sequence pairs

A matrix of homologous segments between two sequences is constructed, and the alignment obtained by the standard *dynamic programming* procedure (Katoh et al. 2002).

Application to nucleotide sequences

For nucleotide sequences, rather than using vectors that represent amino acid side chain properties, four-dimensional vectors of A, C, G, T are used at each column. Otherwise the process is identical (Katoh et al. 2002).

Scoring scheme

`MAFFT` departs from the common approach of using an all-positive-value scoring matrix, instead adopting a normalised similarity matrix. By default, this was originally derived from a 200 PAM log-odds matrix for both protein and nucleotide alignments, and a simplified gap penalty scheme was also employed (Katoh et al. 2002). More recently, the 200 PAM matrix has been replaced as default by BLOSUM62, and the gap penalty scheme was heavily revised (Katoh and Toh 2008a).

Guide tree building

The modified UPGMA tree building method used in the original version of `MAFFT` did not scale well to very large sequence sets, and was replaced in v6 onwards by the PartTree algorithm (Katoh and Toh 2008a, 2008b). This constructs an approximate tree from unaligned sequences by partitioning the dataset and is $O(NlogN)$ rather than $O(N^2)$, like UPGMA.

RNA and protein alignment incorporating structural information

In `MAFFT` v6, options were introduced to include structural information when aligning RNA sequences. This takes base pairing probability into account, and applies a novel four-way alignment consistency function (Katoh and Toh 2008a).

`MAFFT` v7 introduced the ability to inform an alignment by using the `ASH` structural alignment program and, in particular, the inter-residue distance between alpha carbons for amino acids in the alignment. This is likely to be most useful when aligning sequence-diverse, but structurally similar, proteins (Katoh and Standley 2013).

`MAFFT` has been integrated with the `DASH` structural alignment database, to use structural alignment information as a set of constraints on sequence alignment. This integration is available through the official `MAFFT` alignment server (Rozewicki et al. 2019).

Extension of existing alignments with new sequences

Further improvements were introduced in MAFFT v7 to enable addition of unaligned sequences into an existing multiple sequence alignment (Katoh and Standley 2013).

Parallelisation

Threading was added to MAFFT v7. Progressive alignment method outputs are identical in non-threaded and threaded runs, but iterative refinement-based alignments may generate different output when threaded (Katoh and Standley 2013).

Where can I find out more about MAFFT?

- MAFFT v7 tips
- MAFFT v7 algorithm

Part V

Biological Networks

This section of the book will introduce network representations in biology.

Part VI

Biological Structures

This section of the book will introduce structural data and visualisation.

PyMOL

PyMOL is a widely-used, open source molecular visualisation package, with paid-for ("incentive") support and updates.

Where can I download PyMOL?

The official home of PyMOL is at Schrodinger Software, where you can download the "Incentive" version. This requires a licence, which is free for educational use, but not free for academic/commercial/research use.

- https://pymol.org/2/

There is an open-source, freely-licensed version of PyMOL, also provided by Schrodinger Software. This can be downloaded from GitHub and self-compiled, or obtained through homebrew:

- https://github.com/schrodinger/pymol-open-source
- https://formulae.brew.sh/formula/pymol

Using PyMOL

Start the program from the application icon, or from the command-line (depending on which version you have installed), to obtain the landing screen (Figure 2.1).

PyMOL layout

Pymol is laid out approximately into four quadrants. Clockwise from top left in Figure 2.1 we have:

- The **interactive command window**. At the bottom of this window is the command prompt PyMOL >, where commands to control the visualisation can be typed and executed.
- The next quadrant (top right) contains a set of **function buttons** providing a range of actions for control of the visualisation.

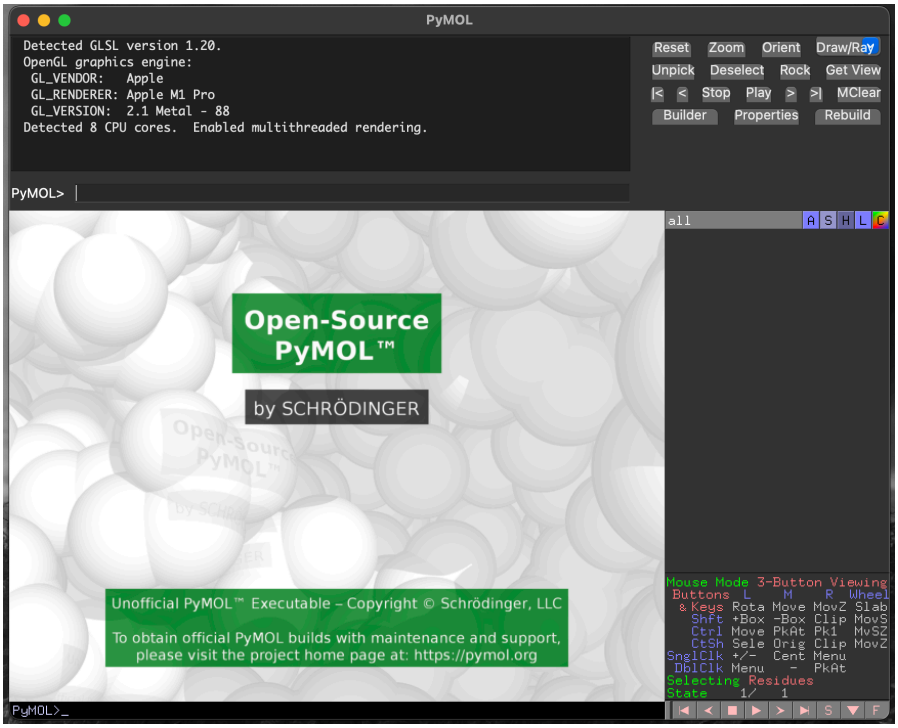

Figure 2.1: PyMOL landing screen, for the Open Source version, on macOS.

- Below this is the **object control panel**, which is the main point-and-click interface for changing the way that molecules in the visualisation window appear, and below this is the mouse control legend to explain what your mouse/trackpad actions will achieve.
- Finally, at bottom left, we have the **main viewer window**, where your molecule will be shown.

Obtaining structures

PyMOL can connect directly to RCSB/PDB to download structural data, and can load structure files from your local storage.

fetching structures from RCSB/PDB

We will download a structure directly from RCSB/PDB. To do this we will use the **interactive command window**. We also need to know the RCSB/PDB accession for the structure we want to view. Here, we will use the structure 4I61, a trimer of PduB, a bacterial microcompartment protein.

> **i** PduB makes biochemical factories inside cells.
>
> PduB is a protein that trimerises to form a structural unit that then combines with other similar structural units to enclose a volume within bacterial cells, to make a kind of "factory" for chemical reactions.
> The UniProt entry for the protein corresponding to this structure is F8DQ39

To fetch this structure from RCSB/PDB, we enter the command fetch 4I61 into the command prompt, and hit Return (Figure 2.2).

Figure 2.2: PyMOL command prompt including fetch command.

Executing this command will produce a short report to the interactive command window, and show a rendering of the structure in the **main viewer window** (Figure 2.3).

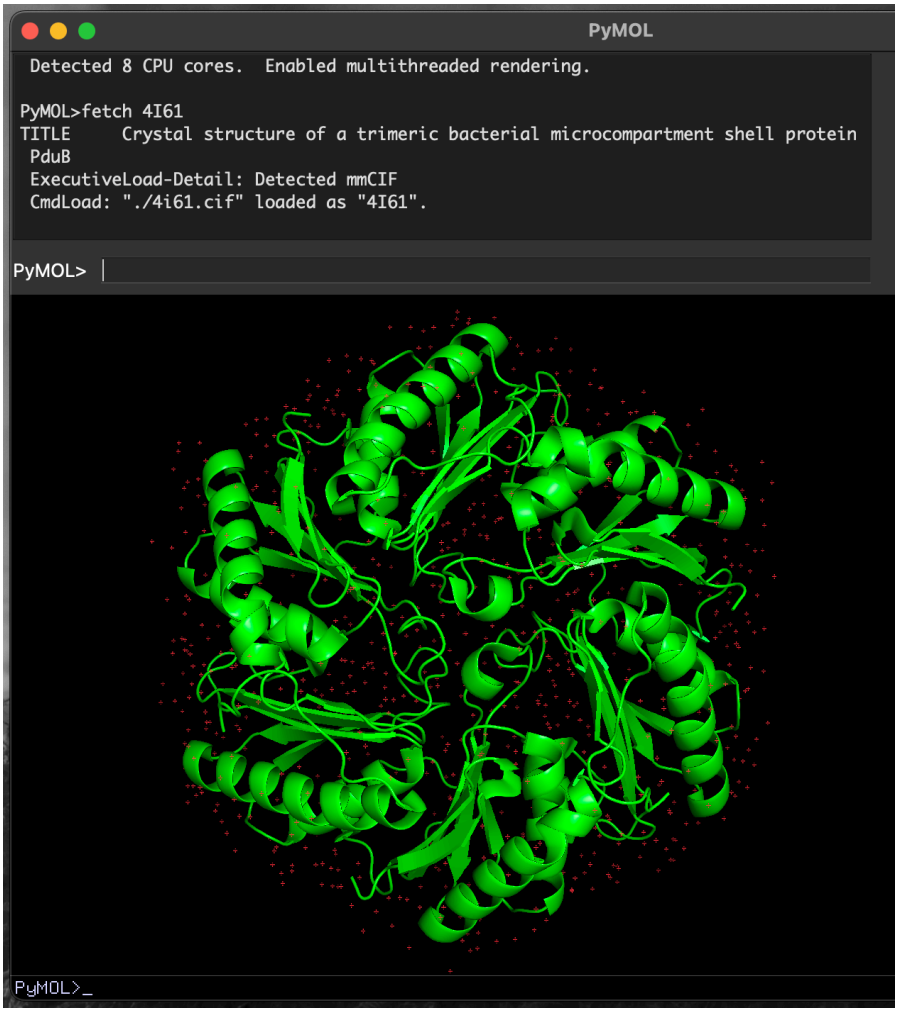

Figure 2.3: Initial render of `4I61` PduB structure in `PyMOL`.

PyMOL downloads the fetched file.

When executed, this command will also download the file `4I61.cif` to the current working directory. You can load this file into PyMOL without requiring a live network connection.

Changing the appearance of the structure

Rotating the molecule

By default, left-clicking on the molecule in the main viewer window and moving your mouse will rotate the molecule. You can use this to obtain a viewing position that helps you understand the structure better or that, when saved as a figure, will communicate your message to a reader.

Changing molecule colours

The **object control panel** provides buttons (`A`, `S`, `H`, `L`, `C`) that control aspects of the molecule's appearance:

- `A`/Action
- `S`/Show
- `H`/Hide
- `L`/Label
- `C`/Colour

There are three distinct protein chains in this trimeric structure. We will colour the protein differently by chain so that we can see them more clearly.

Important

- Click on the `C` button for `4I61` in the **object control panel**
- Click on `by chain` in the menu that appears
- Click on `by chain` in the new menu

This colours each chain in the structure differently, and also the water molecules (small dots) associated with each chain (Figure 2.4).

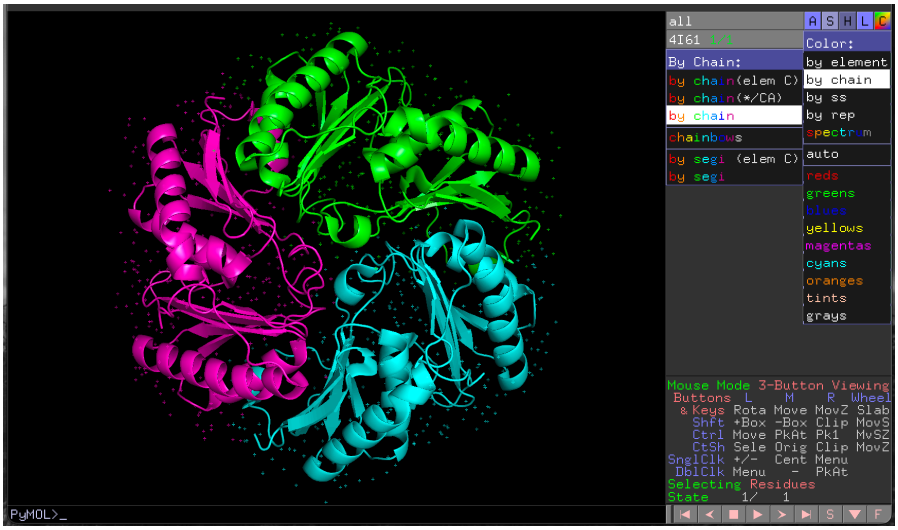

Figure 2.4: PyMOL rendering of 4I61 with a different colour for each chain.

Hiding elements of the structure

We sometimes want to focus attention on particular parts of a structure. To aid in this we can *hide* parts of the visualisation, using the H/Hide menu in the object control panel.

For example, to hide the water molecules that surround each chain, we would:

> **!** Important
>
> - Click on the H button for 4I61 in the object control panel
> - Click on `waters`

This removes the water molecules from our visualisation (Figure 2.5).

Selecting part of the structure

It is possible to select parts of the structure in PyMOL by pointing and clicking using the mouse. For complicated selections especially, this can be difficult, tedious, and error-prone. It is usually better to use the **interactive command window** to select structural components explicitly.

For example, to select only the first chain of the trimeric PduB structure (chain A), we would execute the command `indicate chain A`, which highlights that chain in the main visualisation window (Figure 2.6).

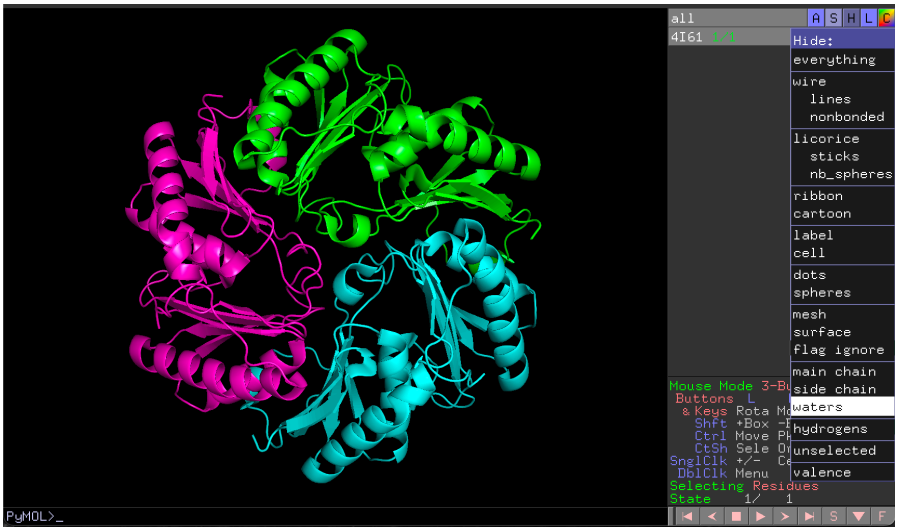

Figure 2.5: `4I61` structure with waters removed.

See that using `indicate` has produced a new row in the **object control panel** called `(indicate)`. This allows us to control the appearance of the selected element.

With the chain selected, we can then change the way it is drawn using the `S`/Show menu for the `(indicate)` selection, and the colours used for rendering, using the corresponding `C`/Colour menu (Figure 2.7, Figure 2.8).

> **!** Important
>
> Click on the `(indicate)` label in the object control panel to cancel the selection.

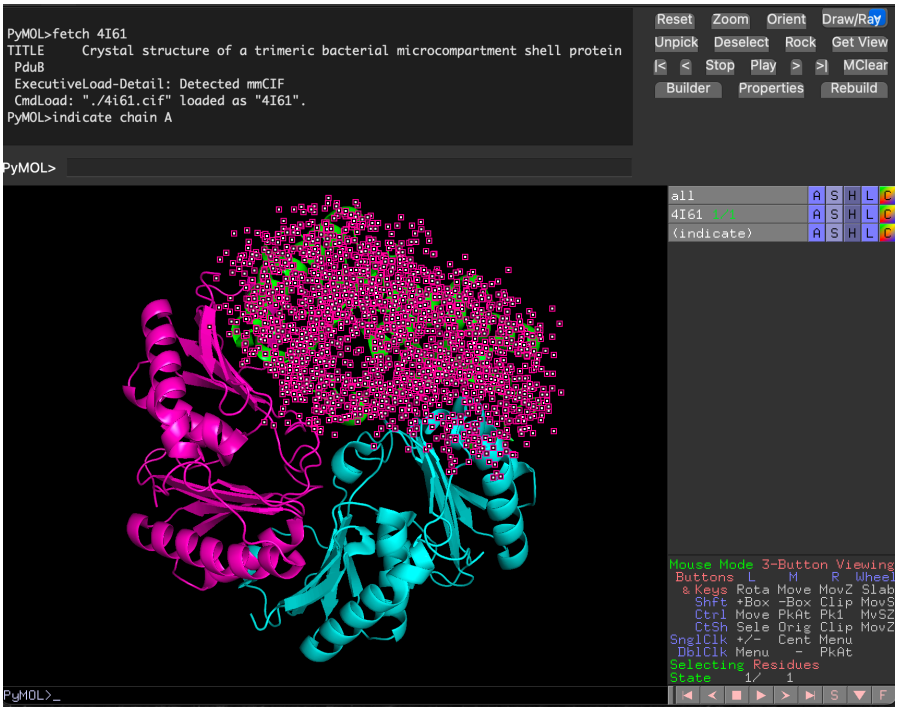

Figure 2.6: `4I61` structure with chain A highlighted.

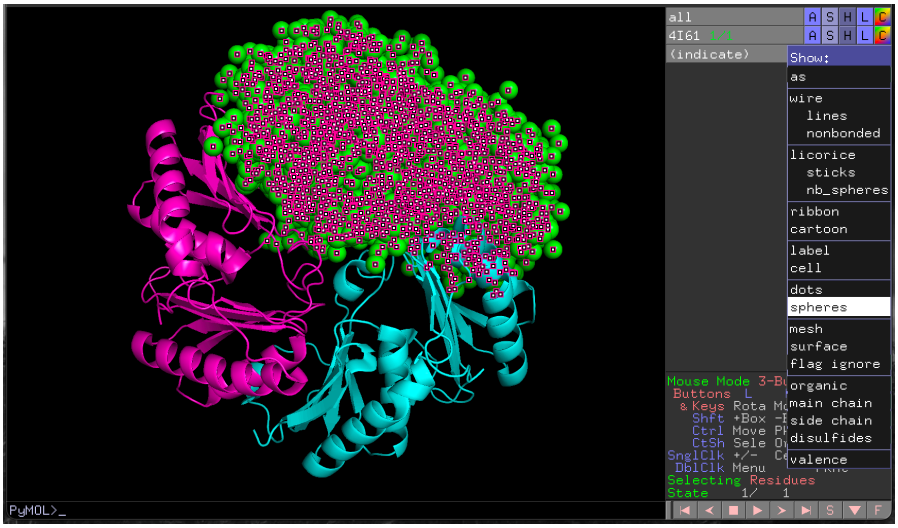

Figure 2.7: Rendering `4I61` chain A structure as spheres.

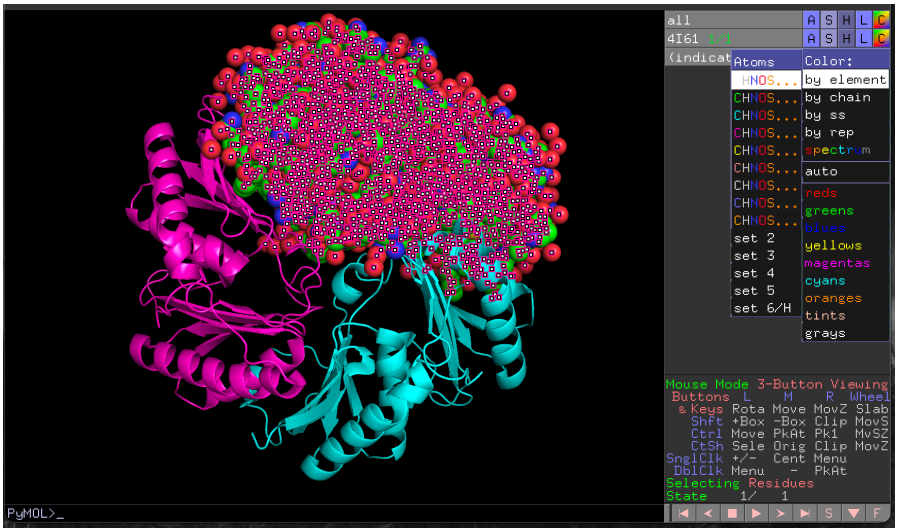

Figure 2.8: Rendering `4I61` chain A structure with atom colouring.

ChimeraX

ChimeraX is a widely-used, open source software package for macromolecular visualisation and analysis. It is free for academic, government, nonprofit, and personal users.

Where can I download ChimeraX?

ChimeraX is available from the UC San Francisco website.

- https://www.cgl.ucsf.edu/chimerax/download.html

Part VII

Signatures of Evolution

This section of the book will introduce phylogenetics and related approaches.

References

Feng, Da-Fei, and Russell F Doolittle. 1987. "Progressive Sequence Alignment as a Prerequisite to Correct Phylogenetic Trees." *J. Mol. Evol.* 25 (4): 351–60. https://doi.org/10.1007/BF02603120.

Katoh, Kazutaka, Kazuharu Misawa, Kei-Ichi Kuma, and Takashi Miyata. 2002. "MAFFT: A Novel Method for Rapid Multiple Sequence Alignment Based on Fast Fourier Transform." *Nucleic Acids Res.* 30 (14): 3059–66. https://doi.org/10.1093/nar/gkf436.

Katoh, Kazutaka, John Rozewicki, and Kazunori D Yamada. 2019. "MAFFT Online Service: Multiple Sequence Alignment, Interactive Sequence Choice and Visualization." *Brief. Bioinform.* 20 (4): 1160–66. https://doi.org/10.1093/bib/bbx108.

Katoh, Kazutaka, and Daron M Standley. 2013. "MAFFT Multiple Sequence Alignment Software Version 7: Improvements in Performance and Usability." *Mol. Biol. Evol.* 30 (4): 772–80. https://doi.org/10.1093/molbev/mst010.

Katoh, Kazutaka, and Hiroyuki Toh. 2008a. "Recent Developments in the MAFFT Multiple Sequence Alignment Program." *Brief. Bioinform.* 9 (4): 286–98. https://doi.org/10.1093/bib/bbn013.

———. 2008b. "Recent Developments in the MAFFT Multiple Sequence Alignment Program." *Brief. Bioinform.* 9 (4): 286–98. https://doi.org/10.1093/bib/bbn013.

Knuth, Donald E. 1984. "Literate Programming." *Comput. J.* 27 (2): 97–111. https://doi.org/10.1093/comjnl/27.2.97.

Rozewicki, John, Songling Li, Karlou Mar Amada, Daron M Standley, and Kazutaka Katoh. 2019. "MAFFT-DASH: Integrated Protein Sequence and Structural Alignment." *Nucleic Acids Res.* 47 (W1): W5–10. https://doi.org/10.1093/nar/gkz342.