pyGLAS version 1 documentation

Naushad AL Velgy

Prof Philip C Biggin

Department of Biochemistry University of Oxford

December 8, 2016

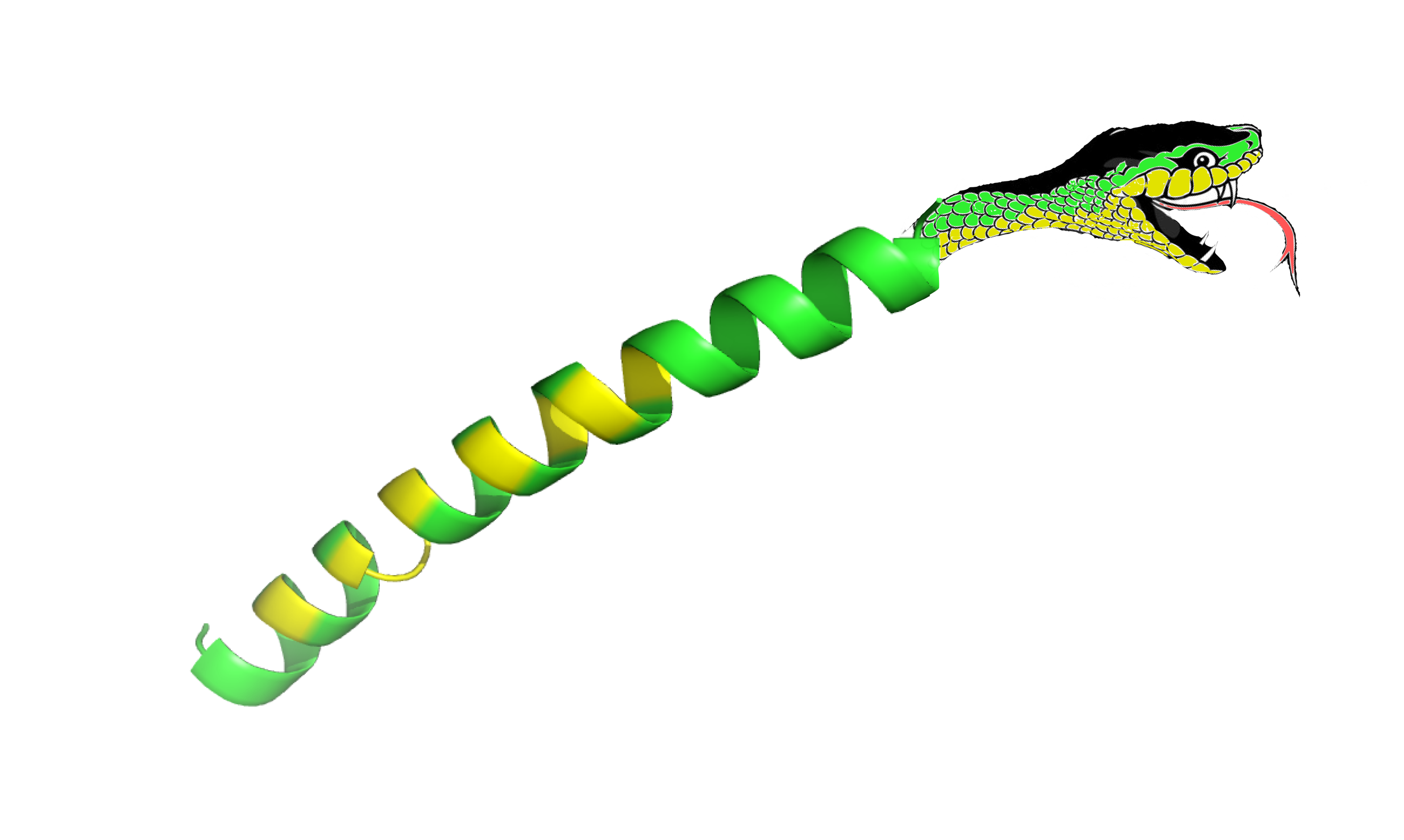


Figure 1: pyGLAS logo

**Abstract**

Here we present pyGLAS, a software designed to calculate the GPCR Likeness Assessment Score (GLAS) of a GPCR. We make use of common Python packages, such as MDAnalysis, for fast calculation of specific contacts found in crystallographically determined structures of GPCRs. pyGLAS allows for calculation of GLAS for both static snapshots (aiding in modelling of GPCRs using modelling software), or for trajectories generated from molecular dynamics (to assess the stability of GPCRs in different environments).

**Introduction**

GPCRs (G Protein Coupled Receptors) are cell surface receptors responsible for transmitting signals across the membrane, from the extracellular matrix to the intracellular milieu. As such, they mediate a variety of processes, including nociception1. So important are these receptors that approximately 50% of all drugs in the market target GPCRs2.

To better understand the nature of GPCRs, it is important to correctly model the canonical 7 transmembrane architecture, as well as the orientation of residues in critical structural motifs. Software packages, such as Modeller3 and Medeller4, are commonly used to create homology models of GPCRs prior to more extensive studies into receptor-ligand interactions, allosteric modulation, free energy calculations, activation mechanism, among others.

However, excellent though these may be, Modeller and other software packages yield generic scoring functions, such as the DOPE score5, which attempt to distinguish good models among those generated. Further steps can be taken by evaluating models using the QMEAN server6; however, as with the DOPE, this score is somewhat generic. It may be important, therefore, to include in our studies of GPCRs a score that is specific to the canonical structure of receptors, and that can be used to better qualify good models from poor ones.

A thorough analysis of structures deposited in the Protein Data Bank, in 2013, yielded a list of 24 interhelical contacts that are present in all activity states7. A later revision of structures, in 2016, reinforced the presence of these in crystal structures where receptors existed in active states (i.e. bound to G proteins/β-arrestins), as well as in receptors in an inactive state (i.e. incapable of intracellular binding)2.

These were later termed CHICOs, or Conserved Inter Helical COntacts3. The presence of these contacts was first exploited by Heifetz and colleagues8 in exploring agonist binding to the human Orexin-1 and Orexin-2 receptors, where the scoring variable GLAS (GPCR Likeness Assessment Score) was first introduced.

Here we introduce the first (to our knowledge) open source script to calcu- late the variable GLAS. We termed it pyGLAS, as it is written, and exploits well known libraries in the Python programming language. These include MDAnalysis9, BioPython10, Numpy11, Pandas12 and GGPlot.

**Input**

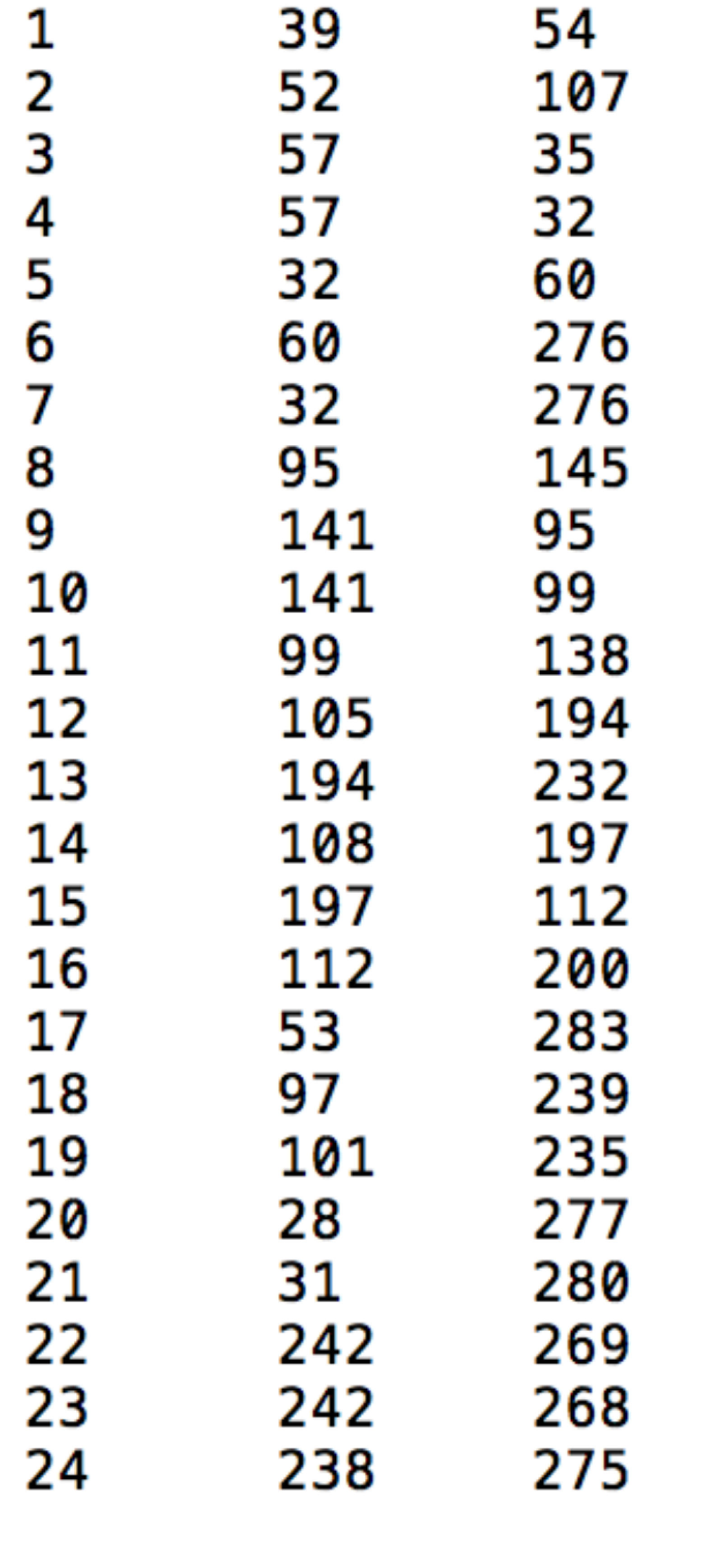
pyGLAS takes the following input:

Figure 2: Sample pyGLAS input. The first column refers to the pair number and the remaining column refer to the corresponding residues. Please consult reference 2 for details.

* A topology file (in the PDB or GRO formats; compulsory). Input using the “-r” flag.
* The type of output required, triggered by the “--format” flag. The default output is “short”; see Output type for more on this.
* A trajectory file (in the XTC format; op- tional). Input using the “-x” flag.
* An option to provide a list of CHICOs and NACHOs using the “--chicos” and/or “--nachos” flags, respectively. Refer to Figure 2 for an example input CHICOs file. Please be aware that, if neither flag is triggered, pyGLAS will attempt to find the CHICOs and the NACHOs from the input file. This is done using the output from the GPCR database (reference). See Appendix 1 for more on this.
* Alternatively, “--no chicos” stops pyGLAS from calculating and outputting the CHICOs. “--no nachos” is the same for the NACHOs.

**Output type**

pyGLAS was designed with one philosophy in mind: simplicity of output. As such, there are three output types that can be specified using the “--format” flag.

**“plain”**

The “plain” output was created with the goal of being used with Modeller. It will produce a text file with the GLAS of the protein and exit. The output filename is based on the input filename, such that, if your input filename is “Receptor.pdb”, the output will be “Receptor.txt”.

If the GLAS of “Receptor.pdb” is 19,

$ cat Receptor.txt

will output the number 19 to the terminal.

A quick for loop using all generated models can then be used to compare GPCR Likeness Assessment Scores. For example, in a directory with the Modeller output:

$ forf in \*.pdb; do echo -e $f "\t" $(cat $(basename $f ‘.pdb’)\_GLAS.txt) >> \ GLAS\_comparison.tsv; done

The file “GLAS\_comparison.tsv” should now have a list of all models generated and their respective GLAS, separated by a tab.

**“short”**

This is the default output. There are 2 streams of output based on whether a trajectory is part of the input.

**“short”; No Trajectory**

If the input is a single topology, pyGLAS will calculate 2 scores: the GLAS, based on the aforementioned CHICOs, and the SAS, based on the NACHOs (see Introduction). pyGLAS will output 2 files based on the input receptor filename: “filename\_GLAS.short” and “filename\_SAS.short”. Figure 3 shows the output for a receptor with a GLAS of 19.

Note that this output format does not support trajectories (please see below for trajectory based output).

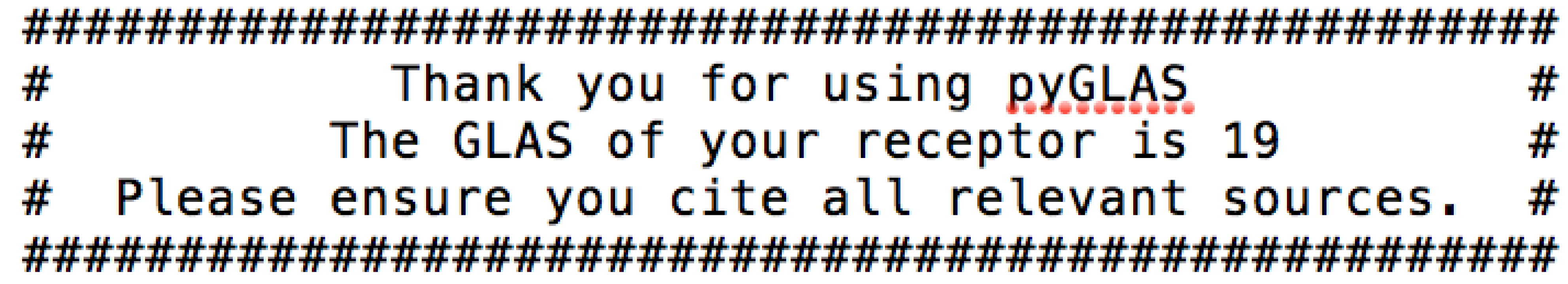


Figure 3: Sample pyGLAS output. The relevant sources can be found in pyGLAS/Documentation/references.txt.

**“short”: Trajectory**

If the input includes trajectory data, the output will consist of 2 plots: “filename\_GLAS\_overTime.svg” and “filename\_SAS\_overTime.svg” (see Figure 4 for an example of the former). These will track the change in the GLAS and SAS over the trajectory, and are stored in SVG files (which can be viewed using software packages such as Inkscape, on Linux platforms, or Gapplin, on Mac platforms).

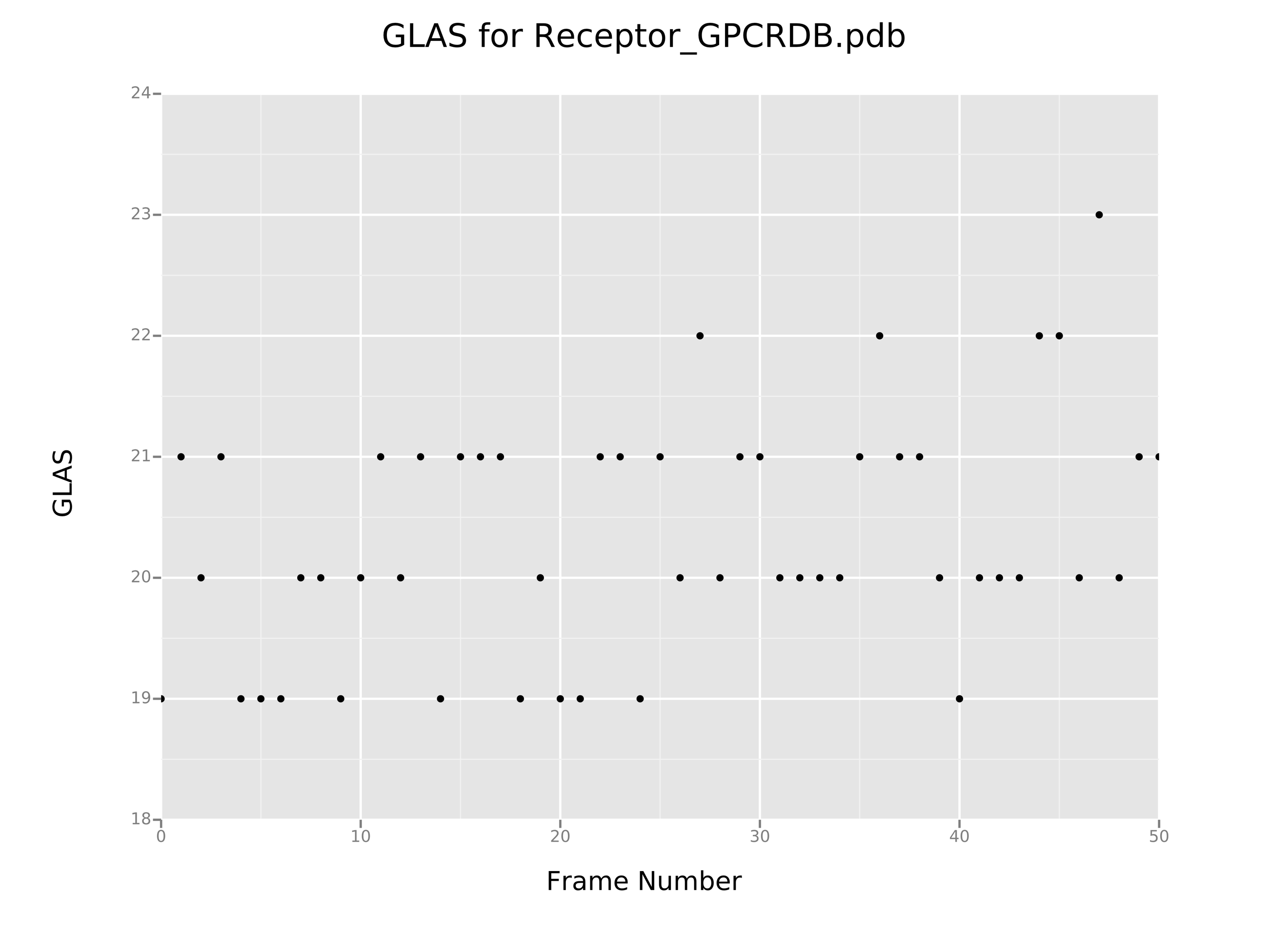


Figure 4: Variation of GLAS over simulation frame. Note that pyGLAS defines the vertical axis to be between one point below lowest GLAS and 24 (so, if the lowest GLAS is 19, the vertical axis will range from 18 to 24). All plots generated by pyGLAS are saved as SVG files.

**“long”**

Triggering the “–format long” flag will change the output. Again, 2 streams of output will be generated based on whether trajectory data is present.

**“long”: No Trajectory**

This output is like the “short” output (see Figure 5). However, it will also write to the output files the minimum distances between the residue pairs (in Ångstrom).

**“long”: Trajectory**

Using the python library ggplot, pyGLAS will output the same graphs pro- duced using the “short” option, and will additionally plot all minimum dis- tances between residue pairs over time. This consists of a 4x6 canvas for GLAS residue pairs, and a 3x5 canvas for the SAS residue pairs.

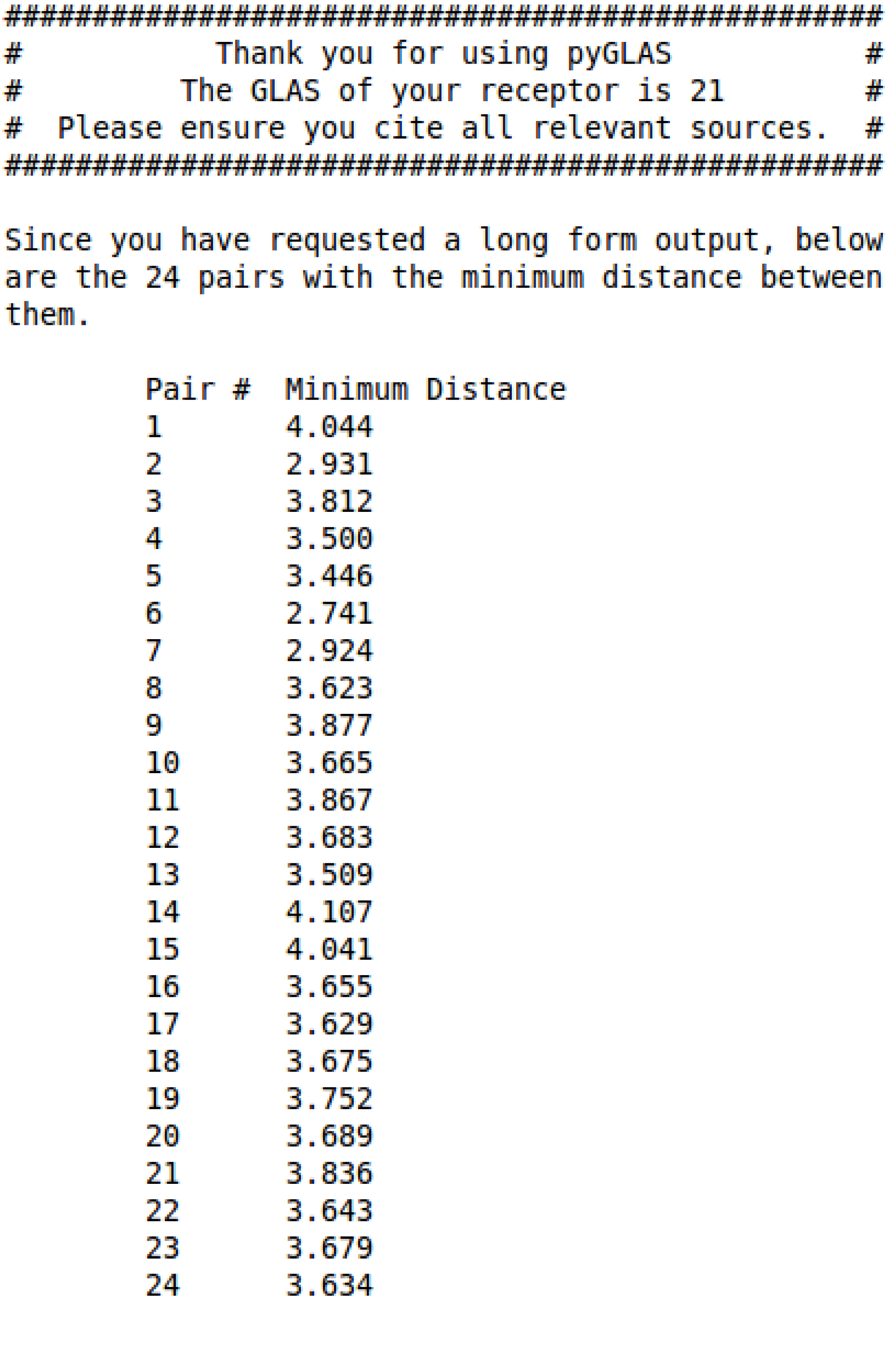


Figure 5: Sample pyGLAS output. The first column refers to the pair number and the second to the minimum distance between the residues.

**Tutorials**

There are several directories provided with the github download: Documentation (where we are), scripts (where pyGLAS is), examples/Single Topology, examples/1\_Trajectory, examples/Modeller, examples/Test Cases and examples/High Temp. The latter is divided into 400K and 500K.

Each example directory has an iPython Notebook (now Jupyter Notebook) file that serves as a tutorial for using that specific branch of pyGLAS, and how each flag works.

**References**

1 - Fenalti, G., Giguere, P. M., Katritch, V., Huang, X.-P., Thompson, A. A., Cherezov, V., Roth, B. L., and Stevens, R. C. Molecular control of $$-opioid receptor signalling. Nature 506, 7487 (2014), 191–196.

2 - Cvicek, V., Goddard, W. A., and Abrol, R. Structure-Based Sequence Alignment of the Transmembrane Domains of All Human GPCRs: Phylogenetic, Structural and Functional Implications. PLoS Computational Biology (2016).

3 - Sali, A., and Blundell, T. Comparative Protein Modelling by Sat- isfaction of Spatial Restraints. J Mol Bio 234 (1993), 779–815.

4 - Ebejer, J. P., Hill, J. R., Kelm, S., Shi, J., and Deane, C. M. Memoir: template-based structure prediction for membrane proteins. Nucleic acids research 41, Web Server issue (2013), W379–W383.

5 - Shen, M.-Y., and Sali, A. Statistical potential for assessment and prediction of protein structures. Protein Science 15, 11 (2006), 2507– 2524.

6 - Benkert, P., Tosatto, S. C. E., and Schomburg, D. QMEAN: A comprehensive scoring function for model quality assessment. Proteins: Structure, Function, and Bioinformatics 71, 1 (2008), 261–277.

7 - Venkatakrishnan, A. J., Deupi, X., Lebon, G., Tate, C. G., Schertler, G. F., and Babu, M. M. Molecular signatures of G- protein-coupled receptors. Nature 494, 7436 (2013), 185–194.

8 - Heifetz, A., Barker, O., Morris, G. B., Law, R. J., Slack, M., and Biggin, P. C. Toward an Understanding of Agonist Binding to Human Orexin1 and Orexin2 Receptors with GProtein-Coupled Re- ceptor Modeling and Site-Directed Mutagenesis. Biochem 52, 46 (2013), 8246–826.

9 - Michaud-Agrawal, N., Denning, E. J., Woolf, T. B., and Beckstein, O. MDAnalysis: A Toolkit for the Analysis of Molecu- lar Dynamics Simulations. J Comp Chem 32 (2011), 2319–2327.

10 - Cock, P. J. A., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B., and De Hoon, M. J. L. Biopython: Freely available Python tools for computational molecular biology and bioin- formatics. Bioinformatics 25, 11 (2009), 1422–1423.

11 - Walt, S., Colbert, S. C., and Varoquaux, G. The NumPy Array: A Structure for Efficient Numerical Computation. Comp Sci Eng (2011), 22–30.

12 - McKinney, W. Data Structures for Statistical Computing in Python. Proceedings of the 9th Python in Science Conference 1697900, Scipy (2010), 51–56.