pyGLAS version 1 documentation

Naushad AL Velgy
Prof Philip C Biggin
Department of Biochemistry
University of Oxford

December 8, 2016

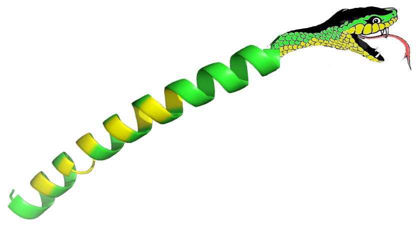


Figure 1: pyGLAS logo

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).

1 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 nociception[5]. So important are these receptors that approximately 50% of all drugs in the market target GPCRs[3].

In order 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 Modeller[9] and Medeller[4], 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.

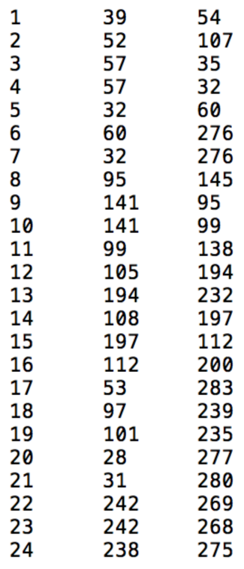
However, excellent though these may be, Modeller and other software packages yield generic scoring functions, such as the DOPE score[10], which attempt to distinguish good models among those generated. Further steps can be taken by evaluating models using the QMEAN server[1]; however, as with the DOPE, 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 states[11]. 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)[3].

These were later termed CHICOs, or Conserved Inter Helical COntacts[3]. The presence of these contacts was first exploited by Heifetz and colleagues[6] 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 calculate the variable GLAS. We termed it pyGLAS, as it is written, and exploits well known libraries in the Python programming language. These include MDAnalysis[8], Biopython[2], Numpy[12], Pandas[7] and GGPlot.

2 Input



```
1       39      54
2       52      107
3       57      35
4       57      32
5       32      60
6       60      276
7       32      276
8       95      145
9       141     95
10      141     99
11      99      138
12      105     194
13      194     232
14      108     197
15      197     112
16      112     200
17      53      283
18      97      239
19      101     235
20      28      277
21      31      280
22      242     269
23      242     268
24      238     275
```

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

pyGLAS takes the following input:

- 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; optional). 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.

3 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.

3.1 ”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,

Listing 1: Some Bash

```
$ 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:

Listing 2: Sample Bash loop for GLAS comparison

```
$ for f in *.pdb; echo $f "\t" \
$(cat $(basename $f '.pdb').txt) \
>> GLAS_comparison.tsv
```

The file ”GLAS_comparison.txt” should now have a list of all models generated and their respective GLAS, separated by a tab.

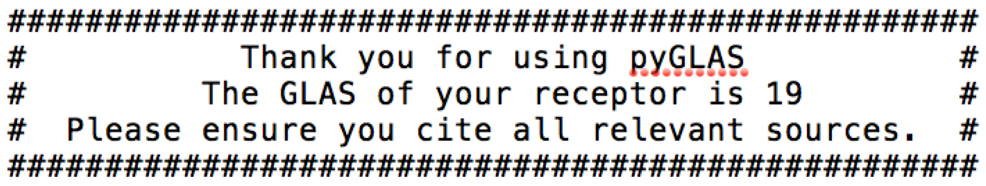
3.2 ”short”

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

3.2.1 ”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 "short" and "long" for trajectory based output).



```
##################################################
#              Thank you for using pyGLAS            #
#            The GLAS of your receptor is 19         #
#   Please ensure you cite all relevant sources.     #
##################################################
```

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

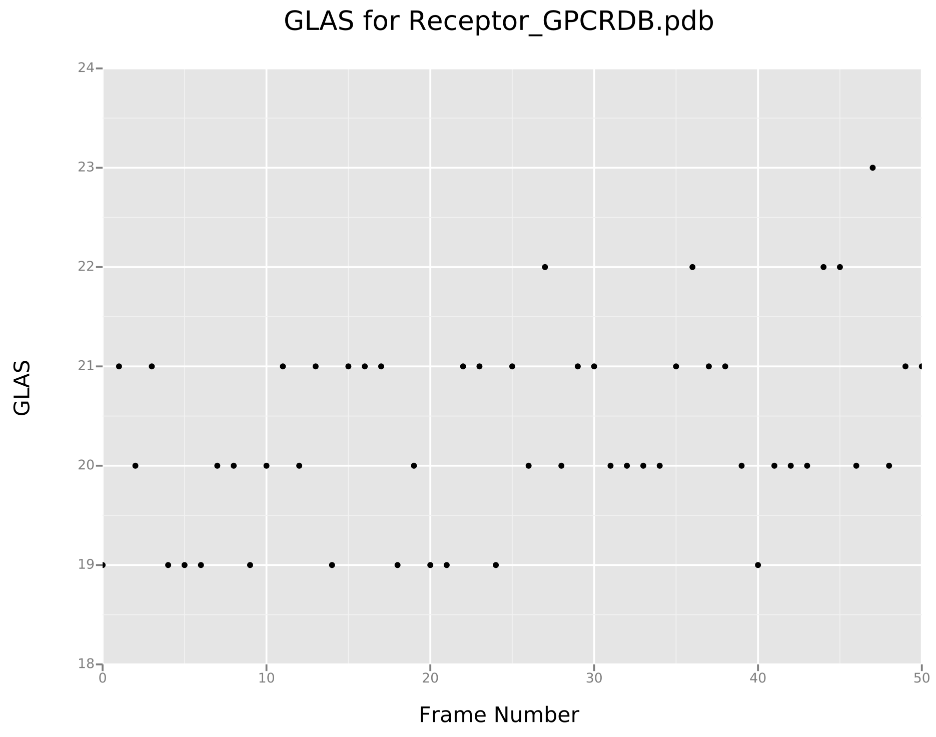
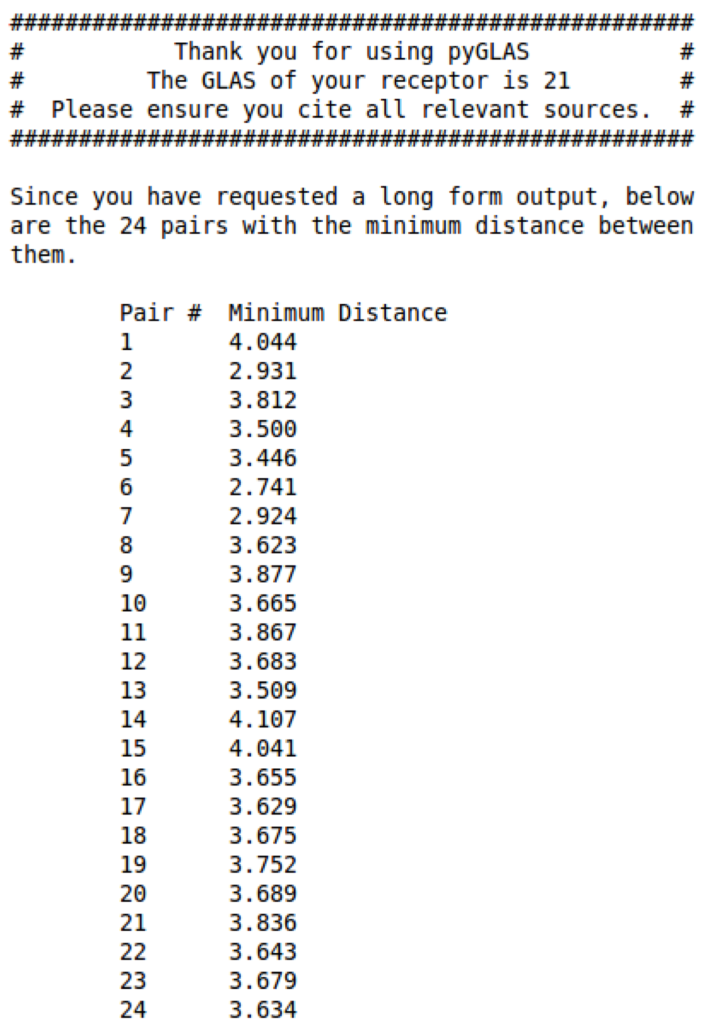


Figure 4: Variation of GLAS over simulation frame. Note that pyGLAS defines the vertical axis to be between one point below lowest GLAS and 24 (so, if the lowest GLAS is 19, the vertical axis will range from 18 to 24). All plots generated by pyGLAS are saved as SVG files.



```
################################################
#          Thank you for using pyGLAS          #
#        The GLAS of your receptor is 21       #
#  Please ensure you cite all relevant sources.  #
################################################

Since you have requested a long form output, below
are the 24 pairs with the minimum distance between
them.

        Pair #  Minimum Distance
        1       4.044
        2       2.931
        3       3.812
        4       3.500
        5       3.446
        6       2.741
        7       2.924
        8       3.623
        9       3.877
        10      3.665
        11      3.867
        12      3.683
        13      3.509
        14      4.107
        15      4.041
        16      3.655
        17      3.629
        18      3.675
        19      3.752
        20      3.689
        21      3.836
        22      3.643
        23      3.679
        24      3.634
```

Figure 5: Sample pyGLAS input. The first column refers to the pair number and the remaining column refer to the corresponding residues. Please consult [Cvicek] for details.

3.2.2 "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).

3.3 "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.

3.3.1 "long": No Trajectory

This output is similar to the "short" output (see FIgure 5). However, it will also write to the output files the minimum distances between the residue pairs (in Angstrom).

3.3.2 "long": Trajectory

Using the python library ggplot, pyGLAS will output the same graphs produced using the "short" option, and will additionally plot all minimum distances between residue pairs over time. This consists of a 4x6 canvas for GLAS residue pairs, and a 3x5 canvas for the SAS residue pairs.

4 Test cases

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] 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.

[2] 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 bioinformatics. *Bioinformatics 25*, 11 (2009), 1422–1423.

[3] 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).

[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] 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.

[6] 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 Receptor Modeling and Site-Directed Mutagenesis. *Biochem 52*, 46 (2013), 8246–826.

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

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

[9] Sali, A., and Blundell, T. Comparative Protein Modelling by Satisfaction of Spatial Restraints. *J Mol Bio 234* (1993), 779–815.

[10] Shen, M.-y., and Sali, A. Statistical potential for assessment and prediction of protein structures. *Protein Science 15*, 11 (2006), 2507–2524.

[11] 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.

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