

pygen-structures: A Python package to generate 3D molecular structures for simulations using the CHARMM forcefield

Travis Hesketh¹

¹ University of Strathclyde, Department of Pure and Applied Chemistry

DOI: [10.21105/joss.02157](https://doi.org/10.21105/joss.02157)

Software

- [Review](#) ↗
- [Repository](#) ↗
- [Archive](#) ↗

Editor: Richard Gowers ↗

Reviewers:

- [@dotsdl](#)
- [@rmeli](#)
- [@amandadumi](#)

Submitted: 29 February 2020

Published: 13 April 2020

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC-BY](#)).

Statement of Need

Forcefields used in molecular dynamics (MD) or Monte Carlo simulations (such as the CHARMM forcefield, Huang & MacKerell (2013)) require a great deal of information in order to ensure that the correct harmonic force constants are chosen for particular bonds or angles. As such, protein structure files (PSF files), containing information about connectivity and relevant parameters, are often used by simulation packages to contain this information. These files are typically generated using *psfgen*, (Ribeiro et al., 2019) a Tcl package bundled with *VMD* (Humphrey, Dalke, & Schulten, 1996) which requires a fully annotated Protein Data Bank (PDB) format file (wwPDB, 2012) with correct names for atoms and residues, as defined in the forcefield. This annotation is challenging to do in an automated fashion. For structures from the Protein Data Bank itself, this is expected to some degree, as experimentally resolved structures are often missing the coordinates of certain residues (Brünger & Nilges, 1993) and can contain bound ligands which may not be parameterised in the forcefield. At present, the normal solution to this problem is to download a structure from the database, manually fix residue and atom names, and generate a PSF file using *psfgen*. In other words, obtaining a 3D structure, annotating the structure and generating an input file in the correct format for a simulation package are treated as separate tasks.

For simulations involving combinatorial searches of smaller sequences, however, the level of human intervention required can be far greater. There may be no experimentally determined structures for a given sequence, so these must be generated. Existing structure generation tools must be applicable to more general chemistry than molecular dynamics forcefields, and as a result are far less specific about the identities of particular atoms. As such, these packages often leave residue and atom names labelled as unknowns or element symbols. Additionally, if structures are generated based only on connectivity or drawn by the user, it can be challenging to ensure that tetrahedral chirality (the *handedness* of carbon centres with four different substituents) is properly respected. A structure generation method for small molecules which is aware of the forcefield is necessary in order to automate this process. The workflow which *pygen-structures* hopes to automate is shown in Figure 1. As with all structure generation methods, applicability is limited to cases where secondary structure is unimportant. Generation of 3D coordinates is currently limited to structures with around 15 protein residues.

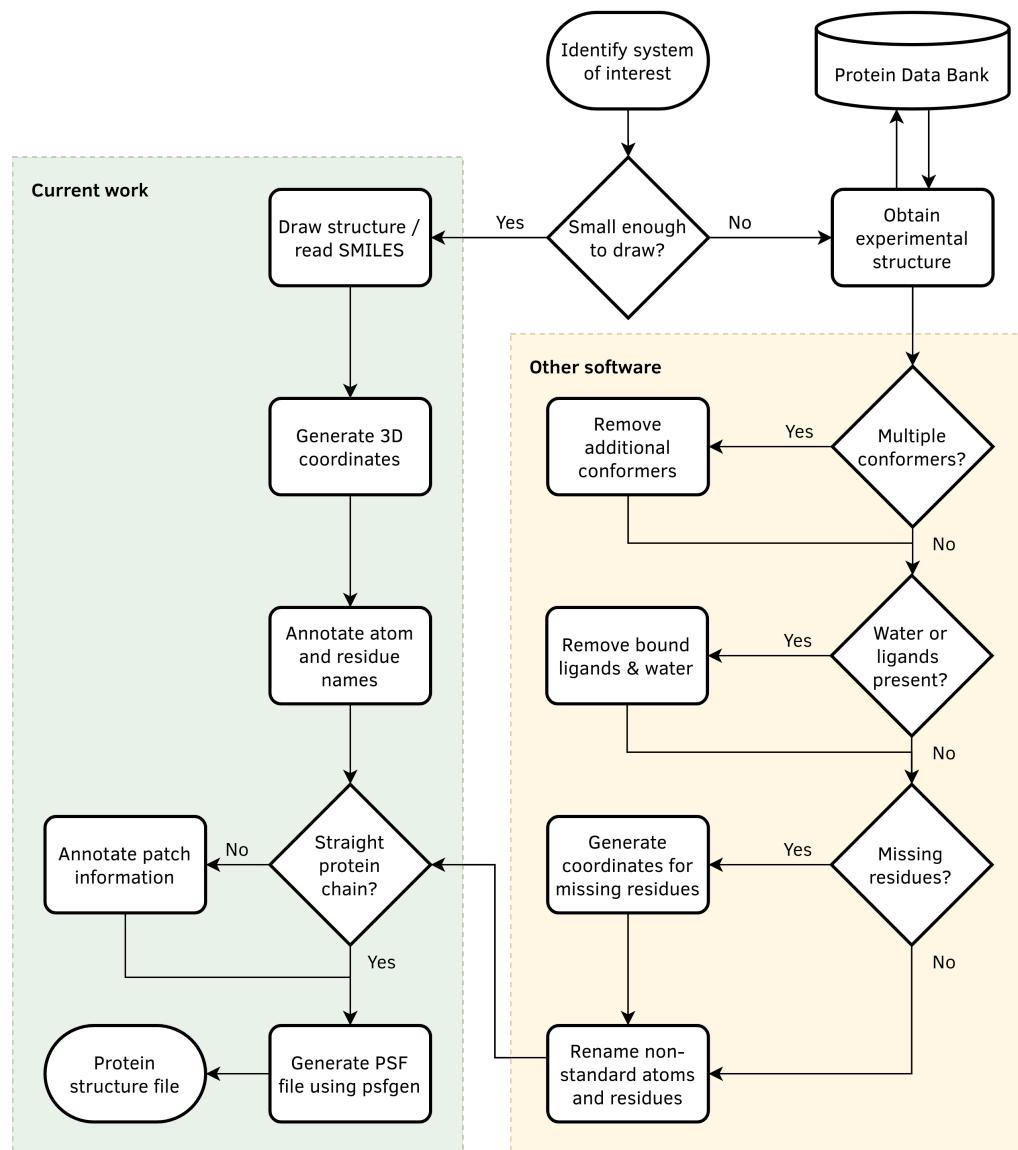


Figure 1: An example workflow used to prepare structures for molecular dynamics or Monte Carlo simulation. For smaller polymeric structures, `pygen-structures` can be used to prevent the need for significant manual intervention.

Summary

`pygen-structures` is an open source (3-clause BSD license) Python library to generate 3 dimensional structures for molecules from the CHARMM forcefield. Coordinates are generated using an embedding method based on empirical data (Riniker & Landrum (2015), as implemented in the *RDKit*, The RDKit Contributors (n.d.)), and are written out as standard PSF and PDB files. The package contains convenience functions for generating molecules from a collection of CHARMM residues and patches, or from one letter amino acid codes (usable by calling `pygen-structures` as a command line application) and a series of classes and functions for representing and manipulating CHARMM data. The chirality of tetrahedral centres is set using internal coordinate data from the residue topology file.

Classes provided by `pygen-structures` include representations of CHARMM residue topol-

ogy files, which contain information about residues, their atoms and their connectivity; and parameter files, containing the relevant force constants involving sets of atoms. There are also classes which describe residues and patches from topology files, and representations of atoms and molecules. Residues from the forcefield (and molecules created from those residues) can be represented as RDKit Mols, enabling pattern matching of residues to molecules.

Demonstrating the Benefits of pygen-structures

To illustrate the current necessity of human involvement in annotation of small-molecule structures, it is useful to examine structures which are created by other software packages. The following is a PDB file generated by drawing the structure of phenylalanine in *MarvinSketch* 20.3 and cleaning the structure in 3D. This is representative of many chemical drawing packages, which aim to work with as many areas of general chemistry as possible. Similar behaviour is observed in PerkinElmer's ChemOffice suite (version 19), (PerkinElmer Informatics, 2019) which does not write out residue names upon PDB export. Avogadro 1.2.0 (Hanwell et al., 2012) and PyMol 2.3.0 (Schrödinger LLC, 2019) cope better with biological molecules, and annotate the residue and atom names for protein residues correctly according to the PDB standard, but fail to differentiate between D and L isomers.

HEADER	PROTEIN								25-FEB-20	NONE	
TITLE	NULL										
COMPND	NULL										
SOURCE	NULL										
KEYWDS	NULL										
EXPDTA	NULL										
AUTHOR	Marvin										
REVDAT	1	25-FEB-20		0							
HETATM	1	C	UNK	0	1.322	4.682	0.738	0.00	0.00		\
...		C+0									
HETATM	2	C	UNK	0	-0.092	4.630	0.732	0.00	0.00		\
...		C+0									
HETATM	3	C	UNK	0	-0.852	5.626	1.374	0.00	0.00		\
...		C+0									
HETATM	4	C	UNK	0	-0.210	6.682	2.043	0.00	0.00		\
...		C+0									
HETATM	5	C	UNK	0	1.194	6.742	2.065	0.00	0.00		\
...		C+0									
HETATM	6	C	UNK	0	1.956	5.751	1.415	0.00	0.00		\
...		C+0									
HETATM	7	C	UNK	0	2.119	3.650	0.016	0.00	0.00		\
...		C+0									
HETATM	8	C	UNK	0	2.390	4.043	-1.467	0.00	0.00		\
...		C+0									
HETATM	9	C	UNK	0	3.204	3.058	-2.216	0.00	0.00		\
...		C+0									
HETATM	10	N	UNK	0	1.121	4.261	-2.184	0.00	0.00		\
...		N+1									
HETATM	11	O	UNK	0	3.531	3.283	-3.404	0.00	0.00		\
...		O+0									
HETATM	12	O	UNK	0	3.645	1.907	-1.656	0.00	0.00		\
...		O-1									
MASTER	0	0	0	0	0	0	0	0	12	0	0
END										0	0

Note the lack of annotation in the PDB atom names (labelled as their element symbol) and residue names ('UNK'). This information would need to be annotated manually in order to run an MD simulation. In the RDKit, structures for simple protein sequences can be generated by reading an amino acid sequence:

```

>>> from rdkit import Chem
>>> from rdkit.Chem import AllChem
>>> mol = Chem.MolFromSequence('F')
>>> AllChem.EmbedMolecule(mol, AllChem.ETKDGv2())
0
>>> print(Chem.MolToPDBBlock(mol))
ATOM      1  N   PHE A   1      -2.664  -1.853  0.236  1.00  0.00  \
...        N
ATOM      2  CA  PHE A   1      -1.340  -1.105  0.789  1.00  0.00  \
...        C
ATOM      3  C   PHE A   1      -1.982   0.347  0.875  1.00  0.00  \
...        C
ATOM      4  O   PHE A   1      -3.045   0.666  0.443  1.00  0.00  \
...        O
ATOM      5  CB  PHE A   1      -0.777  -0.919  -0.748  1.00  0.00  \
...        C
ATOM      6  CG  PHE A   1       0.505  -0.247  -0.694  1.00  0.00  \
...        C
ATOM      7  CD1 PHE A   1       1.621  -1.079  -0.546  1.00  0.00  \
...        C
ATOM      8  CD2 PHE A   1       0.787   1.081  -0.783  1.00  0.00  \
...        C
ATOM      9  CE1 PHE A   1       2.896  -0.556  -0.497  1.00  0.00  \
...        C
ATOM     10  CE2 PHE A   1       2.054   1.623  -0.737  1.00  0.00  \
...        C
ATOM     11  CZ  PHE A   1       3.120   0.793  -0.592  1.00  0.00  \
...        C
ATOM     12  OXT PHE A   1      -1.176   1.249   1.519  1.00  0.00  \
...        O
CONECT   1   2
CONECT   2   3   5
CONECT   3   4   4   12
CONECT   5   6
CONECT   6   7   7   8
CONECT   7   9
CONECT   8   10  10
CONECT   9   11  11
CONECT  10  11
END

```

The level of annotation is greater, but hydrogen atom positions are not added (were these to be added using `Chem.AddHs`, they would be incorrectly annotated) and the atom names do not match the CHARMM atom names exactly (though they do follow the standard PDB naming conventions, so it would be trivial to correct them). `psfgen` is capable of assigning the positions of hydrogen atoms and setting the correct charge states, so this approach works reasonably well for simple L-amino or D-amino sequences. It is, however, nontrivial to generate mixtures of D and L amino acids, as acid chirality is set using a flag argument to `Chem.MolFromSequence`. As with Avogadro and PyMol, D-amino acids are labelled in the same way as the corresponding L-amino acid.

With pygen-structures, it is easily possible to generate the structures of free amino acids, peptides, or more complex structures such as glycans and glycopeptides, in an automated fashion. A significant advantage is that it is no longer necessary to use psfgen to generate the PSF file. This greatly simplifies combinatorial searches of small molecule sequences.

```

>>> from pygen_structures import (
...     code_to_mol,
...     sequence_to_mol,
...     load_charmm_dir
... )
>>> ## pygen_structures is distributed with the CHARMM35/36 files.
>>> rtf, prm = load_charmm_dir()
>>> mol = code_to_mol('AdAF', rtf, segid='PROT') # L-ALA, D-ALA, L-PHE
>>> print(mol.to_pdb_block())
COMPND    AdAF
AUTHOR    pygen-structures v0.2.4
REMARK 42
REMARK 42 TOPOLOGY FILES USED
REMARK 42      toppar_all36_prot_c36_d_aminoacids.str
REMARK 42      top_all36_prot.rtf
ATOM      1  N  ALA      1      -6.082  -0.735   1.391  1.00  0.00  \
... PROT N1+
ATOM      2  HT1 ALA      1      -5.807  -0.157   2.244  1.00  0.00  \
... PROT H
ATOM      3  HT2 ALA      1      -7.098  -0.635   1.251  1.00  0.00  \
... PROT H
ATOM      4  HT3 ALA      1      -5.757  -1.714   1.511  1.00  0.00  \
... PROT H
ATOM      5  CA  ALA      1      -5.408  -0.121   0.253  1.00  0.00  \
... PROT C
<truncated>
>>> patches = {"RAFF", [0, 1, 2]}
>>> sequence = ['AGLC', 'BFRU', 'AGAL']
>>> mol = sequence_to_mol(
...     sequence,
...     rtf,
...     patches=patches,
...     name='Raffinose'
...     segid="RAFF"
... )
>>> print(mol.to_pdb_block())
COMPND    Raffinose
AUTHOR    pygen-structures v0.2.4
REMARK 42
REMARK 42 TOPOLOGY FILES USED
REMARK 42      top_all36_carb.rtf
ATOM      1  C1  AGLC     1      -1.645  -0.345  -0.614  1.00  0.00  \
... RAFF C
ATOM      2  H1  AGLC     1      -1.845   0.725  -0.413  1.00  0.00  \
... RAFF H
ATOM      3  O1  AGLC     1      -2.764  -1.026  -0.170  1.00  0.00  \
... RAFF O
ATOM      4  C5  AGLC     1       0.657  -0.321  -0.142  1.00  0.00  \
... RAFF C
ATOM      5  H5  AGLC     1      1.185  -1.104  -0.707  1.00  0.00  \

```

```
... RAFF H
<truncated>
>>> print(mol.to_psf_block())
PSF EXT CMAP XPLOR

    4 !NTITLE
* Generated procedurally by pygen-structures v0.2.4
* Molecule: Raffinose
* Topology files used:
* - top_all36_carb.rtf

    66 !NATOM
    1 RAFF    1      AGLC    C1      CC3162  0.290000  \
... 12.0110    0
    2 RAFF    1      AGLC    H1      HCA1    0.090000  \
... 1.0080    0
    3 RAFF    1      AGLC    O1      OC302   -0.360000  \
... 15.9994    0
    4 RAFF    1      AGLC    C5      CC3163  0.110000  \
... 12.0110    0
    5 RAFF    1      AGLC    H5      HCA1    0.090000  \
... 1.0080    0
<truncated>
```

A downside to this approach is that knowledge of the available CHARMM residues and the required patches is necessary, but this is true of psfgen as well. Some initial work is always required to discover whether CHARMM can represent a molecule of interest. This could be improved through use of automated pattern matching from provided structures, but is challenging due to the incomplete nature of X-ray crystal data.

Future Work

pygen-structures is not yet a fully featured replacement for psfgen. In particular, the present structure generation method fails to fill coordinates for missing residues or missing atoms of large structures (making it impractical to use with protein structures from the PDB).

In future, the aim is to support all of psfgen's current functionality. Further advantageous features would include the generation of simulation boxes containing water, the ability to handle multiple molecules in a single system, and an automatic bead-generating method for coarse-grained simulations.

Further documentation of the polymeric structures (and relevant linkages) which are already parameterised in the CHARMM forcefield would be advantageous to encourage further combinatorial work.

Dependencies

pygen-structures depends upon the RDKit for 3D coordinate generation and uses NumPy arrays for representations of the molecular adjacency matrix (Oliphant, 2015; van der Walt, Colbert, & Varoquaux, 2011).

Unit and integration tests (using `pytest`, Krekel et al. (2004)) are supplied to test the functionality provided by the package. Tests can be run using `pytest --pyargs pygen_structures`. Tests rely upon `OpenMM` (Eastman et al., 2017) as an additional dependency.

Python 3.6 and 3.7 are supported, and pip can be used for installation: `pip install pygen-structures`.

Acknowledgements

TH thanks the Strathclyde Computational and Theoretical Chemistry Hub (SCoTCH) at the University of Strathclyde for helpful conversations and enthusiasm, and those that are working to increase recognition of research software engineers.

MarvinSketch was used for 3D coordinate generation in “Problems with Current Workflows”, MarvinSketch 20.3, ChemAxon (<https://www.chemaxon.com>).

References

Brünger, A. T., & Nilges, M. (1993). Computational challenges for macromolecular structure determination by X-ray crystallography and solution NMRspectroscopy. *Quarterly Reviews of Biophysics*, 26(1), 49–125. doi:[10.1017/S0033583500003966](https://doi.org/10.1017/S0033583500003966)

Eastman, P., Swails, J., Chodera, J. D., McGibbon, R. T., Zhao, Y., Beauchamp, K. A., Wang, L.-P., et al. (2017). OpenMM 7: Rapid development of high performance algorithms for molecular dynamics. *PLoS Computational Biology*, 13(7), e1005659. doi:[10.1371/journal.pcbi.1005659](https://doi.org/10.1371/journal.pcbi.1005659)

Hanwell, M. D., Curtis, D. E., Lonié, D. C., Vandermeersch, T., Zurek, E., & Hutchison, G. R. (2012). Avogadro: An advanced semantic chemical editor, visualization, and analysis platform. *Journal of Cheminformatics*, 4(1), 17. doi:[10.1186/1758-2946-4-17](https://doi.org/10.1186/1758-2946-4-17)

Huang, J., & MacKerell, A. D. (2013). CHARMM36 all-atom additive protein force field: Validation based on comparison to NMR data. *Journal of Computational Chemistry*, 34(25), 2135–2145. doi:[10.1002/jcc.23354](https://doi.org/10.1002/jcc.23354)

Humphrey, W., Dalke, A., & Schulten, K. (1996). VMD - Visual Molecular Dynamics. *Journal of Molecular Graphics*, 14, 33–38. doi:[10.1016/0263-7855\(96\)00018-5](https://doi.org/10.1016/0263-7855(96)00018-5)

Krekel, H., Oliveira, B., Pfannschmidt, R., Bruynooghe, F., Laugher, B., & Bruhin, F. (2004). Pytest 5.0. Retrieved from <https://github.com/pytest-dev/pytest>

Oliphant, T. (2015). *Guide to NumPy* (Second.). CreateSpace Independent Publishing Platform. Retrieved from <http://doi.org/10.5555/2886196>

PerkinElmer Informatics. (2019). *The ChemOffice Suite*.

Ribeiro, J. V., Radak, B., Stone, J., Gullingsrud, J., Saam, J., & Phillips, J. (2019, August 27). VMD psfgen Plugin, Version 2.0. Retrieved February 25, 2020, from <https://www.ks.uiuc.edu/Research/vmd/plugins/psfgen/>

Riniker, S., & Landrum, G. A. (2015). Better informed distance geometry: Using what we know to improve conformation generation. *Journal of Chemical Information and Modeling*, 55(12), 2562–2574. doi:[10.1021/acs.jcim.5b00654](https://doi.org/10.1021/acs.jcim.5b00654)

Schrödinger LLC. (2019). *The PyMOL Molecular Graphics System*.

The RDKit Contributors. (n.d.). RDKit: Open-Source Cheminformatics Software. Retrieved February 25, 2020, from <https://www.rdkit.org/>

van der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science Engineering*, 13(2), 22–30. doi:[10.1109/MCSE.2011.37](https://doi.org/10.1109/MCSE.2011.37)

wwPDB. (2012). Protein Data Bank Contents Guide, version 3.30. wwPDB. Retrieved from
<https://www.wwpdb.org/documentation/file-format>