Privileged Residues Documentation

Release 0.5.0

Brian D. Weitzner

Jul 24, 2018

Contents

Contents:

CHAPTER 1

Privileged Residues

Privileged Residues contains methods for placing residues on the surface of a target protein that can be added to a RIF.

- Free software: Apache Software License 2.0
- Documentation: https://privileged-residues.readthedocs.io.

1.1 Features

- TODO

1.2 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install Privileged Residues, run this command in your terminal:

```
$ pip install privileged_residues
```

This is the preferred method to install Privileged Residues, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for Privileged Residues can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/weitzner/privileged_residues
```

Or download the tarball:

```
$ curl  -OL https://github.com/weitzner/privileged_residues/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

To use Privileged Residues in a project:

```
import privileged_residues
```

CHAPTER 4

API Documentation

4.1 Privileged Residues Package

4.1.1 Submodules

privileged_residues.chemical module

class privileged_residues.chemical.**FunctionalGroup**
Bases: tuple

Store hydrogen bonding information about a functional group as well as information that can be used to position it in three-space.

resName
str – Name of the functional group.

donor
bool – True if the functional group can be a donor in a hydrogen bond.

acceptor
bool – True if the functional group can be an acceptor in a hydrogen bond.

atoms
list of str – List of three atom names that are used to construct a coordinate frame to describe the position of the functional group in three-space.

acceptor
Alias for field number 2

atoms
Alias for field number 3

donor
Alias for field number 1

resName
Alias for field number 0

class privileged_residues.chemical.**ResInfo**
Bases: tuple

Store functional group information about an amino acid as well as information that can be used to position it in three-space.

grp
str – Name of a functional group.

atoms
list of str – List of three atom names that are used to construct a coordinate frame to describe the position of the functional group of the amino acid in three-space.

atoms
Alias for field number 1

grp
Alias for field number 0

privileged_residues.chemical.**acceptor_acceptor_rays**(*pose*, *selector*)
Get acceptor-acceptor network ray pairs for the residues indicated by the provided residue selector.

Parameters

- **pose** (*pyrosetta.pose*) – Target structure.
- **selector** (*pyrosetta.rosetta.core.select.residue_selector.ResidueSelector*) – Residue selector to apply to the pose.

Returns All of the ray pairs corresponding to possible acceptor-acceptor network interactions in the selected subset of the pose.

Return type list of tuple of np.ndarray

privileged_residues.chemical.**donor_acceptor_rays**(*pose*, *selector*)
Get donor-acceptor network ray pairs for the residues indicated by the provided residue selector.

Parameters

- **pose** (*pyrosetta.pose*) – Target structure.
- **selector** (*pyrosetta.rosetta.core.select.residue_selector.ResidueSelector*) – Residue selector to apply to the pose.

Returns All of the ray pairs corresponding to possible donor-acceptor network interactions in the selected subset of the pose.

Return type list of tuple of np.ndarray

privileged_residues.chemical.**donor_donor_rays**(*pose*, *selector*)
Get donor-donor network ray pairs for the residues indicated by the provided residue selector.

Parameters

- **pose** (*pyrosetta.pose*) – Target structure.
- **selector** (*pyrosetta.rosetta.core.select.residue_selector.ResidueSelector*) – Residue selector to apply to the pose.

Returns All of the ray pairs corresponding to possible donor-donor network interactions in the selected subset of the pose.

Return type list of tuple of np.ndarray

privileged_residues.chemical.**sc_bb_rays**(*pose*, *selector*)
Get sidechain-to-backbone ray pairs for the residues indicated by the provided residue selector.

Parameters

- **pose** (*pyrosetta.pose*) – Target structure.
- **selector** (*pyrosetta.rosetta.core.select.residue_selector.ResidueSelector*) – Residue selector to apply to the pose.

Returns All of the ray pairs corresponding to possible sidechain-to-backbone interactions in the selected subset of the pose.

Return type list of tuple of np.ndarray

privileged_residues.chemical.**sc_sc_rays**(*pose*, *selector*)
Get sidechain-to-sidechain ray pairs for the residues indicated by the provided residue selector.

Parameters

- **pose** (*pyrosetta.pose*) – Target structure.
- **selector** (*pyrosetta.rosetta.core.select.residue_selector.ResidueSelector*) – Residue selector to apply to the pose.

Returns All of the ray pairs corresponding to possible sidechain-to-sidechain interactions in the selected subset of the pose.

Return type list of tuple of np.ndarray

privileged_residues.chemical.**sc_scbb_rays**(*pose*, *selector*)
Get sidechain-to-sidechain-and-backbone ray pairs for the residues indicated by the provided residue selector.

Parameters

- **pose** (*pyrosetta.pose*) – Target structure.
- **selector** (*pyrosetta.rosetta.core.select.residue_selector.ResidueSelector*) – Residue selector to apply to the pose.

Returns All of the ray pairs corresponding to possible sidechain-to-sidechain-and-backbone interactions in the selected subset of the pose.

Return type list of tuple of np.ndarray

privileged_residues.geometry module

privileged_residues.geometry.**coords_to_transform**(*coords*)
From a set of coordinates, construct a homogeneous transform.

Parameters **coords** (*np.ndarray*) –

Returns Homogeneous transform constructed from the input coordinates.

Return type np.ndarray

privileged_residues.geometry.**create_ray**(*center*, *base*)
Create a ray of unit length from two points in space and return it.

Notes

The ray is constructed such that:

1. The direction points from *base* to *center* and is unit length.
2. The point at which the ray is centered is at *center*.

Parameters

- **center** (*numpy.ndarray*) – A (1, 3) array representing the coordinate at which to center the resulting ray.
- **base** (*numpy.ndarray*) – A (1, 3) array representing the base used to determine the direction of the resulting ray.

Returns A (2,4) array representing a ray in space with a point and a unit direction.

Return type numpy.ndarray

privileged_residues.geometry.**rays_to_transform**(*first*, *second*)

From two rays, construct a homogeneous transform.

Parameters

- **first** (*np.ndarray*) –
- **second** (*np.ndarray*) –

Returns Homogeneous transform constructed from the input rays.

Return type np.ndarray

privileged_residues.postproc module

privileged_residues.postproc.**filter_clash_minimize**(*pose*, *hits*, *clash_cutoff=35.0*, *rmsd_cutoff=0.5*, *sfx=None*, *mmap=None*, *limit=0*)

Filter match output for clashes, then minimize the remaining structures against the target pose.

Parameters

- **pose** (*pyrosetta.Pose*) – Target structure.
- **hits** (*np.ndarray*) – Set of functional group matches against positions in the target.
- **clash_cutoff** (*float*) – Maximum tolerated increase in score terms during clash checking.
- **sfx** (*pyrosetta.rosetta.core.scoring.ScoreFunction, optional*) – Scorefunction to use during minimization. If left as None, a default scorefunction is constructed.
- **mmap** (*pyrosetta.rosetta.protocols.minimization_packing.MinMover, optional*) – Movemap to use during minimization. If left as None, a default movemap is constructed.

Yields *pyrosetta.Pose* – Next target structure and matched functional group, minimized and in complex.

privileged_residues.privileged_residues module

class privileged_residues.privileged_residues.**PrivilegedResidues**(*path='/home/onalant/dump/2018-05-07_datatables/database.h5'*)

Bases: object

match(*ray1*, *ray2*, *group*)
Construct all of the matched structures for a given ray pair and group.

Notes

The following are the available search groups.

Bidentates:

- "sc_sc"
- "sc_scbb"
- "sc_bb"

Networks:

- "acceptor_acceptor"
- "acceptor_donor"
- "donor_acceptor"
- "donor_donor"

Parameters

- **ray1** (*np.ndarray*) –
- **ray2** (*np.ndarray*) – Rays used to search in the underlying database.
- **group** (*str*) – Dataset to search in.

Yields *pyrosetta.Pose* – Functional group as placed by transform from table.

search(*pose*, *groups*, *selector*)
Search for privileged interactions in a pose.

Parameters

- **pose** (*pyrosetta.Pose*) – Target structure.
- **groups** (*list of str*) – Datasets or groups to search for matches in.
- **selector** (*pyrosetta.rosetta.core.select.residue_selector.ResidueSelector*) – Residue selector to apply to the pose.

Yields *tuple of np.uint64 and pyrosetta.Pose* – Target ray pair hash and output pose.

privileged_residues.table module

class privileged_residues.table.**GenericTable**(*dbpath*)
Bases: object

Indexed key-value store implementation.

Best used on large datasets that do not fit into memory.

fetch(*key*, *findgroup=''*)

Parameters

- **key** (*np.uint64*) – A hash value.

- **findgroup** (*str, optional*) – A named group to search for hashes in. Defaults to "", which searches in all named groups.

Returns Concatenated list of matches for a hash and group query.

Return type np.ndarray

privileged_residues.util module

privileged_residues.util.**models_from_pdb**(*fname*)
Get models from a PDB as individual poses.

Parameters **fname** (*str*) – Path to a PDB.

Yields *pyrosetta.Pose* – The next model in the PDB.

privileged_residues.util.**numpy_to_rif**(*r*)
Convert from NumPy ray representation to RIF ray representation.

Parameters **r** (*np.ndarray*) – Input NumPy ray.

Returns

Return type rif.geom.Ray

4.1.2 Module contents

class privileged_residues.**PrivilegedResidues**(*path='/home/onalant/dump/2018-05-07_datatables/database.h5'*)
Bases: object

match(*ray1*, *ray2*, *group*)
Construct all of the matched structures for a given ray pair and group.

Notes

The following are the available search groups.

Bidentates:

- "sc_sc"
- "sc_scbb"
- "sc_bb"

Networks:

- "acceptor_acceptor"
- "acceptor_donor"
- "donor_acceptor"
- "donor_donor"

Parameters

- **ray1** (*np.ndarray*) –
- **ray2** (*np.ndarray*) – Rays used to search in the underlying database.

- **group** (*str*) – Dataset to search in.

Yields *pyrosetta.Pose* – Functional group as placed by transform from table.

search (*pose*, *groups*, *selector*)
Search for privileged interactions in a pose.

Parameters

- **pose** (*pyrosetta.Pose*) – Target structure.
- **groups** (*list of str*) – Datasets or groups to search for matches in.
- **selector** (*pyrosetta.rosetta.core.select.residue_selector.ResidueSelector*) – Residue selector to apply to the pose.

Yields *tuple of np.uint64 and pyrosetta.Pose* – Target ray pair hash and output pose.

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/RosettaCommons/privileged_residues/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

5.1.4 Write Documentation

Privileged Residues could always use more documentation, whether as part of the official Privileged Residues docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/RosettaCommons/privileged_residues/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *privileged_residues* for local development.

1. Fork the *privileged_residues* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/privileged_residues.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv privileged_residues
$ cd privileged_residues/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 privileged_residues tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/RosettaCommons/privileged_residues/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ py.test tests.test_privileged_residues
```

CHAPTER 6

Credits

6.1 Development Lead

- Brian D. Weitzner <weitzner@uw.edu>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

History

7.1 0.1.0 (2018-02-05)

- First release on PyPI.
- This package is used to position amino acid residues at a protein interface in a “privileged” position.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index