SPyCi-PDB

Release 0.3.13

Zi Hao Liu

Jul 27, 2023

CONTENTS

Structural Python (Back) Calculator Interface for PDBs

Goal: User friendly Python3 based interface to generate back-calculated experimental data for singular and multiple (ensembles of) PDB structures.

Back-Calculators: The current back calculators integrated are:

1. PRE, NOE, J-Coupling, smFRET: Internal
2. Chemical shifts (CS): UCBShift
3. SAXS: CRYSOLv3
4. Hydrodynamic Radius (Rh): HullRad
5. Residual Dipolar Couplings (RDC): PALES

Please note for third-party software, installation instructions have been fully documented and tested for Linux Ubuntu Focal Fossa (20.04.X LTS) and Bionic Beaver (18.04.X LTS).

To make new requests and/or additions, please see `docs/contributing.rst`.

Developer Notes: project CI based on @joaomcteixeira's Python-Project-Skeleton template. Developed as a standalone program with integration into the IDPConformerGenerator platform in mind.

CHAPTER

ONE

CONTENTS

1.1 Installation

SPyCi-PDB uses only Python-based APIs for which we expect it to run native on any system Python can run, as long as third-party back-calculator installation requirements are met.

Please note that the IDPConformerGenerator library is required for parsing PDB structure coordinates. Details for IDPConformerGenerator can be found here.

Full installation instructures are highlighted below.

Note: As highlighted in `requirements.txt`, listed below are all of the Python packages required. The following list will be updated if more packages are added:

```
numpy >= 1.22.4, < 2
numba >= 0.53.0, < 1
pybind11 >= 2.9.2, < 3
pandas >= 1.2.4, < 2
scipy >= 1.8.1, < 2
matplotlib >= 3.5.2, < 4
natsort >= 8, < 9
tox >= 3.25.0, < 4
```

As mentioned above, `IDPConformerGenerator` libraries are required.

1.1.1 If you already have an `idpconfgen` environment

It is recommended to install IDPConformerGenerator first to have the environment prepared for SPyCi-PDB. Please visit the IDPConformerGenerator repository to install IDPConformerGenerator environment first.

Activate the `idpconfgen` environment (conda or virtualenv) and update the environment with the dependencies seen in SPyCi-PDB:

```
pip install -r existing_requirements.txt --upgrade
```

Install SPyCi-PDB on top of IDPConformerGenerator:

```
python setup.py develop --no-deps
```

1.1.2 Standalone From Source

Clone from the official repository:

```
git clone https://github.com/julie-forman-kay-lab/SPyCi-PDB
```

Navigate to the new `SPyCi-PDB` folder:

```
cd SPyCi-PDB
```

Run the following commands to install `spycipdb` dependencies if Anaconda is used as your Python package manager:

```
conda env create -f requirements.yml
conda activate spycipdb
```

Note: If you don't use Anaconda to manage your Python installations, you can use `virtualenv` and the `requirements.txt` file following the commands:

```
virtualenv spycienv --python=3.8
source spycienv/bin/activate
pip install -r requirements.txt
```

If you have difficulties installing `spycipdb`, raise an issue in the main GitHub repository, and we will help you.

Install `spycipdb` in development mode in order for your installation to be always up-to-date with the repository:

```
python setup.py develop --no-deps
```

Note: The above applies also if you used `virtualenv` instead of `conda`.

Remember to activate the `spycipdb` environment every time you open a new terminal window, from within the repository folder, choose yours:

```
# Installation with Anaconda
conda activate spycipdb

# Installation with virtualenv
source spycienv/bin/activate
```

To update to the latest version, navigate to the repository folder, activate the `spycipdb` python environment as described above, and run the commands:

```
git pull

# if you used anaconda to create the python environment, run:
```

(continues on next page)

(continued from previous page)

```
conda env update -f requirements.yml

# if you used virtual env to create the python environment, run:
pip install -r requirements.txt  --upgrade

python setup.py develop --no-deps
```

Your installation will become up to date with the latest developments.

Please note that IDPConformerGenerator will be installed alongside SPyCi-PDB as the library is required for processing PDB files. You can find the installation of IDPConformerGenerator in `src/idpconfgen/`.

1.1.3 Updating SPyCi-PDB

When new back-calculators are added or when dependencies change from the GitHub, please update your environment with the latest dependencies after performing a `git pull`.

To update an anaconda environment:

```
conda env update --file requirements.yml --prune
```

To update a virtualenv environment:

```
pip install -r requirements.txt --upgrade
```

Then perform `setup.py` again for good measure:

```
python setup.py develop --no-deps
```

1.1.4 Installing Third-party Software

Some functionalities of `SPyCi-PDB` require third-party software. These are not mandatory to install unless you want to use such operations.

UCBShift

Note: Module is only required if you wish to perform Chemical Shift back-calculations. The following installation will revert Python to version 3.8.15.

You should be in the parent directory where `SPyCi-PDB` was cloned to with the Python environment is deactivated.

Clone the UCBShift repository from Zi Hao Liu's fork on GitHub.:

```
git clone https://github.com/menoliu/CSpred
```

Enter the `CSpred` folder and make a `models` directory, then download and extract the latest trained models to `CSpred/models` directory.

Move back into the `SPyCi-PDB` directory and enter the `thirdparty/ucbshift_reqs` folder:

```
cd ..
cd SPyCi-PDB
cd ./thirdparty/ucbshift_reqs
```

Run the following commands to install `UCBShift` dependencies if Anaconda is used as your Python package manager:

```
conda env update --name spycipdb --file ucbshift_requirements.yml --prune
```

Run the following commands to install `UCBShift` dependencies if virtualenv was used to install SPyCi-PDB:

```
pip install -r ucbshift_requirements.txt
```

Go back to the `SPyCi-PDB` directory and reinstall `spycipdb` and `idpconfgen` if needed.:

```
conda activate spycipdb
cd ../..
python setup.py develop --no-deps
cd ./src/idpconfgen/
python setup.py develop --no-deps
```

The following is the same with virtualenv:

```
source spycienv/bin/activate
cd ../..
python setup.py develop --no-deps
cd ./src/idpconfgen/
python setup.py develop --no-deps
```

Currently, reinstallation is required as UCBShift changes the Python version. We will be working on a fix to streamline this process soon by using package handlers such as `PyPi`. Thank you for your patience.

ATSAS v3.1.1 - CRYSOL v3.0

Note: ATSAS installation is only required for the `saxs` module.

Please visit the ATSAS website to download v3.1.1 of ATSAS. Theoretically, SPyCi-PDB will work if you already have ATSAS v3.X installed.

Test your installation via:

```
crysol -h
```

PALES v6.0

Note: PALES installation is only required for the `rdc` module.

A package of PALES v6.0 for Linux is already included in the `thirdparty/` folder. Downloaded and extracted from the Ad Bax Group.

For use with x64 bit Linux Ubuntu 20.04.X LTS and 18.04.X, you must install the i386 architecture along with required package libraries:

```
sudo dpkg --add-architecture i386
sudo apt update
sudo apt install libc6:i386 libncurses5:i386 libstdc++6:i386 libx11-6:i386
```

If the last command above fails, run the following instead:

```
sudo apt install multiarch-support
```

1.2 Usage

SPyCi-PDB runs entirely through command-lines. Theoretically, it is compatible with any machine that can un Python. However, it's only been thoroughly tested on WSL 2.0 on Windows 11, Linux Ubuntu 20.04.X LTS and 18.04.X LTS.

Please follow the explanations in this page plus the documentation on the command-line themselves via:

```
spycipdb <MODULE> -h
```

Please note that SPyCi-PDB can also be used as a library as back-calculator functions are completely modular:

```
import spycipdb
```

1.2.1 Command-Lines

To execute `spycipdb` command-line, run `spycipdb` in your terminal window, after *installation*:

```
spycipdb
```

or:

```
spycipdb -h
```

Both will output the help menu.

Note: All subclients have the `-h` option to show help information.

1.2.2 Formatting for Input and Output Files

Conformers will be required in the PDB format with the `.pdb` file extension. For tarballs and folders, only the `.pdb` files in the folder/tarball will be used. Accepted tarballs include `.tar`, `.tar.gz`, and `.tar.xz` file extensions.

Most modules will require a sample experimental results template to base the back-calculations off of. Please note that experimental result `values` are not required. For the mathematical equations for the default internal calculators (`pre`, `noe`, `jc`, `smfret`) please refer to the paper.

All input files can be in the `.txt` file format with comma-delimitation for values. Examples can be found in the `example/drksh3_exp_data/` folder. The required header formatting for the `.txt` file for each experimental module is highlighted below.

Output files are saved in a standard human-readable `.JSON` format. In most cases, the first key-value pair gives the format for each of the values in subsequent key-value pairs.

Paramagnetic Resonance Entropy (PRE) module

Input: `res1,atom1,res2,atom2`

Where res1/atom1 is the residue number and atom name respectively for the first residue and res2/atom2 is the residue number and atom name respectively for the second residue.

Output: `'format': {'res1': [], 'atom1': [], 'res2': [], 'atom2': []}`

Subsequent keys are the names of the PDB file with a value of: `[dist_values]`.

About: The default back-calculator is a simple distance based interpretation of PRE data. Scalar distances between pairs of atoms according to the pairs derived from the experimental template are used.

Nuclear Overhauser Effect (NOE) module

Input: `res1,atom1,atom1_multiple_assignments,res2,atom2,atom2_multiple_assignments`

Where res1/atom1 is the residue number and atom name respectively for the first residue and res2/atom2 is the residue number and atom name respectively for the second residue. Multiple assignments are either 0 or 1 (no/yes respectively).

Output: `'format': {'res1': [], 'atom1': [], 'atom1_multiple_assignments': [], 'res2': [], 'atom2': [], 'atom2_multiple_assignments': []}`

Subsequent keys are the names of the PDB file with a value of: `[dist_values]`.

About: The default back-calculator is a simple distance based interpretation of NOE data. Scalar distances between pairs of atoms according to the pairs derived from the experimental template are used.

3J-HNHA coupling (JC) module

Input: `resnum`

Where `resnum` indicates the JC for a specific residue.

Output: `'format': [resnum]`

Subsequent keys are the names of the PDB file with a value of: `[jc_values]`.

About: The default back-calculator uses the Karplus curve, a cosine function, to back-calculate desired J-couplings according to the residue number as provided by the experimental template file.

single-molecule Fluoresence Resonance Energy Transfer (smFRET) module

Input: `res1,res2,scaler`

Where res1/res2 is the residue number for the first and second residue respectively. Scaler is the r0 Foster radius of the dye pair.

Output: `'format': { 'res1': [], 'res2': [], 'scale': []}`

Subsequent keys are the names of the PDB file with a value of: `[smfret_values]`.

About: The default back-calculator, abliet distance based, takes into consideration residue pairs and a scale factor to adjust for dye size from the experimental setup to back-calculate distances between two CA backbone atoms.

Residual Dipolar Coupling (RDC) module

Input: The default back-calculator for RDCs is PALES. To run PALES, however, you will need the provide an experimental file template that is the same format as for PALES. For more information, please see the Data Format section. | **Output:** `'format': {resnum1: [], resname1: [], atomname1: [], resnum2: [], resname2: [], atomname2: []}` | Subsequent keys are the names of the PDB file with a value of: `[rh_values]`. | **About:** The default back-calculator uses the third-party program PALES, which uses the steric obstruction model to derive the RDC. It has also been chosen due to its popularity in the field for RDC back-calculations.

Note: SAXS, Rh, and CS modules do not require input files. The following is formatting for the output.

Small Angle X-ray Scattering (SAXS) module

Output: Each PDB name is assigned a list of `index` and `value` where `index` represents the X-axis and `value` represents the Y-axis with units `I_abs(s)[cm^-1]/c[mg/ml]`.

About: The default back-calculator uses the third-party program CRYSOLv3, from the ATSAS suite of biological softwares. it has been chosen due to its popularity as well as its ability to evaluate the hydration shell using dummy water parameters.

Hydrodynamic Radius (Rh) module

Output: Each PDB name is assigned a singular hydrodynamic radius value.

About: The default back-calculator uses the third-party program HullRad. It has been chosen for its speed and usage of the convex hull model to estimate hydrodynamic properties.

Chemical Shift (CS) module

Output: `'format': {'res': [], 'resname': []}`. Subsequent keys are the names of the PDB file with a value of: `{'H': [], 'HA': [], 'C': [], 'CA': [], 'CB': [], 'N': []}`.

About: The default back-calculator uses the third-party program UCBShift. It has been chosen for its two-pronged machine learning approach for both feature and sequence alignment in order to provide an accurate chemical shift prediction.

1.2.3 Basic Usage Examples

The `example/` folder contains instructions to test native back-calculators on 100 test conformers of the unfolded state of the drkN SH3 domain.

To get started with using the different modules for back-calculating experimental data, we will be using the unfolded state of drkN SH3, an intensively studied intrinsically disordered protein.

The goal of these examples is to walk you through expected experimental data formatting styles, as well as how to get started with each module. The experimental files could be found in the `example/drksh3_exp_data` folder. A set of 100 conformers generated using IDPConformerGenerator has been provided as a tarball: `example/drksh3_csss_100.tar.xz`.

Note that these experimental data files are comma-delimited per CSV formatting for ease of use in `pandas` dataframe as well as Microsoft Excel usage.

To use the bare-bones version of spycipdb, the PRE, NOE, JC, Rh, and smFRET modules do not require third-party installation instructions.

Every module is equipped with `--help` sections with detailed usage examples and documentation for each functionality. For customized `--output` file names, the flag `-o` can be used with every module.

Please note for large number of PDB ensembles, it is recommended to specify the number of CPU cores you are comfortable with using to maximize speed. Having just `-n` will utilize all but one CPU thread.

1.2.4 PRE, NOE, and JC Modules

To perform the back-calculation using the default internal calculators, give the tarball of provided structures as the first argument as well as the path to the experimental data file to use as a template. For example, using the PRE module:

```
spycipdb pre ./drksh3_csss_100.tar.xz -e ./drksh3_exp_data/drksh3_PRE.txt -n
```

Likewise to the PRE module, the tarball of the provided structures and sample NOE experimental data file are required:

```
spycipdb noe ./drksh3_csss_100.tar.xz -e ./drksh3_exp_data/drksh3_NOE.txt -n
```

Since the J-Coupling back-calculator is also internal, the same format is as follows:

```
spycipdb jc ./drksh3_csss_100.tar.xz -e ./drksh3_exp_data/drksh3_JC.txt -n
```

1.2.5 CS Module - Using UCBShift

After ensuring UCBShift is installed, the CS module does not require experimental file samples. Furthermore, you could also adjust the pH value to be considered. The default pH value is 5.0. A sample command is as follows:

```
spycipdb cs ./drksh3_csss_100.tar.xz --ph 7 -n
```

The above command sets a custom pH of 7.0. Please also note that UCBShift is fairly RAM intensive, it's recommended to run with less than 10 CPUs (can be changed with the flag `-n 10` for example).

1.2.6 SAXS Module - Using CRYSOLv3

After ensuring the proper ATSAS/CRYSOL version is installed, the following command can be used to run the SAXS module. Please note again that an experimental template is not required:

```
spycipdb saxs ./drksh3_csss_100.tar.xz -n
```

The SAXS module is equipped with a `--lm` flag, as CRYSOLv3 uses an adjustable number of harmonics to perform the appropriate SAXS back-calculation. The default value is 20 (from 1-100). Increasing the number of harmonics will also increase the cost of computational time.

1.2.7 Rh Module - Using HullRad 8.1

HullRad should be working out of the box with the basic installation instructions. An experimental file template is not required:

```
spycipdb rh ./drksh3_csss_100.tar.xz -n
```

1.3 Contributing

There are several strategies on how to contribute to a project on GitHub. Here, I explain the one I use for all the project I am participating. You can use this same strategy to contribute to this template or to suggest contributions to your project.

1.3.1 Fork this repository

Fork this repository before contributing. It is a better practice, possibly even enforced, that only pull request from forks are accepted. In my opinion enforcing forks creates a cleaner representation of the contributions to the project.

Clone the main repository

Next, clone the main repository to your local machine:

```
git clone https://github.com/julie-forman-kay-lab/SPyCi-PDB
cd SPyCi-PDB
```

Add your fork as an upstream repository:

```
git remote add myfork git://github.com/YOUR-USERNAME/SPyCi-PDB
git fetch myfork
```

1.3.2 Install for developers

Create a dedicated Python environment where to develop the project.

If you are using `pip` follow the official instructions on Installing packages using pip and virtual environments, most likely what you want is:

```
python3 -m venv newenv
source newenv/bin/activate
```

If you are using Anaconda go for:

```
conda create --name newenv python=3.9
conda activate newenv
```

Where `newenv` is the name you wish to give to the environment dedicated to this project.

Either under *pip* or *conda*, install the package in `develop` mode. Install also *tox*.

```
python setup.py develop
pip install tox
```

This configuration, together with the use of the `src` folder layer, guarantees that you will always run the code after installation. Also, thanks to the `develop` flag, any changes in the code will be automatically reflected in the installed version.

1.3.3 Make a new branch

From the `main` branch create a new branch where to develop the new code.

```
git checkout main
git checkout -b new_branch
```

Note the `main` branch is from the main repository.

Develop the feature and keep regular pushes to your fork with comprehensible commit messages.

```
git status
git add (the files you want)
git commit (add a nice commit message)
git push myfork new_branch
```

While you are developing, you can execute `tox` as needed to run your unit tests or inspect lint, or other integration tests. See the last section of this page.

Update your branch

It is common that you need to keep your branch update to the latest version in the `main` branch. For that:

```
git checkout main  # return to the main branch
git pull  # retrieve the latest source from the main repository
git checkout new_branch  # return to your devel branch
git merge --no-ff main  # merge the new code to your branch
```

At this point you may need to solve merge conflicts if they exist. If you don't know how to do this, I suggest you start by reading the official docs

You can push to your fork now if you wish:

```
git push myfork new_branch
```

And, continue doing your developments are previously discussed.

Update CHANGELOG

Update the changelog file under `CHANGELOG.rst` with an explanatory bullet list of your contribution. Add that list right after the main title and before the last version subtitle:

```
Changelog
=========

* here goes my new additions
* explain them shortly and well

vX.X.X (1900-01-01)
-------------------
```

Also add your name to the authors list at `docs/AUTHORS.rst`.

Pull Request

Once you finished, you can create a pull request to the main repository, and engage with the community.

Before submitting a Pull Request, verify your development branch passes all tests as *described below* . If you are developing new code you should also implement new test cases.

1.3.4 Uniformed Tests with tox

Thanks to Tox we can have a unified testing platform that runs all tests in controlled environments and that is reproducible for all developers. In other words, it is a way to welcome (*force*) all developers to follow the same rules.

The `tox` testing setup is defined in a configuration file, the tox.ini, which contains all the operations that are performed during the test phase. Therefore, to run the unified test suite, developers just need to execute `tox`, provided tox is installed in the Python environment in use.

```
pip install tox
# or
conda install tox -c conda-forge
```

One of the greatest advantages of using `tox` together with the src layout is that unit test actually perform on the installed source (our package) inside an isolated deployment environment. In order words, tests are performed in an environment simulating a post-installation state instead of a pre-deploy/development environment. Under this setup, there is no need, in general cases, to distribute unit test scripts along with the actual source, in my honest opinion - see MANIFEST.in.

Before creating a Pull Request from your branch, certify that all the tests pass correctly by running:

```
tox
```

These are exactly the same tests that will be performed online in the Github Actions.

Also, you can run individual testing environments if you wish to test only specific functionalities, for example:

```
tox -e lint  # code style
tox -e build  # packaging
tox -e docs  # only builds the documentation
tox -e test  # runs unit tests
```

1.4 How to Cite

If you use SPyCi-PDB, please cite:

```
Liu et al., (2023). SPyCi-PDB: A modular command-line interface for back-calculating␣
↪experimental datatypes of protein structures.. Journal of Open Source Software, 8(85),␣
↪4861, https://doi.org/10.21105/joss.04861
```

1.5 Authors

1.5.1 Main Developers/Maintainers

- Zi Hao (Nemo) Liu

1.5.2 Contributors

- João M. C. Teixeira (Postdoctoral researcher, University of Padua, Italy)
- Jie (Jerry) Li (PhD student, University of California Berkeley, USA)

1.5.3 Acknowledgements

- *Oufan Zhang*
- *Mickaël Krzeminski*
- Julie D. Forman-Kay
- Teresa Head-Gordon

1.6 Changelog

1.6.1 v0.3.13 (2023-07-27)

- Correct formatting error in documentation

1.6.2 v0.3.12 (2023-07-27)

- Update MANIFEST to include existing_requirements.txt

1.6.3 v0.3.11 (2023-07-27)

- Update requirements for existing IDPConformerGenerator environment
- Update documentations for installation for clarity

1.6.4 v0.3.10 (2023-05-11)

- Update README to include citation and JOSS button

1.6.5 v0.3.9 (2023-05-10)

- Correct repository for versioning and Git actions

1.6.6 v0.3.8 (2023-05-10)

- Update minor “co-ordinates” to “coordinates” in docs

1.6.7 v0.3.7 (2023-05-10)

- Update changelog and minor formatting for README

1.6.8 v0.3.6 (2023-05-10)

- Update paper affiliations

1.6.9 v0.3.5 (2023-05-02)

- Implement minor editorial suggestions from issue #51

1.6.10 v0.3.4 (2023-04-26)

- Minor corrections to documentation

1.6.11 v0.3.3 (2023-04-15)

- Update acknowledgements section of manuscript

1.6.12 v0.3.2 (2023-04-13)

- Update and make references uniform for manuscript

1.6.13 v0.3.1 (2023-04-12)

- Correct ‘ATSAS 2.8’ formatting in references of paper
- Include in RtD documentation explanations of back-calculators
- Combine formatting for input/output in usage section of RtD documentation

1.6.14 v0.3.0 (2023-04-10)

- Update results in the figure to include PRE and NOE as per #43
- Add plotting feature to `rh`, `jc`, `pre`, `noe` modeules

1.6.15 v0.2.3 (2023-03-27)

- Update example usage documentation

1.6.16 v0.2.2 (2023-03-07)

- Update installation documentation
- Temporarily change requirements for default IDPConformerGenerator repository link

1.6.17 v0.2.1 (2023-03-06)

- Update client setup
- Add warning for updating the environment when pulling

1.6.18 v0.2.0 (2023-01-27)

- Adds `natsort` as dependency to yield ordered results
- Results will be sorted as how they will appear on your OS

1.6.19 v0.1.16 (2023-01-18)

- Fixes input file extension validataion for PDB files
- Implements data validation for input experimental files
- Clarifies documentation for these changes
- Addresses issues in #35

1.6.20 v0.1.15 (2023-01-16)

- Install IDPConformerGenerator while resolving the conda env
- Addresses bug in issue #33

1.6.21 v0.1.14 (2022-11-28)

- Automatically catch `CSpred` missing import issue
- Addresses issue #31

1.6.22 v0.1.13 (2022-09-30)

- Modify SAXS module to follow “format” in output

1.6.23 v0.1.12 (2022-09-29)

- Minor edits for the paper

1.6.24 v0.1.11 (2022-09-28)

- Bugfix for smFRET module with output formatting error

1.6.25 v0.1.10 (2022-09-20)

- Re-test for PR

1.6.26 v0.1.9 (2022-09-20)

- Add buttons on README

1.6.27 v0.1.8 (2022-09-19)

- Update authors
- Add ASCII art for SPyCi-PDB

1.6.28 v0.1.7 (2022-09-19)

- Create unit tests for internal calculators and parsers

1.6.29 v0.1.6 (2022-09-14)

- Edits to manuscript per Sept 7 comments from Dr. Julie Forman-Kay
- Add figure 1 to paper

1.6.30 v0.1.5 (2022-09-08)

- Edits to the manuscript per Aug 29 comments from Dr. Julie Forman-Kay
- Fix documentation error for RDC module
- Fix small issue of CS module output

1.6.31 v0.1.4 (2022-09-01)

- Update failing tests
- Upload manuscript, bibliography, and tests for JOSS (#21)

1.6.32 v0.1.3 (2022-09-01)

- Update RtD link in README.rst

1.6.33 v0.1.2 (2022-08-31)

- Minor fix to gitworkflows for tests

1.6.34 v0.1.1 (2022-08-31)

- Modularize all calculator components
- Remove Python 3.7 from requirements

1.6.35 v0.1.0 (2022-08-24)

- Lint everything

1.6.36 v0.0.15 (2022-08-24)

- Update README documentation
- Update ReadTheDocs format and associated docs

1.6.37 v0.0.14 (2022-08-23)

- Upgrade CS module for multiprocessing with UCBShift
- Update installation instructions for UCBShift

1.6.38 v0.0.13 (2022-08-22)

- Logic/module to link PALES v6.0 for RDC back-calculator (#14)
- Documentation for installing dependencies for PALES v6.0 for Ubuntu 20.04 LTS

1.6.39 v0.0.12 (2022-08-12)

- Logic/module to link HullRad for Rh back-calculator (#13)

1.6.40 v0.0.11 (2022-08-12)

- Logic/module to link CRYSOL 3.0 for SAXS back-calculator (#12)
- Documentation for installing CRYSOL 3.0 on top of SPyCi-PDB

1.6.41 v0.0.10 (2022-08-12)

- Logic/module to link UCBShift for CS back-calculator (#10)
- Documentation for installing UCBShift on top of SPyCi-PDB

1.6.42 v0.0.9 (2022-08-10)

- Logic/module for smFRET back-calculator (#9)

1.6.43 v0.0.8 (2022-08-10)

- Logic/module for NOE back-calculator (#8)
- Refractor get_pdb_paths

1.6.44 v0.0.7 (2022-08-10)

- Examples folder and some usage documentation (#7)

1.6.45 v0.0.6 (2022-08-10)

- Logic/module for JC back-calculator (#6)

1.6.46 v0.0.5 (2022-08-09)

- Logic/module for PRE back-calculator (#5)

1.6.47 v0.0.4 (2022-08-08)

- Documentation for installing IDPConformerGenerator as a library (#4)

1.6.48 v0.0.3 (2022-08-08)

- Core CLI backbone and base libs required (#2)
- Basic documentation for installation and updates

1.6.49 v0.0.2 (2022-08-08)

- Fix reference to python-project-skeleton (#3)

1.6.50 v0.0.1 (2022-07-28)

- Housekeeping items (#1)
- Building based on python-project-skeleton
- Renaming and changing base structure

CHAPTER TWO

INDICES AND TABLES

- genindex
- modindex
- search