electronDensityAnalysis Documentation

Release 1.0

Sen Yao, Hunter N.B. Moseley

Jan 17, 2022

Contents:

CHAPTER 1

User Guide

1.1 Description

The *pdb_eda* package provides a simple Python tool for parsing and analyzing electron density maps data available from the world wide Protein Data Bank (PDB).

The *pdb_eda* package currently provides facilities that can:

- Parse .ccp4 format file into their object representation.
- Parse .pdb format file to get information that is complimentary to the Bio.PDB module in BioPython package.
- Analyze the electron density maps at the atom/residue/domain levels and interpret the electron densities in terms of number of electrons by estimating a density-electron ratio.

1.2 Citation

Please cite the following papers when using pdb_eda:

Sen Yao and Hunter N.B. Moseley. "A chemical interpretation of protein electron density maps in the worldwide protein data bank" PLOS One 15, e0236894 (2020). https://doi.org/10.1371/journal.pone.0236894

Sen Yao and Hunter N.B. Moseley. "Finding high-quality metal ion-centric regions across the worldwide Protein Data Bank" Molecules 24, 3179 (2019). https://doi.org/10.3390/molecules24173179

1.3 Installation

pdb_eda runs under Python 3.4+ and is available through python3-pip. Install via pip or clone the git repo and install the following dependencies and you are ready to go!

1.3.1 Install on Linux, Mac OS X

```
python3 -m pip install pdb_eda
```

1.3.2 GitHub Package installation

Make sure you have git installed:

```
git clone https://github.com/MoseleyBioinformaticsLab/pdb_eda.git
```

1.3.3 Dependencies

pdb_eda requires the following Python libraries:

- Biopython for creating and analyzing the *pdb_eda* atom objects.
- **Cython for cythonizing low-level utility functions to improve computational performance.**
  - Requires gcc to be installed for the cythonization process.
- numpy and scipy for mathmatical calculations.
- docopt for better command line interface.
- jsonpickle for formatted and reusable output.
- **PyCifRW for reading Cif formatted files.**
  - Requires gcc to be installed for compiling components of the package.
- pymol for calculating crystal contacts. (This package is not required, except for this functionality).

To install dependencies manually:

```
pip3 install biopython
pip3 install cython
pip3 install numpy
pip3 install scipy
pip3 install docopt
pip3 install jsonpickle
pip3 install PyCifRW
```

1.4 Basic usage

The *pdb_eda* package can be used in several ways:

- As a library for accessing and manipulating data in PDB and CCP4 format files.
  - Use the *fromPDBid* generator function that will generate (yield) a single *densityAnalysis* instance at a time.
  - Process each *densityAnalysis* instance:
  - Generate symmetry atoms.
  - Generate red (negative density) or green (positive density) blob lists.
  - Process PDB structures to aggregate cloud.

  - Calculate atom blob list and statistics.
  - Calculate atom regional discrepancies and statistics.
  - Calculate residue regional discrepancies and statistics.
- As a command-line tool using the pdb_eda command (or "python3 -m pdb_eda").
  - The command-line interface has multiple modes.
  - **single - single-structure mode:**
    - Convert electron density map CCP4 files into its equivalent JSON file format.
    - Aggregate electron density map by atom, residue, and domain, and return the results in either JSON or csv format.
    - Aggregate difference electron density map into green (positive) or red (negative) blobs, and return the object or statistics results in either JSON or csv format.
    - Aggregate difference electron density map for atom and residue specific regions and return results in either JSON or csv format.
    - Return traditional quality metrics and statistics for atoms and residues.
  - **multiple - multiple-structure mode:**
    - Analyze and return cumulative statistics for a given list of PDB IDs.
    - Filter list of PDB IDs by cumulative statistic criteria.
    - Check and redownload problematic PDB entries.
    - Run single structure mode with multicore processing.
    - Run crystal contacts mode with multicore processing.
  - **contacts - crystal contacts mode:**
    - Analyze and return atoms with crystal contacts.
    - This mode requires pymol to be installed.
  - **generate - parameter generation mode: (rarely used mode)**
    - Downloads PDB chemical component list and extracts information to create atom type parameters.
    - Analyzes list of PDB IDs for specific atom types.
    - Generates atom type parameter file and list of PDB IDs for their optimization.
  - **optimize - parameter optimization mode: (rarely used mode)**
    - Optimizes atom type radii and b-factor density correction slopes using a given list of PDB IDs.

1.5 CHANGELOG

Since version 1.0.1, over 2200 lines of additional code has been written and most of the code base has been revised and refactored. Computationally intensive parts of the code have been cythonized to improve execution performance. Many variables and functions have been renamed to greatly improve readability and understanding of the code base, API, and CLI.

The application programming interface (API) has been greatly expanded and much of the functionality streamlined.

The command line interface has been greatly expanded and now includes single, multiple, contacts, generate, and optimize modes.

Optimize mode has a new penalty function being optimized that both minimizes differences in density-electron ratio estimates and maximizes electron cloud aggregation. The optimization is also roughly 10-fold faster than the previous generation of algorithm.

The atom types have been systematically generated from the wwPDB master chemical components file. Both amino acid and nucleic acid type parameters have been optimized. So both protein and nucleic acid PDB entries can be analyzed now.

1.6 License

CHAPTER 2

The pdb_eda Tutorial

The *pdb_eda* package provides classes and other methods for analyzing electron density maps data available from the worldwide Protein Data Bank (PDB). It also provides simple command-line interface.

2.1 Using pdb_eda as a library

2.1.1 Importing pdb_eda package

If the *pdb_eda* is installed, it can be imported:

```
import pdb_eda
```

2.1.2 Constructing densityAnalysis instance

The densityAnalysis module provides the *fromPDBid()* function that returns *densityAnalysis* instance. Constructing a *densityAnalysis* instance only requires a PDB id:

```
pdbid = '1cbs'
analyzer = densityAnalysis.fromPDBid(pdbid)
```

The analyzer will only be generated if its .pdb and .ccp4 files exist (valid PDB id), either locally or can be download on the fly. Or otherwise it will return zero.

2.1.3 Accessing the PDB data

The PDB data can be accessed through the **biopdbObj** and **pdbObj**:

```
analyzer.biopdbObj
analyzer.pdbObj
```

The **biopdbObj** is a Biopython data member instance, and the **pdbObj** is a *pdb_eda.pdbParser.PDBentry* instance that includes some information that is not available in the Biopython instance, such as space group, or rotational matrices.

The information about how to use and access data from the **biopdbObj** instance can be found at Biopython.

The header information in **pdbObj** can be accessed through *header* attribute as a data member:

```
rValue = analyzer.pdbObj.header.rValue
spaceGroup = analyzer.pdbObj.header.spaceGroup
```

The available keys include date, method, pdbid, rFree, rValue, resolution, rotationMats, and spaceGroup. Atom information is optional if running in *lite* mode.

2.1.4 Accessing the CCP4 data

The CCP4 data can be accessed through the **densityObj** and **diffDensityObj** data members:

```
analyzer.densityObj
analyzer.diffDensityObj
```

They both contain the header information and the density map from the CCP4 standard map file. Their header information should be the same, while **densityObj** contains the 2Fo - Fc density map and **diffDensityObj** contains Fo - Fc density map. The header information can be accessed through *header* attribute as a data member:

```
alpha = analyzer.densityObj.header.alpha
xlength = analyzer.densityObj.header.xlength
```

The density map is available in both 1-d and 3-d array:

```
oneDmap = analyzer.densityObj.densityArray
threeDmap = analyzer.densityObj.density
```

You also have access to several methods that help manipulate the ccp4 data, for example, to get the point density from a set of given xyz coordinates:

```
analyzer.densityObj.getPointDensityFromXyz([10.1, 15.2, 24.4])
# 1.3517704010009766
```

A full list of methods can be found at the **API Reference**.

2.1.5 Analyzing the electron density data

There are several methods you can use to perform on the electron density data. To aggregate the electron density map (2Fo - Fc) by atom, residue, and domain:

```
analyzer.aggregateCloud()
medians = analyzer.medians
densityElectronRatio = analyzer.densityElectronRatio
```

To aggregate the difference electron density map (Fo - Fc) into positive (green) and negative (red) blobs:

```
greenBlobList = analyzer.greenBlobList
redBlobList = analyzer.redBlobList
```

To aggregate the electron density map (2Fo - Fc) into positive (blue) blobs:

```
blueBlobList = analyzer.blueBlobList
```

To acquire a list all nearby symmetry, symmetry-only, or asymmetry atoms:

```
symmetryAtoms = analyzer.symmetryAtoms
symmetryOnlyAtoms = analyzer.symmetryOnlyAtoms
asymmetryAtoms = analyzer.asymmetryAtoms
```

To acquire a list all nearby symmetry, symmetry-only, or asymmetry coordinate lists:

```
symmetryAtomCoords = analyzer.symmetryAtomCoords
symmetryOnlyAtomCoords = analyzer.symmetryOnlyAtomCoords
asymmetryAtomCoords = analyzer.asymmetryAtomCoords
```

The result is a list of `pdb_eda.densityAnalysis.symAtom` instances.

To calculate the summary statistics of the above positive and negative density blobs with respect to their closest symmetry atom:

```
diffMapAtomBlobStatistics = analyzer.calcAtomSpecificBlobStatistics()
```

For more detailed information, check the **API Reference**.

2.2 Using pdb_eda in the command-line interface

Some of the above functions can be accessed from the command line interface:

```
Either the "pdb_eda" command or "python3 -m pdb_eda" can be used to run the command␣
↪line interface.

> pdb_eda -h

pdb_eda command-line interface

Usage:
    pdb_eda -h | --help     for this screen.
    pdb_eda --full-help     help documentation on all modes.
    pdb_eda --version       for the version of pdb_eda.
    pdb_eda single ...      for single structure analysis mode. (Most useful command␣
↪line mode).
    pdb_eda multiple ...    for multiple structure analysis mode. (Second most useful␣
↪command line mode).
    pdb_eda contacts ...    for crystal contacts analysis mode.  (Third most useful␣
↪command line mode).
    pdb_eda generate ...    for generating starting parameters file that then needs␣
↪to be optimized. (Rarely used mode).
    pdb_eda optimize ...    for parameter optimization mode. (Rarely used mode).

For help on a specific mode, use the mode option -h or --help.
For example:
    pdb_eda single --help   for help documentation about single structure analysis␣
↪mode.
```

Using single mode to sum significant (> 3 std.dev) deviations in a 3.5 angstrom spherical region around atoms:

```
pdb_eda single 3UBK 3ubk.txt difference --atom --radius=3.5 --num-sd=3 --out-
↪format=csv --include-pdbid
```

Using single mode to sum significant (> 3 std.dev) deviations in a 5 angstrom spherical region around residues:

```
pdb_eda single 3UBK 3ubk.txt difference --residue --radius=5 --num-sd=3 --out-
↪format=csv --include-pdbid
```

Using single mode to return all green difference blobs and their closest symmetry atom:

```
pdb_eda single 3UBK 3ubk.green_blobs.txt blob --green --out-format=csv --include-pdbid
```

Using multiple mode to return summative analysis results for a list of PDB IDs:

```
pdb_eda multiple pdbids.txt results/result.txt
```

Using multiple mode to run single mode with multiprocessing:

```
pdb_eda multiple pdbids.txt results/ --single-mode="--atom --radius=3.5 --num-sd=3 --
↪out-format=csv --include-pdbid"
```

Using multiple mode to check and redownload entry and ccp4 files for a given set of PDB IDs:

```
pdb_eda multiple pdbids.txt --reload
```

CHAPTER 3

The pdb_eda API Reference

3.1 Electron Density Analysis

3.1.1 PDB Electron Density Analysis (pdb_eda.densityAnalysis)

This module provides methods for the creation of the *pdb_eda.densityAnalysis* class given a PDB id, along with methods to analyze its 2Fo-Fc and Fo-Fc electron density maps.

pdb_eda.densityAnalysis.**setGlobals**(*params*)
Sets global parameters. Typically used for optimizing parameters.

Parameters **params** (dict) – dictionary of parameters needed for pdb_eda calculations.

pdb_eda.densityAnalysis.**loadF000Parameters**()
Loads and assigns global parameters needed for F000 estimation.

pdb_eda.densityAnalysis.**fromPDBid**(*pdbid*, *ccp4density=True*, *ccp4diff=True*, *pdbbio=True*, *pdbi=True*, *downloadFile=True*, *mmcif=False*)
Creates *pdb_eda.densityAnalysis.DensityAnalysis* object given the PDB id if the id is valid and the structure has electron density file available.

Parameters

- **pdbid** (str) – PDB entry ID.
- **ccp4density** (bool) – Whether to generate ccp4 density object, defaults to True.
- **ccp4diff** (bool) – Whether to generate in default of ccp4 difference density object, defaults to True.
- **pdbbio** (bool) – Whether to generate in default of bio.PDB object, defaults to True.
- **pdbi** (bool) – Whether to generate in default of PDB object, defaults to True.
- **downloadFile** (bool) – Whether to save the downloaded ccp4 density, ccp4 difference density, and PDB file, defaults to True.
- **mmcif** (bool) – Whether to download the mmCif file, defaults to False.

Returns densityAnalysis object

Return type *pdb_eda.densityAnalysis.DensityAnalysis*

pdb_eda.densityAnalysis.**fromFile**(*pdbFile*, *ccp4DensityFile=None*, *ccp4DiffDensityFile=None*)

Creates *pdb_eda.densityAnalysis.DensityAnalysis* object given the appropriate PDB and CCP4 files.

Parameters

- **pdbFile** (str, io.IOBase) – PDB entry file.
- **ccp4DensityFile** (str, io.IOBase, optional) – ccp4 density file.
- **ccp4DiffDensityFile** (str, io.IOBase, optional) – ccp4 difference density file.

Returns densityAnalysis object

Return type *pdb_eda.densityAnalysis.DensityAnalysis*

pdb_eda.densityAnalysis.**cleanPDBid**(*pdbid*)

Removes PDB entry, ccp4, and mmcif files associated with a PDB id.

Parameters **pdbid** (str) – PDB entry ID.

Returns bool whether the operation was successful.

Return type bool

pdb_eda.densityAnalysis.**testCCP4URL**(*pdbid*)

Test whether the pdbid has electron density maps by querying if the PDBe API has electron density statistics. :param pdbid: PDB entry ID. :type pdbid: str

Returns bool on test success.

Return type bool

class pdb_eda.densityAnalysis.**DensityAnalysis**(*pdbid*, *densityObj=None*, *diffDensityObj=None*, *biopdbObj=None*, *pdbObj=None*)

DensityAnalysis class that stores the density, difference density, bio.PDB, and PDB objects.

__init__(*pdbid*, *densityObj=None*, *diffDensityObj=None*, *biopdbObj=None*, *pdbObj=None*)

densityAnalysis initializer. Leave *densityObj*, *diffDensityObj*, *biopdbObj* and *pdbObj* as None to be created. They are not required for initialization but could be required for some methods.

Parameters

- **pdbid** (str) – PDB entry ID.
- **densityObj** (*pdb_eda.ccp4.DensityMatrix*, optional) – DensityMatrix object.
- **diffDensityObj** (*pdb_eda.ccp4.DensityMatrix*, optional) – optional DensityMatrix object.
- **biopdbObj** (Bio.PDB.Structure.Structure, optional) – optional Bio.PDB Structure object.
- **pdbObj** (*pdb_eda.pdbParser.PDBentry*, optional) – optional PDBentry object.

symmetryAtoms

Returns list of symmetry atoms.

Returns symmetryAtoms

Return type list

symmetryOnlyAtoms
Returns list of non-[0,0,0,0] symmetry atoms.

Returns symmetryAtoms

Return type list

asymmetryAtoms
Returns list of [0,0,0,0] symmetry atoms.

Returns symmetryAtoms

Return type list

symmetryAtomCoords
Returns list of symmetry atom xyz coordinates.

Returns symmetryAtomCoords

Return type list

symmetryOnlyAtomCoords
Returns list of non-[0,0,0,0] symmetry atom xyz coordinates.

Returns symmetryAtomCoords

Return type list

asymmetryAtomCoords
Returns list of [0,0,0,0] symmetry atom xyz coordinates.

Returns symmetryAtomCoords

Return type list

greenBlobList
Returns list of all positive significant difference density blobs.

Returns greenBlobList

Return type list

redBlobList
Returns list of all negative significant difference density blobs.

Returns redBlobList

Return type list

blueBlobList
Returns list of all positive significant density blobs.

Returns blueBlobList

Return type list

fc
Returns the Fc map = 2Fo-Fc - Fo-Fc.

Returns fc

Return type *pdb_eda.ccp4.DensityMatrix*

fo
Returns the Fo map = 2Fo-Fc.

Returns fo

Return type *pdb_eda.ccp4.DensityMatrix*

F000
Returns estimated F000. This estimate may not be that accurate.

Returns F000

Return type float

medians
Returns median field values calculated per atom type.

Returns medians

Return type numpy.array

atomCloudDescriptions
Returns aggregated atom cloud descriptions.

Returns atomCloudDescriptions

Return type numpy.array

residueCloudDescriptions
Returns aggregated residue cloud descriptions.

Returns residueCloudDescriptions

Return type list

domainCloudDescriptions
Returns aggregated domain cloud descriptions.

Returns domainCloudDescriptions

Return type list

numVoxelsAggregated
Returns number of aggregated voxels in cloud analysis.

Returns numVoxelsAggregated

Return type int

totalAggregatedElectrons
Returns total amount of aggregated electrons in cloud analysis.

Returns totalAggregatedElectrons

Return type float

totalAggregatedDensity
Returns total amount of aggregated density in cloud analysis.

Returns totalAggregatedDensity

Return type float

densityElectronRatio
Returns the density-electron ratio estimated from cloud analysis.

Returns densityElectronRatio

Return type float

atomTypeOverlapCompleteness
Returns atom-type overlap completeness counts.

Returns atomTypeOverlapCompleteness

Return type dict

atomTypeOverlapIncompleteness
Returns atom-type overlap incompleteness counts.

Returns atomTypeOverlapIncompleteness

Return type dict

aggregateCloud(*minCloudElectrons=25.0*, *minTotalElectrons=400.0*)
Aggregate the electron density map clouds by atom, residue, and domain. Calculate and populate *densityAnalysis.densityElectronRatio* and *densityAnalysis.medians* data members.

Parameters

- **minCloudElectrons** (float) – minimum number of electrons needed to aggregate a residue or domain cloud., defaults to 25.0
- **minTotalElectrons** (float) – mininum number of electrons required to calculate a density-electron ratio., defaults to 400.0

medianAbsFoFc()
Calculates median absolute values for the Fo and Fc maps less than 1 sigma. These values should be comparable, i.e. low relative difference, for RSCC and RSR metric calculations.

Returns tuple of median abs values from fo and fc maps respectively.

Return type tuple

residueMetrics(*residueList=None*)
RETURNS rscc and rsr statistics for each residue using the Fo and Fc density maps.

Parameters **residueList** (list, optional) –

Returns results

Return type list

atomMetrics(*atomList=None*)
RETURNS rscc and rsr statistics for each residue using the Fo and Fc density maps.

Parameters **atomList** (list, optional) –

Returns results

Return type list

calculateRsccRsrMetrics(*crsList*)
Calculates and returns RSCC and RSR metrics. This method of calculating RSCC and RSR assumes that the Fo and Fc maps are appropriately scaled. Comparison of median absolute values below one sigma should be quite similar between Fo and Fc maps.

Parameters **crsList** (list, set) –

Returns rscc_rsr_arrays_tuple

Return type tuple

_calculateSymmetryAtoms()
Calculate all the symmetry and nearby cells and keep those have at least on atom within 5 grid points

of the non-repeating crs boundary. Ref: Biomolecular Crystallography: Principles, Practice, and Application to Structural Biology by Bernhard Rupp. Orthogonalization matrix O and deororthogonalization matrix O' are from *pdb_eda.ccp4* object. Rotation matrix R and Translation matrix T is from *pdb_eda.pdbParser* object. The neighbouring cells can be calculated using formula, X' = O(O'(RX + T) + T') = OO'(RX+T) + OT' = RX + T + O[-1/0/1,-1/0/1,-1/0/1]. Assign the list of `pdb_eda.densityAnalysis.symAtom` instances to *densityAnalysis.symmetryAtoms* data member

calculateAtomSpecificBlobStatistics(*blobList*)
Calculate atom-specific blob statistics.

Parameters **blobList** (`list`) – list of blobs to calculate statistics for.

Return blobStats Difference density map statistics.

Return type `list`

calculateAtomRegionDensity(*radius*, *numSD=1.5*, *type=''*, *useOptimizedRadii=False*)
Calculates significant region density in a given radius of each atom.

Parameters

- **radius** (`float`) – the search radius.
- **numSD** (py:class:*float*) – number of standard deviations of significance, defaults to 1.5
- **type** (`str`) – atom type to filter on., defaults to ""

Return diffMapRegionStats density map region statistics per atom.

Return type `list`

calculateSymmetryAtomRegionDensity(*radius*, *numSD=1.5*, *type=''*, *useOptimizedRadii=False*)
Calculates significant region density in a given radius of each symmetry atom.

Parameters

- **radius** (`float`) – the search radius.
- **numSD** (`float`) – number of standard deviations of significance., defaults to 1.5
- **type** (`str`) – atom type to filter on., defaults to ""

Return diffMapRegionStats density map region statistics per atom.

Return type `list`

calculateResidueRegionDensity(*radius*, *numSD=1.5*, *type=''*, *atomMask=None*, *useOptimizedRadii=False*)
Calculates significant region density in a given radius of each residue.

Parameters

- **radius** (`float`) – the search radius.
- **numSD** (`float`) – number of standard deviations of significance., defaults to 1.5
- **type** (`dict`, optional) – atom type to filter on., defaults to ""
- **atomMask** – residue specific atom mask.

Return diffMapRegionStats density map region statistics per residue.

Return type `list`

calculateRegionDensity(*xyzCoordList*, *radius*, *numSD=1.5*, *testValidCrs=False*)
Calculate region-specific density from the electron density matrix.

Parameters

- **xyzCoordLists** – single xyz coordinate or a list of xyz coordinates.
- **radius** (float or list) – the search radius or list of search radii.
- **numSD** (float) – number of standard deviations of significance., defaults to 1.5
- **testValidCrs** (bool) – whether to test crs are valid and return the results., defaults to False

Return diffMapRegionStats density map region statistics and optional validCrs result.

Return type list, tuple

calculateAtomRegionDiscrepancies(*radius*, *numSD=3.0*, *type=''*)
Calculates significant region discrepancies in a given radius of each atom.

Parameters

- **radius** (float) – the search radius.
- **numSD** (py:class:*float*) – number of standard deviations of significance, defaults to 3.0
- **type** (str) – atom type to filter on., defaults to ''

Return diffMapRegionStats Difference density map region statistics per atom.

Return type list

calculateSymmetryAtomRegionDiscrepancies(*radius*, *numSD=3.0*, *type=''*)
Calculates significant region discrepancies in a given radius of each symmetry atom.

Parameters

- **radius** (float) – the search radius.
- **numSD** (float) – number of standard deviations of significance., defaults to 3.0
- **type** (str) – atom type to filter on., defaults to ''

Return diffMapRegionStats Difference density map region statistics per atom.

Return type list

calculateResidueRegionDiscrepancies(*radius*, *numSD=3.0*, *type=''*, *atomMask=None*)
Calculates significant region discrepancies in a given radius of each residue.

Parameters

- **radius** (float) – the search radius.
- **numSD** (float) – number of standard deviations of significance., defaults to 3.0
- **type** (dict, optional) – atom type to filter on., defaults to ''
- **atomMask** – residue specific atom mask.

Return diffMapRegionStats Difference density map region statistics per residue.

Return type list

calculateRegionDiscrepancy(*xyzCoordList*, *radius*, *numSD=3.0*, *testValidCrs=False*)
Calculate region-specific discrepancy from the difference density matrix.

Parameters

- **xyzCoordLists** – single xyz coordinate or a list of xyz coordinates.
- **radius** (float) – the search radius.

- **numSD** (float) – number of standard deviations of significance., defaults to 3.0
- **testValidCrs** (bool) – whether to test crs are valid and return the results., defaults to False

Return diffMapRegionStats Difference density map region statistics and optional validCrs result.

Return type list, tuple

estimateF000()
Estimate the F000 term as sum of all electrons over the unit cell volume

Returns estimatedF000

Return type float

pdb_eda.densityAnalysis.**residueAtomName**(*atom*)
Returns a combined residue and atom name used to select an atom type.

Parameters **atom** (Bio.PDB.atom) –

Returns fullAtomName

Return type str

3.2 CCP4 Parser

3.2.1 CCP4 Parser (pdb_eda.ccp4)

This module provides methods to read and parse the CCP4 format files, returning ccp4 objects. Format details of ccp4 can be found in http://www.ccp4.ac.uk/html/maplib.html.

pdb_eda.ccp4.**readFromPDBID**(*pdbid*, *verbose=False*)
Creates *pdb_eda.ccp4.DensityMatrix* object.

Parameters

- **pdbid** (str) – PDB entry ID.
- **verbose** (bool) – verbose mode, defaults to False

Returns densityMatrix

Return type *pdb_eda.ccp4.DensityMatrix*

pdb_eda.ccp4.**readFromURL**(*url*, *pdbid=None*, *verbose=False*)
Creates *pdb_eda.ccp4.DensityMatrix* object.

Parameters

- **url** (str) –
- **pdbid** (str, optional) – PDB entry ID.
- **verbose** (bool) – verbose mode, defaults to False

Returns densityMatrix

Return type *pdb_eda.ccp4.DensityMatrix*

pdb_eda.ccp4.**read**(*ccp4Filename*, *pdbid=None*, *verbose=False*)
Creates *pdb_eda.ccp4.DensityMatrix* object.

Parameters

- **ccp4Filename** (str) – .ccp4 filename including path.
- **pdbid** (str, optional) – PDB entry ID.
- **verbose** (bool) – verbose mode, defaults to False

Returns densityMatrix

Return type *pdb_eda.ccp4.DensityMatrix*

pdb_eda.ccp4.**parse**(*handle*, *pdbid*, *verbose=False*)

Creates *pdb_eda.ccp4.DensityMatrix* object.

Parameters

- **handle** (io.IOBase) – an I/O handle for .ccp4 file.
- **pdbid** (str) – PDB entry ID.
- **verbose** (bool) – verbose mode, defaults to False

Returns densityMatrix

Return type *pdb_eda.ccp4.DensityMatrix*

class pdb_eda.ccp4.**DensityHeader**(*headerTuple*, *labels*, *endian*)

pdb_eda.ccp4.DensityHeader that stores information about ccp4 header.

classmethod fromFileHeader(*fileHeader*)

RETURNS *pdb_eda.ccp4.DensityHeader* object given the fileHeader.

Parameters **fileHeader** (bytes) – ccp4 file header.

Returns densityHeader

Return type *pdb_eda.ccp4.DensityHeader*

__init__(*headerTuple*, *labels*, *endian*)

Initialize the DensityHeader object, assign values to data members accordingly, and calculate some metrics that will be used frequently.

Parameters

- **headerTuple** (tuple) – The ccp4 header information (excluding labels) in a tuple.
- **labels** (bytes) – The labels field in a ccp4 header.
- **endian** (str) – The endianness of the file.

_calculateOrigin()

Calculate the xyz coordinates from the header information.

Returns xyz coordinates.

Return type list of float.

xyz2crsCoord(*xyzCoord*)

Convert the xyz coordinates into crs coordinates.

Parameters **xyzCoord** (list of float) – xyz coordinates.

Returns crs coordinates.

Return type A list of int.

crs2xyzCoord(*crsCoord*)

Convert the crs coordinates into xyz coordinates.

Parameters **crsCoord** (A list of int) – crs coordinates.

Returns xyz coordinates.

Return type A list of float.

class pdb_eda.ccp4.**DensityMatrix**(*header*, *origin*, *density*, *pdbid*)

pdb_eda.ccp4.DensityMatrix that stores data and methods of a ccp4 file.

__init__(*header*, *origin*, *density*, *pdbid*)

Initialize the *pdb_eda.ccp4.DensityMatrix* object.

Parameters

- **header** (*pdb_eda.ccp4.DensityHeader*) –
- **origin** (list) – the xyz coordinates of the origin of the first number of the density data.
- **density** (tuple) – the density data as a 1-d list.
- **pdbid** (str) – PDB entry ID

meanDensity

Returns mean of the density.

Returns mean

Return type float

stdDensity

Returns the standard deviation of the density.

Returns std

Return type float

getTotalAbsDensity(*densityCutoff*)

Returns total absolute Density above a densityCutoff

Parameters **densityCutoff** (float) –

Returns totalAbsDensity

Return type float

getPointDensityFromCrs(*crsCoord*)

Get the density of a point.

Parameters **crsCoord** (A list of int) – crs coordinates.

Returns pointDensity

Return type float

getPointDensityFromXyz(*xyzCoord*)

Get the density of a point.

Parameters **xyzCoord** (A list of float) – xyz coordinates.

Returns pointDensity

Return type float

getSphereCrsFromXyz(*xyzCoord*, *radius*, *densityCutoff=0*)

Calculate a list of crs coordinates that within a given distance of a point.

Parameters

- **xyzCoord** (A list of float) – xyz coordinates.

- **radius** (float) –
- **densityCutoff** (float) – a density cutoff for all the points wants to be included, defaults to 0 Default 0 means include every point within the radius. If cutoff < 0, include only points with density < cutoff. If cutoff > 0, include only points with density > cutoff.

Returns crsList

Return type list

getTotalDensityFromXyz (*xyzCoord*, *radius*, *densityCutoff=0*)
Calculate the total density of a sphere.

Parameters

- **xyzCoord** (list of float) – xyz coordinates.
- **radius** (float) –
- **densityCutoff** (float) – a density cutoff for all the points to include, defaults to 0 Default 0 means include every point within the radius. If cutoff < 0, include only points with density < cutoff. If cutoff > 0, include only points with density > cutoff.

Returns density

Return type float

findAberrantBlobs (*xyzCoords*, *radius*, *densityCutoff=0*)
Within a given radius, find and aggregate all neighbouring aberrant points into blobs (red/green meshes).

Parameters

- **xyzCoords** (list) – single xyz coordinate or a list of xyz coordinates.
- **radius** (float or list) – search radius or list of search radii
- **densityCutoff** (float) – A density cutoff for all the points wants to be included, defaults to 0 Default 0 means include every point within the radius. If cutoff < 0, include only points with density < cutoff. If cutoff > 0, include only points with density > cutoff.

Returns blobList of aberrant blobs described by their xyz centroid, total density, and volume.

Return type list of *pdb_eda.ccp4.DensityBlob* objects.

createFullBlobList (*cutoff*)
Aggregate the density map into positive (green or blue) or negative (red) blobs.

Parameters **cutoff** (float) – density cutoff to use to filter voxels.

Return blobList list of DensityBlobs

Return type list of *pdb_eda.ccp4.DensityBlob* objects.

createBlobList (*crsList*)
Calculates a list of blobs from a given crsList.

Parameters **crsList** (list, set) – a crs list.

Returns blobList

Return type list of *pdb_eda.ccp4.DensityBlob* objects.

class pdb_eda.ccp4.**DensityBlob** (*centroid*, *coordCenter*, *totalDensity*, *volume*, *crsList*, *densityMatrix*, *atoms=None*)
pdb_eda.ccp4.DensityBlob that stores data and methods of a electron density blob.

__init__ (*centroid*, *coordCenter*, *totalDensity*, *volume*, *crsList*, *densityMatrix*, *atoms=None*)
Initialize a *pdb_eda.ccp4.DensityBlob* object.

Parameters

- **centroid** (list) – the centroid of the blob.
- **totalDensity** (float) – the totalDensity of the blob.
- **volume** (float) – the volume of the blob = number of density units * unit volumes.
- **crsList** (list) – the crs list of the blob.
- **densityMatrix** (*pdb_eda.ccp4.DensityMatrix*) – the entire density map that the blob belongs to.
- **atoms** (list, optional) – list of atoms for the blob.

Returns densityBlob

Return type *pdb_eda.ccp4.DensityBlob*

static fromCrsList(*crsList*, *densityMatrix*)
The creator of a A *pdb_eda.ccp4.DensityBlob* object.

Parameters

- **crsList** (list) – the crs list of the blob.
- **densityMatrix** (*pdb_eda.ccp4.DensityMatrix*) – the 3-d density matrix to use for calculating centroid etc, so the object does not have to have a density list data member.

Returns densityBlob

Return type *pdb_eda.ccp4.DensityBlob*

__eq__(*otherBlob*)
Check if two blobs are the same, and overwrite the '==' operator for the *pdb_eda.ccp4.DensityBlob* object.

Parameters **otherBlob** (*pdb_eda.ccp4.DensityBlob*) –

Returns bool

Return type :py:class'bool'

testOverlap(*otherBlob*)
Check if two blobs overlaps or right next to each other.

Parameters **otherBlob** (*pdb_eda.ccp4.DensityBlob*) –

Returns bool

Return type :py:class'bool'

merge(*otherBlob*)
Merge the given blob into the original blob.

Parameters **otherBlob** (*pdb_eda.ccp4.DensityBlob*) –

clone()
Returns a copy of the density blob.

Returns densityBlob

Return type *pdb_eda.ccp4.DensityBlob*

3.3 PDB Parser

3.3.1 PDB Parser (pdb_eda.pdbParser)

This module provides methods to read and parse the PDB format files and returns PDB objects. Format details of PDB can be found in ftp://ftp.wwpdb.org/pub/pdb/doc/format_descriptions/Format_v33_Letter.pdf.

pdb_eda.pdbParser.**readPDBfile**(*file*)
Creates *pdb_eda.pdbParser.PDBentry* object from file name.

Parameters **file** (str, io.IOBase) – The name of a PDB formated file or a file handle.

pdb_eda.pdbParser.**parse**(*handle*, *mode='lite'*)
Creates *pdb_eda.pdbParser.PDBentry* object from file handle object.

Parameters

- **handle** (io.IOBase) – The file handle of a PDB formatted file.
- **mode** (str) – Whether of not to parse all the atoms. 'lite' means do not to parse., defaults to 'lite'

Returns pdbEntry

Return type *pdb_eda.pdbParser.PDBentry*

class pdb_eda.pdbParser.**PDBentry**(*header*, *atoms*)
pdb_eda.pdbParser.PDBentry class that stores the *pdb_eda.pdbParser.PDBheader* and/or *pdb_eda.pdbParser.Atom* class.

__init__(*header*, *atoms*)
pdb_eda.pdbParser.PDBentry initializer.

Parameters

- **header** (*pdb_eda.pdbParser.PDBheader*) –
- **atoms** (list) – list of *pdb_eda.pdbParser.Atom* objects

class pdb_eda.pdbParser.**PDBheader**(*PDBid*, *date*, *method*, *resolution*, *rValue*, *rFree*, *program*, *spaceGroup*, *rotationMats*)
pdb_eda.pdbParser.PDBheader that stores information about PDB header.

__init__(*PDBid*, *date*, *method*, *resolution*, *rValue*, *rFree*, *program*, *spaceGroup*, *rotationMats*)
pdb_eda.pdbParser.PDBheader initializer.

Parameters

- **PDBid** (str) – PDB entry ID.
- **date** (str) – PDB structure publish date.
- **method** (str) – Experiment method, i.e. X-ray, NMR, etc.
- **resolution** (float) – Structure resolution if applicable.
- **rValue** (float) – Structure's R value.
- **rFree** (float) – Structure's R free value.
- **program** (str) – Software for acquiring the structure.
- **spaceGroup** (str) – Structure's space group if applicable.
- **rotationMats** (list) – Structure's rotation matrix and translation matrix if applicable.

class pdb_eda.pdbParser.**Atom**(*keyValues*)
pdb_eda.pdbParser.Atom that stores information about PDB atoms.

__init__(*keyValues*)
pdb_eda.pdbParser.Atom initializer.

Parameters **keyValues** (dict) – key-value pairs for atom information.

3.4 Crystal Contacts Analysis

pdb_eda (crystal) contacts analysis mode command-line interface Analyzes atoms in a PDB entry for the closest crystal contacts. This mode requires the pymol package and associated python library to be installed.

Usage: pdb_eda contacts -h | –help pdb_eda contacts <pdbid> <out-file> [–distance=<cutoff>] [–symmetry-atoms] [–include-pdbid] [–out-format=<format>]

Options:

-h, --help Show this screen.

<pdbid> The PDB ID to download and analyze. <out-file> Output filename. "-" will write to standard output. –distance=<cutoff> Distance cutoff in angstroms for detecting crystal contacts [default: 5.0]. –symmetry-atoms Calculate crystal contacts to symmetry atoms too. –include-pdbid Include PDB ID at the beginning of each result. –out-format=<format> Output file format, available formats: csv, json [default: json].

pdb_eda.crystalContacts.**findCoordContacts**(*coordList1*, *coordList2*, *distanceCutoff=5.0*)
Find contacts in coordList1 to coordList2 at the given distance cutoff.

Parameters

- **coordList1** (list) – list of coordinates.
- **coordList2** (py:class:*list*) – list of coordinates.
- **distanceCutoff** (float) – distance cutoff., defaults to 5.0

Returns contactList of index,minDistance tuples.

Return type list

pdb_eda.crystalContacts.**simulateCrystalNeighborCoordinates**(*filename*, *distanceCutoff=5.0*)
RETURN a list of atom coordinates of the simulated crystal environment surrounding the X-Ray Diffraction asymmetric unit (excluding heteroatoms). Requires a file path instead of a structure because the bulk of this is handled by Pymol. NOTE: This will only work with PDB structures resolved with X-RAY DIFFRACTION.

Parameters

- **filename** (str) –
- **distanceCutoff** (float) – distance cutoff., defaults to 5.0

Returns coordList

Return type list

3.5 Utilities

3.5.1 Utilities (pdb_eda.utils)

Contains low-level functions used in pdb_eda.ccp4 and pdb_eda.densityAnalysis. Should not be used, but is a fall-back if cutils.pyx cannot be cythonized. The cythonized version provide a 3- to 4-fold improvement in execution performance.

pdb_eda.utils.**testOverlap**(*selfBlob*, *otherBlob*)
Check if two blobs overlaps or right next to each other.

Parameters

- **selfBlob** (*pdb_eda.ccp4.DensityBlob*) –
- **otherBlob** (*pdb_eda.ccp4.DensityBlob*) –

Returns bool

Return type bool

pdb_eda.utils.**sumOfAbs**(*array*, *cutoff*)
Return sum of absolute values above a cutoff.

Parameters

- **array** (collections.abc.Iterable) –
- **cutoff** (float) –

Returns value

Return type float

pdb_eda.utils.**createCrsLists**(*crsList*)
Calculates a list of crsLists from a given crsList. This is a preparation step for creating blobs.

Parameters **crsList** (list) – a crs list.

Returns crsLists is a list of disjoint crsLists.

Return type list

pdb_eda.utils.**createSymmetryAtoms**(*atomList*, *rotationMats*, *orthoMat*, *xs*, *ys*, *zs*)
Creates and returns a list of all symmetry atoms.

Parameters

- **atomList** (list) –
- **rotationMats** (list) –
- **orthoMat** (list) –
- **xs** (list) –
- **ys** (list) –
- **zs** (py:class:*list*) –

Returns allAtoms

Return type list

class pdb_eda.utils.**SymAtom**(*atom*, *coord*, *symmetry*)
A wrapper class to the *BioPDB.atom* class, delegating all BioPDB atom class methods and data members except having its own symmetry and coordination.

__init__(*atom*, *coord*, *symmetry*)
pdb_eda.densityAnalysis.symAtom initializer.

Parameters

- **atom** (Bio.PDB.atom) – atom object.
- **coord** (list) – x,y,z coordinates.
- **symmetry** (list) – i,j,k,r symmetry

pdb_eda.utils.**getPointDensityFromCrs**(*densityMatrix*, *crsCoord*)
Returns the density of a point.

Parameters

- **densityMatrix** (*pdb_eda.ccp4.DensityMatrix*) –
- **crsCoord** – crs coordinates.

Type list, set

Returns density

Return type float

pdb_eda.utils.**testValidCrs**(*densityMatrix*, *crsCoord*)
Tests whether the crs coordinate is valid.

Parameters

- **densityMatrix** (*pdb_eda.ccp4.DensityMatrix*) –
- **crsCoord** (list) – crs coordinates.

Returns bool

Return type bool

pdb_eda.utils.**testValidCrsList**(*densityMatrix*, *crsList*)
Tests whether all of the crs coordinates in the list are valid.

Parameters

- **densityMatrix** (*pdb_eda.ccp4.DensityMatrix*) –
- **crsList** – list of crs coordinates.

Type list, set

Returns bool

Return type bool

pdb_eda.utils.**createFullCrsList**(*densityMatrix*, *cutoff*)
Returns full crs list for the density matrix.

Parameters

- **densityMatrix** (*pdb_eda.ccp4.DensityMatrix*) –
- **cutoff** (float) –

Returns crsList

Return type list

pdb_eda.utils.**_testXyzWithinDistance**(*xyzCoord1*, *xyzCoord2*, *distance*)
Tests whether two xyzCoords are within a certain distance.

Parameters

- **xyzCoord1** (list) –
- **xyzCoord2** (list) –
- **distance** (float) –

Returns bool

Return type bool

pdb_eda.utils.**getSphereCrsFromXyz**(*densityMatrix*, *xyzCoord*, *radius*, *densityCutoff=0*)
Calculate a list of crs coordinates that within a given distance of a xyz point.

Parameters

- **densityMatrix** (*pdb_eda.ccp4.DensityMatrix*) –
- **xyzCoord** (list) – xyz coordinates.
- **radius** (float) –
- **densityCutoff** (float) – a density cutoff for all the points wants to be included, defaults to 0 Default 0 means include every point within the radius. If cutoff < 0, include only points with density < cutoff. If cutoff > 0, include only points with density > cutoff.

Returns crsCoordList of crs coordinates

Return type list

pdb_eda.utils.**getSphereCrsFromXyzList**(*densityMatrix*, *xyzCoordList*, *radius*, *densityCutoff=0*)
Calculates a list of crs coordinates that within a given distance from a list of xyz points.

Parameters

- **densityMatrix** (*pdb_eda.ccp4.DensityMatrix*) –
- **xyzCoordList** (list) – xyz coordinates.
- **radius** (float or list) – search radius or list of search radii
- **densityCutoff** (float) – a density cutoff for all the points wants to be included., defaults to 0 Default 0 means include every point within the radius. If cutoff < 0, include only points with density < cutoff. If cutoff > 0, include only points with density > cutoff.

Returns crsCoordList of crs coordinates

Return type set

pdb_eda.utils.**testValidXyz**(*densityMatrix*, *xyzCoord*, *radius*)
Tests whether all crs coordinates within a given distance of a xyzCoord is within the densityMatrix.

Parameters

- **densityMatrix** (*pdb_eda.ccp4.DensityMatrix*) –
- **xyzCoord** (list) – xyz coordinates.
- **radius** (float) –

Returns bool

Return type bool

pdb_eda.utils.**testValidXyzList**(*densityMatrix*, *xyzCoordList*, *radius*)
Tests whether all crs coordinates within a given distance of a set of xyzCoords is within the densityMatrix.

Parameters

- **densityMatrix** (*pdb_eda.ccp4.DensityMatrix*) –
- **xyzCoordList** (list) – list of xyz coordinates.
- **radius** (float) –

Returns bool

Return type bool

3.6 File Utilities

3.6.1 File Utilities (pdb_eda.fileUtils)

Common file utility functions used across pdb_eda's CLI.

pdb_eda.fileUtils.**createTempJSONFile**(*data*, *filenamePrefix*)
Creates a temporary JSON file and returns its filename.

Parameters

- **data** (dict, list) – data to save into the JSON file.
- **filenamePrefix** (str) – temporary filename prefix.

Returns filename

Return type str

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

Symbols

A

B

C

D

E

F

G

L

M

N

P

R

S

T

X