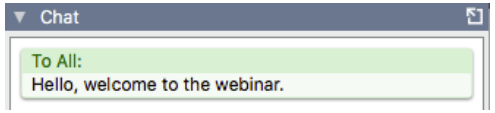
Welcome - webinar instructions

- GoToTraining works best in **Chrome** or on Linux, **Firefox**
- To access the full features of GoToTraining, use the desktop version by clicking **“switch to desktop version”**
- All **microphones will be muted** whilst the trainer is speaking 
- If you have a question please use the **chat box** at the bottom of the GoToTraining box 

- Please complete the **feedback survey** which will launch at the end of the webinar.
- This presentation is available in the materials section

PDBe API webinar series:
5) PDBe tools in GitHub

github.com/pdbeurope

Lukáš Pravda

pdbhelp@ebi.ac.uk
proteindatabank
@PDBeurope
pdbeurope
pdbart

PDBe
Protein Data Bank in Europe

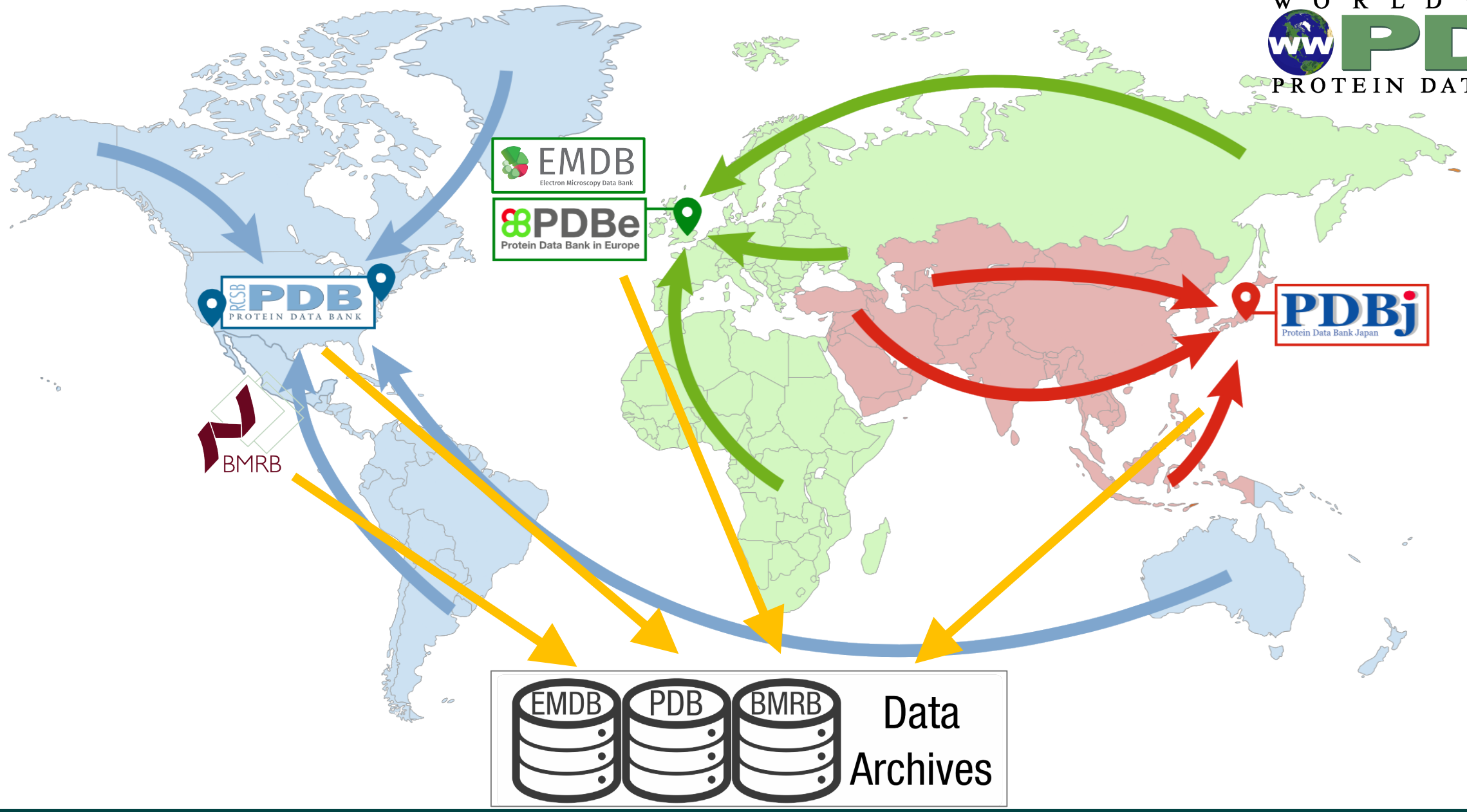
EMBL-EBI

PDBe API webinar series

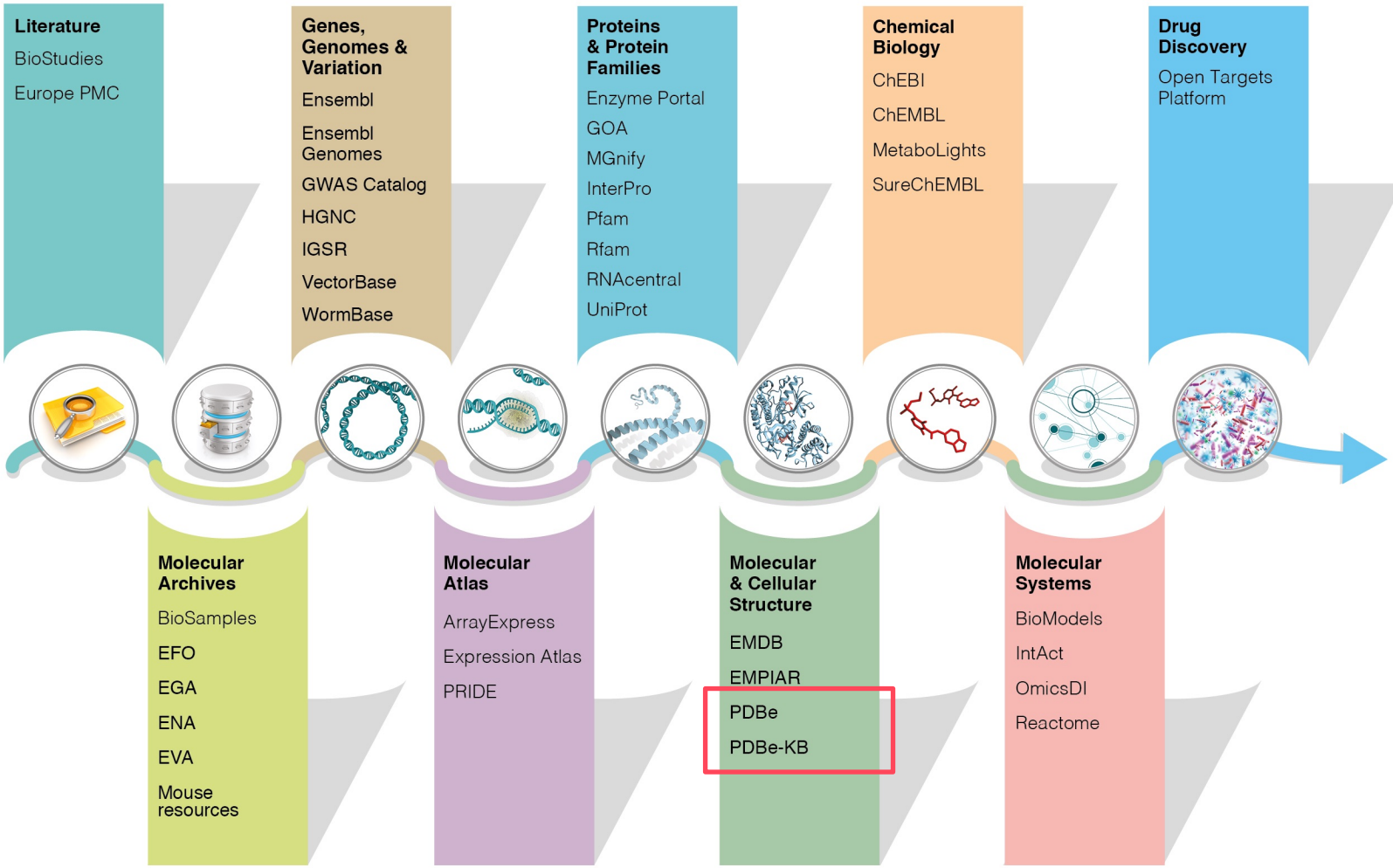
Sep 15th	Introduction to PDBe programmatic access
Sep 22nd	Searching with the PDBe API
Sep 29th	Creating complex PDBe API queries
Oct 6th	Using the PDBe graph API
Oct 13th	PDBe tools in GitHub
Oct 20th	Data visualisation at PDBe

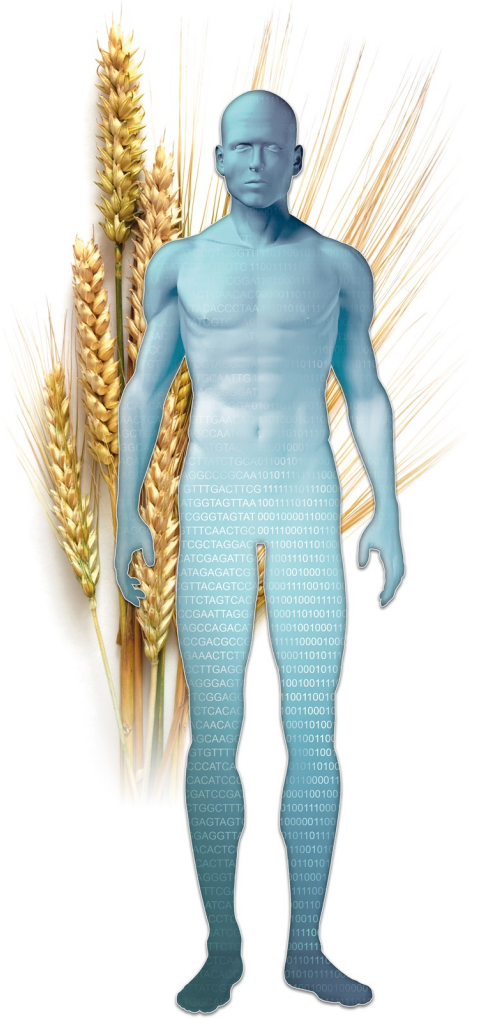
Summary of the webinar

- PDBe in the wwPDB
- pdbecif – mmcif parser
- pdbeccdutils – small molecule chemistry toolkit
- arpeggio – molecular interactions


WORLDWIDE
PDB
PROTEIN DATA BANK
EMDB
Electron Microscopy Data Bank
PDBe
Protein Data Bank in Europe
RCSB PDB
PROTEIN DATA BANK
BMRB
PDBj
Protein Data Bank Japan
EMDB
PDB
BMRB
Data Archives

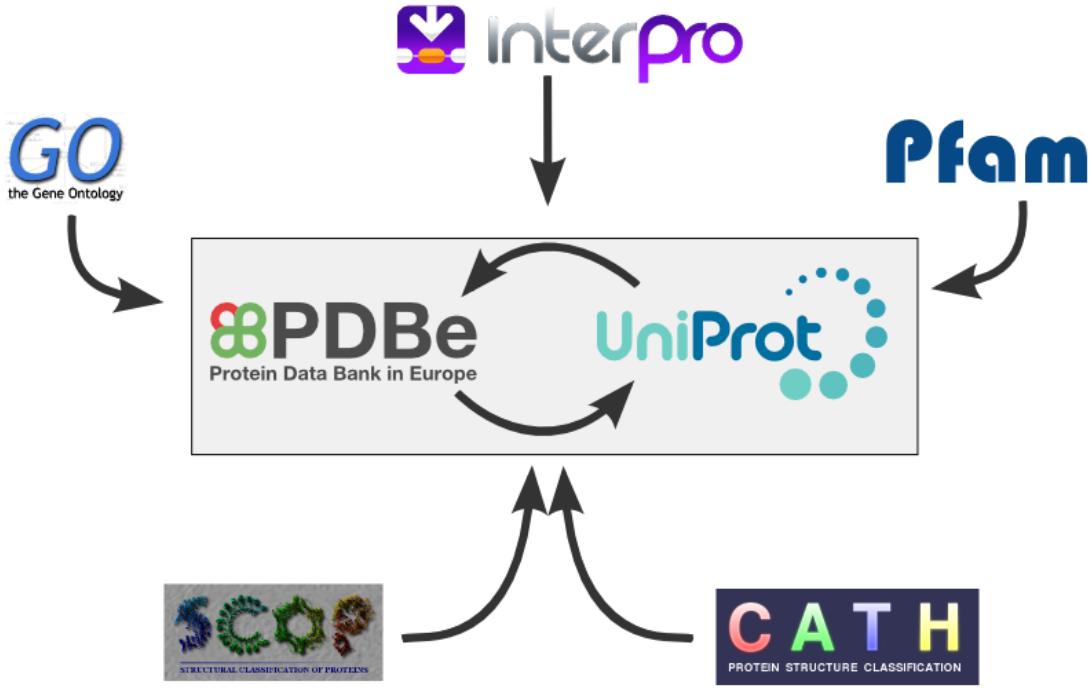
At the heart of EMBL-EBI resources





PDBe – add additional data

- At PDBe we integrate data from other data resources, including from within EBI
- An example
  - SIFTS - Structure Integration with Function, Taxonomy and Sequence



Before we start

- The following examples are available also in a form of jupyter notebooks.
- The best way to handle dependencies is using conda environment manager.
- Install the packages directly from repositories or from PYPi.

- Feel free to fork any of the repository and contribute, open issues in repositories with your feature requests/suggestions and bug reports.

PDBeurope

Repositories 23 | Packages | People 6 | Teams 2 | Projects

Find a repository... | Type: All | Language: All | New

pdbecif

A lightweight pure python package for reading, writing and manipulating mmCIF files distributed by the wwPDB"

Python Apache-2.0 0 1 0 0 Updated on 29 Jul

ccdutils

A set of python tools to deal with PDB chemical components definitions for small molecules, taken from the wwPDB Chemical Component Dictionary, uses RDKit

Python Apache-2.0 0 2 0 0 Updated 24 days ago

arpeggio

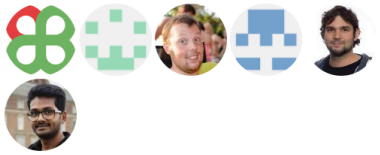
Calculation of interatomic interactions in molecular structures

Python GPL-3.0 0 10 0 0 Updated on 19 Jul

Top languages

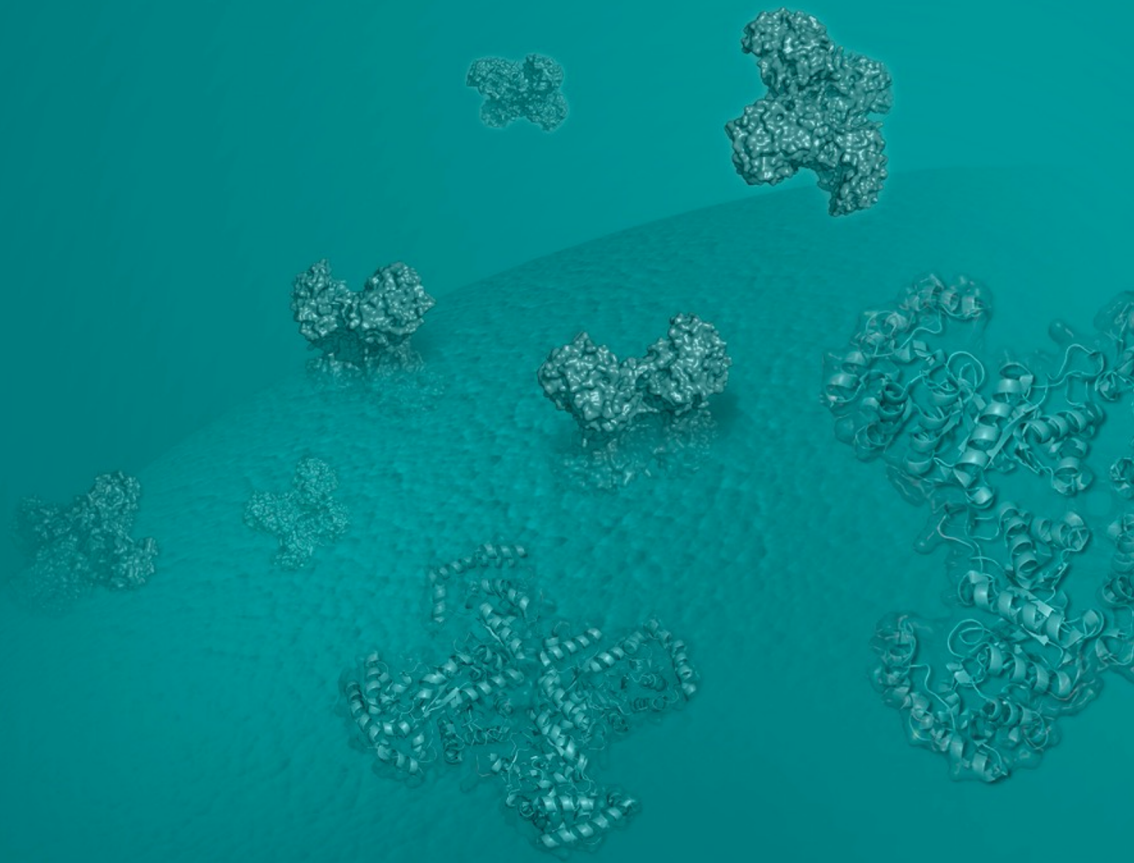
TypeScript JavaScript Python
Jupyter Notebook CSS

People 6 >



PDBecif

Python mmCIF parser



EMBL-EBI

PDBeCIF

GΦL
Global Phasing Limited

- Lightweight pure python 2/3 mmCif/CIF/STAR parser
- No external dependencies, but internally uses Global Phasing tokenizer.
- Allows I/O operations on mmCIF files distributed by the wwPDB members.

- Repository: https://github.com/pdbeurope/pdbecif
- Documentation: https://pdbeurope.github.io/pdbecif/

```
pip install pdbecif
pip install git+https://github.com/PDBeurope/pdbecif.git@master#egg=pdbecif
```

mmCIF file

- Master format of the PDB archive since 2014. Required in xray depositions since 2020.
- Addresses PDB shortcomings
- Flexible and extensible key-value format representing macromolecular data.
  - Value is either a string or an array of strings
- All data items are identified by name, beginning with underscore ('_').

- Syntax description: http://mmcif.wwpdb.org/docs/tutorials/mechanics/pdbx-mmcif-syntax.html
- PDB-mmCIF field correspondence: http://mmcif.wwpdb.org/docs/pdb_to_pdbx_correspondences.html
- mmCIF parsers for different languages: http://mmcif.wwpdb.org/docs/software-resources.html

mmCIF file - schema

```
data_1CBS
 ...
#
```

Data block id

Data block

```
loop
_entity.id
_entity.type
_entity.src_method
_entity.pdbx_description
_entity.formula_weight
_entity.pdbx_number_of_molecules
_entity.pdbx_ec
_entity.pdbx_mutation
_entity.pdbx_fragment
_entity.details
1 polymer man 'CELLULAR RETINOIC ACID BINDING PROTE
2 non-polymer syn 'RETINOIC ACID' 300.435 1 ? ? ? ?
3 water nat water 18.015 100 ? ? ? ?
#
```

Data item

Category name

Attribute name

```
_symmetry.entry_id 1CBS
_symmetry.space_group_name_H-M 'P 21 21 21'
_symmetry.pdbx_full_space_group_name_H-M ?
_symmetry.cell_setting ?
_symmetry.Int_Tables_number 19
#
```

Reading files – option 1 (python dictionary)

1)

```python
from pdbecif.mmcif_io import CifFileReader

reader = CifFileReader()
cif_dict = reader.read(cif_path, output='cif_dictionary')
```

2)

```python
# you can limit data categories by listing the names you are interested in
short_dict = reader.read(cif_path, output='cif_dictionary',
                         only=['_entry', '_symmetry'])

# or listing those that should be discarded
ignored_categories = reader.read(cif_path, output='cif_dictionary',
                                 ignore=['_atom_site'])
```

```
data_1CBS
#
_entry.id        1CBS

#
_symmetry.entry_id                              1CBS
_symmetry.space_group_name_H-M                  "P 21 21 21"
_symmetry.pdbx_full_space_group_name_H-M        ?
_symmetry.cell_setting                          ?
_symmetry.Int_Tables_number                     19

#
```

3)

```
{
    "1CBS": {
        "_entry": {
            "id": "1CBS"
        },
        "_symmetry": {
            "entry_id": "1CBS",
            "space_group_name_H-M": "P 21 21 21",
            "pdbx_full_space_group_name_H-M": "?",
            "cell_setting": "?",
            "Int_Tables_number": "19",
            "space_group_name_Hall": "?"
        }
}
```

Reading files – option 2 (CIFWrapper object)

```
cif_wrapper_result = reader.read(cif_path, output='cif_wrapper')
cif_wrapper_result
```

```
{'1CBS': <pdbecif.mmcif.CIFWrapper at 0x7fc4075f2dd0>}
```

```
cif_wrapper = list(cif_wrapper_result.values())[0]

# access data objects using dot notation
print(cif_wrapper._entity.pdbx_description)
['CELLULAR RETINOIC ACID BINDING PROTEIN TYPE II', 'RETINOIC ACID', 'water']

# or by indexing, the result is the same
print(cif_wrapper['_entity']['pdbx_description'])
['CELLULAR RETINOIC ACID BINDING PROTEIN TYPE II', 'RETINOIC ACID', 'water']
```

```
components = cif_wrapper._chem_comp
non_polymer_components = components.search('type', 'non-polymer')
non_polymer_components
```


```
{8: {"id": "HOH",
  "type": "non-polymer",
  "mon_nstd_flag": ".",
  "name": "WATER",
  "pdbx_synonyms": "?",
  "formula": "H2 O",
  "formula_weight": "18.015"},
 15: {"id": "REA",
  "type": "non-polymer",
  "mon_nstd_flag": ".",
  "name": "RETINOIC ACID",
  "pdbx_synonyms": "?",
  "formula": "C20 H28 O2",
  "formula_weight": "300.435"}}
```

Reading – option 3 (CifFile object)

```
cif_file = reader.read(cif_path, output='cif_file')
cif_file
```

```
<CifFile "1cbs.cif">
```

```
loop_
_entity.id
_entity.type
_entity.src_method
_entity.pdbx_description
_entity.formula_weight
_entity.pdbx_number_of_molecules
_entity.pdbx_ec
_entity.pdbx_mutation
_entity.pdbx_fragment
_entity.details
1     polymer man "CELLULAR RETINOIC ACID BINDING PROTEIN TYPE II" 15581.802   1 ? ? ? ?
2 non-polymer syn                                    "RETINOIC ACID"   300.435   1 ? ? ? ?
3       water nat                                             water    18.015 100 ? ? ? ?
#
_entity_poly.entity_id                      1
_entity_poly.type                           polypeptide(L)
_entity_poly.nstd_linkage                   no
_entity_poly.nstd_monomer                   no
_entity_poly.pdbx_seq_one_letter_code
;PNFSGNWKIIRSENFEELLKVLGVNVMLRKIAVAAASKPAVEIKQEGDTFYIKTSTTVRTTEINFKVGEEFEEQTVDGRP
CKSLVKWESENKMVCEQKLLKGEGPKTSWTRELTNDGELILTMTADDVVCTRVYVRE
;

_entity_poly.pdbx_seq_one_letter_code_can   XYZVAL
_entity_poly.pdbx_strand_id                 A
_entity_poly.pdbx_target_identifier         ?
#
_new_category.new_item       "some value"

#
```

```
block = cif_file.getDataBlock('1CBS')
entity_poly = block.getCategory('_entity_poly')

# add category
category = block.setCategory("new_category")
new_item = category.setItem('new_item')
new_item.setValue('some value')

item = entity_poly.getItem('pdbx_seq_one_letter_code_can')

item.getFormattedValue()
'\n;PNFSGNWKIIRSENFEELLKVLGVNVMLRKIAVAAASKPAVEIKQEGDTFYIKTSTTVRTTEINFKV
```

```
item.getRawValue()
'PNFSGNWKIIRSENFEELLKVLGVNVMLRKIAVAAASKPAVEIKQEGDTFYIKTSTTVRTTEINFKVGEE
```

```
# modify value
item.reset()
item.setValue('XYZVAL')

item.getRawValue()
'XYZVAL'
```

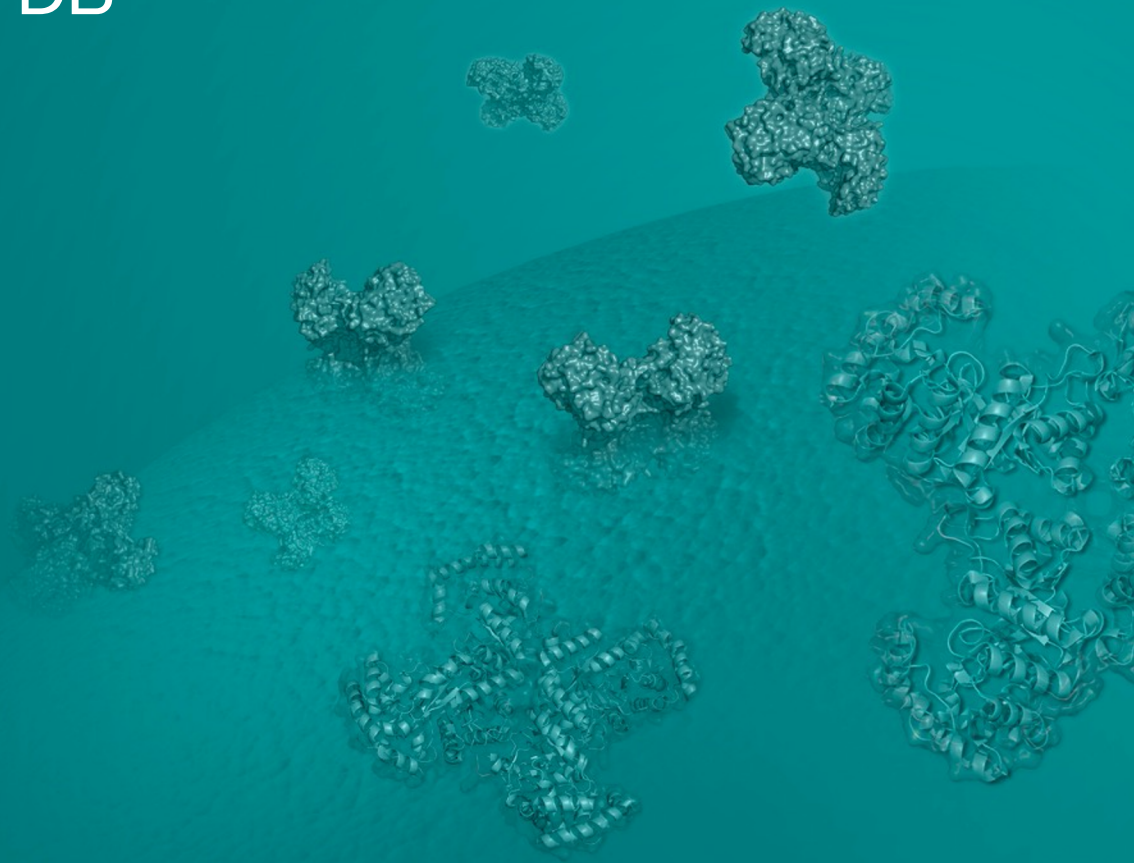
Writing data

```
from pdbecif.mmcif_io import CifFileWriter

writer = CifFileWriter('/Users/lpravda/my_cif.cif')
writer
<pdbecif.mmcif_io.CifFileWriter at 0x7fa3e57936d0>

obj = {
    "root": {
        "category1": {
            "subcatA": "val1",
            "subcatB": "val2"
        },
        "category2": {
            "subcat1": [0,1,2],
            "subcat2": ["a", "b", "c"]
        }
    }
}

writer.write(obj)
```

```
data_root
#
_category1.subcatA          val1
_category1.subcatB          val2

#
loop_
_category2.subcat1
_category2.subcat2
. a
1 b
2 c
#
```

pdbeccdutils

Python tools for small molecules in the PDB



EMBL-EBI

pdbeccdutils

RDKit
Open-Source Cheminformatics and Machine Learning

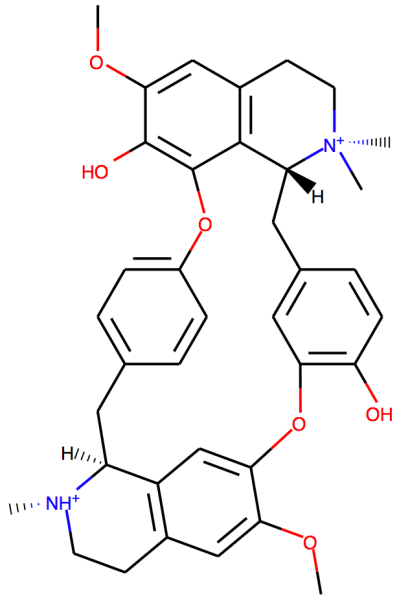
- A set of python tools to deal with PDB chemical components definitions for small molecules, taken from the wwPDB Chemical component dictionary
- Depends on RDKit (https://www.rdkit.org/); other dependencies are installed automatically.

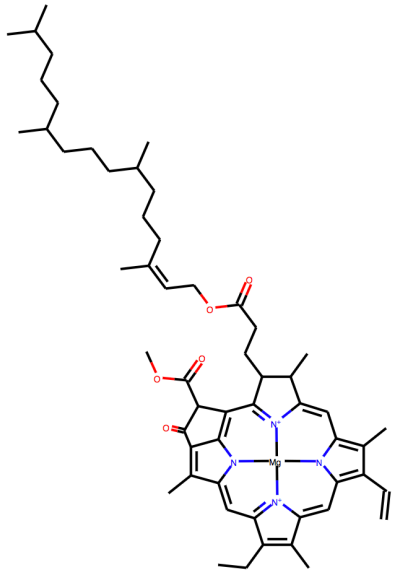
- Repository: https://github.com/pdbeurope/ccdutils
- Documentation: https://pdbeurope.github.io/ccdutils/

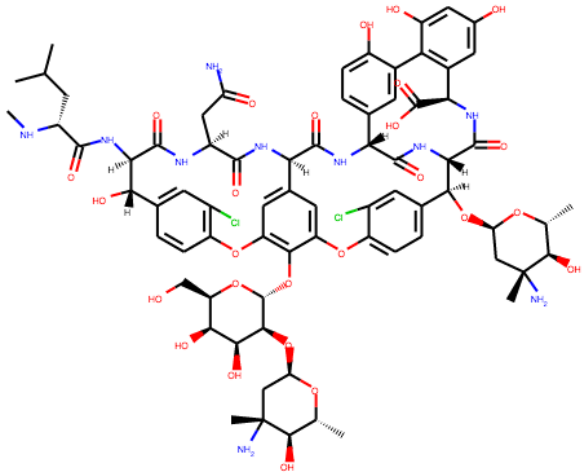
```
conda create -c conda-forge -n rdkit-env rdkit python=3.7
conda activate rdkit-env

pip install pdbeccdutils
pip install git+https://github.com/PDBeurope/ccdutils.git@master#egg=pdbeccdutils
```

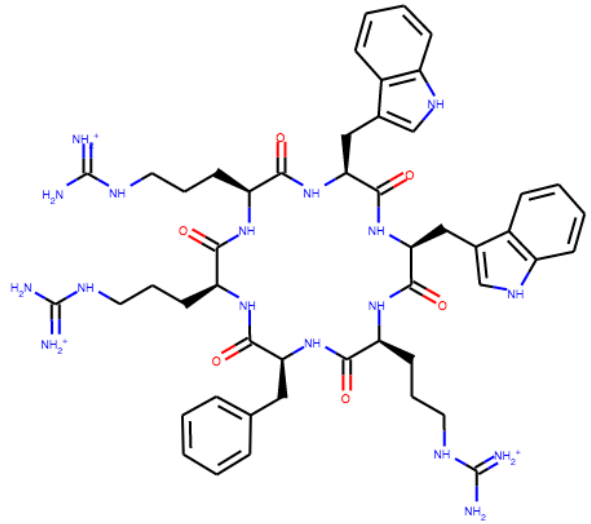
Source of the data

- wwPDB Chemical component dictionary (CCD het-codes)
  - http://www.wwpdb.org/data/ccd
- wwPDB biologically interesting Molecule Reference Dictionary (BIRD)
  - http://www.wwpdb.org/data/bird









pdbeccdutils API

```
pdbeccdutils
  computations
    parity_method.py
  core
    ccd_reader.py
    ccd_writer.py
    component.py
    depictions.py
    fragment_library.py
  scripts
    process_components_cif_cli.py
    setup_pubchem_library.py
  utils
    pubchem_downloader.py
    web_services.py
```

Contains convenient read methods for creating rdkit.Mol objects from raw mmCIF files.

Reading in structures

```
from pdbeccdutils.core import ccd_reader

result = ccd_reader.read_pdb_cif_file('/Users/lpravda/HEM.cif', sanitize=False)
result
CCDReaderResult(warnings=[], errors=[], component=<pdbeccdutils.core.component.Component object at 0x7ffe0534bfd0>)

components = ccd_reader.read_pdb_components_file('/Users/lpravda/HEM.cif', sanitize=False)
{'HEM': CCDReaderResult(warnings=[], errors=[], component=<pdbeccdutils.core.component.Component object at 0x7ffe053
```

pdbeccdutils API

```
pdbeccdutils
  computations
    parity_method.py
  core
    ccd_reader.py
    ccd_writer.py
    component.py
    depictions.py
    fragment_library.py
  scripts
    process_components_cif_cli.py
    setup_pubchem_library.py
  utils
    pubchem_downloader.py
    web_services.py
```

The root of the package with shorthand methods enabling integration with RDKit

Structure of the Components object

```
Component
  id
  name
  formula
  pdbx_release_status
  modified_data
  inchi
  inchikey
  ...

  mol
  mol_no_h

  physchem_properties
  fragments
  scaffolds
```

```
component
<pdbeccdutils.core.component.Component at 0x7ffe0538a390>

component.formula
'C34 H32 Fe N4 O4'

component.inchikey
'KABFMIBPWCXCRK-RGGAHWMASA-L'

component.mol
<rdkit.Chem.rdchem.Mol at 0x7ffe052538f0>

component.mol_no_h
<rdkit.Chem.rdchem.Mol at 0x7ffe05a304e0>
```

- \+ convenience methods

Shorthand functions - scaffolds

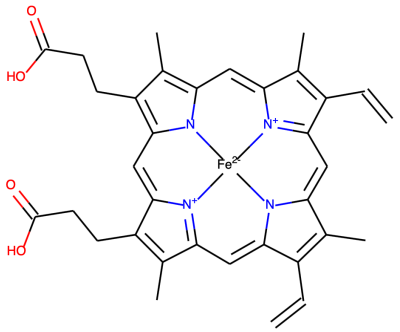
```
component.scaffolds
[]

component.get_scaffolds()
[<rdkit.Chem.rdchem.Mol at 0x7ffe05adf580>]

scaffold_details = component.scaffolds[0]
scaffold_details
SubstructureMapping(name='MurckoScaffold', smiles='C1=CC2=[N+]3C1=Cc1ccc4n1[Fe-2]31n3c

scaffold_details.smiles
'C1=CC2=[N+]3C1=Cc1ccc4n1[Fe-2]31n3c(ccc3=CC3=[N+]1C(=C4)C=C3)=C2'
```

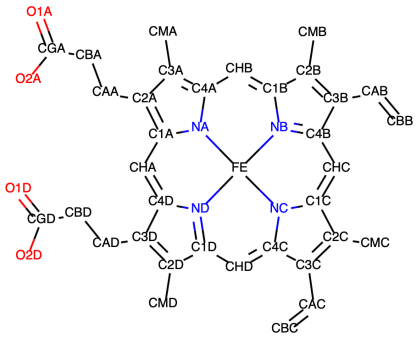
Shorthand functions - fragments

```
from pdbeccdutils.core.fragment_library import FragmentLibrary

library = FragmentLibrary()
fragments = component.library_search(library)
len(fragments)
2

fragments[1]
SubstructureMapping(name='pyrrole', smiles='c1cc[nH]c1', source='PDBe', mappings=((4, 5, 6, 7,


component.fragments
[SubstructureMapping(name='porphin-like', smiles='C1~C~C2~C~C3~C~C~C(~C~C4~C~C~C(~C~C5~C~C~C(~C
SubstructureMapping(name='pyrrole', smiles='c1cc[nH]c1', source='PDBe', mappings=[['C1A', 'C2A'
```

Shorthand functions - depictions

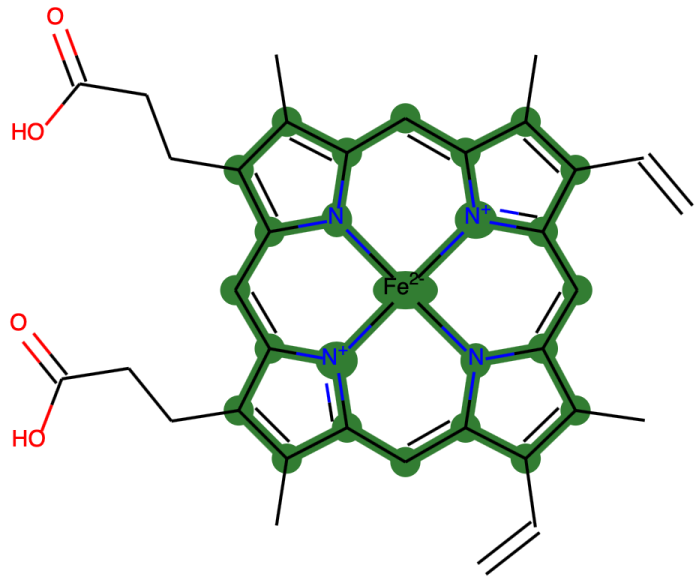
```
from pdbeccdutils.core.depictions import DepictionManager

d = DepictionManager()
result = component.compute_2d(d)
result
DepictionResult(source=<DepictionSource.Template: 2>, template_name='hem', mol=<rdkit.

result.score
0.0

component.export_2d_svg('/Users/lpravda/HEM.svg')
component.export_2d_svg('/Users/lpravda/HEM.svg', names=True)
```





Shorthand functions – depictions with highlight

```
scaffold = component.scaffolds[0]
atom_names = scaffold.mappings[0]
atom_color_mapping = {x: (0.2, 0.5, 0.2) for x in atom_names}

bond_color_highlight = {}
bonds = component.mol_no_h.GetBonds()
for bond in bonds:
    begin = bond.GetBeginAtom().GetProp('name')
    end = bond.GetEndAtom().GetProp('name')

    if begin in atom_names and end in atom_names:
        bond_color_highlight[(begin, end)] = ((0.2, 0.5, 0.2))

component.export_2d_svg('/Users/lpravda/HEM_with_scaffold.svg',
                        atom_highlight=color_mapping,
                        bond_highlight=bond_color_highlight)
```



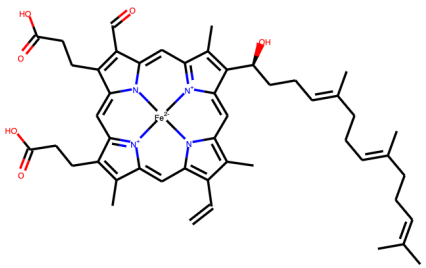
pdbeccdutils API

```
pdbeccdutils
  computations
    parity_method.py
  core
    ccd_reader.py
    ccd_writer.py
    component.py
    depictions.py
    fragment_library.py
  scripts
    process_components_cif_cli.py
    setup_pubchem_library.py
  utils
    pubchem_downloader.py
    web_services.py
```

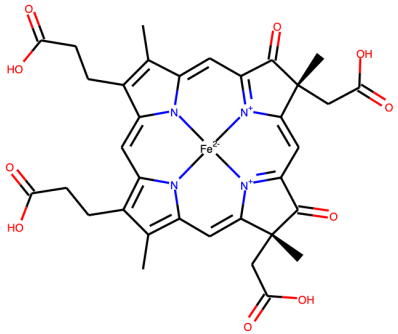
Implementation of a molecular similarity method by Jon Tyzack https://doi.org/10.1016/j.str.2018.02.009

parity_method.py

```
from pdbeccdutils.core import ccd_reader
from pdbeccdutils.computations.parity_method import compare_molecules

hem_a = ccd_reader.read_pdb_cif_file('HEA.cif').component
hem_d = ccd_reader.read_pdb_cif_file('DHE.cif').component

result = compare_molecules(hem_a.mol_no_h, hem_d.mol_no_h)
result

ParityResult(mapping={0: 0, 5: 5, 6: 6, 1: 1, 35: 42, 31: 38, 32: 39, 4: 4, 27

result.similarity_score
0.6029411764705882
```



HEME-A



HEME-D

pdbeccdutils API

```
pdbeccdutils
  computations
    parity_method.py
  core
    ccd_reader.py
    ccd_writer.py
    component.py
    depictions.py
    fragment_library.py
  scripts
    process_components_cif_cli.py
    setup_pubchem_library.py
  utils
    pubchem_downloader.py
    web_services.py
```

Contains convenient methods for writing out components object and its content in number of different file formats.

Writing molecules

```
from pdbeccdutils.core import ccd_writer
from pdbeccdutils.core.models import ConformerType

# write idealized coordinates in the SDF format.
ccd_writer.write_molecule('HEM.sdf', component)

# write model coordinates in the mmCIF
ccd_writer.write_molecule('HEM.cif', component, conf_type=ConformerType.Model)

# write model coordinates in the PDB format without hydrogens
ccd_writer.write_molecule('HEM.pdb', component, remove_hs=True, conf_type=ConformerType
```

pdbeccdutils API

```
pdbeccdutils
  computations
    parity_method.py
  core
    ccd_reader.py
    ccd_writer.py
    component.py
    depictions.py
    fragment_library.py
  scripts
    process_components_cif_cli.py
    setup_pubchem_library.py
  utils
    pubchem_downloader.py
    web_services.py
```

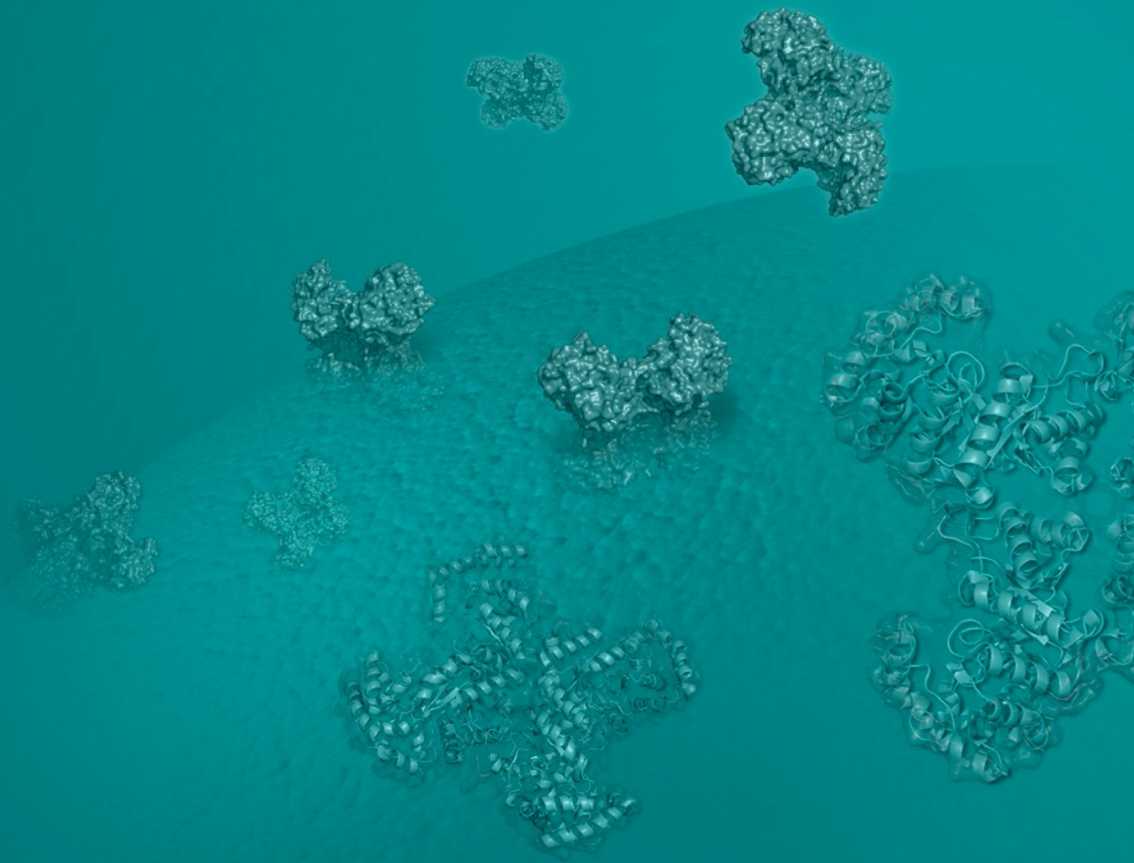
Process that we use internally during our weekly release process to generate all the chemistry data.

process_components_cif_cli.py

- Generates content for the http://ftp.ebi.ac.uk/pub/databases/msd/pdbechem_v2
- ATP example
  - * `ATP.cif` - Standard wwPDB CCD file with the | mmCIF.
  - * `ATP_ideal.pdb` - ideal cooordinates | PDB.
  - * `ATP_ideal_alt.pdb` - ideal cooordinates with atom alternate names | PDB.
  - * `ATP_model.pdb` - model cooordinates | PDB.
  - * `ATP_model_alt.pdb` - model cooordinates with atom alternate names | PDB.
  - * `ATP_N.svg` - 2D depiction in N x N resolution. Where N is (100,200,300,400,500) pixels.
  - * `ATP_N_names.svg` - 2D depiction in N x N resolution with atom names. Where N in (100,200,300,400,500) pixels.
  - * `ATP_model.sdf` - model coordinates | MOL
  - * `ATP_ideal.sdf` - ideal cooordinates | MOL.
  - * `ATP.cml` - component representation | CML.
  - * `ATP_annotation.json` - 2D depiction in 'natural format' (i.e. 50px per 1Å) with some additional annotation. This file is consumed by the protein-ligand interaction viewer.

arpeggio

Molecular interactions



EMBL-EBI

Arpeggio

- Originally developed by Harry Jubb at the Blundell's lab (University of Cambridge)
- Depends on biopython, openbabel (<3.0), and pdbecif

- Original repository: https://github.com/harryjubb/arpeggio
- Web version: http://biosig.unimelb.edu.au/arpeggioweb/

- Our fork: https://github.com/PDBeurope/arpeggio
- Documentation: https://github.com/PDBeurope/arpeggio/blob/master/README.md

```
conda create conda -n arpeggio-env python=3.7
conda activate arpeggio-env
conda install -c openbabel openbabel

pip install git+https://github.com/PDBeurope/arpeggio.git@master#egg=arpeggio
```

Differences to the original arpeggio

- Python 3 support
- Modular architecture
- support for mmCIF structures
- results in the JSON format
- bugfixes

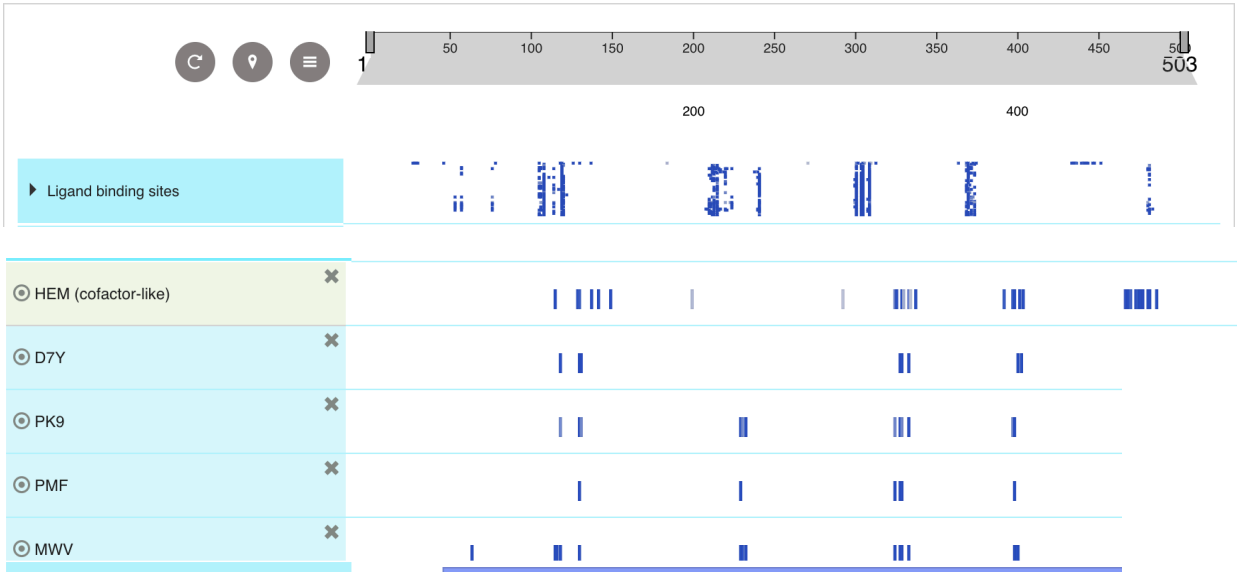
Interactions

- Atom-atom interactions
  - Covalent
  - Electrostatic (ionic, hydrogen-bonds, polar, xbond, carbonyl)
  - Aromatic
  - Vdw interactions
  - Clashes
- Atom-plane
  - E.g. Carbon-PI, Cation-PI
- Plane-plane interactions
  - 9 mutual ring positions - https://doi.org/10.1016/j.pbiomolbio.2007.03.016
- Group-plane, group-group interactions
  - Amide-ring; amide-amide

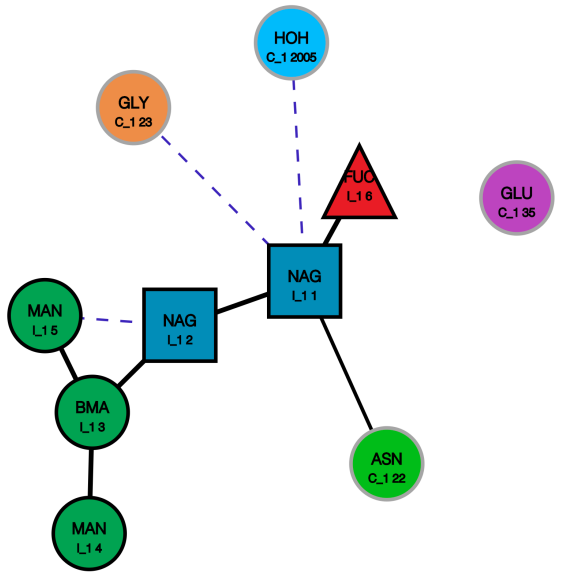
Data availability

- Protonated structures – quaternary structure (using ChimeraX)
  - https://www.ebi.ac.uk/pdbe/model-server/v1/:pdb_id/full?encoding=cif&data_source=pdb-h
  - E.g. https://www.ebi.ac.uk/pdbe/model-server/v1/1cbs/full?encoding=cif&data_source=pdb-h
- Interactions data:
  - pdbe.org/aggregated-api
    - Bound molecules (pdb id)
    - Bound molecule interactions (pdb id, bound molecule id)
    - Bound ligand interactions (pdb id, ligand chain id, ligand residue id)

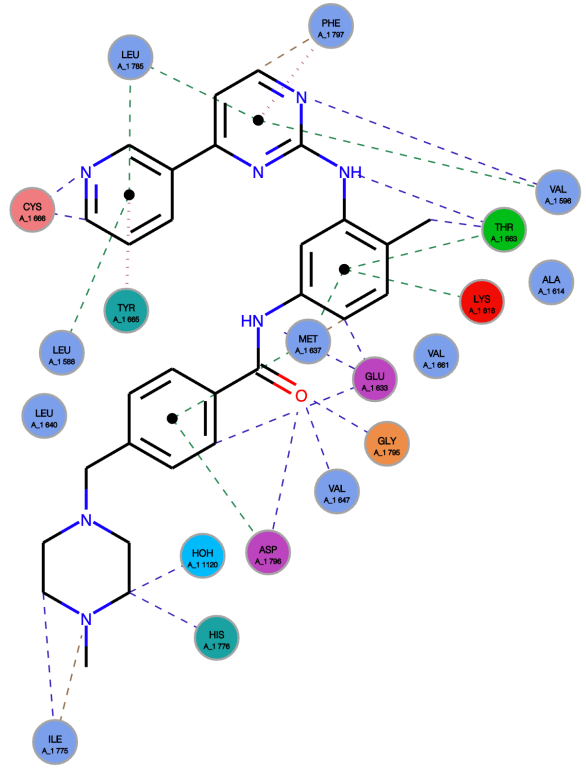
Interactions - visualization



http://pdbekb.org/proteins/P08684



Oligosaccharide interactions

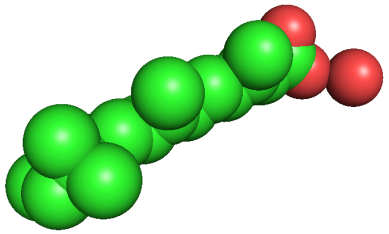


Imatinib (cancer drug) interactions

github.com/PDBeurope/arpeggio

EMBL-EBI

Script



```
arpeggio -s /A/200/ -o arpeggio_result 1cbs.cif

INFO//14:59:04.545//Program begin.
INFO//14:59:04.545//Selection perceived: ['/A/200/']
DEBUG//14:59:04.605//Loaded PDB structure (BioPython)
DEBUG//14:59:04.667//Loaded MMCIF structure (OpenBabel)
DEBUG//14:59:04.674//Mapped OB to BioPython atoms and vice-versa.
DEBUG//14:59:04.674//Detected that the input structure contains hydrogens. Hydrogen additic
DEBUG//14:59:04.787//Determined atom explicit and implicit valences, bond orders, atomic nu
DEBUG//14:59:04.810//Initialised SIFts.
DEBUG//14:59:04.812//Determined polypeptide residues, chain breaks, termini
DEBUG//14:59:04.858//Percieved and stored rings.
DEBUG//14:59:04.869//Perceived and stored amide groups.
DEBUG//14:59:04.882//Added hydrogens to BioPython atoms.
DEBUG//14:59:04.887//Added VdW radii.
DEBUG//14:59:04.892//Added covalent radii.
DEBUG//14:59:04.910//Completed NeighborSearch.
DEBUG//14:59:04.912//Assigned rings to residues.
DEBUG//14:59:04.918//Made selection.
DEBUG//14:59:05.112//Expanded to binding site.
DEBUG//14:59:05.113//Flagged selection rings.
DEBUG//14:59:05.114//Completed new NeighbourSearch.
INFO//14:59:05.438//Program End. Maximum memory usage was 77.32 MB.
```


```
[...{
      "bgn": {
          "auth_asym_id": "A",
          "auth_atom_id": "O",
          "auth_seq_id": 309,
          "label_comp_id": "HOH",
          "pdbx_PDB_ins_code": " "
      },
      "contact": [
          "vdw_clash",
          "hbond",
          "polar"
      ],
      "distance": 2.71,
      "end": {
          "auth_asym_id": "A",
          "auth_atom_id": "O2",
          "auth_seq_id": 200,
          "label_comp_id": "REA",
          "pdbx_PDB_ins_code": " "
      },
      "interacting_entities": "SELECTION_WATER",
      "type": "atom-atom"
  },...]
```

Arpeggio API

```
from arpeggio.core import InteractionComplex

selection = ['/A/1210/', '/D/1310/']

complex = InteractionComplex('/Users/lpravda/3d12_h.cif')
complex.structure_checks()
complex.address_ambiguities()

complex.initialize()
complex.run_arpeggio(selection, interacting_cutoff=5,
                                vdw_comp=0.1,
                                include_sequence_adjacent=False)
contacts = complex.get_contacts()
contacts
```

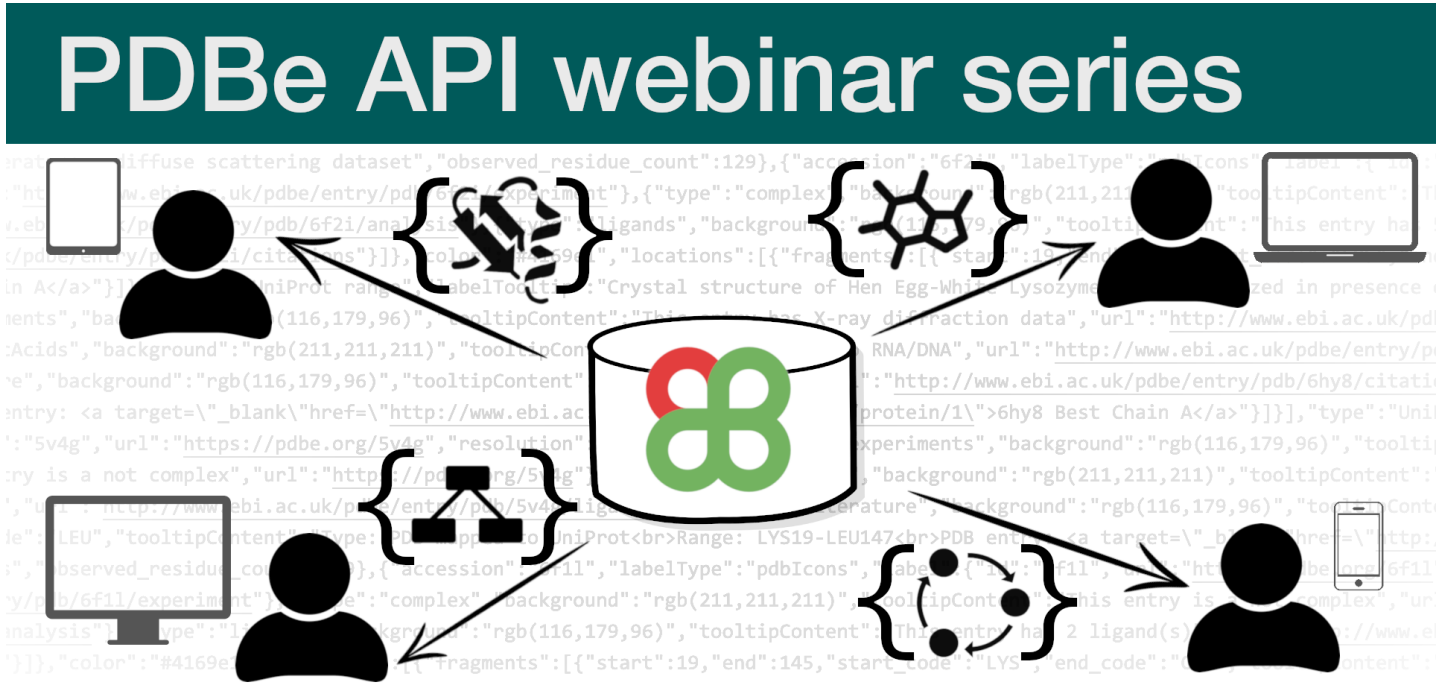
PDBe API webinar series

Sep 15th	Introduction to PDBe programmatic access
Sep 22nd	Searching with the PDBe API
Sep 29th	Creating complex PDBe API queries
Oct 6th	Using the PDBe graph API
Oct 13th	PDBe tools in GitHub
Oct 20th	Data visualisation at PDBe

Mandar Deshpande

Webinars

- See the list of webinars at bit.ly/PDBe_API_webinars
- Or visit the PDBe events pages at PDBe.org/events
- The last webinar is full



Thank you for your attention!
Any questions?

PDBe.org/API

Lukáš Pravda

pdbhelp@ebi.ac.uk

proteindatabank

@PDBeurope

pdbeurope

pdbart

PDBe
Protein Data Bank in Europe

EMBL-EBI