Flexible searching of small molecule and protein structural data with the CSD Python API toolkit

Juliette Pradon, Ilenia Giangreco

The Cambridge Crystallographic Data Centre (CCDC), Cambridge, CB2 1EZ, UK; Email: pradon@ccdc.cam.ac.uk

Introduction

Successful modern drug discovery research makes extensive use of structural data – from target proteins, candidate drug molecules, and complexes of the two. The value of protein-ligand structural information is well accepted. In addition, knowledge of molecular conformations and interactions derived from small molecule structures alone can have a significant impact in drug discovery. The CSD Python API scripting interface has been developed to mine the world's two most important structural databases, the Cambridge Structural Database (CSD) [1] and the Protein Data Bank (PDB) [2], and extract data driven insight, which can inform the design, development and identification of new and better pharmaceutical products.

Overview of search capabilities in the CSD Python API

1. Text/numeric searching (of data associated with CSD entries only)
2. Similarity searching (using the CCDC fingerprint) of ligands
3. Fast (sub)structure searching (with support for atom constraints) of ligands. Substructure searching also supports geometric constraints to mine 3D intermolecular interaction patterns between any of: protein binding sites, waters, metals and small molecules, with support for nucleotides.

Note: these three search types can be combined using 'And', 'Or' and 'Not', and can be constrained using search filters, e.g. filtering by maximum R-factor, organic/organometallic, disorder, etc.

4. Sequence-independent protein cavity & sub-pocket searching methods to identify similar protein binding sites, useful in many areas of pharmaceutical drug design (e.g. bioisosteric replacement, polypharmacology, off-target prediction). Three methods, varying on speed & accuracy:
   - fast cavity graph comparison (based on Local Cliques [3] algorithm)
   - cavity graph comparison (based on CavBase [4] algorithm)
   - cavity histograms comparison (based on RAPMAD [5] algorithm)

5. Pharmacophore searching of CSD & PDB data simultaneously, to mine for structural motifs that bind in similar environments and generate new ideas for ligand mimicking, scaffold hopping or growing into a sub-pocket.

Note: optional protein/small molecule components and intra/intermolecular constraints on pharmacophore features to tailor searches; and ability to filter search results based on database annotations.

Structural databases available to search with the CSD Python API

1. The CSD: over 1 million organic & metal-organic, fully curated and validated small molecule crystal structures, enriched with additional data accessible to the API, e.g. chemical name, melting point, polymorphic form, any disorder, publication details, etc.

2. Two ready-to-search CSD SQL Fast Binary structure databases: one with ~285,000 protein-ligand binding sites and one with ~5,000 protein-ligand-nucleic acid binding sites from ~60,000 high (≤3Å) resolution PDB entries.

3. The CSD-CrossMiner [6] feature database, with these PDB protein-ligand & protein-ligand-nucleic acid binding sites, plus ~381,000 organic structures from the CSD. Annotations from crystallographic structure information, accessible for hit filtering with API, e.g. CSD refcode, EC_number, PDB code, type of molecule, resolution, protein target, etc.

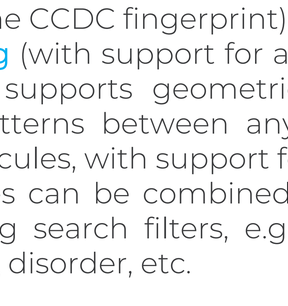
4. For associated collaborators, a ready-to-search exemplar SQLite database containing, in an XML-based representation, LIGSITE[7]-detected cavities extracted from high (≤3Å) resolution X-ray crystallography and electron microscopy structures from the PDB.

5. Create and search your own structural, feature and/or cavity databases, to search any proprietary small molecule and/or protein structures alongside public CSD and PDB structures. Add any annotations to your own CSD-CrossMiner feature database, for further tailored filtering of hits with the API.

Example 1 – mining for ligand chemistry that binds in a specific protein environment

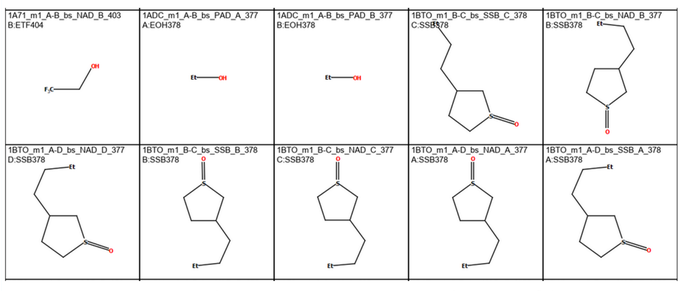
How to search for ligands that bind to a Zinc ion, where it coordinates a histidine and two cysteine residues, in PDB protein-ligand binding sites? Write a complex 3D substructure query:

```
query = ccdc.search.SubstructureSearch()
query.settings.max_hits_per_structure = 1
zinc = ccdc.search.SMARTSSubstructure('[Zn]')
zinc.atoms[0].add_protein_atom_type_constraint('Metal')
sub = query.add_substructure(zinc)
histidine = ccdc.search.SMARTSSubstructure('[#7]1~[#6]~[#7]~[#6]~[#6]1')
for a in histidine.atoms:
    a.add_protein_atom_type_constraint('Amino')
sub1 = query.add_substructure(histidine)
cysteine = ccdc.search.SMARTSSubstructure('SC')
for a in cysteine.atoms:
    a.add_protein_atom_type_constraint('Amino')
sub2 = query.add_substructure(cysteine)
sub3 = query.add_substructure(cysteine)
query.add_distance_constraint('DIST1', (sub, 0), (sub1, 0), (0, 3.0), vdw_corrected=False, type='any')
query.add_distance_constraint('DIST2', (sub, 0), (sub2, 0), (0, 3.0), vdw_corrected=False, type='any')
query.add_distance_constraint('DIST3', (sub, 0), (sub3, 0), (0, 3.0), vdw_corrected=False, type='any')
any_ligand_atom = ccdc.search.SMARTSSubstructure('[!H]')
any_ligand_atom.atoms[0].add_protein_atom_type_constraint('Ligand')
sub4 = query.add_substructure(any_ligand_atom)
query.add_distance_constraint('DIST4', (sub, 0), (sub4, 0), (0, 3.0), vdw_corrected=False, type='any')
```



Output the superimposed hits as mol2 (note that NAD is classified as a co-factor) and the hit ligands as 2D diagrams, highlighting the Zinc-coordinating ligand atom:

```
hit_writer = WriteHit(outfile='superimposed_hits.mol2')
hit_writer.search(query, db)
```

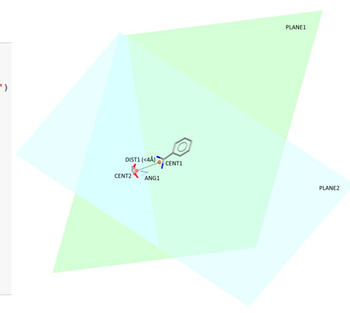


Use these results to generate ideas of how to modify your compound to bind in this protein environment.

Example 2 – identifying the geometric interaction preferences for a protein-ligand interaction pair

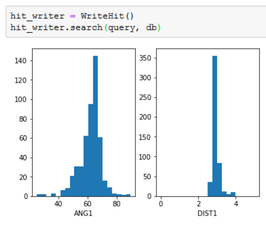
How to search for benzamidine fragment in ligand interacting with carboxylate of a residue sidechain (with no specified amino acid type) in PDB protein -ligand binding sites? Write a complex 3D substructure query:

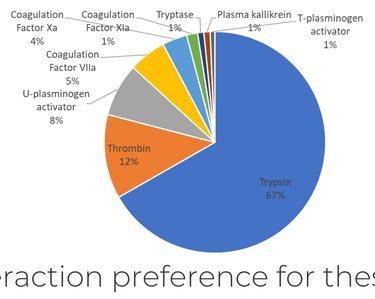
```
query = ccdc.search.SubstructureSearch()
query.settings.max_hits_per_structure = 1

benzamidine = ccdc.search.SMARTSSubstructure('[#7]~[#6](~[#7])c1ccccc1')
for a in benzamidine.atoms:
    a.add_protein_atom_type_constraint('Ligand')

carboxylate = ccdc.search.SMARTSSubstructure('c~[o]~o')
for a in carboxylate.atoms:
    a.add_protein_atom_type_constraint('Amino')

sub1 = query.add_substructure(benzamidine)
sub2 = query.add_substructure(carboxylate)

query.add_centroid('CENT1', (sub1, 0), (sub1, 2))
query.add_centroid('CENT2', (sub2, 1), (sub2, 2))
query.add_distance_constraint('DIST1', 'CENT1', 'CENT2', (0, 4.0))

query.add_plane('PLANE1', (sub1, 1), (sub1, 0), (sub1, 2))
query.add_plane('PLANE2', (sub2, 0), (sub2, 1), (sub2, 2))
query.add_plane_angle_measurement('ANG1', 'PLANE1', 'PLANE2')
```



Plot the centroid-centroid distance and angle of interaction between the carboxylate and the benzamidine fragments, and access the protein name (via EC_number) where interaction is seen:

```
hit_writer = WriteHit()
hit_writer.search(query, db)
```





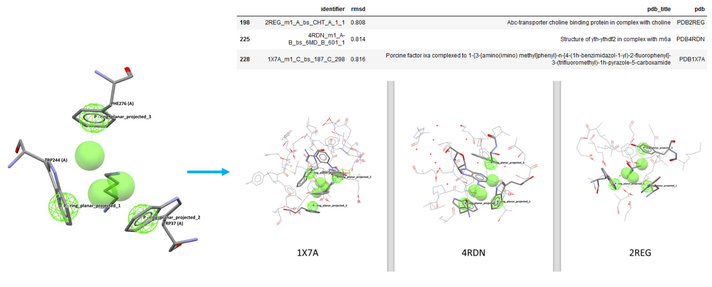
Observed marked interaction preference for these two functional groups, with peak of distance at ~2.8 Å and angle at ~60°, in the first 500 hits.

Example 3 – identifying similar protein environments and their existing ligand binders

How to detect aromatic cages in the PDB database and identify what ligand functionality binds inside these pockets? Write a pharmacophore query from the putrescine receptor binding site, selecting three features:

```
# Let's define a ring_planar_projected feature of Trp244 and Trp37 by choosing only the one on the 6-membered ring
# and the one that points towards the ligand. Similarly for a ring_planar_projected feature of Phe276

six_membered_ring_atoms_trp244 = [a for a in protein_binding_site["A:TRP244"].atoms
                if a.label=='CD2' or a.label=='CE2' or a.label=='CZ2' or a.label=='CH2' or a.label=='CZ3' or a.label=='CE3']
trp244_centroid = MolecularDescriptors.atom_centroid(*six_membered_ring_atoms_trp244)
trp244_ring_feature = select_aromatic_feature(trp244_centroid)

six_membered_ring_atoms_trp37 = [a for a in protein_binding_site["A:TRP37"].atoms
                if a.label=='CD2' or a.label=='CE2' or a.label=='CZ2' or a.label=='CH2' or a.label=='CZ3' or a.label=='CE3']
trp37_centroid = MolecularDescriptors.atom_centroid(*six_membered_ring_atoms_trp37)
trp37_ring_feature = select_aromatic_feature(trp37_centroid)

ring_atoms_phe276 = [a for a in protein_binding_site["A:PHE276"].atoms
                if a.label=='CD2' or a.label=='CE2' or a.label=='CZ' or a.label=='CE1' or a.label=='CD1' or a.label=='CG
phe276_centroid = MolecularDescriptors.atom_centroid(*ring_atoms_phe276)
phe276_ring_feature = select_aromatic_feature(phe276_centroid)

# Create the pharmacophore query from this set of features

complete_set_of_features = [trp244_ring_feature, trp37_ring_feature, phe276_ring_feature]
query = Pharmacophore.Query(complete_set_of_features)
query.write('model.cm')
```

Access the annotations to find out in what other protein classes do such aromatic cages occur, with hits in many unrelated proteins such as:



Can then identify existing ligand chemistry that binds in such aromatic cages by adding a small molecule heavy_atom feature.

References

[1] C. R. Groom, *et al.*, *Acta Cryst.* (2016) B**72**, 171-179.
[2] H. M. Berman, *et al.*, *Nucleic Acids Res.* (2000) **28**, 235-242.
[3] T. Krotzky, *et al.*, *IEEE/ACM Trans. Comput. Biol. Bioinf.* (2014) **11**, 878-890.
[4] S. Schmitt, *et al.*, *J. Mol. Biol.* (2002) **323**, 387-406.
[5] T. Krotzky, *et al.*, *J. Chem. Inf. Model.* (2015) **55**, 165-179.
[6] O. Korb, *et al.*, *J. Med. Chem.* (2016) **59**, 4257-4266.
[7] M. Hendlich, *et al.*, *J. Mol. Graphics. Mod.* (1997) **15**, 359-363
CSD Python API: downloads.ccdc.cam.ac.uk/documentation/API/index.html

Conclusions

The CSD Python API is essentially feature complete with regards to functionality exposed through all CCDC software components for Discovery and Materials users. This includes API access to: protein-ligand docking; virtual screening of a compound library against a pharmacophore query obtained from one or multiple overlaid ligands; generation of interaction maps around small molecules or within a protein binding site; conformer generation and molecular minimisation; calculation of probabilities for the formation of H-bonds; calculation of a crystal packing similarity metric; molecular geometry analysis; ...
Please let us know what else you would like to see exposed in the CSD Python API!