ProteinLigandBenchmarks Documentation

PLBenchmarks

Feb 17, 2020

CONTENTS:

CHAPTER

ONE

INSTALLING THE PROTEIN LIGAND BENCHMARK SET

The Protein Ligand Benchmark Set is currently only installable from source.

1.1 Installation from Source

To install PLBenchmarks from source, clone the repository from github:

```
git clone https://github.com/openforcefield/PLBenchmarks.git
cd PLBenchmarks
```

Create a custom conda environment which contains the required dependencies and activate it:

```
conda env create --name plbenchmarks --file devtools/conda-envs/test_env.yaml
conda activate plbenchmarks
```

The final step is to install PLBenchmarks itself:

```
python setup.py develop
```

CHAPTER

TWO

GETTING STARTED WITH PLBENCHMARKS

```
from PLBenchmarks import targets
from IPython.core.display import HTML
```

2.1 Get the whole set of targets in the dataset

```
# it is initialized from the `PLBenchmarks/data/targets.yml` file
tgtset = targets.targetSet()
# to see which targets are available, one can get a list of names
tgtset.getNames()
```

```
['jnk1',
 'pde2',
 'thrombin',
 'p38',
 'ptp1b',
 'galectin',
 'cdk2',
 'cmet',
 'mcl1']
```

The `targetSet` is a `dict`, but can be converted to a `pandas.DataFrame` or a `html` string via `targetSet.getDF(columns=None)` or `targetSet.getHTML(columns=None)`. The default `None` for `columns` means that all columns are printed. One can also define a subset of columns as a `list`:

```
HTML(tgtset.getHTML(columns=['name', 'fullname', 'pdb', 'references', 'numLigands',
↪'minDG', 'maxDG', 'associated_sets']))
```

A `target` can be accessed with its name in two ways

```
jnk1 = tgtset['jnk1']
pde2 = tgtset.getTarget('pde2')
```

2.2 The `target` class

contains all the available information about one target of PLBenchmarks. It also has two member variables, `_ligandSet` and `_edgeSet`, which contain the information about the available ligand and edges of the respective target. A `target` can either be accessed from the `targetSet` (see cell before) or initialized using its name via

```
jnk1 = targets.target('jnk1')
# The data in the column is stored in a pandas.Series and can be accessed via
jnk1.getDF(columns=None)
```

```
id                                                              1
name                                                         jnk1
fullname                                  c-Jun N-terminal kinase 1
netcharge                                                      xx
pdb                                                          2GMX
ligands            [lig_17124-1, lig_18624-1, lig_18625-1, lig_18...
references         [{'measurement': None}, {'calculation': ['10.1...
comments                                                     None
associated_sets                                 [Schrodinger JACS]
dtype: object
```

Access to the `edgeSet` and `ligandSet` in different formats is achieved by

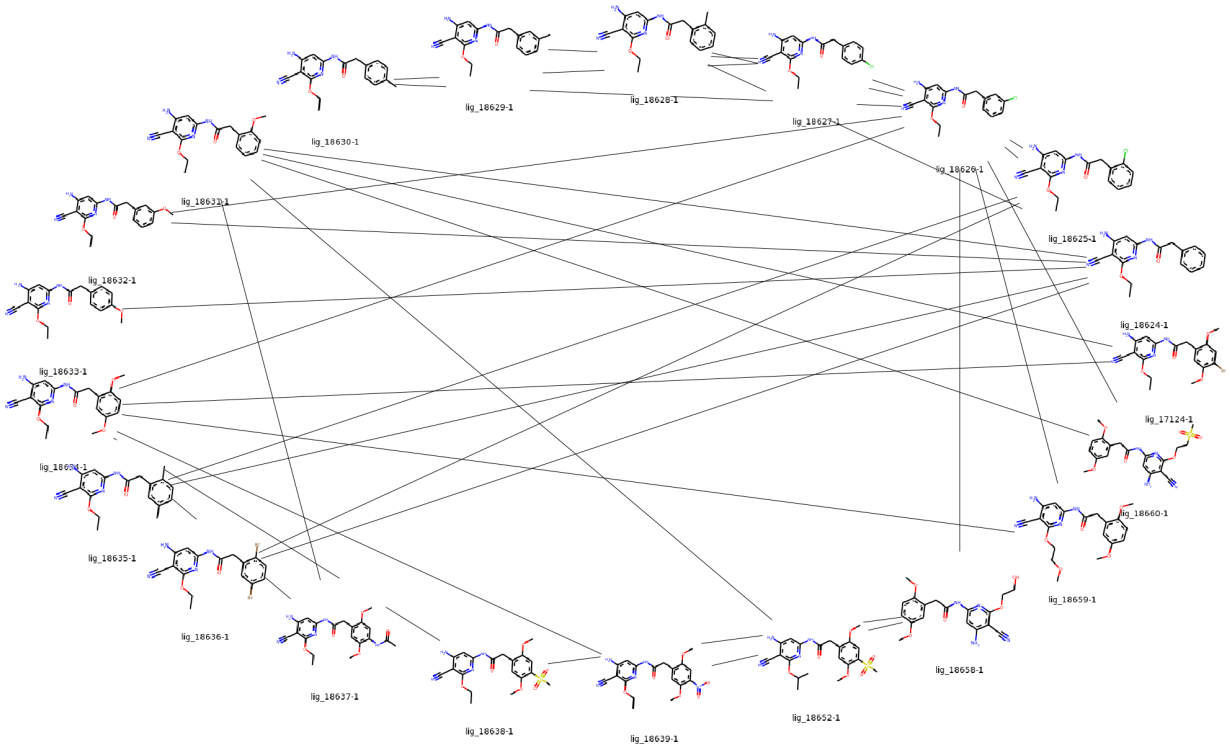
```
jnk1_ligands = jnk1.getLigandSet()
jnk1_ligands_df = jnk1.getLigandSetDF()
HTML(jnk1.getLigandSetHTML(columns = ['name', 'ROMol', 'measurement',
↪'DerivedMeasurement']))
```

```
jnk1_edges = jnk1.getEdgeSet()
jnk1_edges_df = jnk1.getEdgeSetDF()
HTML(jnk1.getEdgeSetHTML())
```

Finally, the set out of ligands and edges can be visualized in a graph:

```
graph = jnk1.getGraph()
```

```
/opt/anaconda3/envs/off-demo/lib/python3.7/site-packages/networkx/drawing/nx_pylab.
↪py:579: MatplotlibDeprecationWarning:
The iterable function was deprecated in Matplotlib 3.1 and will be removed in 3.3.␣
↪Use np.iterable instead.
  if not cb.iterable(width):
```



2.3 The `ligandSet` and `ligand` class

The `ligandSet` consists of a `dict` of `ligands` which are availabe for one target. It is accessible via `target.getLigandSet()`, but can also be initialized directly.

```
from PLBenchmarks import ligands
```

```
thrombin_ligands = ligands.ligandSet('thrombin')
HTML(thrombin_ligands.getHTML())
```

The `ligand` classes can be accessed from the `ligandSet` by their name. Each `ligand` has information about experimental data, references, SMILES string and SDF file path of the docked structure. Additionally, there are functions to derive and process the primary data, which is then added to the `pandas.Series` as a new entry.

```
lig_6e = thrombin_ligands['lig_6e']
lig_1a = thrombin_ligands.getLigand('lig_6e')
```

2.4 The edgeSet and edge class

The edgeSet contains a dict of edges which are availabe for one target. It is accessible via target.getEdgeSet(), but can also be initialized directly.

```
from PLBenchmarks import edges
```

```
pde2_edges = edges.edgeSet('pde2')
HTML(pde2_edges.getHTML())
```

```
pde2_edges.keys()
```

```
dict_keys(['edge_49220392_49137530', 'edge_49932714_49137530', 'edge_49582468_49137530
↪', 'edge_49396360_49137530', 'edge_50181001_49137530', 'edge_49585367_49137530',
↪'edge_49220392_49175828', 'edge_49220548_49220392', 'edge_49220548_49932129', 'edge_
↪49582468_49932129', 'edge_49396360_49175828', 'edge_49175828_49580115', 'edge_
↪49220548_49137374', 'edge_49220548_49580115', 'edge_49396360_49220548', 'edge_
↪49932714_49582390', 'edge_49396360_49582390', 'edge_50181001_49582390', 'edge_
↪50107616_49582390', 'edge_48168913_48271249', 'edge_49072088_48271249', 'edge_
↪50107616_48271249', 'edge_49137374_48271249', 'edge_49932714_49175789', 'edge_
↪49932714_49580115', 'edge_49932714_49582468', 'edge_48168913_49585367', 'edge_
↪50107616_49585367', 'edge_48168913_48022468', 'edge_43249674_48022468', 'edge_
↪48009208_43249674', 'edge_43249674_49175789', 'edge_49175789_49072088', 'edge_
↪48009208_49137374'])
```

The edge classes can be accessed from the edgeSet by their name. They are lightweight and provide only access to a pandas.DataFrame and a dict:

```
edge_49220392_49137530 = pde2_edges.getEdge('edge_49220392_49137530')
df = edge_49220392_49137530.getDF()
edge_49220392_49137530.getDict()
```

```
{'edge_49220392_49137530': ['lig_49220392', 'lig_49137530']}
```

CHAPTER

THREE

DATA

CHAPTER

FOUR

API DOCUMENTATION

4.1 Targets

4.2 Ligands

4.3 Edges

4.4 Utils

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search