
ProteinLigandBenchmarks Documentation

plbenchmark

Jun 01, 2022

CONTENTS:

1	Installing the Protein Ligand Benchmark Set	1
1.1	Installation from Source	1
2	Example Notebook: protein-ligand-benchmark	3
2.1	Related Publication	3
2.2	Get the whole set of targets in the dataset	3
2.3	The Target class	4
2.4	The LigandSet and Ligand class	5
2.5	The EdgeSet and Edge class	6
3	Data	7
3.1	Data file tree and file description	7
4	Contributions	9
4.1	Related Publication	9
5	API Documentation	11
5.1	Targets	11
5.2	Ligands	13
5.3	Edges	15
5.4	Utils	17
6	Indices and tables	19
	Python Module Index	21
	Index	23

INSTALLING THE PROTEIN LIGAND BENCHMARK SET

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

1.1 Installation from Source

The repository uses [git-lfs \(large file storage\)](#) for the storage of all the data file. Ideally git-lfs is installed first before cloning the repository.

```
conda create -n plbenchmark python=3.7 git-lfs
conda activate plbenchmark
git lfs clone https://github.com/openforcefield/protein-ligand-benchmark.git
cd protein-ligand-benchmark
conda env update --file environment.yml
pip install -e .
```


EXAMPLE NOTEBOOK: PROTEIN-LIGAND-BENCHMARK

2.1 Related Publication

The [preprint](#) on “Best practices for constructing, preparing, and evaluating protein-ligand binding affinity benchmarks” provides accompanying information to this benchmark dataset and how to use it for alchemical free energy calculations. For any suggestions of improvements please raise an issue in its [GitHub repository](#).

```
[1]: from plbenchmark import targets
      from IPython.core.display import HTML
```

```
Warning: Unable to load toolkit 'OpenEye Toolkit'. The Open Force Field Toolkit does not
↳ require the OpenEye Toolkits, and can use RDKit/AmberTools instead. However, if you
↳ have a valid license for the OpenEye Toolkits, consider installing them for faster
↳ performance and additional file format support: https://docs.eyesopen.com/toolkits/
↳ python/quickstart-python/linuxosx.html OpenEye offers free Toolkit licenses for
↳ academics: https://www.eyesopen.com/academic-licensing
```

2.2 Get the whole set of targets in the dataset

```
[2]: # it is initialized from the `plbenchmark/sample_data/targets.yml` file
      target_set = targets.TargetSet()
      # to see which targets are available, one can get a list of names
      target_set.get_names()
```

```
[2]: ['mcl1_sample']
```

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

```
[3]: HTML(target_set.get_html(columns=['name', 'fullname', 'pdb', 'references', 'numLigands',
↳ 'minDG', 'maxDG', 'associated_sets']))
```

```
/home/dhahn3/miniconda3/envs/plbenchmark/lib/python3.9/site-packages/pandas/core/dtypes/
↳ cast.py:1638: UnitStrippedWarning: The unit of the quantity is stripped when
↳ downcasting to ndarray.
      result[:] = values
```

```
[3]: <IPython.core.display.HTML object>
```

A target can be accessed with its name in two ways

```
[4]: mcl1 = target_set['mcl1_sample']
mcl1_2 = target_set.get_target('mcl1_sample')
```

2.3 The Target class

contains all the available information about one target of plbenchmark. It also has two member variables, `_ligand_set` and `_edge_set`, 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

```
[5]: mcl1 = targets.Target('mcl1_sample')
# The data in the column is stored in a pandas.Series and can be accessed via
mcl1.get_dataframe(columns=None)

/home/dhahn3/miniconda3/envs/plbenchmark/lib/python3.9/site-packages/pandas/core/dtypes/
↳ cast.py:1638: UnitStrippedWarning: The unit of the quantity is stripped when
↳ downcasting to ndarray.
result[:] = values
```

```
[5]: associated_sets          [Schrodinger JACS]
comments          hydrophobic interactions contributing to binding
date              2020-08-21
fullname          Induced myeloid leukemia cell differentiation ...
id                99
ligands           [lig_23, lig_26, lig_27, lig_28, lig_29, lig_3...
name              mcl1_sample
netcharge         xx
pdb              4HW3
references        {'calculation': ['10.1021/ja512751q', '10.1021...
numLigands        15
maxDG             -6.1 kilocalorie / mole
minDG             -9.0 kilocalorie / mole
std(DG)           0.9 kilocalorie / mole
calculation       REP1http://dx.doi.org/10.1021/ja512751qREP2Wan...
pdblinks          REP1http://www.rcsb.org/structure/4HW3REP24HW3...
dtype: object
```

Access to the EdgeSet and LigandSet in different formats is achieved by

```
[6]: mcl1_ligands = mcl1.get_ligand_set()
mcl1_ligands_df = mcl1.get_ligand_set_dataframe()
HTML(mcl1.get_ligand_set_html(columns = ['name', 'ROMol', 'measurement',
↳ 'DerivedMeasurement']))
```

```
[6]: <IPython.core.display.HTML object>
```

```
[7]: mcl1_edges = mcl1.get_edge_set()
mcl1_edges_df = mcl1.get_edge_set_dataframe()
HTML(mcl1.get_edge_set_html())

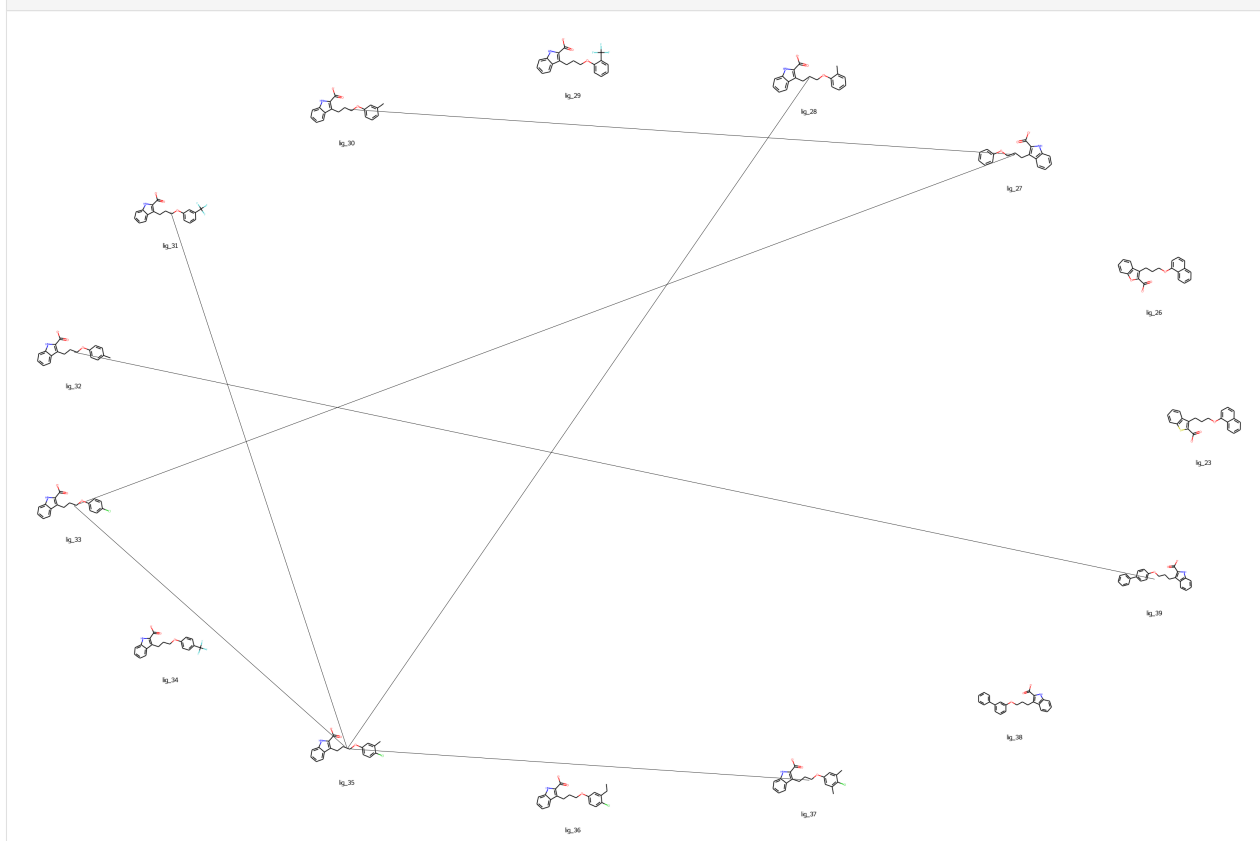
/home/dhahn3/miniconda3/envs/plbenchmark/lib/python3.9/site-packages/pandas/core/dtypes/
↳ cast.py:1638: UnitStrippedWarning: The unit of the quantity is stripped when
↳ downcasting to ndarray.
result[:] = values
```



```
[7]: <IPython.core.display.HTML object>
```

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

```
[8]: graph = mcl1.get_graph()
```



2.4 The LigandSet and Ligand class

The LigandSet consists of a Dict of Ligands which are available for one target. It is accessible via `Target.get_ligand_set()`, but can also be initialized directly.

```
[9]: from plbenchmark import ligands
```

```
[10]: mcl1_ligands = ligands.LigandSet('mcl1_sample')
      HTML(mcl1_ligands.get_html())
```

```
/home/dhahn3/miniconda3/envs/plbenchmark/lib/python3.9/site-packages/pandas/core/dtypes/
↳ cast.py:1638: UnitStrippedWarning: The unit of the quantity is stripped when
↳ downcasting to ndarray.
      result[:] = values
```

```
[10]: <IPython.core.display.HTML object>
```

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.

```
[11]: lig_30 = mcl1_ligands['lig_30']  
      lig_27 = mcl1_ligands.get_ligand('lig_27')
```

2.5 The EdgeSet and Edge class

The `EdgeSet` contains a dict of `Edges` which are available for one target. It is accessible via `Target.get_edge_set()`, but can also be initialized directly.

```
[12]: from plbenchmark import edges
```

```
[13]: mc11_edges = edges.EdgeSet('mc11_sample')
HTML(mc11_edges.get_html())
```

```

/home/dhahn3/miniconda3/envs/plbenchmark/lib/python3.9/site-packages/pandas/core/dtypes/
↳ cast.py:1638: UnitStrippedWarning: The unit of the quantity is stripped when
↳ downcasting to ndarray.
    result[:] = values

```

```
[13]: <IPython.core.display.HTML object>
```

```
[14]: mcl1_edges.keys()
```

```
[14]: dict_keys(['edge_28_35', 'edge_30_27', 'edge_31_35', 'edge_33_27', 'edge_35_33', 'edge_35_37', 'edge_39_32'])
```

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`:

```
[15]: edge_30_27 = mcl1_edges.get_edge('edge_30_27')
df = edge_30_27.get_dataframe()
edge_30_27.get_dict()
```

```
[15]: {'ligand_a': 'lig_30',  
      'ligand_b': 'lig_27',  
      'name': 'edge_30_27',  
      'Mol1': <rdkit.Chem.rdchem.Mol at 0x7f1a3046e8e0>,  
      'Smiles1': '[H]c1c(c(c2c(c1[H])C(=C(N2[H])C(=O)[O-  
→]))C([H])([H])C([H])([H])C([H])([H])O)c3c(cc(c(c3[H])C([H])([H])[H])[H])[H])[H]  
→',  
      'Mol2': <rdkit.Chem.rdchem.Mol at 0x7f1a30460700>,  
      'Smiles2':  
→ '[H]c1c(cc(cc(c1[H])[H])OC([H])([H])C([H])([H])C([H])([H])C2=C(N(c3c2c(cc(c3[H])[H])[H]  
→)) [H])[H]',  
      'exp. DeltaG [kcal/mol]': 1.73 <Unit('kilocalorie / mole')>,  
      'exp. Error [kcal/mol]': 0.22 <Unit('kilocalorie / mole')>>
```

3.1 Data file tree and file description

The data is organized as follows:

```

data
├── targets.yml                                # list of all targets and their directories
├── <date>_<target_name_1>                   # directory for target 1
│   ├── 00_data                               # metadata for target 1
│   │   ├── edges.yml                         # edges/perturbations
│   │   ├── ligands.yml                       # ligands and activities
│   │   └── target.yml                        # target
│   ├── 01_protein                            # protein data
│   │   ├── crd                              # coordinates
│   │   │   ├── cofactors_crystalwater.pdb   # cofactors and crystal waters
│   │   │   └── protein.pdb                  # aminoacid residues
│   │   ├── top                              # topology(s)
│   │   │   ├── amber99sb-star-ildn-mut.ff    # force field spec.
│   │   │   └── cofactors_crystalwater.top    # Gromacs TOP file of
│   │   └── protein.top                      # Gromacs TOP file of
│   ├── cofactors and crystal water
│   ├── amino acid residues
│   │   └── *.itp                            # Gromacs ITP file(s) to
│   └── be included in TOP files
│       ├── 02_ligands                        # ligands
│       │   ├── lig_<name_1>                 # ligand 1
│       │   │   ├── crd                      # coordinates
│       │   │   ├── lig_<name_1>.sdf         # SDF file
│       │   │   ├── top                      # topology(s)
│       │   │   ├── openff-1.0.0.offxml      # force field spec.
│       │   │   └── fflig_<name_1>.itp       # Gromacs ITP file :
│       │   ├── lig_<name_1>.itp             # Gromacs ITP file
│       │   ├── lig_<name_1>.top             # Gromacs TOP file
│       │   └── posre_lig_<name_1>.itp       # Gromacs ITP file :
│       ├── atom types
│       ├── position restraint file
│       │   ├── lig_<name_2>                 # ligand 2
│       │   ├── ...
│       │   ├── 03_hybrid                     # edges (perturbations)
│       │   └── edge_<name_1>_<name_2>       # edge between ligand 1 and ligand
└── 2

```

(continues on next page)

(continued from previous page)

```
| | | | |└─ water # edge in water
| | | | |└─ crd # coordinates
| | | | |└─ mergedA.pdb # merged conf based on┐
└─┐coords of ligand 1
| | | | |└─ mergedB.pdb # merged conf based on┐
└─┐coords of ligand 2
| | | | |└─ pairs.dat # atom mapping
| | | | |└─ score.dat # similarity score
| | | | |└─ top # topology(s)
| | | | |└─ openff-1.0.0.offxml # force field spec.
| | | | |└─ ffmerged.itp # Gromacs ITP file
| | | | |└─ ffMOL.itp # Gromacs ITP file
| | | | |└─ merged.itp # Gromacs ITP file
| | | | |...
└─ <date>_<target_name_2> # directory for target 2
...
```

CONTRIBUTIONS

- **Authors** David Hahn
- **Data Contributors** The authors of the following publications, especially Vytautas Gapsys and Christina E. M. Schindler.
 - V. Gapsys et al., Large scale relative protein ligand binding affinities using non-equilibrium alchemy, Chem. Sci., 2020,11, 1140-1152
 - Christina E. M. Schindler et al., Large-Scale Assessment of Binding Free Energy Calculations in Active Drug Discovery Projects, J. Chem. Inf. Model. 2020, 60, 11, 5457–5474
 - Laura Perez Benito et al., Predicting Activity Cliffs with Free-Energy Perturbation, J. Chem. Theory Comput. 2019, 15, 3, 1884–1895
- **Discussions and Suggestions** Christopher I. Bayly, Marko Breznik, Hannah E. Bruce Macdonald, John D.Chodera, Katharina Meier, Antonia S. J. S. Mey, David L. Mobley, Laura Perez Benito, Gary Tresadern, Gregory L. Warren and all members of the Open Force Field Initiative
- **Code review and discussions** Matt Thompson, Jeffrey Wagner

4.1 Related Publication

The [preprint](#) on “Best practices for constructing, preparing, and evaluating protein-ligand binding affinity benchmarks” provides accompanying information to this benchmark dataset and how to use it for alchemical free energy calculations. For any suggestions of improvements please raise an issue in its [GitHub repository](#).

API DOCUMENTATION

5.1 Targets

targets.py Functions and classes for handling the target data.

class plbenchmark.targets.Target(*name: str*)

Class to store the data of one target.

add_ligand_data()

Adds data from ligands to plbenchmark.targets.target. Molecule images and the minimum and maximum affinity are added.

Returns

None

find_links()

Processes primary data to have links in the html string of the target data

Returns

None

get_dataframe(*columns=None*)

Access the target data as a pandas.DataFrame

Parameters

cols – list of columns which should be returned in the pandas.DataFrame

Returns

pandas.DataFrame

get_edge_set()

Get plbenchmark:edges:edgeSet associated with the target

Returns

plbenchmark:edges:edgeSet object

get_edge_set_dataframe(*columns=None*)

Get plbenchmark:edges:edgeSet associated with the target as a pandas.DataFrame

Parameters

columns – list of columns which should be returned in the pandas.DataFrame

Returns

plbenchmark:edges:edgeSet object

get_edge_set_html(*columns=None*)

Get `plbenchmark.edges:edgeSet` associated with the target in a html string

Parameters

columns – list of edge which should be returned

Returns

html string

get_graph()

Get a graph representation of the ligand perturbations associated with the target in a `matplotlib.figure`

Returns

`matplotlib.figure`

get_ligand_set()

Get `ligandSet` associated with the target

Returns

`plbenchmark.ligands.ligandSet` object

get_ligand_set_dataframe(*columns=None*)

Get `ligandSet` associated with the target in a `pandas.DataFrame`

Parameters

columns – list of columns which should be returned in the `pandas.DataFrame`

Returns

`pandas.DataFrame`

get_ligand_set_html(*columns=None*)

Get `ligandSet` associated with the target in a html string

Parameters

columns – list of columns which should be returned

Returns

html string

get_name()

Access the name of the target.

Returns

name as a string

class `plbenchmark.targets.TargetSet`(*arg, **kw)

Class inherited from dict to store all available targets in `plbenchmark`.

get_dataframe(*columns=None*)

Convert `targetSet` class to `pandas.DataFrame`

Parameters

columns – list of columns which should be returned in the `pandas.DataFrame`

Returns

`pandas.DataFrame`

get_html(*columns=None*)

Access the `plbenchmark.targets:targetSet` as a HTML string

Parameters

cols – list of columns which should be returned in the `pandas.DataFrame`

Returns

HTML string

get_names()

Get a list of available target names

Returns

list of strings

get_target(name)

Accesses one target of the targetSet

Parameters**name** – string name of the target**Returns**

plbenchmark.targets.target class

plbenchmark.targets.**get_target_data_path(target)**

Gets the file path of the target data

Parameters**target** – string with target name**Returns**

list of directories (have to be joined with '/' to get the file path relative to the plbenchmark repository)

plbenchmark.targets.**get_target_dir(target)**

Gets the directory name of the target

Parameters**target** – string with target name**Returns**

string with directory name

plbenchmark.targets.**set_data_dir(path='/home/docs/checkouts/readthedocs.org/user_builds/plbenchmarks/checkouts/stable/pl')**

Gets the directory name of the target

Parameters**path** – string with path to data directory

5.2 Ligands

ligands.py Functions and classes for handling the ligand data.

class plbenchmark.ligands.**Ligand**(*d: dict, target: Optional[str] = None*)

Store and convert the data of one ligand in a pandas.Series.

add_mol_to_frame()

Adds a image file of the ligand to the pandas.DataFrame

Returns

None

derive_observables(*derived_type='dg', destination='DerivedMeasurement', out_unit=None*)

Derive observables from (stored) primary data, which is then stored in the pandas.DataFrame

Parameters

- **derived_type** – type of derived observable, can be any of ‘dg’ ‘ki’, ‘ic50’ or ‘pic50’
- **destination** – string with column name for ‘pandas.DataFrame’ where the derived observable should be stored.
- **out_unit** – unit of type `pint` unit of derived coordinate

Returns

None

find_links()

Processes primary data to have links in the html string of the ligand data

Returns

None

get_coordinate_file_path()

Get file path relative to the plbenchmark repository of the SDF coordinate file of the docked ligand

Returns

file path as string

get_dataframe(*columns=None*)

Access the ligand data as a `pandas.DataFrame`

Parameters

columns – list of columns which should be returned in the `pandas.DataFrame`

Returns

`pandas.DataFrame`

get_html(*columns=None*)

Access the ligand as a HTML string

Parameters

columns – list of columns which should be returned in the `pandas.DataFrame`

Returns

HTML string

get_image()

Creates a molecule image.

Returns

`PIL.Image` object

get_molecule()

Get molecule object with coordinates of the docked ligand

Returns

file path as string

get_name()

Access the name of the ligand.

Returns

name: string

class `plbenchmark.ligands.LigandSet(target, *arg, **kw)`

Class inherited from dict to store all available ligands of one target.

get_dataframe(*columns=None*)

Access the `ligandSet` as a `pandas.DataFrame`

Parameters

columns – list of columns which should be returned in the `pandas.DataFrame`

Returns

`pandas.DataFrame`

get_html(*columns=None*)

Access the `plbenchmark.ligands.ligandSet` as a HTML string

Parameters

columns – list of columns which should be returned in the `pandas.DataFrame`

Returns

HTML string

get_ligand(*name*)

Accesses one ligand of the `ligandSet`

Parameters

name – string name of the ligand

Returns

`plbenchmark.ligands.ligand` class

get_list()

Returns list of ligands

Returns

list of ligand names

get_molecules()

Returns a dict with names as keys and values as `py:class:openforcefield:topology:Molecule` objects

Returns

dict

5.3 Edges

edges.py Functions and classes for handling the perturbation edges.

class `plbenchmark.edges.Edge`(*d: dict*)

Store and convert the data of one perturbation (“edge”) in a `pandas.Series`.

Parameters

d – dict with the edge data

Returns

None

add_ligand_data(*ligand_set*)

Adds data from ligands to edge. Molecule images and the affinity difference are added.

Parameters

ligand_set – `plbenchmark.ligands.ligandSet` class of the same target

Returns

None

get_dataframe(*columns=None*)

Access the edge data as a `pandas.DataFrame`

Parameters

cols – list of columns which should be returned in the `pandas.DataFrame`

Returns

`pandas.DataFrame`

get_dict()

Access the edge data as a `dict` which contains the name of the edge as key and the names of the two ligands as list.

Returns

`dict`

get_name()

Access the name of the edge.

Returns

name as string

class `plbenchmark.edges.EdgeSet`(*target, *arg, **kw*)

Class inherited from `dict` to store all available edges of one target.

get_dataframe(*columns=None*)

Access the `plbenchmark:edges.edgeSet` as a `pandas.DataFrame`

Parameters

cols – list of columns which should be returned in the `pandas.DataFrame`

Returns

`pandas.DataFrame`

get_dict()

Access the `plbenchmark:edges.edgeSet` as a `dict` which contains the name of the edges as key and the names of the two ligands in a list as items.

Returns

`dict`

get_edge(*name*)

Accesses one edge of the `plbenchmark.edges.edgeSet`

Parameters

name – string name of the edge

Returns

`plbenchmark:edges:edge` class

get_html(*columns=None*)

Access the `plbenchmark:edges.edgeSet` as a HTML string

Parameters

cols – list of columns which should be returned in the `pandas.DataFrame`

Returns

HTML string

5.4 Utils

utils.py Contains utility functions

`plbenchmark.utils.convert_error(error_value, value, original_type, final_type, temperature=300.0, out_unit=None)`

Converts an experimental value into another derived quantity with specified unit.

Parameters

- **error_value** – float, error of val, numerical value
- **value** – float, numerical value
- **original_type** – string, code for the original observable. Can be *dg*, *ki*, *ic50*, *pic50*
- **final_type** – string, code for the desired derived quantity. Can be *dg*, *ki*, *ic50*, *pic50*
- **temperature** – float, temperature in kelvin
- **out_unit** – unit of type `pint`, output unit of `final_type`, needs to fit to the requested `final_type`

Returns

`pint.Quantity` with desired unit

`plbenchmark.utils.convert_value(value, original_type, final_type, temperature=300.0, out_unit=None)`

Converts an experimental value into another derived quantity with specified unit.

Parameters

- **value** – float, numerical value
- **original_type** – string, code for the original observable. Can be *dg*, *ki*, *ic50*, *pic50*
- **final_type** – string, code for the desired derived quantity. Can be *dg*, *ki*, *ic50*, *pic50*
- **temperature** – float, temperature in kelvin
- **out_unit** – unit of type `pint`, output unit of `final_type`, needs to fit to the requested `final_type`

Returns

`pint.Quantity` with desired unit

`plbenchmark.utils.find_doi_url(doi)`

Finds the links to a digital object identifier (doi).

Parameters

doi – string

Returns

string compiled string including the urls to the publication

`plbenchmark.utils.find_pdb_url(pdb)`

Finds the links to a pdb or a list of pdb codes.

Parameters

pdb – string or list of strings

Returns

string compiled string including the urls to the pdb entries

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

p

`plbenchmark.edges`, [15](#)
`plbenchmark.ligands`, [13](#)
`plbenchmark.targets`, [11](#)
`plbenchmark.utils`, [17](#)

A

add_ligand_data() (plbenchmark.edges.Edge method), 15
 add_ligand_data() (plbenchmark.targets.Target method), 11
 add_mol_to_frame() (plbenchmark.ligands.Ligand method), 13

C

convert_error() (in module plbenchmark.utils), 17
 convert_value() (in module plbenchmark.utils), 17

D

derive_observables() (plbenchmark.ligands.Ligand method), 13

E

Edge (class in plbenchmark.edges), 15
 EdgeSet (class in plbenchmark.edges), 16

F

find_doi_url() (in module plbenchmark.utils), 17
 find_links() (plbenchmark.ligands.Ligand method), 14
 find_links() (plbenchmark.targets.Target method), 11
 find_pdb_url() (in module plbenchmark.utils), 17

G

get_coordinate_file_path() (plbenchmark.ligands.Ligand method), 14
 get_dataframe() (plbenchmark.edges.Edge method), 15
 get_dataframe() (plbenchmark.edges.EdgeSet method), 16
 get_dataframe() (plbenchmark.ligands.Ligand method), 14
 get_dataframe() (plbenchmark.ligands.LigandSet method), 14
 get_dataframe() (plbenchmark.targets.Target method), 11

get_dataframe() (plbenchmark.targets.TargetSet method), 12
 get_dict() (plbenchmark.edges.Edge method), 16
 get_dict() (plbenchmark.edges.EdgeSet method), 16
 get_edge() (plbenchmark.edges.EdgeSet method), 16
 get_edge_set() (plbenchmark.targets.Target method), 11
 get_edge_set_dataframe() (plbenchmark.targets.Target method), 11
 get_edge_set_html() (plbenchmark.targets.Target method), 11
 get_graph() (plbenchmark.targets.Target method), 12
 get_html() (plbenchmark.edges.EdgeSet method), 16
 get_html() (plbenchmark.ligands.Ligand method), 14
 get_html() (plbenchmark.ligands.LigandSet method), 15
 get_html() (plbenchmark.targets.TargetSet method), 12
 get_image() (plbenchmark.ligands.Ligand method), 14
 get_ligand() (plbenchmark.ligands.LigandSet method), 15
 get_ligand_set() (plbenchmark.targets.Target method), 12
 get_ligand_set_dataframe() (plbenchmark.targets.Target method), 12
 get_ligand_set_html() (plbenchmark.targets.Target method), 12
 get_list() (plbenchmark.ligands.LigandSet method), 15
 get_molecule() (plbenchmark.ligands.Ligand method), 14
 get_molecules() (plbenchmark.ligands.LigandSet method), 15
 get_name() (plbenchmark.edges.Edge method), 16
 get_name() (plbenchmark.ligands.Ligand method), 14
 get_name() (plbenchmark.targets.Target method), 12
 get_names() (plbenchmark.targets.TargetSet method), 13
 get_target() (plbenchmark.targets.TargetSet method), 13
 get_target_data_path() (in module plbenchmark.targets), 13
 get_target_dir() (in module plbenchmark.targets), 13

L

Ligand (*class in plbenchmark.ligands*), [13](#)

LigandSet (*class in plbenchmark.ligands*), [14](#)

M

module

plbenchmark.edges, [15](#)

plbenchmark.ligands, [13](#)

plbenchmark.targets, [11](#)

plbenchmark.utils, [17](#)

P

plbenchmark.edges

module, [15](#)

plbenchmark.ligands

module, [13](#)

plbenchmark.targets

module, [11](#)

plbenchmark.utils

module, [17](#)

S

set_data_dir() (*in module plbenchmark.targets*), [13](#)

T

Target (*class in plbenchmark.targets*), [11](#)

TargetSet (*class in plbenchmark.targets*), [12](#)