QCArchive Documentation

Release 1.0

Daniel G. A. Smith

Oct 15, 2020

BASIC EXAMPLES

These examples demonstrate how to use QCPortal to access data on the QCArchive in a variety of situations.

All examples are available in your browser through Binder. Alternatively, you may run these examples locally. To do so, please install QCPortal with either `conda` or `pip`.

CHAPTER

ONE

LOCAL INSTALLATION

Install `qcportal` using conda:

```
conda install qcportal -c conda-forge
```

or with `pip`:

```
pip install qcportal
```

You can run this notebook online in a session or view it on Github.

1.1 First Steps

The Molecular Sciences Software Institute hosts the Quantum Chemistry Archive (QCArchive) and makes this data available to the entire Computational Molecular Sciences community free of charge. The QCArchive is both a database to view, analyze, and explore existing data as well as a live instance that continuously generates new data as directed by the community.

The primary interface to this database in Python is a through a `FractalClient` from the `qcportal` package which can be downloaded via pip (`pip install -e qcportal`) or conda (`conda install qcportal -c conda-forge`). A new `FractalClient` automatically connects to MolSSI's central server and has access to all data contained within the QCArchive.

[1]:
```
import qcportal as ptl
client = ptl.FractalClient()
client
```

[1]:
```
FractalClient(server_name='The MolSSI QCArchive Server', address='https://api.
↪qcarchive.molssi.org:443/', username='None')
```

1.1.1 Finding Collections

One of the main ways to explore the QCArchive is to examine `Collections` which are structures that allow easy manipulation of data in preset ways. Several example of `Collections` contained within the QCArchive are as follows:

- `Dataset` - A dataset where each record corresponds to a single molecule, with one or more QM methods applied to that molecule.
- `ReactionDataset` - A dataset where each record is a combination of molecules (e.g. interaction and reaction energies). Each record contains data from one or more QM methods.
- `OptimizationDataset` - A dataset where each record represents geometry optimization of a molecule.
- `TorsionDriveDataset` - A dataset which organizes many molecular torsion scans together for data exploration, analysis, and methodology comparison (see the TorsionDrive Dataset example for more details).

[2]:
```
client.list_collections().head()
```

[2]:
```
                                          tagline
collection name
Dataset    GDB13-T                    In progress
           OpenFF Discrepancy Benchmark 1    None
           OpenFF NCI250K Boron 1            None
           OpenFF Optimization Set 1         None
           OpenFF VEHICLe Set 1              None
```

Specific `Collection` types can be queried to limit the amount of collections to browse through:

[3]:
```
client.list_collections("reactiondataset").head()
```

[3]:
```
                                                                tagline
collection      name
ReactionDataset A21        Equilibrium complexes from A24 database of sma...
                A24        Interaction energies for small bimolecular com...
                ACONF                     Conformation energies for alkanes
                AlkBind12  Binding energies of saturated and unsaturated ...
                AlkIsod14     Isodesmic reaction energies for alkanes N=3--8
```

1.1.2 Exploring Collections

Collections can be obtained by pulling their data from the central server. A collection is primarily metadata and extremely large collections can be pulled in a few seconds. For this example, we will explore S22 dataset which is a small interaction energy dataset of 22 common dimers such as the water dimer, methane dimer, and more. To obtain this collection:

[4]:
```
ds = client.get_collection("ReactionDataset", "S22")
print(ds)
```

```
ReactionDataset(name=`S22`, id='184', client='https://api.qcarchive.molssi.org:443/')
```

1.1.3 Statistics and Visualization

Visual statics and plotting can be generated by the `visualize` command:

[5]:
```
ds.visualize(method="B2PLYP", basis=["def2-svp", "def2-tzvp"], bench="S220", kind=
↪"violin")
```

Data type cannot be displayed: application/vnd.plotly.v1+json, text/html

1.1.4 Next steps

Congratulations! You have taken the first steps to exploring the data within the QCArchive. Please consider viewing the Reaction Dataset and the TorsionDrive Dataset examples for a more in depth look at these Collections and what you can do with them.

Feel free to explore the data you access through these examples in detail. When you connect a `FractalClient` to the server without a username and password, the data is open to explore and cannot alter what is saved on the server itself. So if you change your local data, the server data remains untouched!

You can run this notebook online in a session or view it on Github.

1.2 Reaction Datasets

ReactionDatasets are datasets where the primary index represents a chemical reaction, made up of stoichiometrically weighted linear combinations of individual computations. For example, an interaction energy dataset would have an index of the complex subtracted by the individual monomers to obtain a final interaction energy. This idea can extended to standard reaction energies, conformational defect energies, and more.

This dataset type has been developed by the QCArchive Team in conjunction with:

- David Sherrill
- Lori Burns
- Daniel Nascimento
- Dom Sirianni

To begin, we can connect to the MolSSI QCArchive server:

[1]:
```
import qcportal as ptl
client = ptl.FractalClient()
client
```

[1]:
```
FractalClient(server_name='The MolSSI QCArchive Server', address='https://api.
↪qcarchive.molssi.org:443/', username='None')
```

The current `ReactionDatasets` can be explored below:

[2]:
```
client.list_collections("ReactionDataset").head()
```

[2]:

		tagline
collection	name	
ReactionDataset	A21	Equilibrium complexes from A24 database of sma...
	A24	Interaction energies for small bimolecular com...

(continues on next page)

(continued from previous page)

```
                ACONF                       Conformation energies for alkanes
                AlkBind12  Binding energies of saturated and unsaturated ...
                AlkIsod14     Isodesmic reaction energies for alkanes N=3--8
```

1.2.1 Exploring a Dataset

For this example, we will explore S22 dataset which is a small interaction energy dataset of 22 common dimers such as the water dimer, methane dimer, and more. To obtain this collection:

[3]:
```
ds = client.get_collection("ReactionDataset", "S22")
print(ds)
```

```
ReactionDataset(name=`S22`, id='184', client='https://api.qcarchive.molssi.org:443/')
```

The reactions in the dataset – dimerization reactions in the case of S22 – can be listed:

[4]:
```
ds.get_index()
```

[4]:
```
['2-Pyridone-2-Aminopyridine Complex',
 'Adenine-Thymine Complex Stack',
 'Adenine-Thymine Complex WC',
 'Ammonia Dimer',
 'Benzene-Ammonia Complex',
 'Benzene Dimer PD',
 'Benzene Dimer T-Shape',
 'Benzene-HCN Complex',
 'Benzene-Methane Complex',
 'Benzene-Water Complex',
 'Ethene Dimer',
 'Ethene-Ethine Complex',
 'Formamide Dimer',
 'Formic Acid Dimer',
 'Indole-Benzene Complex Stack',
 'Indole-Benzene Complex T-Shape',
 'Methane Dimer',
 'Phenol Dimer',
 'Pyrazine Dimer',
 'Uracil Dimer HB',
 'Uracil Dimer Stack',
 'Water Dimer']
```

Datasets contain two types of data, those computed through QCArchive ("native") and those that are provided from external sources ("contributed"). Contributed data often come from experiments or very costly benchmarks taken from literature.

`Datasets` and `ReactionDatasets` provide a list of all data that has been computed or contributed through the `list_values` method.

[5]:
```
ds.list_values().head()
```

[5]:
```
                                                    stoichiometry  \
native driver  program method  basis       keywords
False  Unknown Unknown Unknown Unknown     Unknown           default
                                          Unknown           default
                                          Unknown           default
True   energy  psi4    b2plyp  aug-cc-pvdz scf_default           cp
```

(continues on next page)

(continued from previous page)

```
                                          scf_default       default

                                                                                name
native driver  program method  basis       keywords
False  Unknown Unknown Unknown Unknown     Unknown                          S220
                                          Unknown                          S22a
                                          Unknown                          S22b
True   energy  psi4    b2plyp  aug-cc-pvdz scf_default  cp-B2PLYP/aug-cc-pvdz
                                          scf_default     B2PLYP/aug-cc-pvdz
```

Here, we have listed the first five available data sources. The first three are contributed, marked by `native=False` and correspond to benchmarks. The last two are computed data (`native=True`).

There are six primary keys to describe data:

- `native` - Whether a computation was done using QCArchive.
- `driver` - The type of computation, this can be energy, gradient, Hessian, and properties.
- `program` - The program used in the computation.
- `method` - The quantum chemistry, semiempirical, AI-model, or force field used in the computation.
- `basis` - The basis used in the computation.
- `keywords` - A keywords alias used in the computation, specific to the details of the program or procedure.

In addition, there is also the `stoichiometry` field which is unique to `ReactionDatasets`. There exist several ways to compute the interaction energy: counterpoise-corrected (`cp`), non-counterpoise-corrected (`default`), and Valiron–Mayer function counterpoise (`vmfc`). The `stoichiometry` field allows for the selection of this particular form.

Searches in `list_values` may be narrowed by specifying some or all of the keys. In this case, we will filter our history by the DFT method `B2PLYP` and the basis set `def2-SVP`.

[6]:
```
ds.list_values(method="B2PLYP", basis="def2-SVP")
```

[6]:
```
                                                 stoichiometry  \
native driver program method basis    keywords
True   energy psi4    b2plyp def2-svp scf_default            cp
                                      scf_default       default

                                                                     name
native driver program method basis    keywords
True   energy psi4    b2plyp def2-svp scf_default  cp-B2PLYP/def2-svp
                                      scf_default     B2PLYP/def2-svp
```

1.2.2 Querying Data

To obtain the data for the computations we must query them from the server. For example, we can pull all `B3LYP-D3M` interaction energies:

[7]:
```
ds.get_values(method="B3LYP-D3M")
```

[7]:
```
                                  B3LYP-D3M/def2-tzvp  \
2-Pyridone-2-Aminopyridine Complex          -18.536530
Adenine-Thymine Complex Stack               -12.149707
Adenine-Thymine Complex WC                  -17.833451
```

(continues on next page)

(continued from previous page)

```
Ammonia Dimer                                -4.049052
Benzene Dimer PD                             -2.556100
Benzene Dimer T-Shape                        -3.072012
Benzene-Ammonia Complex                      -2.934200
Benzene-HCN Complex                          -5.279021
Benzene-Methane Complex                      -1.555573
Benzene-Water Complex                        -4.613285
Ethene Dimer                                 -1.668464
Ethene-Ethine Complex                        -1.878851
Formamide Dimer                             -17.436781
Formic Acid Dimer                           -20.668411
Indole-Benzene Complex Stack                 -4.398316
Indole-Benzene Complex T-Shape               -6.363817
Methane Dimer                                -0.522247
Phenol Dimer                                 -8.032781
Pyrazine Dimer                               -4.096590
Uracil Dimer HB                             -21.922461
Uracil Dimer Stack                          -10.781041
Water Dimer                                  -6.427460

                                 B3LYP-D3M/aug-cc-pvdz  B3LYP-D3M/def2-svp  \
2-Pyridone-2-Aminopyridine Complex          -19.005121          -22.831506
Adenine-Thymine Complex Stack               -12.897930          -15.577143
Adenine-Thymine Complex WC                  -18.449484          -22.574701
Ammonia Dimer                                -3.509980           -6.248386
Benzene Dimer PD                             -3.058981           -3.459984
Benzene Dimer T-Shape                        -3.617173           -3.597379
Benzene-Ammonia Complex                      -2.833251           -3.251346
Benzene-HCN Complex                          -5.479076           -5.480155
Benzene-Methane Complex                      -1.830850           -1.917835
Benzene-Water Complex                        -3.924570           -4.926573
Ethene Dimer                                 -2.050798           -2.294543
Ethene-Ethine Complex                        -2.114814           -2.330609
Formamide Dimer                             -17.546706          -21.689185
Formic Acid Dimer                           -20.536286          -25.933297
Indole-Benzene Complex Stack                 -5.056658           -5.736506
Indole-Benzene Complex T-Shape               -6.796603           -7.081938
Methane Dimer                                -0.813825           -0.672244
Phenol Dimer                                 -7.974372          -10.977429
Pyrazine Dimer                               -4.664210           -5.443984
Uracil Dimer HB                             -22.497729          -25.623412
Uracil Dimer Stack                          -11.125815          -13.223797
Water Dimer                                  -5.539915           -9.002674

                                 B3LYP-D3M/aug-cc-pvtz
2-Pyridone-2-Aminopyridine Complex          -18.238308
Adenine-Thymine Complex Stack               -11.778090
Adenine-Thymine Complex WC                  -17.687043
Ammonia Dimer                                -3.328184
Benzene Dimer PD                             -2.467563
Benzene Dimer T-Shape                        -3.016720
Benzene-Ammonia Complex                      -2.572470
Benzene-HCN Complex                          -5.221790
Benzene-Methane Complex                      -1.552191
Benzene-Water Complex                        -3.727725
Ethene Dimer                                 -1.678959
Ethene-Ethine Complex                        -1.823828
```

(continues on next page)

(continued from previous page)

```
Formamide Dimer                              -17.104115
Formic Acid Dimer                            -20.385421
Indole-Benzene Complex Stack                  -4.213569
Indole-Benzene Complex T-Shape                -6.083396
Methane Dimer                                 -0.511469
Phenol Dimer                                  -7.523356
Pyrazine Dimer                                -4.036813
Uracil Dimer HB                              -21.904878
Uracil Dimer Stack                           -10.486322
Water Dimer                                   -5.417591
```

The units of these energies are stored in `ds.units`:

[8]:
```
ds.units
```

[8]:
```
'kcal / mol'
```

1.2.3 Statistics and Visualization

Visual statistics and plotting can be generated by the `visualize` command:

[9]:
```
ds.visualize(method=["B3LYP", "B3LYP-D3", "B3LYP-D3M"], basis=["def2-tzvp"], groupby=
↪"D3")
```

Data type cannot be displayed: application/vnd.plotly.v1+json, text/html

[10]:
```
ds.visualize(method=["B3LYP", "B3LYP-D3", "B2PLYP", "B2PLYP-D3"], basis="def2-tzvp",␣
↪groupby="D3", kind="violin")
```

Data type cannot be displayed: application/vnd.plotly.v1+json, text/html

1.2.4 Next steps

The next sections cover other collections that are used for organizing workflows, such as geometry optimization. There are more examples using `Dataset` and `ReactionDataset` in the Cookbook. Full documentation of Dataset and ReactionDataset are available in the QCPortal documentation.

You can run this notebook online in a session or view it on Github.

1.3 Optimization Datasets

An `OptimizationDataset` represents geometry optimizations performed on a set of molecules.

[1]:
```
import qcportal as ptl
client = ptl.FractalClient()
client
```

[1]:
```
FractalClient(server_name='The MolSSI QCArchive Server', address='https://api.
↪qcarchive.molssi.org:443/', username='None')
```

[2]:
```
client.list_collections("OptimizationDataset")
```

[2]:

collection	name	tagline
OptimizationDataset	FDA Optimization Dataset 1	None
	JGI Metabolite Set 1	None
	OpenFF Discrepancy Benchmark 1	None
	OpenFF Full Optimization Benchmark 1	None
	OpenFF NCI250K Boron 1	None
	OpenFF Optimization Set 1	None
	OpenFF Primary Optimization Benchmark 1	None
	OpenFF VEHICLe Set 1	None
	PEI	None
	Pfizer Discrepancy Optimization Dataset 1	None
	QM8-T	None
	SMIRNOFF Coverage Set 1	None

[3]:
```
ds = client.get_collection("OptimizationDataset", "SMIRNOFF Coverage Set 1")
```

1.3.1 Exploring the Dataset

Each row of the dataset is comprised of a `Entry` which corresponds to a molecule.

[4]:
```
ds.df.head()
```

[4]:
```
Empty DataFrame
Columns: []
Index: [COC(O)OC-0, C[S-]-0, CS-0, CO-0, CCO-0]
```

New computations are based off specifications which contain many additional parameters to tune the geometry optimization as well as the underlying computational method. Here, we can list all specifications that are attributed to this dataset.

[5]:
```
ds.list_specifications()
```

[5]:

Name	Description
default	Standard OpenFF optimization quantum chemistry...

In this case, there is one specification corresponding to a single level of theory. It is important to recall that these Collections are "live": new specifications can be added and individual optimizations can be under computation. To see the current status of each specification the `status` function is provided:

[6]:
```
ds.status(["default"])
```

[6]:
```
             default
COMPLETE        1118
INCOMPLETE         7
ERROR              7
```

The number of geometry steps for each molecule can be shown:

[7]:
```
ds.counts()
```

[7]:
```
                default
COC(O)OC-0         11.0
C[S-]-0             6.0
CS-0                5.0
CO-0                4.0
CCO-0               8.0
...                 ...
CSSCCN=C=S-7       26.0
CSSCCN=C=S-8       45.0
CSSCCN=C=S-9       48.0
CSSCCN=C=S-10      60.0
CSSCCN=C=S-11      38.0

[1118 rows x 1 columns]
```

Individual records can be pulled for molecules:

[8]:
```
optrec = ds.get_record(name="CCO-0", specification="default")
```

These records contain the geometries and energies of the optimization trajectory. Below are some example data that may be pulled from an `OptimizationRecord`. The initial and final molecules may be extracted:

[9]:
```
optrec.get_initial_molecule()
```

```
_ColormakerRegistry()
```

```
NGLWidget()
```

[10]:
```
optrec.get_final_molecule()
```

```
NGLWidget()
```

And the energy trajectory of the optimization can be plotted:

[11]:
```
optrec.show_history()
```

Data type cannot be displayed: application/vnd.plotly.v1+json, text/html

You can run this notebook online in a session or view it on Github.

1.4 TorsionDrive Datasets

An individual TorsionDrive is specific workflow in the QCArchive ecosystem that computes the energy profile of rotatable dihedral (torsion) for use in Force Field fitting. This workflow has been developed by the QCArchive Team in conjunction with:

- Lee-Ping Wang
- John Chodera
- Chaya Stern
- Yudong Qiu
- The Open Force Field Initiative

The top-level TorsionDrive code can be found here.

The TorsionDrive Dataset organizes many TorsionDrives together for data exploration, analysis, and methodology comparison. To begin, we can connect to the MolSSI QCArchive server and query all known TorsionDrive Datasets:

[1]:
```
import qcportal as ptl
client = ptl.FractalClient()
client
```

[1]:
```
FractalClient(server_name='The MolSSI QCArchive Server', address='https://api.
↪qcarchive.molssi.org:443/', username='None')
```

[2]:
```
client.list_collections("TorsionDriveDataset")
```

[2]:
```
↪             tagline
collection          name
TorsionDriveDataset Fragment Stability Benchmark
↪                None
                      OpenFF Fragmenter Phenyl Benchmark       Phenyl substituent␣
↪torsional barrier heights.
                      OpenFF Full TorsionDrive Benchmark 1
↪                None
                      OpenFF Group1 Torsions
↪                None
                      OpenFF Primary TorsionDrive Benchmark 1
↪                None
                      OpenFF Substituted Phenyl Set 1
↪                None
                      Pfizer Discrepancy Torsion Dataset 1
↪                None
                      SMIRNOFF Coverage Torsion Set 1
↪                None
                      TorsionDrive Paper
↪                None
```

One of the datasets can be obtained in the canonical manner:

[3]:
```
ds = client.get_collection("TorsionDriveDataset", "OpenFF Fragmenter Phenyl Benchmark
↪")
```

1.4.1 Exploring the Dataset

Each row of the dataset is comprised of a `Entry` which contains a list of starting molecules for the TorsionDrive, the dihedral of interest (in this case zero-indexed), and the scan resolution of the dihedral. For the purposes of the underlying DataFrame each row is the name of this `Entry`.

[4]:
```
ds.df.head()
```

[4]:
```
Empty DataFrame
Columns: []
Index: [c1c[cH:1][c:2](cc1)[C:3](=[O:4])O, c1[cH:1][c:2](cnc1)[C:3](=[O:4])O, [cH:
↪1]1cncc[c:2]1[C:3](=[O:4])O, [cH:1]1cc(nc[c:2]1[C:3](=[O:4])O)[O-], Cc1c[cH:1][c:
↪2](cn1)[C:3](=[O:4])O]
```

New computations are based off specifications which contain many additional parameters to tune the torsiondrive as well as the underlying computational method. Here, we can list all specifications that are attributed to this dataset.

[5]:
```
ds.list_specifications()
```

[5]:
```
                                Description
Name
UFF        UFF gradient evaluation with RDKit
B3LYP-D3       B3LYP-D3 evaluation with Psi4
```

As we can see, we have several specifications at different levels of theory. It is important to recall that these Collections are "live": new specifications can be added and individual TorsionDrives can be under computation. To see the current status of each specification the `status` function is provided:

[6]:
```
ds.status(["uff", "b3lyp-d3"])
```

[6]:
```
             UFF  B3LYP-D3
COMPLETE     165       226
INCOMPLETE    62         1
```

[7]:
```
ds.status(["b3lyp-d3"], collapse=False, status="COMPLETE").head()
```

[7]:
```
                                          B3LYP-D3
c1c[cH:1][c:2](cc1)[C:3](=[O:4])O       COMPLETE
c1[cH:1][c:2](cnc1)[C:3](=[O:4])O       COMPLETE
[cH:1]1cncc[c:2]1[C:3](=[O:4])O         COMPLETE
[cH:1]1cc(nc[c:2]1[C:3](=[O:4])O)[O-]   COMPLETE
Cc1c[cH:1][c:2](cn1)[C:3](=[O:4])O      COMPLETE
```

1.4.2 Visualizing the TorsionDrives

TorsionDrives can be visualized via the `visualize` command. Multiple torsiondrives can be plotted on the same graph for comparison.

[8]:
```
ds.visualize(["[CH3:4][O:3][c:2]1[cH:1]cccc1", "[CH3:4][O:3][c:2]1[cH:1]ccnc1"],
↪"B3LYP-D3", units="kJ / mol")
```

Data type cannot be displayed: application/vnd.plotly.v1+json, text/html

1.4.3 Computational Requirements

We can check the number of optimizations and individual gradient evaluations by using the `count` method. By default, the `count` method shows the number of individual geometry optimizations:

[9]:
```
ds.counts(["[CH3:4][O:3][c:2]1[cH:1]cccc1", "[CH3:4][O:3][c:2]1[cH:1]ccnc1"])
```

[9]:
```
                                UFF  B3LYP-D3
[CH3:4][O:3][c:2]1[cH:1]cccc1    49         49
[CH3:4][O:3][c:2]1[cH:1]ccnc1    49         53
```

In addition to individual optimization runs, we can also find a sum of how many gradient evaluations were performed in the course of the run.

[10]:
```
ds.counts(["[CH3:4][O:3][c:2]1[cH:1]cccc1", "[CH3:4][O:3][c:2]1[cH:1]ccnc1"], count_
↪gradients=True)
```

[10]:
```
                                UFF  B3LYP-D3
[CH3:4][O:3][c:2]1[cH:1]cccc1   606        601
[CH3:4][O:3][c:2]1[cH:1]ccnc1   538        680
```

1.4.4 Deep data inspection

The TorsionDriveDataset also allows exploration of every single computation and molecule created during the individual TorsionDrive executions. Examples below this point will only be for those who wish to explore all of the individual computations in a TorsionDrive Dataset.

These TorsionDrive Datasets are different from canonical `ReactionDataset` and `Dataset` collections as each item in the underlying Pandas DataFrame is a TorsionDriveRecord object which contains links to all data used in the TorsionDrive. We can observe the these in the underlying DataFrame:

[11]:
```
ds.df.head()
```

[11]:
```
                                                                                 ⌐
↪UFF  \
c1c[cH:1][c:2](cc1)[C:3](=[O:4])O      id=1761595 hash_
↪index=973a82dee50895e16d0a1099...
c1[cH:1][c:2](cnc1)[C:3](=[O:4])O      id=1761596 hash_
↪index=52cd24c390a25eee94e2ce1b...
[cH:1]1cncc[c:2]1[C:3](=[O:4])O        id=1761597 hash_
↪index=50db63b5d61dbaa0b326d588...
[cH:1]1cc(nc[c:2]1[C:3](=[O:4])O)[O-]  id=1761598 hash_
↪index=aa028d05f9aac9984667e4e2...
Cc1c[cH:1][c:2](cn1)[C:3](=[O:4])O     id=1761599 hash_
↪index=16aeb7c248405ebd9ea1d117...

                                                                        B3LYP-
↪D3
c1c[cH:1][c:2](cc1)[C:3](=[O:4])O      id=1761822 hash_
↪index=028379f6fcec47dd5a41b7c4...
c1[cH:1][c:2](cnc1)[C:3](=[O:4])O      id=1761823 hash_
↪index=205e930c36ae28eb2dd5153d...
[cH:1]1cncc[c:2]1[C:3](=[O:4])O        id=1761824 hash_
↪index=381175f46c75a36d1bdec6c2...
[cH:1]1cc(nc[c:2]1[C:3](=[O:4])O)[O-]  id=1761825 hash_
↪index=f11d261f1ba21ac280706a36...
Cc1c[cH:1][c:2](cn1)[C:3](=[O:4])O     id=1761826 hash_
↪index=93cf4b08356a79aef7f70be6...
```

We can then begin to explore an individual TorsionDriveRecord by first pulling it from the DataFrame:

[12]:
```
td = ds.df.loc["[CH3:4][O:3][c:2]1[cH:1]cccc1", "B3LYP-D3"]
```

We can then request a variety of attributes off this TorsionDriveRecord:

[13]:
```
print("Torsion of interest                : {}".format(td.keywords.dihedrals))
print("Final optimization energy in hartree: {}".format(td.get_final_energies(180)))
```

```
Torsion of interest                : [(3, 5, 7, 6)]
Final optimization energy in hartree: -346.5319986074462
```

[14]:
```
td.get_final_molecules(90)
```

```
_ColormakerRegistry()
```

```
NGLWidget()
```

The `Molecule` object has a `measure` attribute so that we check the dihedral angle is in fact 180 degrees.

[15]:
```
"3-5-7-6 dihedral in degrees: {}".format(td.get_final_molecules(180).measure([3, 5, 7,
↪ 6]))
```

[15]:
```
'3-5-7-6 dihedral in degrees: 179.99999995886662'
```

You can also get a dictionary of all final molecules:

[16]:
```
td.get_final_molecules()
```

[16]:
```
{(-90,): Molecule(schema_name=qcschema_molecule, schema_version=2, validated=True,␣
↪symbols=['C' 'C' 'C' 'C' 'C' 'C' 'C' 'O' 'H' 'H' 'H' 'H' 'H' 'H' 'H' 'H'],␣
↪geometry=[[ 3.40718507 -2.21840696 -0.22958297]
  [ 4.97493788 -3.48334777  1.48334476]
  [ 1.7716505  -0.3298455   0.63640014]
  [ 4.90787611 -2.87176474  4.05190433]
  [ 1.69441113  0.29082009  3.20260133]
  [ 3.26680019 -0.98070424  4.91013927]
  [ 1.5337185  -1.70492748  8.94696498]
  [ 3.24498132 -0.32624151  7.42021119]
  [ 3.46717648 -2.69676568 -2.23750076]
  [ 6.26600112 -4.95093418  0.81459893]
  [ 0.55229994  0.67321911 -0.69586245]
  [ 6.13043938 -3.82147864  5.41754483]
  [ 0.44593371  1.77186336  3.91617642]
  [ 1.72079846 -0.98831032 10.88831694]
  [ 1.96264592 -3.7518771   8.93520479]
  [-0.44384326 -1.44836098  8.3163972 ]], name=C7H8O, identifiers={'molecule_hash':
↪'34e0a1476c201686f17fe576609083dc7d8bb187', 'molecular_formula': 'C7H8O', 'smiles':␣
↪None, 'inchi': None, 'inchikey': None, 'canonical_explicit_hydrogen_smiles': None,
↪'canonical_isomeric_explicit_hydrogen_mapped_smiles': None, 'canonical_isomeric_
↪explicit_hydrogen_smiles': None, 'canonical_isomeric_smiles': None, 'canonical_
↪smiles': None}, comment=None, molecular_charge=0.0, molecular_multiplicity=1,␣
↪masses=[12.         12.         12.         12.         12.         12.
  12.         15.99491462  1.00782503  1.00782503  1.00782503  1.00782503
   1.00782503  1.00782503  1.00782503  1.00782503], real=[ True  True  True  True␣
↪True  True  True  True  True  True  True  True
   True  True  True  True], atom_labels=['' '' '' '' '' '' '' '' '' '' '' '' '' '' ''
↪''], atomic_numbers=[6 6 6 6 6 6 6 8 1 1 1 1 1 1 1 1], mass_numbers=[12 12 12 12 12␣
↪12 12 16  1  1  1  1  1  1  1  1], connectivity=[(0, 2, 2.0), (0, 1, 1.0), (0, 8, 1.
↪0), (1, 3, 2.0), (1, 9, 1.0), (2, 4, 1.0), (2, 10, 1.0), (3, 5, 1.0), (3, 11, 1.0),␣
↪(4, 5, 2.0), (4, 12, 1.0), (5, 7, 1.0), (6, 7, 1.0), (6, 13, 1.0), (6, 14, 1.0), (6,
↪ 15, 1.0)], fragments=[array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,␣
↪13, 14, 15],
```

(continues on next page)

(continued from previous page)

```
       dtype=int32)], fragment_charges=[0.0], fragment_multiplicities=[1], fix_
↪com=True, fix_orientation=True, fix_symmetry=None, provenance={'creator':
↪'QCElemental', 'version': 'v0.11.0', 'routine': 'qcelemental.molparse.from_schema'},
↪ id=460608, extras=None),
 (180,): Molecule(schema_name=qcschema_molecule, schema_version=2, validated=True,␣
↪symbols=['C' 'C' 'C' 'C' 'C' 'C' 'C' 'O' 'H' 'H' 'H' 'H' 'H' 'H' 'H' 'H'],␣
↪geometry=[[ 3.61027365e+00 -1.98030605e+00 -3.18740370e-01]
  [ 5.52088757e+00 -3.08165509e+00  1.14994779e+00]
  [ 1.54199251e+00 -8.74617230e-01  8.85238100e-01]
  [ 5.36125372e+00 -3.07470080e+00  3.77580127e+00]
  [ 1.35317133e+00 -8.52024700e-01  3.52622139e+00]
  [ 3.27349763e+00 -1.95850076e+00  4.98449807e+00]
  [ 1.25868666e+00 -9.66527320e-01  8.90037066e+00]
  [ 3.28505965e+00 -2.04442029e+00  7.55013272e+00]
  [ 3.73979950e+00 -1.98801841e+00 -2.37887261e+00]
  [ 7.15429448e+00 -3.95597241e+00  2.35669430e-01]
  [ 3.66345200e-02 -7.97205000e-03 -2.33700310e-01]
  [ 6.82865599e+00 -3.92206287e+00  4.95325767e+00]
  [-2.84620420e-01  2.56175000e-02  4.41539709e+00]
  [ 1.66683083e+00 -1.25550003e+00  1.09153783e+01]
  [-5.62174440e-01 -1.88506626e+00  8.44355628e+00]
  [ 1.07941944e+00  1.08370374e+00  8.53662357e+00]], name=C7H8O, identifiers={
↪'molecule_hash': '454b9110a607f34201c736a791f2a438ef7eab3b', 'molecular_formula':
↪'C7H8O', 'smiles': None, 'inchi': None, 'inchikey': None, 'canonical_explicit_
↪hydrogen_smiles': None, 'canonical_isomeric_explicit_hydrogen_mapped_smiles': None,
↪'canonical_isomeric_explicit_hydrogen_smiles': None, 'canonical_isomeric_smiles':␣
↪None, 'canonical_smiles': None}, comment=None, molecular_charge=0.0, molecular_
↪multiplicity=1, masses=[12.         12.         12.         12.         12.        ␣
↪12.
  12.         15.99491462  1.00782503  1.00782503  1.00782503  1.00782503
   1.00782503  1.00782503  1.00782503  1.00782503], real=[ True  True  True  True ␣
↪True  True  True  True  True  True  True  True
   True  True  True  True], atom_labels=['' '' '' '' '' '' '' '' '' '' '' '' '' '' ''
↪''], atomic_numbers=[6 6 6 6 6 6 8 1 1 1 1 1 1 1 1], mass_numbers=[12 12 12 12 12␣
↪12 12 16  1  1  1  1  1  1  1], connectivity=[(0, 2, 2.0), (0, 1, 1.0), (0, 8, 1.
↪0), (1, 3, 2.0), (1, 9, 1.0), (2, 4, 1.0), (2, 10, 1.0), (3, 5, 1.0), (3, 11, 1.0),␣
↪(4, 5, 2.0), (4, 12, 1.0), (5, 7, 1.0), (6, 7, 1.0), (6, 13, 1.0), (6, 14, 1.0), (6,
↪ 15, 1.0)], fragments=[array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,␣
↪13, 14, 15],
       dtype=int32)], fragment_charges=[0.0], fragment_multiplicities=[1], fix_
↪com=True, fix_orientation=True, fix_symmetry=None, provenance={'creator':
↪'QCElemental', 'version': 'v0.11.0', 'routine': 'qcelemental.molparse.from_schema'},
↪ id=458506, extras=None),
 (135,): Molecule(schema_name=qcschema_molecule, schema_version=2, validated=True,␣
↪symbols=['C' 'C' 'C' 'C' 'C' 'C' 'C' 'O' 'H' 'H' 'H' 'H' 'H' 'H' 'H' 'H'],␣
↪geometry=[[ 3.71074266 -1.78005411 -0.28373136]
  [ 5.60347076 -2.81108719  1.25014161]
  [ 1.41712809 -1.04780261  0.80318487]
  [ 5.2116725  -3.10100381  3.84211902]
  [ 0.99788049 -1.34633653  3.39535384]
  [ 2.89417641 -2.3994055   4.92058236]
  [ 1.52117769 -0.80706242  8.90148939]
  [ 2.60669985 -2.77544922  7.4566502 ]
  [ 4.02516342 -1.54291228 -2.31039311]
  [ 7.41160882 -3.37483763  0.42488076]
  [-0.07297217 -0.23671814 -0.37601593]
  [ 6.66649345 -3.88431511  5.0779687 ]
```

(continues on next page)

(continued from previous page)

```
  [-0.8050157  -0.78350122  4.22440334]
  [ 1.86678347 -1.26394897 10.89874498]
  [-0.5432106  -0.64765122  8.61880289]
  [ 2.39551084  1.03754822  8.45893749]], name=C7H8O, identifiers={'molecule_hash':
↪'a181fc4f7143c3e59d6db6a67f68ec2d6dca7d01', 'molecular_formula': 'C7H8O', 'smiles':␣
↪None, 'inchi': None, 'inchikey': None, 'canonical_explicit_hydrogen_smiles': None,
↪'canonical_isomeric_explicit_hydrogen_mapped_smiles': None, 'canonical_isomeric_
↪explicit_hydrogen_smiles': None, 'canonical_isomeric_smiles': None, 'canonical_
↪smiles': None}, comment=None, molecular_charge=0.0, molecular_multiplicity=1,␣
↪masses=[12.         12.         12.         12.         12.         12.
  12.         15.99491462  1.00782503  1.00782503  1.00782503  1.00782503
   1.00782503  1.00782503  1.00782503  1.00782503], real=[ True  True  True  True␣
↪True  True  True  True  True  True  True  True
   True  True  True  True], atom_labels=['' '' '' '' '' '' '' '' '' '' '' '' '' ''
↪''], atomic_numbers=[6 6 6 6 6 6 6 8 1 1 1 1 1 1 1 1], mass_numbers=[12 12 12 12 12␣
↪12 12 16  1  1  1  1  1  1  1  1], connectivity=[(0, 2, 2.0), (0, 1, 1.0), (0, 8, 1.
↪0), (1, 3, 2.0), (1, 9, 1.0), (2, 4, 1.0), (2, 10, 1.0), (3, 5, 1.0), (3, 11, 1.0),␣
↪(4, 5, 2.0), (4, 12, 1.0), (5, 7, 1.0), (6, 7, 1.0), (6, 13, 1.0), (6, 14, 1.0), (6,
↪ 15, 1.0)], fragments=[array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,␣
↪13, 14, 15],
      dtype=int32)], fragment_charges=[0.0], fragment_multiplicities=[1], fix_
↪com=True, fix_orientation=True, fix_symmetry=None, provenance={'creator':
↪'QCElemental', 'version': 'v0.11.0', 'routine': 'qcelemental.molparse.from_schema'},
↪ id=458654, extras=None),
 (165,): Molecule(schema_name=qcschema_molecule, schema_version=2, validated=True,␣
↪symbols=['C' 'C' 'C' 'C' 'C' 'C' 'C' 'O' 'H' 'H' 'H' 'H' 'H' 'H' 'H' 'H'],␣
↪geometry=[[ 3.65418749 -1.90093631 -0.31404815]
  [ 5.56021506 -2.99808714  1.16271943]
  [ 1.51514808 -0.91626636  0.87283114]
  [ 5.32801018 -3.10570501  3.78188   ]
  [ 1.25147303 -1.01602504  3.50484657]
  [ 3.16118484 -2.13031622  4.97086769]
  [ 1.31253922 -0.89145213  8.89765096]
  [ 3.08963919 -2.33077078  7.53054258]
  [ 3.84376413 -1.81267554 -2.36768675]
  [ 7.25298638 -3.76720236  0.26197523]
  [ 0.01480673 -0.0501931  -0.25324579]
  [ 6.79530961 -3.94250746  4.9669548 ]
  [-0.4400024  -0.23597809  4.38548001]
  [ 1.7612165  -1.14516869 10.90913746]
  [-0.64769987 -1.53804325  8.5690084 ]
  [ 1.43117256  1.13977679  8.42246929]], name=C7H8O, identifiers={'molecule_hash':
↪'2c67957d95a44c137d7746d2ecd1c4f28c655849', 'molecular_formula': 'C7H8O', 'smiles':␣
↪None, 'inchi': None, 'inchikey': None, 'canonical_explicit_hydrogen_smiles': None,
↪'canonical_isomeric_explicit_hydrogen_mapped_smiles': None, 'canonical_isomeric_
↪explicit_hydrogen_smiles': None, 'canonical_isomeric_smiles': None, 'canonical_
↪smiles': None}, comment=None, molecular_charge=0.0, molecular_multiplicity=1,␣
↪masses=[12.         12.         12.         12.         12.         12.
  12.         15.99491462  1.00782503  1.00782503  1.00782503  1.00782503
   1.00782503  1.00782503  1.00782503  1.00782503], real=[ True  True  True  True␣
↪True  True  True  True  True  True  True  True
   True  True  True  True], atom_labels=['' '' '' '' '' '' '' '' '' '' '' '' '' ''
↪''], atomic_numbers=[6 6 6 6 6 6 6 8 1 1 1 1 1 1 1 1], mass_numbers=[12 12 12 12 12␣
↪12 12 16  1  1  1  1  1  1  1  1], connectivity=[(0, 2, 2.0), (0, 1, 1.0), (0, 8, 1.
↪0), (1, 3, 2.0), (1, 9, 1.0), (2, 4, 1.0), (2, 10, 1.0), (3, 5, 1.0), (3, 11, 1.0),␣
↪(4, 5, 2.0), (4, 12, 1.0), (5, 7, 1.0), (6, 7, 1.0), (6, 13, 1.0), (6, 14, 1.0), (6,
↪ 15, 1.0)], fragments=[array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,␣
↪13, 14, 15],
```

(continues on next page)

(continued from previous page)

```
      dtype=int32)], fragment_charges=[0.0], fragment_multiplicities=[1], fix_
↪com=True, fix_orientation=True, fix_symmetry=None, provenance={'creator':
↪'QCElemental', 'version': 'v0.11.0', 'routine': 'qcelemental.molparse.from_schema'},
↪ id=458889, extras=None),
 (-165,): Molecule(schema_name=qcschema_molecule, schema_version=2, validated=True,␣
↪symbols=['C' 'C' 'C' 'C' 'C' 'C' 'C' 'O' 'H' 'H' 'H' 'H' 'H' 'H' 'H' 'H'],␣
↪geometry=[[ 3.56477391 -2.06268365 -0.31933034]
  [ 5.46806545 -3.1646068   1.15740257]
  [ 1.56234862 -0.83094627  0.87518466]
  [ 5.36641548 -3.0359742   3.78393169]
  [ 1.43705914 -0.68051491  3.51500416]
  [ 3.35702811 -1.77598282  4.98183602]
  [ 1.22152767 -1.05569795  8.89252899]
  [ 3.42101743 -1.73127425  7.549176  ]
  [ 3.64518515 -2.17194564 -2.37912539]
  [ 7.04316019 -4.14656642  0.25010762]
  [ 0.0611202   0.03347081 -0.2509852 ]
  [ 6.82376306 -3.89074244  4.96846074]
  [-0.14605238  0.29553537  4.40164763]
  [ 1.62176362 -1.39719575 10.90124846]
  [-0.42219723 -2.21053475  8.31784924]
  [ 0.73508868  0.96218359  8.64722648]], name=C7H8O, identifiers={'molecule_hash':
↪'cf816486557e9146f6a71606f1c8ec975073de08', 'molecular_formula': 'C7H8O', 'smiles':␣
↪None, 'inchi': None, 'inchikey': None, 'canonical_explicit_hydrogen_smiles': None,
↪'canonical_isomeric_explicit_hydrogen_mapped_smiles': None, 'canonical_isomeric_
↪explicit_hydrogen_smiles': None, 'canonical_isomeric_smiles': None, 'canonical_
↪smiles': None}, comment=None, molecular_charge=0.0, molecular_multiplicity=1,␣
↪masses=[12.         12.         12.         12.         12.         12.
  12.         15.99491462  1.00782503  1.00782503  1.00782503  1.00782503
   1.00782503  1.00782503  1.00782503  1.00782503], real=[ True  True  True  True ␣
↪True  True  True  True  True  True  True  True
   True  True  True  True], atom_labels=['' '' '' '' '' '' '' '' '' '' '' '' '' '' ''
↪''], atomic_numbers=[6 6 6 6 6 6 8 1 1 1 1 1 1 1 1], mass_numbers=[12 12 12 12 12␣
↪12 12 16  1  1  1  1  1  1  1], connectivity=[(0, 2, 2.0), (0, 1, 1.0), (0, 8, 1.
↪0), (1, 3, 2.0), (1, 9, 1.0), (2, 4, 1.0), (2, 10, 1.0), (3, 5, 1.0), (3, 11, 1.0),␣
↪(4, 5, 2.0), (4, 12, 1.0), (5, 7, 1.0), (6, 7, 1.0), (6, 13, 1.0), (6, 14, 1.0), (6,
↪ 15, 1.0)], fragments=[array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,␣
↪13, 14, 15],
      dtype=int32)], fragment_charges=[0.0], fragment_multiplicities=[1], fix_
↪com=True, fix_orientation=True, fix_symmetry=None, provenance={'creator':
↪'QCElemental', 'version': 'v0.11.0', 'routine': 'qcelemental.molparse.from_schema'},
↪ id=458861, extras=None),
 (-135,): Molecule(schema_name=qcschema_molecule, schema_version=2, validated=True,␣
↪symbols=['C' 'C' 'C' 'C' 'C' 'C' 'C' 'O' 'H' 'H' 'H' 'H' 'H' 'H' 'H' 'H'],␣
↪geometry=[[ 3.491733   -2.17583439 -0.29637177]
  [ 5.33054359 -3.30455675  1.23430153]
  [ 1.62351776 -0.67457419  0.81459627]
  [ 5.2991748  -2.94298239  3.84661412]
  [ 1.58487459 -0.28523287  3.42824227]
  [ 3.44443861 -1.4048686   4.9512827 ]
  [ 1.26160538 -1.27722055  8.88669334]
  [ 3.54307214 -1.08323406  7.50911694]
  [ 3.51165624 -2.47088682 -2.33965283]
  [ 6.79103733 -4.49672339  0.38935433]
  [ 0.17567664  0.21298338 -0.36216721]
  [ 6.70326276 -3.8180982   5.07954316]
  [ 0.12784967  0.9028205   4.27685668]
```

(continues on next page)

(continued from previous page)

```
  [ 1.77954751 -1.42167327 10.89369992]
  [ 0.17735948 -2.97500059  8.3345377 ]
  [ 0.03575585  0.39865631  8.65060519]], name=C7H8O, identifiers={'molecule_hash':
↪'8175d7667946055f7f16e5bb64997ab5fb0dfb17', 'molecular_formula': 'C7H8O', 'smiles':␣
↪None, 'inchi': None, 'inchikey': None, 'canonical_explicit_hydrogen_smiles': None,
↪'canonical_isomeric_explicit_hydrogen_mapped_smiles': None, 'canonical_isomeric_
↪explicit_hydrogen_smiles': None, 'canonical_isomeric_smiles': None, 'canonical_
↪smiles': None}, comment=None, molecular_charge=0.0, molecular_multiplicity=1,␣
↪masses=[12.         12.         12.         12.         12.         12.
  12.         15.99491462  1.00782503  1.00782503  1.00782503  1.00782503
   1.00782503  1.00782503  1.00782503  1.00782503], real=[ True  True  True  True␣
↪True  True  True  True  True  True  True  True
   True  True  True  True], atom_labels=['' '' '' '' '' '' '' '' '' '' '' '' '' ''
↪''], atomic_numbers=[6 6 6 6 6 6 6 8 1 1 1 1 1 1 1 1], mass_numbers=[12 12 12 12 12␣
↪12 12 16  1  1  1  1  1  1  1  1], connectivity=[(0, 2, 2.0), (0, 1, 1.0), (0, 8, 1.
↪0), (1, 3, 2.0), (1, 9, 1.0), (2, 4, 1.0), (2, 10, 1.0), (3, 5, 1.0), (3, 11, 1.0),␣
↪(4, 5, 2.0), (4, 12, 1.0), (5, 7, 1.0), (6, 7, 1.0), (6, 13, 1.0), (6, 14, 1.0), (6,
↪ 15, 1.0)], fragments=[array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,␣
↪13, 14, 15],
      dtype=int32)], fragment_charges=[0.0], fragment_multiplicities=[1], fix_
↪com=True, fix_orientation=True, fix_symmetry=None, provenance={'creator':
↪'QCElemental', 'version': 'v0.11.0', 'routine': 'qcelemental.molparse.from_schema'},
↪ id=458870, extras=None),
 (150,): Molecule(schema_name=qcschema_molecule, schema_version=2, validated=True,␣
↪symbols=['C' 'C' 'C' 'C' 'C' 'C' 'C' 'O' 'H' 'H' 'H' 'H' 'H' 'H' 'H' 'H'],␣
↪geometry=[[ 3.68950957 -1.83192958 -0.30263301]
  [ 5.58788714 -2.91449311  1.19197383]
  [ 1.47651723 -0.96952926  0.84767887]
  [ 5.27814438 -3.12904808  3.79899959]
  [ 1.13585096 -1.18495302  3.46192866]
  [ 3.03480466 -2.29331251  4.94542969]
  [ 1.40141672 -0.85296282  8.89427796]
  [ 2.86097753 -2.59344088  7.49456838]
  [ 3.94144492 -1.65547154 -2.34397562]
  [ 7.33843648 -3.57937644  0.31914596]
  [-0.01672084 -0.11341923 -0.29527781]
  [ 6.74037439 -3.94979797  5.00135427]
  [-0.60798886 -0.50658543  4.32710875]
  [ 1.88525221 -1.13551231 10.89457591]
  [-0.65152006 -1.17180494  8.66612967]
  [ 1.831008    1.11863797  8.35470128]], name=C7H8O, identifiers={'molecule_hash':
↪'2afce2897a153d1f113c03a8839f12cac3ac0719', 'molecular_formula': 'C7H8O', 'smiles':␣
↪None, 'inchi': None, 'inchikey': None, 'canonical_explicit_hydrogen_smiles': None,
↪'canonical_isomeric_explicit_hydrogen_mapped_smiles': None, 'canonical_isomeric_
↪explicit_hydrogen_smiles': None, 'canonical_isomeric_smiles': None, 'canonical_
↪smiles': None}, comment=None, molecular_charge=0.0, molecular_multiplicity=1,␣
↪masses=[12.         12.         12.         12.         12.         12.
  12.         15.99491462  1.00782503  1.00782503  1.00782503  1.00782503
   1.00782503  1.00782503  1.00782503  1.00782503], real=[ True  True  True  True␣
↪True  True  True  True  True  True  True  True
   True  True  True  True], atom_labels=['' '' '' '' '' '' '' '' '' '' '' '' '' ''
↪''], atomic_numbers=[6 6 6 6 6 6 6 8 1 1 1 1 1 1 1 1], mass_numbers=[12 12 12 12 12␣
↪12 12 16  1  1  1  1  1  1  1  1], connectivity=[(0, 2, 2.0), (0, 1, 1.0), (0, 8, 1.
↪0), (1, 3, 2.0), (1, 9, 1.0), (2, 4, 1.0), (2, 10, 1.0), (3, 5, 1.0), (3, 11, 1.0),␣
↪(4, 5, 2.0), (4, 12, 1.0), (5, 7, 1.0), (6, 7, 1.0), (6, 13, 1.0), (6, 14, 1.0), (6,
↪ 15, 1.0)], fragments=[array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,␣
↪13, 14, 15],
```

(continues on next page)

(continued from previous page)

```
       dtype=int32)], fragment_charges=[0.0], fragment_multiplicities=[1], fix_
↪com=True, fix_orientation=True, fix_symmetry=None, provenance={'creator':
↪'QCElemental', 'version': 'v0.11.0', 'routine': 'qcelemental.molparse.from_schema'},
↪ id=459105, extras=None),
 (-150,): Molecule(schema_name=qcschema_molecule, schema_version=2, validated=True,␣
↪symbols=['C' 'C' 'C' 'C' 'C' 'C' 'C' 'O' 'H' 'H' 'H' 'H' 'H' 'H' 'H' 'H'],␣
↪geometry=[[ 3.52476149e+00 -2.12952483e+00 -3.12109620e-01]
  [ 5.41127251e+00 -3.23366809e+00  1.18174239e+00]
  [ 1.58752395e+00 -7.68590320e-01  8.53755290e-01]
  [ 5.35958136e+00 -2.98227988e+00  3.80332133e+00]
  [ 1.51990512e+00 -4.90890560e-01  3.48340437e+00]
  [ 3.42886502e+00 -1.58133228e+00  4.96735537e+00]
  [ 1.23691058e+00 -1.15109572e+00  8.88478749e+00]
  [ 3.52325946e+00 -1.39670006e+00  7.53161235e+00]
  [ 3.56322781e+00 -2.33889999e+00 -2.36548922e+00]
  [ 6.92801398e+00 -4.32112774e+00  2.95604100e-01]
  [ 9.88758100e-02  9.59362800e-02 -2.88856250e-01]
  [ 6.80025429e+00 -3.84213567e+00  5.00437371e+00]
  [-4.47220000e-03  5.84117900e-01  4.36108482e+00]
  [ 1.68066939e+00 -1.50621313e+00  1.08828092e+01]
  [-1.89240600e-01 -2.53460509e+00  8.24066372e+00]
  [ 4.20982040e-01  7.66566550e-01  8.72640143e+00]], name=C7H8O, identifiers={
↪'molecule_hash': 'f3b67ee06b6b817183b9c0f41a9c6e7ac631c146', 'molecular_formula':
↪'C7H8O', 'smiles': None, 'inchi': None, 'inchikey': None, 'canonical_explicit_
↪hydrogen_smiles': None, 'canonical_isomeric_explicit_hydrogen_mapped_smiles': None,
↪'canonical_isomeric_explicit_hydrogen_smiles': None, 'canonical_isomeric_smiles':␣
↪None, 'canonical_smiles': None}, comment=None, molecular_charge=0.0, molecular_
↪multiplicity=1, masses=[12.         12.         12.         12.         12.        ␣
↪12.
  12.         15.99491462  1.00782503  1.00782503  1.00782503  1.00782503
   1.00782503  1.00782503  1.00782503  1.00782503], real=[ True  True  True  True ␣
↪True  True  True  True  True  True  True  True
   True  True  True  True], atom_labels=['' '' '' '' '' '' '' '' '' '' '' '' '' '' ''
↪''], atomic_numbers=[6 6 6 6 6 6 8 1 1 1 1 1 1 1 1], mass_numbers=[12 12 12 12 12␣
↪12 12 16  1  1  1  1  1  1  1], connectivity=[(0, 2, 2.0), (0, 1, 1.0), (0, 8, 1.
↪0), (1, 3, 2.0), (1, 9, 1.0), (2, 4, 1.0), (2, 10, 1.0), (3, 5, 1.0), (3, 11, 1.0),␣
↪(4, 5, 2.0), (4, 12, 1.0), (5, 7, 1.0), (6, 7, 1.0), (6, 13, 1.0), (6, 14, 1.0), (6,
↪ 15, 1.0)], fragments=[array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,␣
↪13, 14, 15],
       dtype=int32)], fragment_charges=[0.0], fragment_multiplicities=[1], fix_
↪com=True, fix_orientation=True, fix_symmetry=None, provenance={'creator':
↪'QCElemental', 'version': 'v0.11.0', 'routine': 'qcelemental.molparse.from_schema'},
↪ id=459114, extras=None),
 (105,): Molecule(schema_name=qcschema_molecule, schema_version=2, validated=True,␣
↪symbols=['C' 'C' 'C' 'C' 'C' 'C' 'C' 'O' 'H' 'H' 'H' 'H' 'H' 'H' 'H' 'H'],␣
↪geometry=[[ 3.69545538 -1.70634993 -0.23938507]
  [ 5.58436954 -2.54998336  1.40623546]
  [ 1.25836456 -1.25614547  0.68418211]
  [ 5.04976205 -2.9314534   3.96508707]
  [ 0.70724096 -1.64444833  3.23706462]
  [ 2.60724282 -2.48633636  4.87951087]
  [ 1.86172856 -0.77318807  8.93121643]
  [ 2.08977467 -2.94052016  7.37904178]
  [ 4.11789466 -1.4081985  -2.23853071]
  [ 7.4899185  -2.91207376  0.69511821]
  [-0.22967914 -0.61125939 -0.59562359]
  [ 6.49168507 -3.5949349   5.28423518]
```

(continues on next page)

(continued from previous page)

```
  [-1.19362524 -1.33018691  3.97905554]
  [ 1.41574336 -1.43372593 10.85002017]
  [ 0.33246541  0.49775271  8.28516983]
  [ 3.6428032   0.32145572  8.98528528]], name=C7H8O, identifiers={'molecule_hash':
↪'a72a6127282ba5336fb37ae76ef40ea1e446a21c', 'molecular_formula': 'C7H8O', 'smiles':␣
↪None, 'inchi': None, 'inchikey': None, 'canonical_explicit_hydrogen_smiles': None,
↪'canonical_isomeric_explicit_hydrogen_mapped_smiles': None, 'canonical_isomeric_
↪explicit_hydrogen_smiles': None, 'canonical_isomeric_smiles': None, 'canonical_
↪smiles': None}, comment=None, molecular_charge=0.0, molecular_multiplicity=1,␣
↪masses=[12.         12.         12.         12.         12.         12.
 12.         15.99491462  1.00782503  1.00782503  1.00782503  1.00782503
  1.00782503  1.00782503  1.00782503  1.00782503], real=[ True  True  True  True ␣
↪True  True  True  True  True  True  True  True
  True  True  True  True], atom_labels=['' '' '' '' '' '' '' '' '' '' '' '' '' '' ''
↪''], atomic_numbers=[6 6 6 6 6 6 6 8 1 1 1 1 1 1 1 1], mass_numbers=[12 12 12 12 12␣
↪12 12 16  1  1  1  1  1  1  1  1], connectivity=[(0, 2, 2.0), (0, 1, 1.0), (0, 8, 1.
↪0), (1, 3, 2.0), (1, 9, 1.0), (2, 4, 1.0), (2, 10, 1.0), (3, 5, 1.0), (3, 11, 1.0),␣
↪(4, 5, 2.0), (4, 12, 1.0), (5, 7, 1.0), (6, 7, 1.0), (6, 13, 1.0), (6, 14, 1.0), (6,
↪ 15, 1.0)], fragments=[array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,␣
↪13, 14, 15],
      dtype=int32)], fragment_charges=[0.0], fragment_multiplicities=[1], fix_
↪com=True, fix_orientation=True, fix_symmetry=None, provenance={'creator':
↪'QCElemental', 'version': 'v0.11.0', 'routine': 'qcelemental.molparse.from_schema'},
↪ id=459232, extras=None),
 (-105,): Molecule(schema_name=qcschema_molecule, schema_version=2, validated=True,␣
↪symbols=['C' 'C' 'C' 'C' 'C' 'C' 'C' 'O' 'H' 'H' 'H' 'H' 'H' 'H' 'H' 'H'],␣
↪geometry=[[ 3.42039406 -2.20354666 -0.25271686]
  [ 5.09605067 -3.43333395  1.38060355]
  [ 1.7185079  -0.4240417   0.71262135]
  [ 5.06721158 -2.90095096  3.96800149]
  [ 1.68677973  0.12628289  3.29486938]
  [ 3.36694948 -1.11363064  4.92455346]
  [ 1.41395078 -1.58524678  8.90830178]
  [ 3.41029401 -0.55427638  7.4557179 ]
  [ 3.44543837 -2.62362267 -2.27431064]
  [ 6.43496313 -4.81991356  0.63721685]
  [ 0.4146949   0.55441554 -0.55653123]
  [ 6.36113935 -3.83233623  5.27864259]
  [ 0.38996126  1.53278678  4.07115436]
  [ 1.68407697 -0.94737393 10.86717504]
  [ 1.43485456 -3.67596418  8.86458193]
  [-0.45838733 -0.93599831  8.24185377]], name=C7H8O, identifiers={'molecule_hash':
↪'dda2cb6b2f0782068fc6a0a395c8631962bc24e0', 'molecular_formula': 'C7H8O', 'smiles':␣
↪None, 'inchi': None, 'inchikey': None, 'canonical_explicit_hydrogen_smiles': None,
↪'canonical_isomeric_explicit_hydrogen_mapped_smiles': None, 'canonical_isomeric_
↪explicit_hydrogen_smiles': None, 'canonical_isomeric_smiles': None, 'canonical_
↪smiles': None}, comment=None, molecular_charge=0.0, molecular_multiplicity=1,␣
↪masses=[12.         12.         12.         12.         12.         12.
 12.         15.99491462  1.00782503  1.00782503  1.00782503  1.00782503
  1.00782503  1.00782503  1.00782503  1.00782503], real=[ True  True  True  True ␣
↪True  True  True  True  True  True  True  True
  True  True  True  True], atom_labels=['' '' '' '' '' '' '' '' '' '' '' '' '' '' ''
↪''], atomic_numbers=[6 6 6 6 6 6 6 8 1 1 1 1 1 1 1 1], mass_numbers=[12 12 12 12 12␣
↪12 12 16  1  1  1  1  1  1  1  1], connectivity=[(0, 2, 2.0), (0, 1, 1.0), (0, 8, 1.
↪0), (1, 3, 2.0), (1, 9, 1.0), (2, 4, 1.0), (2, 10, 1.0), (3, 5, 1.0), (3, 11, 1.0),␣
↪(4, 5, 2.0), (4, 12, 1.0), (5, 7, 1.0), (6, 7, 1.0), (6, 13, 1.0), (6, 14, 1.0), (6,
↪ 15, 1.0)], fragments=[array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,␣
↪13, 14, 15],
```

(continues on next page)

(continued from previous page)

```
      dtype=int32)], fragment_charges=[0.0], fragment_multiplicities=[1], fix_
↪com=True, fix_orientation=True, fix_symmetry=None, provenance={'creator':
↪'QCElemental', 'version': 'v0.11.0', 'routine': 'qcelemental.molparse.from_schema'},
↪ id=459394, extras=None),
 (120,): Molecule(schema_name=qcschema_molecule, schema_version=2, validated=True,␣
↪symbols=['C' 'C' 'C' 'C' 'C' 'C' 'C' 'O' 'H' 'H' 'H' 'H' 'H' 'H' 'H' 'H'],␣
↪geometry=[[ 3.68154157 -1.74990256 -0.24461527]
  [ 5.58614604 -2.68218607  1.33400306]
  [ 1.31356562 -1.16345525  0.77506458]
  [ 5.13669762 -3.01249357  3.91491872]
  [ 0.84241752 -1.50501605  3.35088957]
  [ 2.75895471 -2.43763707  4.92670024]
  [ 1.70964993 -0.7538      8.9503509 ]
  [ 2.3670574  -2.86664598  7.44878031]
  [ 4.03838228 -1.48541515 -2.26104614]
  [ 7.44229936 -3.14542008  0.55413637]
  [-0.18938282 -0.44590515 -0.44750504]
  [ 6.59476681 -3.73210344  5.18542651]
  [-1.01536662 -1.08131531  4.14394906]
  [ 1.55620491 -1.42607057 10.91017616]
  [-0.12115984  0.08733461  8.39660453]
  [ 3.17380361  0.7350621   8.85419846]], name=C7H8O, identifiers={'molecule_hash':
↪'2526a43946fc423be4f370a91480eb7d31cbd392', 'molecular_formula': 'C7H8O', 'smiles':␣
↪None, 'inchi': None, 'inchikey': None, 'canonical_explicit_hydrogen_smiles': None,
↪'canonical_isomeric_explicit_hydrogen_mapped_smiles': None, 'canonical_isomeric_
↪explicit_hydrogen_smiles': None, 'canonical_isomeric_smiles': None, 'canonical_
↪smiles': None}, comment=None, molecular_charge=0.0, molecular_multiplicity=1,␣
↪masses=[12.         12.         12.         12.         12.         12.
  12.         15.99491462  1.00782503  1.00782503  1.00782503  1.00782503
   1.00782503  1.00782503  1.00782503  1.00782503], real=[ True  True  True  True ␣
↪True  True  True  True  True  True  True  True
   True  True  True  True], atom_labels=['' '' '' '' '' '' '' '' '' '' '' '' '' '' ''
↪''], atomic_numbers=[6 6 6 6 6 6 8 1 1 1 1 1 1 1 1], mass_numbers=[12 12 12 12 12␣
↪12 12 16  1  1  1  1  1  1  1], connectivity=[(0, 2, 2.0), (0, 1, 1.0), (0, 8, 1.
↪0), (1, 3, 2.0), (1, 9, 1.0), (2, 4, 1.0), (2, 10, 1.0), (3, 5, 1.0), (3, 11, 1.0),␣
↪(4, 5, 2.0), (4, 12, 1.0), (5, 7, 1.0), (6, 7, 1.0), (6, 13, 1.0), (6, 14, 1.0), (6,
↪ 15, 1.0)], fragments=[array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,␣
↪13, 14, 15],
      dtype=int32)], fragment_charges=[0.0], fragment_multiplicities=[1], fix_
↪com=True, fix_orientation=True, fix_symmetry=None, provenance={'creator':
↪'QCElemental', 'version': 'v0.11.0', 'routine': 'qcelemental.molparse.from_schema'},
↪ id=459538, extras=None),
 (-120,): Molecule(schema_name=qcschema_molecule, schema_version=2, validated=True,␣
↪symbols=['C' 'C' 'C' 'C' 'C' 'C' 'C' 'O' 'H' 'H' 'H' 'H' 'H' 'H' 'H' 'H'],␣
↪geometry=[[ 3.45091577 -2.17571528 -0.26546162]
  [ 5.21045061 -3.36877965  1.30595734]
  [ 1.66970257 -0.52870746  0.78601954]
  [ 5.18350251 -2.93404589  3.9102702 ]
  [ 1.64081975 -0.06925252  3.38632468]
  [ 3.40713121 -1.27216144  4.9545363 ]
  [ 1.31775564 -1.46385389  8.91994316]
  [ 3.49582237 -0.83122039  7.50330384]
  [ 3.4695849  -2.52358983 -2.30063283]
  [ 6.60916241 -4.65886601  0.50136998]
  [ 0.2959696   0.42130221 -0.43001301]
  [ 6.53386114 -3.84721516  5.17544277]
  [ 0.2743827   1.24257482  4.20596625]
```

(continues on next page)

(continued from previous page)

```
  [ 1.75350877 -1.06837263 10.91245988]
  [ 0.83775254 -3.48824763  8.71708563]
  [-0.35272618 -0.33384423  8.37353198]], name=C7H8O, identifiers={'molecule_hash':
↪'135dd920d0dbd5dcbbb668246becc22ea8f19dba', 'molecular_formula': 'C7H8O', 'smiles':␣
↪None, 'inchi': None, 'inchikey': None, 'canonical_explicit_hydrogen_smiles': None,
↪'canonical_isomeric_explicit_hydrogen_mapped_smiles': None, 'canonical_isomeric_
↪explicit_hydrogen_smiles': None, 'canonical_isomeric_smiles': None, 'canonical_
↪smiles': None}, comment=None, molecular_charge=0.0, molecular_multiplicity=1,␣
↪masses=[12.         12.         12.         12.         12.         12.
  12.         15.99491462  1.00782503  1.00782503  1.00782503  1.00782503
   1.00782503  1.00782503  1.00782503  1.00782503], real=[ True  True  True  True␣
↪True  True  True  True  True  True  True  True
   True  True  True  True], atom_labels=['' '' '' '' '' '' '' '' '' '' '' '' '' ''
↪''], atomic_numbers=[6 6 6 6 6 6 6 8 1 1 1 1 1 1 1 1], mass_numbers=[12 12 12 12 12␣
↪12 12 16  1  1  1  1  1  1  1  1], connectivity=[(0, 2, 2.0), (0, 1, 1.0), (0, 8, 1.
↪0), (1, 3, 2.0), (1, 9, 1.0), (2, 4, 1.0), (2, 10, 1.0), (3, 5, 1.0), (3, 11, 1.0),␣
↪(4, 5, 2.0), (4, 12, 1.0), (5, 7, 1.0), (6, 7, 1.0), (6, 13, 1.0), (6, 14, 1.0), (6,
↪ 15, 1.0)], fragments=[array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,␣
↪13, 14, 15],
      dtype=int32)], fragment_charges=[0.0], fragment_multiplicities=[1], fix_
↪com=True, fix_orientation=True, fix_symmetry=None, provenance={'creator':
↪'QCElemental', 'version': 'v0.11.0', 'routine': 'qcelemental.molparse.from_schema'},
↪ id=459600, extras=None),
 (90,): Molecule(schema_name=qcschema_molecule, schema_version=2, validated=True,␣
↪symbols=['C' 'C' 'C' 'C' 'C' 'C' 'C' 'O' 'H' 'H' 'H' 'H' 'H' 'H' 'H' 'H'],␣
↪geometry=[[ 3.70026604 -1.68781217 -0.21473939]
  [ 5.55971395 -2.42504226  1.5147141 ]
  [ 1.20886204 -1.3472961   0.60288173]
  [ 4.93860674 -2.81548453  4.05199411]
  [ 0.57469777 -1.73382934  3.13764424]
  [ 2.44233629 -2.47122049  4.8618902 ]
  [ 2.0262802  -0.81298112  8.97297625]
  [ 1.81054902 -2.91990197  7.33754827]
  [ 4.19006652 -1.388848   -2.19839178]
  [ 7.50634899 -2.70687967  0.88300624]
  [-0.25363715 -0.78411793 -0.74315142]
  [ 6.35677177 -3.41129129  5.42873571]
  [-1.36184649 -1.49732344  3.81235793]
  [ 1.46745331 -1.44403575 10.87215158]
  [ 0.77398282  0.75470241  8.38281463]
  [ 3.98719974 -0.08961615  9.04816579]], name=C7H8O, identifiers={'molecule_hash':
↪'a6601a8f8c024493170efc180fb7ce6a3f6ea552', 'molecular_formula': 'C7H8O', 'smiles':␣
↪None, 'inchi': None, 'inchikey': None, 'canonical_explicit_hydrogen_smiles': None,
↪'canonical_isomeric_explicit_hydrogen_mapped_smiles': None, 'canonical_isomeric_
↪explicit_hydrogen_smiles': None, 'canonical_isomeric_smiles': None, 'canonical_
↪smiles': None}, comment=None, molecular_charge=0.0, molecular_multiplicity=1,␣
↪masses=[12.         12.         12.         12.         12.         12.
  12.         15.99491462  1.00782503  1.00782503  1.00782503  1.00782503
   1.00782503  1.00782503  1.00782503  1.00782503], real=[ True  True  True  True␣
↪True  True  True  True  True  True  True  True
   True  True  True  True], atom_labels=['' '' '' '' '' '' '' '' '' '' '' '' '' ''
↪''], atomic_numbers=[6 6 6 6 6 6 6 8 1 1 1 1 1 1 1 1], mass_numbers=[12 12 12 12 12␣
↪12 12 16  1  1  1  1  1  1  1  1], connectivity=[(0, 2, 2.0), (0, 1, 1.0), (0, 8, 1.
↪0), (1, 3, 2.0), (1, 9, 1.0), (2, 4, 1.0), (2, 10, 1.0), (3, 5, 1.0), (3, 11, 1.0),␣
↪(4, 5, 2.0), (4, 12, 1.0), (5, 7, 1.0), (6, 7, 1.0), (6, 13, 1.0), (6, 14, 1.0), (6,
↪ 15, 1.0)], fragments=[array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,␣
↪13, 14, 15],
```

(continues on next page)

(continued from previous page)

```
      dtype=int32)], fragment_charges=[0.0], fragment_multiplicities=[1], fix_
↪com=True, fix_orientation=True, fix_symmetry=None, provenance={'creator':
↪'QCElemental', 'version': 'v0.11.0', 'routine': 'qcelemental.molparse.from_schema'},
↪ id=461369, extras=None),
 (75,): Molecule(schema_name=qcschema_molecule, schema_version=2, validated=True,␣
↪symbols=['C' 'C' 'C' 'C' 'C' 'C' 'C' 'O' 'H' 'H' 'H' 'H' 'H' 'H' 'H' 'H'],␣
↪geometry=[[ 3.70212325 -1.72466955 -0.21585801]
  [ 5.51905339 -2.32444209  1.61008955]
  [ 1.16923478 -1.46408046  0.49415964]
  [ 4.81449559 -2.66039694  4.13241122]
  [ 0.4522114  -1.78473091  3.01641573]
  [ 2.27581552 -2.38406191  4.8372815 ]
  [ 2.15535767 -0.85813513  9.03145663]
  [ 1.52672714 -2.78611098  7.28616169]
  [ 4.25856462 -1.4718899  -2.18825289]
  [ 7.49871004 -2.54995062  1.06400225]
  [-0.2610239  -1.00629602 -0.92454499]
  [ 6.20785963 -3.16338765  5.57048426]
  [-1.51425058 -1.59868966  3.61443722]
  [ 1.4695119  -1.46795424 10.8955    ]
  [ 1.24259402  0.95548078  8.53075546]
  [ 4.21880374 -0.54221236  9.1461044 ]], name=C7H8O, identifiers={'molecule_hash':
↪'b0e45b3d9bfb9763487b040f0053eb857a2140ce', 'molecular_formula': 'C7H8O', 'smiles':␣
↪None, 'inchi': None, 'inchikey': None, 'canonical_explicit_hydrogen_smiles': None,
↪'canonical_isomeric_explicit_hydrogen_mapped_smiles': None, 'canonical_isomeric_
↪explicit_hydrogen_smiles': None, 'canonical_isomeric_smiles': None, 'canonical_
↪smiles': None}, comment=None, molecular_charge=0.0, molecular_multiplicity=1,␣
↪masses=[12.         12.         12.         12.         12.         12.
  12.         15.99491462  1.00782503  1.00782503  1.00782503  1.00782503
   1.00782503  1.00782503  1.00782503  1.00782503], real=[ True  True  True  True ␣
↪True  True  True  True  True  True  True  True
   True  True  True  True], atom_labels=['' '' '' '' '' '' '' '' '' '' '' '' '' '' ''
↪''], atomic_numbers=[6 6 6 6 6 6 8 1 1 1 1 1 1 1 1], mass_numbers=[12 12 12 12 12␣
↪12 12 16  1  1  1  1  1  1  1], connectivity=[(0, 2, 2.0), (0, 1, 1.0), (0, 8, 1.
↪0), (1, 3, 2.0), (1, 9, 1.0), (2, 4, 1.0), (2, 10, 1.0), (3, 5, 1.0), (3, 11, 1.0),␣
↪(4, 5, 2.0), (4, 12, 1.0), (5, 7, 1.0), (6, 7, 1.0), (6, 13, 1.0), (6, 14, 1.0), (6,
↪ 15, 1.0)], fragments=[array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,␣
↪13, 14, 15],
      dtype=int32)], fragment_charges=[0.0], fragment_multiplicities=[1], fix_
↪com=True, fix_orientation=True, fix_symmetry=None, provenance={'creator':
↪'QCElemental', 'version': 'v0.11.0', 'routine': 'qcelemental.molparse.from_schema'},
↪ id=461425, extras=None),
 (-75,): Molecule(schema_name=qcschema_molecule, schema_version=2, validated=True,␣
↪symbols=['C' 'C' 'C' 'C' 'C' 'C' 'C' 'O' 'H' 'H' 'H' 'H' 'H' 'H' 'H' 'H'],␣
↪geometry=[[ 3.43900598 -2.20071294 -0.22823573]
  [ 4.86508261 -3.50716914  1.57581688]
  [ 1.85180055 -0.22953493  0.53510663]
  [ 4.70765322 -2.85320039  4.12901981]
  [ 1.67473259  0.42691458  3.08785967]
  [ 3.10410599 -0.88523154  4.88656201]
  [ 1.63777454 -1.79343168  9.00470539]
  [ 2.9808923  -0.15490425  7.37079048]
  [ 3.57347597 -2.7115241  -2.22444734]
  [ 6.12387823 -5.03664869  0.98911017]
  [ 0.74084538  0.80565965 -0.86563389]
  [ 5.83674841 -3.83404887  5.55225734]
  [ 0.45497214  1.96371446  3.7278884 ]
```

(continues on next page)

(continued from previous page)

```
  [ 1.7397371  -0.9786394  10.9132759 ]
  [ 2.46538192 -3.71275772  9.04941128]
  [-0.36950472 -1.95935811  8.4422741 ]], name=C7H8O, identifiers={'molecule_hash':
↪'6fe82700dc2bbb954cfa35bce052a43ef355e8c3', 'molecular_formula': 'C7H8O', 'smiles':␣
↪None, 'inchi': None, 'inchikey': None, 'canonical_explicit_hydrogen_smiles': None,
↪'canonical_isomeric_explicit_hydrogen_mapped_smiles': None, 'canonical_isomeric_
↪explicit_hydrogen_smiles': None, 'canonical_isomeric_smiles': None, 'canonical_
↪smiles': None}, comment=None, molecular_charge=0.0, molecular_multiplicity=1,␣
↪masses=[12.         12.         12.         12.         12.         12.
  12.         15.99491462  1.00782503  1.00782503  1.00782503  1.00782503
   1.00782503  1.00782503  1.00782503  1.00782503], real=[ True  True  True  True ␣
↪True  True  True  True  True  True  True  True
   True  True  True  True], atom_labels=['' '' '' '' '' '' '' '' '' '' '' '' '' '' ''
↪''], atomic_numbers=[6 6 6 6 6 6 6 8 1 1 1 1 1 1 1 1], mass_numbers=[12 12 12 12 12␣
↪12 12 16  1  1  1  1  1  1  1  1], connectivity=[(0, 2, 2.0), (0, 1, 1.0), (0, 8, 1.
↪0), (1, 3, 2.0), (1, 9, 1.0), (2, 4, 1.0), (2, 10, 1.0), (3, 5, 1.0), (3, 11, 1.0),␣
↪(4, 5, 2.0), (4, 12, 1.0), (5, 7, 1.0), (6, 7, 1.0), (6, 13, 1.0), (6, 14, 1.0), (6,
↪ 15, 1.0)], fragments=[array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,␣
↪13, 14, 15],
      dtype=int32)], fragment_charges=[0.0], fragment_multiplicities=[1], fix_
↪com=True, fix_orientation=True, fix_symmetry=None, provenance={'creator':
↪'QCElemental', 'version': 'v0.11.0', 'routine': 'qcelemental.molparse.from_schema'},
↪ id=461459, extras=None),
 (30,): Molecule(schema_name=qcschema_molecule, schema_version=2, validated=True,␣
↪symbols=['C' 'C' 'C' 'C' 'C' 'C' 'C' 'O' 'H' 'H' 'H' 'H' 'H' 'H' 'H' 'H'],␣
↪geometry=[[ 3.69600344 -1.85447301 -0.30101366]
  [ 5.42957732 -1.96441247  1.68575814]
  [ 1.10571549 -1.80496047  0.24404663]
  [ 4.60860206 -2.02544662  4.19938437]
  [ 0.26273043 -1.85146068  2.73915521]
  [ 2.01117613 -1.95189563  4.73362008]
  [ 2.47935475 -1.15790713  9.14730531]
  [ 1.03798047 -2.10187281  7.10982174]
  [ 4.35246849 -1.80992101 -2.25772808]
  [ 7.4563067  -2.0155252   1.28725824]
  [-0.27319083 -1.72058681 -1.29211501]
  [ 5.9901253  -2.13100274  5.72583171]
  [-1.74683847 -1.81374977  3.20562904]
  [ 1.1989466  -1.02138253 10.77726717]
  [ 3.26020365  0.7331975   8.72723948]
  [ 4.05738793 -2.42970717  9.6581919 ]], name=C7H8O, identifiers={'molecule_hash':
↪'5aba4f48598014250d3b37a70e119aa9d17b76b8', 'molecular_formula': 'C7H8O', 'smiles':␣
↪None, 'inchi': None, 'inchikey': None, 'canonical_explicit_hydrogen_smiles': None,
↪'canonical_isomeric_explicit_hydrogen_mapped_smiles': None, 'canonical_isomeric_
↪explicit_hydrogen_smiles': None, 'canonical_isomeric_smiles': None, 'canonical_
↪smiles': None}, comment=None, molecular_charge=0.0, molecular_multiplicity=1,␣
↪masses=[12.         12.         12.         12.         12.         12.
  12.         15.99491462  1.00782503  1.00782503  1.00782503  1.00782503
   1.00782503  1.00782503  1.00782503  1.00782503], real=[ True  True  True  True ␣
↪True  True  True  True  True  True  True  True
   True  True  True  True], atom_labels=['' '' '' '' '' '' '' '' '' '' '' '' '' '' ''
↪''], atomic_numbers=[6 6 6 6 6 6 6 8 1 1 1 1 1 1 1 1], mass_numbers=[12 12 12 12 12␣
↪12 12 16  1  1  1  1  1  1  1  1], connectivity=[(0, 2, 2.0), (0, 1, 1.0), (0, 8, 1.
↪0), (1, 3, 2.0), (1, 9, 1.0), (2, 4, 1.0), (2, 10, 1.0), (3, 5, 1.0), (3, 11, 1.0),␣
↪(4, 5, 2.0), (4, 12, 1.0), (5, 7, 1.0), (6, 7, 1.0), (6, 13, 1.0), (6, 14, 1.0), (6,
↪ 15, 1.0)], fragments=[array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,␣
↪13, 14, 15],
```

(continues on next page)

(continued from previous page)

```
      dtype=int32)], fragment_charges=[0.0], fragment_multiplicities=[1], fix_
↪com=True, fix_orientation=True, fix_symmetry=None, provenance={'creator':
↪'QCElemental', 'version': 'v0.11.0', 'routine': 'qcelemental.molparse.from_schema'},
↪ id=461723, extras=None),
 (60,): Molecule(schema_name=qcschema_molecule, schema_version=2, validated=True,␣
↪symbols=['C' 'C' 'C' 'C' 'C' 'C' 'C' 'O' 'H' 'H' 'H' 'H' 'H' 'H' 'H' 'H'],␣
↪geometry=[[ 3.71828345 -1.76811666 -0.27338275]
  [ 5.50089579 -2.20886409  1.62864783]
  [ 1.15779938 -1.58361834  0.35850376]
  [ 4.74133601 -2.46356402  4.14507752]
  [ 0.38360523 -1.81737538  2.87169991]
  [ 2.17400832 -2.25536827  4.7722983 ]
  [ 2.28444156 -0.94953994  9.06682808]
  [ 1.32018125 -2.59349763  7.19110947]
  [ 4.32142939 -1.57815469 -2.23890818]
  [ 7.50367599 -2.37607107  1.14971388]
  [-0.25023734 -1.246811   -1.11528944]
  [ 6.12828802 -2.84801683  5.62417659]
  [-1.60352774 -1.68120413  3.41076772]
  [ 1.31483482 -1.42137021 10.84263787]
  [ 1.90592803  1.05132903  8.59565568]
  [ 4.34048833 -1.17965735  9.35939703]], name=C7H8O, identifiers={'molecule_hash':
↪'179c476f669865b56eebca36680d181929e6e5b7', 'molecular_formula': 'C7H8O', 'smiles':␣
↪None, 'inchi': None, 'inchikey': None, 'canonical_explicit_hydrogen_smiles': None,
↪'canonical_isomeric_explicit_hydrogen_mapped_smiles': None, 'canonical_isomeric_
↪explicit_hydrogen_smiles': None, 'canonical_isomeric_smiles': None, 'canonical_
↪smiles': None}, comment=None, molecular_charge=0.0, molecular_multiplicity=1,␣
↪masses=[12.         12.         12.         12.         12.         12.
  12.         15.99491462  1.00782503  1.00782503  1.00782503  1.00782503
   1.00782503  1.00782503  1.00782503  1.00782503], real=[ True  True  True  True ␣
↪True  True  True  True  True  True  True  True
   True  True  True  True], atom_labels=['' '' '' '' '' '' '' '' '' '' '' '' '' '' ''
↪''], atomic_numbers=[6 6 6 6 6 6 8 1 1 1 1 1 1 1 1], mass_numbers=[12 12 12 12 12␣
↪12 12 16  1  1  1  1  1  1  1], connectivity=[(0, 2, 2.0), (0, 1, 1.0), (0, 8, 1.
↪0), (1, 3, 2.0), (1, 9, 1.0), (2, 4, 1.0), (2, 10, 1.0), (3, 5, 1.0), (3, 11, 1.0),␣
↪(4, 5, 2.0), (4, 12, 1.0), (5, 7, 1.0), (6, 7, 1.0), (6, 13, 1.0), (6, 14, 1.0), (6,
↪ 15, 1.0)], fragments=[array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,␣
↪13, 14, 15],
      dtype=int32)], fragment_charges=[0.0], fragment_multiplicities=[1], fix_
↪com=True, fix_orientation=True, fix_symmetry=None, provenance={'creator':
↪'QCElemental', 'version': 'v0.11.0', 'routine': 'qcelemental.molparse.from_schema'},
↪ id=461685, extras=None),
 (-60,): Molecule(schema_name=qcschema_molecule, schema_version=2, validated=True,␣
↪symbols=['C' 'C' 'C' 'C' 'C' 'C' 'C' 'O' 'H' 'H' 'H' 'H' 'H' 'H' 'H' 'H'],␣
↪geometry=[[ 3.48556806 -2.18859899 -0.28454022]
  [ 4.75712849 -3.55422444  1.58838991]
  [ 1.95045647 -0.14980855  0.40552128]
  [ 4.50247118 -2.8965478   4.13335644]
  [ 1.67036486  0.50930549  2.9466371 ]
  [ 2.94379439 -0.86431289  4.81779631]
  [ 1.78466861 -1.85587089  9.04078282]
  [ 2.71212821 -0.07793354  7.2717555 ]
  [ 3.69732353 -2.7058812  -2.2721875 ]
  [ 5.9768439  -5.13753051  1.06503382]
  [ 0.95491809  0.93350363 -1.0447928 ]
  [ 5.52697525 -3.93700323  5.5917871 ]
  [ 0.48365833  2.09295743  3.53111283]
```

(continues on next page)

(continued from previous page)

```
  [ 1.62148788 -0.86984945 10.86215514]
  [ 3.06567807 -3.48777783  9.28933466]
  [-0.097693   -2.57462474  8.48502773]], name=C7H8O, identifiers={'molecule_hash':
↪'b44ad27c199728b72c05ba913385cb2bc642c8a8', 'molecular_formula': 'C7H8O', 'smiles':␣
↪None, 'inchi': None, 'inchikey': None, 'canonical_explicit_hydrogen_smiles': None,
↪'canonical_isomeric_explicit_hydrogen_mapped_smiles': None, 'canonical_isomeric_
↪explicit_hydrogen_smiles': None, 'canonical_isomeric_smiles': None, 'canonical_
↪smiles': None}, comment=None, molecular_charge=0.0, molecular_multiplicity=1,␣
↪masses=[12.         12.         12.         12.         12.         12.
  12.         15.99491462  1.00782503  1.00782503  1.00782503  1.00782503
   1.00782503  1.00782503  1.00782503  1.00782503], real=[ True  True  True  True␣
↪True  True  True  True  True  True  True  True
   True  True  True  True], atom_labels=['' '' '' '' '' '' '' '' '' '' '' '' '' ''
↪''], atomic_numbers=[6 6 6 6 6 6 6 8 1 1 1 1 1 1 1 1], mass_numbers=[12 12 12 12 12␣
↪12 12 16  1  1  1  1  1  1  1  1], connectivity=[(0, 2, 2.0), (0, 1, 1.0), (0, 8, 1.
↪0), (1, 3, 2.0), (1, 9, 1.0), (2, 4, 1.0), (2, 10, 1.0), (3, 5, 1.0), (3, 11, 1.0),␣
↪(4, 5, 2.0), (4, 12, 1.0), (5, 7, 1.0), (6, 7, 1.0), (6, 13, 1.0), (6, 14, 1.0), (6,
↪ 15, 1.0)], fragments=[array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,␣
↪13, 14, 15],
      dtype=int32)], fragment_charges=[0.0], fragment_multiplicities=[1], fix_
↪com=True, fix_orientation=True, fix_symmetry=None, provenance={'creator':
↪'QCElemental', 'version': 'v0.11.0', 'routine': 'qcelemental.molparse.from_schema'},
↪ id=461700, extras=None),
 (-30,): Molecule(schema_name=qcschema_molecule, schema_version=2, validated=True,␣
↪symbols=['C' 'C' 'C' 'C' 'C' 'C' 'C' 'O' 'H' 'H' 'H' 'H' 'H' 'H' 'H' 'H'],␣
↪geometry=[[ 3.54750913 -2.12259724 -0.30742535]
  [ 4.51078875 -3.62629947  1.63566464]
  [ 2.11306417  0.0171931   0.30314439]
  [ 4.05963677 -3.01927365  4.17022987]
  [ 1.63838631  0.63602012  2.81911036]
  [ 2.60133437 -0.88567949  4.76903241]
  [ 2.06262868 -1.91409872  9.1259098 ]
  [ 2.14806198 -0.09606409  7.17464836]
  [ 3.91046327 -2.60827962 -2.28075711]
  [ 5.64288319 -5.29531538  1.18645125]
  [ 1.34830037  1.21278008 -1.19818872]
  [ 4.84397415 -4.20537125  5.66280304]
  [ 0.52497984  2.29437545  3.33637992]
  [ 1.22301635 -0.98084951 10.78088872]
  [ 3.96516229 -2.59916291  9.65479546]
  [ 0.88871412 -3.55824497  8.59566222]], name=C7H8O, identifiers={'molecule_hash':
↪'c5c33ab70cf9df182831507c6ed5daea82827341', 'molecular_formula': 'C7H8O', 'smiles':␣
↪None, 'inchi': None, 'inchikey': None, 'canonical_explicit_hydrogen_smiles': None,
↪'canonical_isomeric_explicit_hydrogen_mapped_smiles': None, 'canonical_isomeric_
↪explicit_hydrogen_smiles': None, 'canonical_isomeric_smiles': None, 'canonical_
↪smiles': None}, comment=None, molecular_charge=0.0, molecular_multiplicity=1,␣
↪masses=[12.         12.         12.         12.         12.         12.
  12.         15.99491462  1.00782503  1.00782503  1.00782503  1.00782503
   1.00782503  1.00782503  1.00782503  1.00782503], real=[ True  True  True  True␣
↪True  True  True  True  True  True  True  True
   True  True  True  True], atom_labels=['' '' '' '' '' '' '' '' '' '' '' '' '' ''
↪''], atomic_numbers=[6 6 6 6 6 6 6 8 1 1 1 1 1 1 1 1], mass_numbers=[12 12 12 12 12␣
↪12 12 16  1  1  1  1  1  1  1  1], connectivity=[(0, 2, 2.0), (0, 1, 1.0), (0, 8, 1.
↪0), (1, 3, 2.0), (1, 9, 1.0), (2, 4, 1.0), (2, 10, 1.0), (3, 5, 1.0), (3, 11, 1.0),␣
↪(4, 5, 2.0), (4, 12, 1.0), (5, 7, 1.0), (6, 7, 1.0), (6, 13, 1.0), (6, 14, 1.0), (6,
↪ 15, 1.0)], fragments=[array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,␣
↪13, 14, 15],
```

(continues on next page)

(continued from previous page)

```
      dtype=int32)], fragment_charges=[0.0], fragment_multiplicities=[1], fix_
↪com=True, fix_orientation=True, fix_symmetry=None, provenance={'creator':
↪'QCElemental', 'version': 'v0.11.0', 'routine': 'qcelemental.molparse.from_schema'},
↪ id=461731, extras=None),
 (45,): Molecule(schema_name=qcschema_molecule, schema_version=2, validated=True,␣
↪symbols=['C' 'C' 'C' 'C' 'C' 'C' 'C' 'O' 'H' 'H' 'H' 'H' 'H' 'H' 'H' 'H'],␣
↪geometry=[[ 3.69639473 -1.79743864 -0.29873657]
  [ 5.45928124 -2.05967591  1.65083976]
  [ 1.11810236 -1.70428504  0.28674669]
  [ 4.67054666 -2.22992127  4.16744948]
  [ 0.3109718  -1.85413007  2.7927833 ]
  [ 2.08514535 -2.10889291  4.74580389]
  [ 2.40545302 -1.06954623  9.11534623]
  [ 1.16885394 -2.37939496  7.14132671]
  [ 4.32497976 -1.67092038 -2.26118922]
  [ 7.47686639 -2.15019919  1.21495776]
  [-0.27892714 -1.5030941  -1.22190147]
  [ 6.05998786 -2.46332476  5.67393071]
  [-1.68935396 -1.78551201  3.29413963]
  [ 1.14500254 -1.13490653 10.76580932]
  [ 2.76366218  0.92393291  8.60563825]
  [ 4.22194341 -1.95474001  9.648864  ]], name=C7H8O, identifiers={'molecule_hash':
↪'a159c8d539fe8935046beac7fdeabf3a66465c1c', 'molecular_formula': 'C7H8O', 'smiles':␣
↪None, 'inchi': None, 'inchikey': None, 'canonical_explicit_hydrogen_smiles': None,
↪'canonical_isomeric_explicit_hydrogen_mapped_smiles': None, 'canonical_isomeric_
↪explicit_hydrogen_smiles': None, 'canonical_isomeric_smiles': None, 'canonical_
↪smiles': None}, comment=None, molecular_charge=0.0, molecular_multiplicity=1,␣
↪masses=[12.         12.         12.         12.         12.         12.
  12.         15.99491462  1.00782503  1.00782503  1.00782503  1.00782503
   1.00782503  1.00782503  1.00782503  1.00782503], real=[ True  True  True  True ␣
↪True  True  True  True  True  True  True  True
   True  True  True  True], atom_labels=['' '' '' '' '' '' '' '' '' '' '' '' '' '' ''
↪''], atomic_numbers=[6 6 6 6 6 6 8 1 1 1 1 1 1 1 1], mass_numbers=[12 12 12 12 12␣
↪12 12 16  1  1  1  1  1  1  1], connectivity=[(0, 2, 2.0), (0, 1, 1.0), (0, 8, 1.
↪0), (1, 3, 2.0), (1, 9, 1.0), (2, 4, 1.0), (2, 10, 1.0), (3, 5, 1.0), (3, 11, 1.0),␣
↪(4, 5, 2.0), (4, 12, 1.0), (5, 7, 1.0), (6, 7, 1.0), (6, 13, 1.0), (6, 14, 1.0), (6,
↪ 15, 1.0)], fragments=[array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,␣
↪13, 14, 15],
      dtype=int32)], fragment_charges=[0.0], fragment_multiplicities=[1], fix_
↪com=True, fix_orientation=True, fix_symmetry=None, provenance={'creator':
↪'QCElemental', 'version': 'v0.11.0', 'routine': 'qcelemental.molparse.from_schema'},
↪ id=461819, extras=None),
 (-45,): Molecule(schema_name=qcschema_molecule, schema_version=2, validated=True,␣
↪symbols=['C' 'C' 'C' 'C' 'C' 'C' 'C' 'O' 'H' 'H' 'H' 'H' 'H' 'H' 'H' 'H'],␣
↪geometry=[[ 3.49939228 -2.15330139 -0.30787911]
  [ 4.60813322 -3.59922556  1.60454938]
  [ 2.03333459 -0.04875028  0.34065581]
  [ 4.26647776 -2.96167543  4.14644088]
  [ 1.66490412  0.59404878  2.87150549]
  [ 2.77323437 -0.86555735  4.78671794]
  [ 1.94936593 -1.89691127  9.0917486 ]
  [ 2.4517307  -0.06107251  7.21585974]
  [ 3.77829643 -2.65860577 -2.29010662]
  [ 5.7696973  -5.23780629  1.12013198]
  [ 1.15934525  1.09887588 -1.13829112]
  [ 5.16374776 -4.08568481  5.62496036]
  [ 0.52934295  2.22651639  3.42188566]
```

(continues on next page)

(continued from previous page)

```
  [ 1.290692   -0.87441177 10.77628966]
  [ 3.65100376 -2.98999082  9.6184854 ]
  [ 0.46628521 -3.23341981  8.47832305]], name=C7H8O, identifiers={'molecule_hash':
↪'5c9839877f910bb7dafc3694046103f7cee32e02', 'molecular_formula': 'C7H8O', 'smiles':␣
↪None, 'inchi': None, 'inchikey': None, 'canonical_explicit_hydrogen_smiles': None,
↪'canonical_isomeric_explicit_hydrogen_mapped_smiles': None, 'canonical_isomeric_
↪explicit_hydrogen_smiles': None, 'canonical_isomeric_smiles': None, 'canonical_
↪smiles': None}, comment=None, molecular_charge=0.0, molecular_multiplicity=1,␣
↪masses=[12.         12.         12.         12.         12.         12.
  12.         15.99491462  1.00782503  1.00782503  1.00782503  1.00782503
   1.00782503  1.00782503  1.00782503  1.00782503], real=[ True  True  True  True␣
↪True  True  True  True  True  True  True  True
   True  True  True  True], atom_labels=['' '' '' '' '' '' '' '' '' '' '' '' '' '' ''
↪''], atomic_numbers=[6 6 6 6 6 6 6 8 1 1 1 1 1 1 1 1], mass_numbers=[12 12 12 12 12␣
↪12 12 16  1  1  1  1  1  1  1  1], connectivity=[(0, 2, 2.0), (0, 1, 1.0), (0, 8, 1.
↪0), (1, 3, 2.0), (1, 9, 1.0), (2, 4, 1.0), (2, 10, 1.0), (3, 5, 1.0), (3, 11, 1.0),␣
↪(4, 5, 2.0), (4, 12, 1.0), (5, 7, 1.0), (6, 7, 1.0), (6, 13, 1.0), (6, 14, 1.0), (6,
↪ 15, 1.0)], fragments=[array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,␣
↪13, 14, 15],
      dtype=int32)], fragment_charges=[0.0], fragment_multiplicities=[1], fix_
↪com=True, fix_orientation=True, fix_symmetry=None, provenance={'creator':
↪'QCElemental', 'version': 'v0.11.0', 'routine': 'qcelemental.molparse.from_schema'},
↪ id=461827, extras=None),
 (0,): Molecule(schema_name=qcschema_molecule, schema_version=2, validated=True,␣
↪symbols=['C' 'C' 'C' 'C' 'C' 'C' 'C' 'O' 'H' 'H' 'H' 'H' 'H' 'H' 'H' 'H'],␣
↪geometry=[[ 3.705481   -2.005779   -0.30916382]
  [ 5.41340407 -1.8578246   1.69359705]
  [ 1.10677377 -1.96699335  0.20622924]
  [ 4.56869423 -1.67222782  4.19620825]
  [ 0.23744147 -1.78335773  2.68235995]
  [ 1.96387315 -1.63440946  4.69802417]
  [ 2.54284232 -1.30378749  9.1626126 ]
  [ 0.93684859 -1.46033225  7.04427198]
  [ 4.38361058 -2.14990825 -2.25347798]
  [ 7.44581918 -1.88579145  1.32253157]
  [-0.25684224 -2.08140433 -1.34162258]
  [ 5.94690775 -1.55958853  5.72300638]
  [-1.77796083 -1.75024478  3.1236002 ]
  [ 1.31441531 -1.17992622 10.8317381 ]
  [ 3.76178701  0.39258319  9.09604855]
  [ 3.76126299 -2.99182153  9.34675257]], name=C7H8O, identifiers={'molecule_hash':
↪'30b86628fd1d834aa7ad1deeb50d962a82a0214d', 'molecular_formula': 'C7H8O', 'smiles':␣
↪None, 'inchi': None, 'inchikey': None, 'canonical_explicit_hydrogen_smiles': None,
↪'canonical_isomeric_explicit_hydrogen_mapped_smiles': None, 'canonical_isomeric_
↪explicit_hydrogen_smiles': None, 'canonical_isomeric_smiles': None, 'canonical_
↪smiles': None}, comment=None, molecular_charge=0.0, molecular_multiplicity=1,␣
↪masses=[12.         12.         12.         12.         12.         12.
  12.         15.99491462  1.00782503  1.00782503  1.00782503  1.00782503
   1.00782503  1.00782503  1.00782503  1.00782503], real=[ True  True  True  True␣
↪True  True  True  True  True  True  True  True
   True  True  True  True], atom_labels=['' '' '' '' '' '' '' '' '' '' '' '' '' '' ''
↪''], atomic_numbers=[6 6 6 6 6 6 6 8 1 1 1 1 1 1 1 1], mass_numbers=[12 12 12 12 12␣
↪12 12 16  1  1  1  1  1  1  1  1], connectivity=[(0, 2, 2.0), (0, 1, 1.0), (0, 8, 1.
↪0), (1, 3, 2.0), (1, 9, 1.0), (2, 4, 1.0), (2, 10, 1.0), (3, 5, 1.0), (3, 11, 1.0),␣
↪(4, 5, 2.0), (4, 12, 1.0), (5, 7, 1.0), (6, 7, 1.0), (6, 13, 1.0), (6, 14, 1.0), (6,
↪ 15, 1.0)], fragments=[array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,␣
↪13, 14, 15],
```

(continues on next page)

(continued from previous page)

```
      dtype=int32)], fragment_charges=[0.0], fragment_multiplicities=[1], fix_
↪com=True, fix_orientation=True, fix_symmetry=None, provenance={'creator':
↪'QCElemental', 'version': 'v0.11.0', 'routine': 'qcelemental.molparse.from_schema'},
↪ id=463178, extras=None),
 (-15,): Molecule(schema_name=qcschema_molecule, schema_version=2, validated=True,␣
↪symbols=['C' 'C' 'C' 'C' 'C' 'C' 'C' 'O' 'H' 'H' 'H' 'H' 'H' 'H' 'H' 'H'],␣
↪geometry=[[ 3.70444484 -2.08536058 -0.30602168]
  [ 5.42171814 -1.81248734  1.67650045]
  [ 1.10845069 -2.04737062  0.22011872]
  [ 4.58579213 -1.50370096  4.16903174]
  [ 0.24847747 -1.74870149  2.6892643 ]
  [ 1.98334563 -1.48072909  4.68387251]
  [ 2.54115462 -1.3800461   9.16114721]
  [ 0.97332592 -1.13872669  7.02046597]
  [ 4.37458448 -2.32456995 -2.24376515]
  [ 7.45230012 -1.83222076  1.29503933]
  [-0.26085794 -2.25792583 -1.31253249]
  [ 5.96644644 -1.28397146  5.68249548]
  [-1.76495913 -1.71423857  3.13922479]
  [ 1.29951636 -1.2873653  10.82289062]
  [ 3.94041723  0.16716183  9.29131427]
  [ 3.56236256 -3.20285849  9.1733056 ]], name=C7H8O, identifiers={'molecule_hash':
↪'6353c768493bd01d50a28569b04b0bd1faeccfb3', 'molecular_formula': 'C7H8O', 'smiles':␣
↪None, 'inchi': None, 'inchikey': None, 'canonical_explicit_hydrogen_smiles': None,
↪'canonical_isomeric_explicit_hydrogen_mapped_smiles': None, 'canonical_isomeric_
↪explicit_hydrogen_smiles': None, 'canonical_isomeric_smiles': None, 'canonical_
↪smiles': None}, comment=None, molecular_charge=0.0, molecular_multiplicity=1,␣
↪masses=[12.         12.         12.         12.         12.         12.
  12.         15.99491462  1.00782503  1.00782503  1.00782503  1.00782503
   1.00782503  1.00782503  1.00782503  1.00782503], real=[ True  True  True  True ␣
↪True  True  True  True  True  True  True  True
   True  True  True  True], atom_labels=['' '' '' '' '' '' '' '' '' '' '' '' '' '' ''
↪''], atomic_numbers=[6 6 6 6 6 6 8 1 1 1 1 1 1 1 1], mass_numbers=[12 12 12 12 12␣
↪12 12 16  1  1  1  1  1  1  1], connectivity=[(0, 2, 2.0), (0, 1, 1.0), (0, 8, 1.
↪0), (1, 3, 2.0), (1, 9, 1.0), (2, 4, 1.0), (2, 10, 1.0), (3, 5, 1.0), (3, 11, 1.0),␣
↪(4, 5, 2.0), (4, 12, 1.0), (5, 7, 1.0), (6, 7, 1.0), (6, 13, 1.0), (6, 14, 1.0), (6,
↪ 15, 1.0)], fragments=[array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,␣
↪13, 14, 15],
      dtype=int32)], fragment_charges=[0.0], fragment_multiplicities=[1], fix_
↪com=True, fix_orientation=True, fix_symmetry=None, provenance={'creator':
↪'QCElemental', 'version': 'v0.11.0', 'routine': 'qcelemental.molparse.from_schema'},
↪ id=463335, extras=None),
 (15,): Molecule(schema_name=qcschema_molecule, schema_version=2, validated=True,␣
↪symbols=['C' 'C' 'C' 'C' 'C' 'C' 'C' 'O' 'H' 'H' 'H' 'H' 'H' 'H' 'H' 'H'],␣
↪geometry=[[ 3.70448247 -1.92662444 -0.31778593]
  [ 5.42169639 -1.90514381  1.68336373]
  [ 1.10848889 -1.88548988  0.20812449]
  [ 4.58571506 -1.84287149  4.19415947]
  [ 0.24846249 -1.81664612  2.69429743]
  [ 1.983275   -1.78854545  4.70667893]
  [ 2.5411861  -1.22859275  9.14992915]
  [ 0.97317763 -1.78202702  7.06812715]
  [ 4.37466574 -1.97588365 -2.26960286]
  [ 7.45227515 -1.94275871  1.3032283 ]
  [-0.26077669 -1.90247038 -1.33886803]
  [ 5.96632233 -1.83779682  5.72352584]
  [-1.76497555 -1.78351301  3.14435528]
```

(continues on next page)

(continued from previous page)

```
  [ 1.29956022 -1.07481746 10.80714566]
  [ 3.56322061  0.57564798  8.8933725 ]
  [ 3.93975196 -2.74036093  9.50672658]], name=C7H8O, identifiers={'molecule_hash':
↪'0fb2d9128eabf27a29c645d57a5ce4d64ce03dc4', 'molecular_formula': 'C7H8O', 'smiles':␣
↪None, 'inchi': None, 'inchikey': None, 'canonical_explicit_hydrogen_smiles': None,
↪'canonical_isomeric_explicit_hydrogen_mapped_smiles': None, 'canonical_isomeric_
↪explicit_hydrogen_smiles': None, 'canonical_isomeric_smiles': None, 'canonical_
↪smiles': None}, comment=None, molecular_charge=0.0, molecular_multiplicity=1,␣
↪masses=[12.         12.         12.         12.         12.         12.
  12.         15.99491462  1.00782503  1.00782503  1.00782503  1.00782503
   1.00782503  1.00782503  1.00782503  1.00782503], real=[ True  True  True  True␣
↪True  True  True  True  True  True  True  True
   True  True  True  True], atom_labels=['' '' '' '' '' '' '' '' '' '' '' '' '' ''
↪''], atomic_numbers=[6 6 6 6 6 6 8 1 1 1 1 1 1 1 1], mass_numbers=[12 12 12 12 12␣
↪12 12 16  1  1  1  1  1  1  1  1], connectivity=[(0, 2, 2.0), (0, 1, 1.0), (0, 8, 1.
↪0), (1, 3, 2.0), (1, 9, 1.0), (2, 4, 1.0), (2, 10, 1.0), (3, 5, 1.0), (3, 11, 1.0),␣
↪(4, 5, 2.0), (4, 12, 1.0), (5, 7, 1.0), (6, 7, 1.0), (6, 13, 1.0), (6, 14, 1.0), (6,
↪ 15, 1.0)], fragments=[array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,␣
↪13, 14, 15],
      dtype=int32)], fragment_charges=[0.0], fragment_multiplicities=[1], fix_
↪com=True, fix_orientation=True, fix_symmetry=None, provenance={'creator':
↪'QCElemental', 'version': 'v0.11.0', 'routine': 'qcelemental.molparse.from_schema'},
↪ id=465210, extras=None)}
```

1.4.5 Exploring connection optimizations

If desired, we can pull each geometry optimization belonging to the torsiondrive with the `get_history` function. In this case, we will pull the lowest energy optimization for the 180 degree dihedral:

[17]:
```
opt = td.get_history(180, minimum=True)
opt
```

[17]:
```
OptimizationRecord(id=1701507, hash_index=effb8d6307ac3cc1a263e674e93342498f10219f,␣
↪procedure=optimization, program=geometric, version=1, protocols={}, extras={},␣
↪stdout=228720, stderr=None, error=None, task_id=5c988a95b6a2de5fb9b9e33f, manager_
↪name=None, status=COMPLETE, modified_on=2019-03-25 08:02:57.063000, created_on=2019-
↪03-22 20:21:05.145000, provenance={'creator': 'geomeTRIC', 'version': '0.9.5',
↪'routine': 'geometric.run_json.geometric_run_json', 'wall_time': 111.76719880104065,
↪ 'qcengine_version': 'v0.6.4', 'cpu': 'Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz',
↪'username': 'lnaden', 'hostname': 'dt039'}, schema_version=1, initial_
↪molecule=458443, qc_spec={'driver': <DriverEnum.gradient: 'gradient'>, 'method':
↪'b3lyp-d3', 'basis': 'def2-svp', 'keywords': '5c954fa6b6a2de5f188ea234', 'program':
↪'psi4'}, keywords={'coordsys': 'tric', 'enforce': 0.1, 'constraints': {'set': [{
↪'type': 'dihedral', 'indices': [3, 5, 7, 6], 'value': 180}]}, 'program': 'psi4'},␣
↪energies=[-346.53147622614085, -346.53177915374135, -346.5318873591642, -346.
↪53194287295247, -346.53197743434515, -346.5319973422999, -346.53199831291954, -346.
↪53199826976714, -346.5319986074462], final_molecule=458506, trajectory=['594805',
↪'594862', '594863', '594864', '594865', '594866', '594867', '594868', '594869'])
```

[18]:
```
opt.energies
```

[18]:
```
[-346.53147622614085,
 -346.53177915374135,
 -346.5318873591642,
 -346.53194287295247,
```

(continues on next page)

(continued from previous page)

```
 -346.53197743434515,
 -346.5319973422999,
 -346.53199831291954,
 -346.53199826976714,
 -346.5319986074462]
```

1.4.6 Exploring individual gradient evaluations

We can go even deeper in the calculations to look at each gradient calculation of the `Optimization` calculation if we so choose and see even more details about how the `TorsionDrive` object was constructed.

[19]:
```
result = opt.get_trajectory()[-1]
```

[20]:
```
print("Program:                   {}".format(result.program))
print("Number of Basis Functions: {}".format(result.properties.calcinfo_nbasis))
print("Total execution time:      {:.2f}s".format(result.provenance.wall_time))
```

```
Program:                   psi4
Number of Basis Functions: 152
Total execution time:      12.15s
```

This example also contains the Wiberg-Lowdin indices. As this is specific to Psi4, this data resides inside the `extras` tag rather than general properties. This data is not yet well curated and currently exists as a 1D list. We will first pull this data and transform it to a 2D array:

[21]:
```
import numpy as np
wiberg = np.array(result.extras["qcvars"]["WIBERG_LOWDIN_INDICES"]).reshape(-1, 16)
```

As this particular example is exploring the `3-5-7-6` dihedral, we would find the most use in the `5-7` bond. This can be acquired as follows:

[22]:
```
wiberg[5, 7]
```

[22]:
```
1.2700940280530457
```

We can continue to explore these results and even obtain the standard Psi4 logging information!

[23]:
```
print(result.get_stdout()[:1000])
```

```

  Memory set to  60.800 GiB by Python driver.
gradient() will perform analytic gradient computation.

*** tstart() called on dt039
*** at Mon Mar 25 04:02:23 2019

   => Loading Basis Set <=

    Name: DEF2-SVP
    Role: ORBITAL
    Keyword: BASIS
    atoms 1-7  entry C          line    90 file /home/lnaden/miniconda3/envs/qca/
↪share/psi4/basis/def2-svp.gbs
    atoms 8    entry O          line   130 file /home/lnaden/miniconda3/envs/qca/
↪share/psi4/basis/def2-svp.gbs
    atoms 9-16 entry H          line    15 file /home/lnaden/miniconda3/envs/qca/
↪share/psi4/basis/def2-svp.gbs
```

(continues on next page)

(continued from previous page)

```
        ---------------------------------------------------------
                                 SCF
               by Justin Turney, Rob Parrish, Andy Simmonett
                          and Daniel G. A. Smith
                              RKS Reference
                        6 Threads,  62259 MiB Core
        ---------------------------------------------------------

  ==> Geometry <==

    Molecular
```

1.5 Overview

This cookbook gathers recipes for manipulating QCArchive data using QCPortal. If you can't find what you're looking for, please reach out to us. Also, please feel free to submit recipes of your own.

Examples in this cookbook use data from the MolSSI QCArchive server for demonstration purposes; all of these examples can be used with any QCFractal instance.

1.5.1 List of Recipes

- *Getting Molecules*
- *All calculations on a molecule*
- *Exploring Datasets and ReactionDatasets*

You can run this notebook online in a session or view it on Github.

1.6 Getting Molecules

This example shows how to get a molecule from QCArchive in a number of contexts.

1.6.1 From an ID

Every molecule computed with QCArchive is assigned a unique ID. If a molecule's ID is known, it can be queried from the Molecules table.

[1]:
```
import qcportal as ptl
client = ptl.FractalClient()
```

For example, molecule 1234 is 1,2,3-trimethylbenzene.

[2]:
```
mol = client.query_molecules(1234)[0]
mol
```

(continued from previous page)

Data type cannot be displayed: application/3dmoljs_load.v0, text/html

[2]:
```
<Molecule(name='C9H12' formula='C9H12' hash='572b510')>
```

[3]:
```
print(mol)
```

```
    Geometry (in Angstrom), charge = 0.0, multiplicity = 1:

       Center              X                 Y                 Z
    ------------   -----------------  -----------------  -----------------
    C                  0.776479871994     1.156134463385     0.121542591228
    C                  0.438429690334     0.679567908122    -1.141595091975
    C                  0.439577078821     0.423533055514     1.255585387764
    C                 -0.363723536834    -0.465178778108    -1.279725991730
    C                 -0.415502828385    -0.685937227907     1.160631416613
    C                 -0.792912983429    -1.170236644458    -0.121804279943
    C                 -0.744392084678    -0.917923156500    -2.666766549983
    C                 -0.856925058179    -1.374181477949     2.427060703777
    C                 -1.703936690413    -2.374380900784    -0.246989621254
    H                  1.380610203168     2.049406423411     0.216714048921
    H                  0.770290662964     1.232461941773    -2.011963177510
    H                  0.769502950936     0.784464203584     2.222141623291
    H                 -0.238962510978    -1.878436765084    -2.898916777516
    H                 -0.447809351101    -0.177691478927    -3.439954373507
    H                 -1.844638825192    -1.050455805875    -2.735084841327
    H                 -1.962016543060    -1.480103641644     2.438815782834
    H                 -0.562925111565    -0.802128403465     3.332572326307
    H                 -0.383242656300    -2.377541231755     2.485353500027
    H                 -2.761425129123    -2.038610393380    -0.229251405356
    H                 -1.542976842368    -3.097214459361     0.578338572599
    H                 -1.519884697209    -2.938478658464    -1.182927991461
```

The following sections show how to find molecule IDs from Collections.

1.6.2 From a Dataset

Load a `Dataset`:

[5]:
```
import qcportal as ptl
client = ptl.FractalClient()

ds = client.get_collection("Dataset", "SMIRNOFF Coverage Set 1")
```

`get_molecules` returns molecules corresponding to row of the `Dataset`:

[6]:
```
molecules = ds.get_molecules()
molecules
```

[6]:
```
                                                      molecule
index
C(CBr)c1n[nH]nn1-1      Geometry (in Angstrom), charge = 0.0, mult...
C(CBr)c1n[nH]nn1-2      Geometry (in Angstrom), charge = 0.0, mult...
C(CBr)c1n[nH]nn1-3      Geometry (in Angstrom), charge = 0.0, mult...
C(CBr)c1n[n-]nn1-0      Geometry (in Angstrom), charge = -1.0, mul...
```

(continues on next page)

(continued from previous page)

```
C(CBr)c1n[n-]nn1-1      Geometry (in Angstrom), charge = -1.0, mul...
...                                                              ...
CSSCCN=C=S-7            Geometry (in Angstrom), charge = 0.0, mult...
CSSCCN=C=S-8            Geometry (in Angstrom), charge = 0.0, mult...
CSSCCN=C=S-9            Geometry (in Angstrom), charge = 0.0, mult...
CSSCCN=C=S-10           Geometry (in Angstrom), charge = 0.0, mult...
CSSCCN=C=S-11           Geometry (in Angstrom), charge = 0.0, mult...

[1109 rows x 1 columns]
```

Individual `Molecule` objects may be picked out of the dataframe:

[8]:
```
molecules.loc["C(CBr)c1n[n-]nn1-0", "molecule"]
```

Data type cannot be displayed: application/3dmoljs_load.v0, text/html

[8]:
```
<Molecule(name='BrC3H4N4' formula='BrC3H4N4' hash='9fd48c6')>
```

For large datasets, you may not want to query all molecules at once. `get_molecules` accepts a subset option for selecting specific molecules:

[9]:
```
ds.get_molecules(subset=['C(CBr)c1n[n-]nn1-0','CSSCCN=C=S-10'])
```

[9]:
```
                                                      molecule
index
C(CBr)c1n[n-]nn1-0      Geometry (in Angstrom), charge = -1.0, mul...
CSSCCN=C=S-10           Geometry (in Angstrom), charge = 0.0, mult...
```

If a single string is provided for `subset`, the `Molecule` object is returned directly.

[10]:
```
ds.get_molecules(subset='CSSCCN=C=S-10')
```

Data type cannot be displayed: application/3dmoljs_load.v0, text/html

[10]:
```
<Molecule(name='C4H7NS3' formula='C4H7NS3' hash='fc1a1d6')>
```

1.6.3 From a ReactionDataset

Load a `ReactionDataset`:

[11]:
```
import qcportal as ptl
client = ptl.FractalClient()

ds = client.get_collection("ReactionDataset", "S22")
```

`get_molecules` returns molecules corresponding to each reaction. By default, the final molecule is returned for every reaction:

[12]:
```
dimers = ds.get_molecules()
dimers
```

[12]:
```
                                                                                ␣
↪         molecule
name                                stoichiometry idx
2-Pyridone-2-Aminopyridine Complex  default       0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
Adenine-Thymine Complex Stack       default       0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
Adenine-Thymine Complex WC          default       0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
Ammonia Dimer                       default       0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
Benzene Dimer PD                    default       0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
Benzene Dimer T-Shape               default       0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
Benzene-Ammonia Complex             default       0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
Benzene-HCN Complex                 default       0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
Benzene-Methane Complex             default       0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
Benzene-Water Complex               default       0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
Ethene Dimer                        default       0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
Ethene-Ethine Complex               default       0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
Formamide Dimer                     default       0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
Formic Acid Dimer                   default       0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
Indole-Benzene Complex Stack        default       0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
Indole-Benzene Complex T-Shape      default       0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
Methane Dimer                       default       0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
Phenol Dimer                        default       0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
Pyrazine Dimer                      default       0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
Uracil Dimer HB                     default       0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
Uracil Dimer Stack                  default       0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
Water Dimer                         default       0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
```

Individual `Molecule` objects may be picked out of the dataframe:

[13]:
```
dimers.loc['Adenine-Thymine Complex WC', 'molecule'][0]
```

Data type cannot be displayed: application/3dmoljs_load.v0, text/html

[13]:
```
<Molecule(name='C10H11N7O2' formula='C10H11N7O2' hash='5357c2c')>
```

Reactants and products (or monomers and complexes) may be picked out with the `stoich` keyword. For the case of

an interaction energy dataset like S22, `stoich="default"` corresponds to complexes and `stoich="default1"` corresponds to the monomers without counterpoise corrections.

[14]:
```
monomers = ds.get_molecules(stoich="default1")
monomers.head(10)
```

[14]:
```
                                                                                ␣
↪          molecule
name                                stoichiometry idx
2-Pyridone-2-Aminopyridine Complex  default1      0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
                                                  1        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
Adenine-Thymine Complex Stack       default1      0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
                                                  1        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
Adenine-Thymine Complex WC          default1      0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
                                                  1        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
Ammonia Dimer                       default1      0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
                                                  1        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
Benzene Dimer PD                    default1      0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
                                                  1        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
```

As before, the individual `Molecule` objects for the monomers may be extracted from the DataFrame:

[15]:
```
monomers.loc['Adenine-Thymine Complex WC', 'molecule'][0]
```

Data type cannot be displayed: application/3dmoljs_load.v0, text/html

[15]:
```
<Molecule(name='C10H11N7O2 ((0,),[])' formula='C5H5N5' hash='c0e7ed3')>
```

[16]:
```
monomers.loc['Adenine-Thymine Complex WC', 'molecule'][1]
```

Data type cannot be displayed: application/3dmoljs_load.v0, text/html

[16]:
```
<Molecule(name='C10H11N7O2 ((1,),[])' formula='C5H6N2O2' hash='a4f9749')>
```

Note that it is possible to get all molecules involved in a reaction by specifying a list for `stoich`:

[17]:
```
ds.get_molecules(stoich=['default', 'default1']).head(15)
```

[17]:
```
                                                                                ␣
↪          molecule
name                                stoichiometry idx
2-Pyridone-2-Aminopyridine Complex  default       0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
                                    default1      0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
                                                  1        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
```

(continues on next page)

(continued from previous page)

```
Adenine-Thymine Complex Stack      default       0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
                                   default1      0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
                                                 1        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
Adenine-Thymine Complex WC         default       0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
                                   default1      0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
                                                 1        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
Ammonia Dimer                      default       0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
                                   default1      0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
                                                 1        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
Benzene Dimer PD                   default       0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
                                   default1      0        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
                                                 1        Geometry (in Angstrom),␣
↪charge = 0.0, mult...
```

Counterpoise-corrected calcuations are available through `stoich="cp"` and `stoich="cp1"`. Counterpoise-corrected monomers contain ghost atoms:

[18]:
```
ds.get_molecules(stoich="cp1").loc['Adenine-Thymine Complex WC', 'molecule'][0]
```

Data type cannot be displayed: application/3dmoljs_load.v0, text/html

[18]:
```
<Molecule(name='C10H11N7O2 ((0,),[1])' formula='C10H11N7O2' hash='d3955aa')>
```

[19]:
```
ds.get_molecules(stoich="cp1").loc['Adenine-Thymine Complex WC', 'molecule'][1]
```

Data type cannot be displayed: application/3dmoljs_load.v0, text/html

[19]:
```
<Molecule(name='C10H11N7O2 ((1,),[0])' formula='C10H11N7O2' hash='e63c41f')>
```

For large datasets, you may not want to query all molecules at once. `get_molecules` accepts a subset option for selecting specific reactions:

[20]:
```
ds.get_molecules(subset='Adenine-Thymine Complex WC')
```

[20]:
```
                                                                              ␣
↪ molecule
name                         stoichiometry idx
Adenine-Thymine Complex WC default       0        Geometry (in Angstrom), charge = 0.
↪0, mult...
```

[21]:
```
ds.get_molecules(subset=['Adenine-Thymine Complex WC', 'Ammonia Dimer', 'Water Dimer
↪'], stoich=['default', 'default1'])
```

[21]:
```
↪ molecule
name                          stoichiometry idx
Adenine-Thymine Complex WC default       0        Geometry (in Angstrom), charge = 0.
↪0, mult...
                              default1     0        Geometry (in Angstrom), charge = 0.
↪0, mult...
                                           1        Geometry (in Angstrom), charge = 0.
↪0, mult...
Ammonia Dimer                 default      0        Geometry (in Angstrom), charge = 0.
↪0, mult...
                              default1     0        Geometry (in Angstrom), charge = 0.
↪0, mult...
                                           1        Geometry (in Angstrom), charge = 0.
↪0, mult...
Water Dimer                   default      0        Geometry (in Angstrom), charge = 0.
↪0, mult...
                              default1     0        Geometry (in Angstrom), charge = 0.
↪0, mult...
                                           1        Geometry (in Angstrom), charge = 0.
↪0, mult...
```

1.6.4 From an OptimizationDataset

Load an `OptimizationDataset`:

[22]:
```
import qcportal as ptl
client = ptl.FractalClient()

client.list_collections()
ds = client.get_collection("OptimizationDataset", "SMIRNOFF Coverage Set 1")
```

Show some available molecules:

[23]:
```
ds.df.head()
```

[23]:
```
Empty DataFrame
Columns: []
Index: [COC(O)OC-0, C[S-]-0, CS-0, CO-0, CCO-0]
```

Show available specifications:

[24]:
```
ds.list_specifications()
```

[24]:
```
                                                Description
Name
default  Standard OpenFF optimization quantum chemistry...
```

Obtain a specific record from a molecule and specification:

[25]:
```
r = ds.get_record("CCO-0","default")
```

Get the optimized molecule:

[26]:
```
r.get_final_molecule()
```

Data type cannot be displayed: application/3dmoljs_load.v0, text/html

[26]:
```
<Molecule(name='C2H6O' formula='C2H6O' hash='422ad57')>
```

Get the optimization trajectory:

[27]:
```
r.get_molecular_trajectory()
```

[27]:
```
[<Molecule(name='C2H6O' formula='C2H6O' hash='29df3ae')>,
 <Molecule(name='C2H6O' formula='C2H6O' hash='93989e4')>,
 <Molecule(name='C2H6O' formula='C2H6O' hash='14261f7')>,
 <Molecule(name='C2H6O' formula='C2H6O' hash='3b6db86')>,
 <Molecule(name='C2H6O' formula='C2H6O' hash='b35d632')>,
 <Molecule(name='C2H6O' formula='C2H6O' hash='c900f12')>,
 <Molecule(name='C2H6O' formula='C2H6O' hash='a1e9d7a')>,
 <Molecule(name='C2H6O' formula='C2H6O' hash='422ad57')>]
```

1.6.5 From a TorsionDriveDataset

[28]:
```
import qcportal as ptl
client = ptl.FractalClient()

ds = client.get_collection("TorsionDriveDataset", "SMIRNOFF Coverage Torsion Set 1")
```

Show some available torsions:

[29]:
```
ds.df.head()
```

[29]:
```
Empty DataFrame
Columns: []
Index: [[CH3:1][O:2][CH:3]([OH:4])OC, [CH3:1][O:2][CH:3](O)[O:4]C, CO[CH:3]([OH:4])[O:
↪2][CH3:1], C[O:4][CH:3](O)[O:2][CH3:1], [H:4][C:3](O)([O:2][CH3:1])OC]
```

Show available specifications:

[30]:
```
ds.list_specifications()
```

[30]:
```
                                        Description
Name
default  Standard OpenFF torsiondrive specification.
```

Get a specific torsiondrive:

[31]:
```
td = ds.get_record("CO[CH:3]([OH:4])[O:2][CH3:1]", "default")
```

Get molecules for each angle along the torsion scan:

[32]:
```
td.get_final_molecules()
```

[32]:
```
{(-75,): <Molecule(name='C3H8O3' formula='C3H8O3' hash='60e16ca')>,
 (-90,): <Molecule(name='C3H8O3' formula='C3H8O3' hash='c337c03')>,
 (-60,): <Molecule(name='C3H8O3' formula='C3H8O3' hash='b4ff4d4')>,
 (-105,): <Molecule(name='C3H8O3' formula='C3H8O3' hash='5b05d3a')>,
 (-30,): <Molecule(name='C3H8O3' formula='C3H8O3' hash='8737c8f')>,
```

(continues on next page)

(continued from previous page)

```
 (-45,): <Molecule(name='C3H8O3' formula='C3H8O3' hash='240c817')>,
 (-120,): <Molecule(name='C3H8O3' formula='C3H8O3' hash='399d214')>,
 (0,): <Molecule(name='C3H8O3' formula='C3H8O3' hash='f1b0dd1')>,
 (-15,): <Molecule(name='C3H8O3' formula='C3H8O3' hash='05c30a0')>,
 (15,): <Molecule(name='C3H8O3' formula='C3H8O3' hash='f329f87')>,
 (-150,): <Molecule(name='C3H8O3' formula='C3H8O3' hash='1c56b54')>,
 (180,): <Molecule(name='C3H8O3' formula='C3H8O3' hash='d299528')>,
 (-165,): <Molecule(name='C3H8O3' formula='C3H8O3' hash='c81a1fc')>,
 (-135,): <Molecule(name='C3H8O3' formula='C3H8O3' hash='530c77d')>,
 (30,): <Molecule(name='C3H8O3' formula='C3H8O3' hash='99156ab')>,
 (150,): <Molecule(name='C3H8O3' formula='C3H8O3' hash='810b759')>,
 (45,): <Molecule(name='C3H8O3' formula='C3H8O3' hash='e1f13fa')>,
 (165,): <Molecule(name='C3H8O3' formula='C3H8O3' hash='df216e3')>,
 (60,): <Molecule(name='C3H8O3' formula='C3H8O3' hash='e69654b')>,
 (75,): <Molecule(name='C3H8O3' formula='C3H8O3' hash='5c12648')>,
 (135,): <Molecule(name='C3H8O3' formula='C3H8O3' hash='35f87a2')>,
 (90,): <Molecule(name='C3H8O3' formula='C3H8O3' hash='cdbfa17')>,
 (120,): <Molecule(name='C3H8O3' formula='C3H8O3' hash='5271be0')>,
 (105,): <Molecule(name='C3H8O3' formula='C3H8O3' hash='c0f46d7')>}
```

[33]:
```
td.get_final_molecules()[(30,)]
```

Data type cannot be displayed: application/3dmoljs_load.v0, text/html

[33]:
```
<Molecule(name='C3H8O3' formula='C3H8O3' hash='99156ab')>
```

You can run this notebook online in a session or view it on Github.

1.7 All calculations on a molecule

[1]:
```
import qcportal as ptl
client = ptl.FractalClient()

mol = client.query_molecules(2)[0]
mol
```

Data type cannot be displayed: application/3dmoljs_load.v0, text/html

[1]:
```
<Molecule(name='H6N2 ((1,),[])' formula='H3N' hash='5ad632d')>
```

Query the results table for all calculations performed on the molecule:

[2]:
```
ret = client.query_results(molecule=mol.id)
ret
```

[2]:
```
[<ResultRecord(id='735838' status='COMPLETE')>,
 <ResultRecord(id='735899' status='COMPLETE')>,
 <ResultRecord(id='970453' status='COMPLETE')>,
 <ResultRecord(id='970514' status='COMPLETE')>,
 <ResultRecord(id='608545' status='COMPLETE')>,
 <ResultRecord(id='735599' status='COMPLETE')>,
```

(continues on next page)

(continued from previous page)

```
 <ResultRecord(id='735489' status='COMPLETE')>,
 <ResultRecord(id='735550' status='COMPLETE')>,
 <ResultRecord(id='1847216' status='COMPLETE')>,
 <ResultRecord(id='1847316' status='COMPLETE')>,
 <ResultRecord(id='624' status='COMPLETE')>,
 <ResultRecord(id='663' status='COMPLETE')>,
 <ResultRecord(id='1846816' status='COMPLETE')>,
 <ResultRecord(id='1846916' status='COMPLETE')>,
 <ResultRecord(id='468' status='COMPLETE')>,
 <ResultRecord(id='507' status='COMPLETE')>,
 <ResultRecord(id='63' status='COMPLETE')>,
 <ResultRecord(id='208187' status='COMPLETE')>,
 <ResultRecord(id='208287' status='COMPLETE')>,
 <ResultRecord(id='1764410' status='COMPLETE')>,
 <ResultRecord(id='1819425' status='COMPLETE')>,
 <ResultRecord(id='1847016' status='COMPLETE')>,
 <ResultRecord(id='1847116' status='COMPLETE')>,
 <ResultRecord(id='546' status='COMPLETE')>,
 <ResultRecord(id='585' status='COMPLETE')>,
 <ResultRecord(id='618857' status='COMPLETE')>,
 <ResultRecord(id='618957' status='COMPLETE')>,
 <ResultRecord(id='621598' status='COMPLETE')>,
 <ResultRecord(id='621698' status='COMPLETE')>]
```

Inspect one of the results:

[3]:
```
r = ret[9]
print(f"Program: {r.program}\nMethod: {r.method}\nBasis: {r.basis}")
```

```
Program: psi4
Method: b2plyp
Basis: aug-cc-pvtz
```

View the output file:

[5]:
```
print(r.get_stdout())
```

```

  Memory set to  60.800 GiB by Python driver.

*** tstart() called on ca131
*** at Fri Jul 12 22:18:51 2019

   => Loading Basis Set <=

    Name: AUG-CC-PVTZ
    Role: ORBITAL
    Keyword: BASIS
    atoms 1   entry N          line   281 file /home/lnaden/miniconda3/envs/qca/share/
↪psi4/basis/aug-cc-pvtz.gbs
    atoms 2-4 entry H          line    36 file /home/lnaden/miniconda3/envs/qca/share/
↪psi4/basis/aug-cc-pvtz.gbs


         ---------------------------------------------------------
                                   SCF
               by Justin Turney, Rob Parrish, Andy Simmonett
                          and Daniel G. A. Smith
```

(continues on next page)

(continued from previous page)

```
                            RKS Reference
                        16 Threads,  62259 MiB Core
        ---------------------------------------------------------

  ==> Geometry <==

    Molecular point group: cs
    Full point group: Cs

    Geometry (in Bohr), charge = 0, multiplicity = 1:

       Center              X                  Y                   Z               Mass
    ------------   -----------------  -----------------  -----------------  -----------
↪-------
         N            0.000538830000     0.000000000000     0.129095460000    14.
↪003074004430
         H           -0.890229660000     1.529856130000    -0.596877300000     1.
↪007825032230
         H            1.772972570000     0.000000000000    -0.599942880000     1.
↪007825032230
         H           -0.890229660000    -1.529856130000    -0.596877300000     1.
↪007825032230

  Running in cs symmetry.

  Rotational constants: A =      9.98044  B =      9.90180  C =      6.34817 [cm^-1]
  Rotational constants: A = 299206.07647  B = 296848.42896  C = 190313.28552 [MHz]
  Nuclear repulsion =   11.947431780818354

  Charge       = 0
  Multiplicity = 1
  Electrons    = 10
  Nalpha       = 5
  Nbeta        = 5

  ==> Algorithm <==

  SCF Algorithm Type is DF.
  DIIS enabled.
  MOM disabled.
  Fractional occupation disabled.
  Guess Type is SAD.
  Energy threshold   = 1.00e-08
  Density threshold  = 1.00e-08
  Integral threshold = 0.00e+00

  ==> Primary Basis <==

  Basis Set: AUG-CC-PVTZ
    Blend: AUG-CC-PVTZ
    Number of shells: 41
    Number of basis function: 115
    Number of Cartesian functions: 130
    Spherical Harmonics?: true
    Max angular momentum: 3

  ==> DFT Potential <==
```

(continues on next page)

(continued from previous page)

```
  => Composite Functional: B2PLYP <=

   B2PLYP Double Hybrid Exchange-Correlation Functional

   S. Grimme, J. Chem. Phys., 124, 034108, 2006


   Deriv               =              1
   GGA                 =           TRUE
   Meta                =          FALSE

   Exchange Hybrid     =           TRUE
   MP2 Hybrid          =           TRUE

  => Exchange Functionals <=

   0.4700     XC_GGA_X_B88

  => Exact (HF) Exchange <=

   0.5300               HF

  => Correlation Functionals <=

   0.7300     XC_GGA_C_LYP

  => MP2 Correlation <=

   0.2700       MP2

  => Molecular Quadrature <=

   Radial Scheme          =       TREUTLER
   Pruning Scheme         =           FLAT
   Nuclear Scheme         =       TREUTLER

   BS radius alpha        =              1
   Pruning alpha          =              1
   Radial Points          =            100
   Spherical Points       =            302
   Total Points           =         120800
   Total Blocks           =            962
   Max Points             =            256
   Max Functions          =            115

  => Loading Basis Set <=

   Name: (AUG-CC-PVTZ AUX)
   Role: JKFIT
   Keyword: DF_BASIS_SCF
   atoms 1   entry N          line   224 file /home/lnaden/miniconda3/envs/qca/share/
↪psi4/basis/aug-cc-pvtz-jkfit.gbs
   atoms 2-4 entry H          line    70 file /home/lnaden/miniconda3/envs/qca/share/
↪psi4/basis/aug-cc-pvtz-jkfit.gbs

  ==> Pre-Iterations <==
```

(continues on next page)

(continued from previous page)

```
 -------------------------------------------------------
  Irrep   Nso     Nmo     Nalpha   Nbeta   Ndocc  Nsocc
 -------------------------------------------------------
   A'        69      69       0       0       0       0
   A"        46      46       0       0       0       0
 -------------------------------------------------------
  Total     115     115       5       5       5       0
 -------------------------------------------------------

==> Integral Setup <==

DFHelper Memory: AOs need 0.029 GiB; user supplied 45.214 GiB. Using in-core AOs.

==> MemDFJK: Density-Fitted J/K Matrices <==

  J tasked:                   Yes
  K tasked:                   Yes
  wK tasked:                   No
  OpenMP threads:              16
  Memory [MiB]:             46299
  Algorithm:                 Core
  Schwarz Cutoff:           1E-12
  Mask sparsity (%):       0.0000
  Fitting Condition:        1E-10

 => Auxiliary Basis Set <=

Basis Set: (AUG-CC-PVTZ AUX)
  Blend: AUG-CC-PVTZ-JKFIT
  Number of shells: 72
  Number of basis function: 242
  Number of Cartesian functions: 296
  Spherical Harmonics?: true
  Max angular momentum: 4

Cached 100.0% of DFT collocation blocks in 0.386 [GiB].

Minimum eigenvalue in the overlap matrix is 2.7046565184E-04.
Using Symmetric Orthogonalization.

SCF Guess: Superposition of Atomic Densities via on-the-fly atomic UHF.

==> Iterations <==

                        Total Energy        Delta E     RMS |[F,P]|

 @DF-RKS iter SAD:   -55.95900007653938   -5.59590e+01   0.00000e+00
 @DF-RKS iter   1:   -56.28269366355451   -3.23694e-01   7.97409e-03 DIIS
 @DF-RKS iter   2:   -56.25795337003415    2.47403e-02   8.17064e-03 DIIS
 @DF-RKS iter   3:   -56.46021265743074   -2.02259e-01   3.01786e-04 DIIS
 @DF-RKS iter   4:   -56.46058235120474   -3.69694e-04   4.20726e-05 DIIS
 @DF-RKS iter   5:   -56.46059082137342   -8.47017e-06   5.25979e-06 DIIS
 @DF-RKS iter   6:   -56.46059098813920   -1.66766e-07   8.96776e-07 DIIS
 @DF-RKS iter   7:   -56.46059099415442   -6.01521e-09   1.68815e-07 DIIS
 @DF-RKS iter   8:   -56.46059099435612   -2.01709e-10   1.85779e-08 DIIS
 @DF-RKS iter   9:   -56.46059099435834   -2.21689e-12   1.47837e-09 DIIS
```

(continues on next page)

(continued from previous page)

```
Energy and wave function converged.


==> Post-Iterations <==

  Orbital Energies [Eh]
  ---------------------

  Doubly Occupied:

     1Ap   -14.819149     2Ap    -0.976248     3Ap    -0.534045
     1App   -0.533186     4Ap    -0.339638

  Virtual:

     5Ap     0.005072     6Ap     0.034962     2App    0.035036
     7Ap     0.089759     8Ap     0.113558     3App    0.134620
     9Ap     0.134671    10Ap     0.169461     4App    0.169498
    11Ap     0.193429     5App    0.193675    12Ap     0.207388
    13Ap     0.306696     6App    0.319665    14Ap     0.330115
     7App    0.330183    15Ap     0.372804    16Ap     0.394143
     8App    0.394171     9App    0.426433    17Ap     0.426544
    18Ap     0.496383    19Ap     0.530978    10App    0.531450
    20Ap     0.658333    11App    0.692667    21Ap     0.692961
    22Ap     0.725184    23Ap     0.787072    12App    0.828871
    24Ap     0.837552    13App    0.838044    25Ap     0.891859
    14App    0.892131    26Ap     0.948626    15App    0.954762
    27Ap     0.954986    16App    1.004354    28Ap     1.020553
    17App    1.020716    29Ap     1.059845    18App    1.060092
    30Ap     1.097281    31Ap     1.134872    19App    1.135737
    32Ap     1.217908    33Ap     1.256436    20App    1.257636
    34Ap     1.496074    35Ap     1.634208    21App    1.635259
    22App    1.645091    36Ap     1.726835    37Ap     1.734060
    23App    1.734740    38Ap     1.869625    24App    1.894604
    39Ap     1.895043    25App    1.922060    40Ap     1.922836
    41Ap     2.121233    26App    2.214693    27App    2.232937
    42Ap     2.233925    28App    2.288278    43Ap     2.288892
    44Ap     2.313244    45Ap     3.184724    46Ap     3.357067
    29App    3.358506    47Ap     3.788945    30App    3.791293
    48Ap     3.804931    49Ap     3.957783    31App    3.958851
    32App    4.008523    33App    4.044058    50Ap     4.050021
    34App    4.052268    51Ap     4.134546    52Ap     4.153090
    35App    4.153732    53Ap     4.281663    54Ap     4.326653
    36App    4.328826    55Ap     4.479902    37App    4.535739
    56Ap     4.537064    57Ap     4.758429    38App    4.761317
    39App    4.806924    58Ap     4.846585    40App    4.847183
    59Ap     4.852472    60Ap     5.024442    41App    5.425674
    61Ap     5.428564    62Ap     5.493979    42App    5.650005
    63Ap     5.656241    64Ap     5.792429    43App    6.095473
    65Ap     6.100774    44App    6.175525    66Ap     6.371456
    45App    6.689084    67Ap     6.689587    68Ap     7.415202
    46App    7.422249    69Ap    16.716513

  Final Occupation by Irrep:
           Ap   App
  DOCC [     4,    1 ]
```

(continues on next page)

(continued from previous page)

```
  @DF-RKS Final Energy:   -56.46059099435834

   => Energetics <=

    Nuclear Repulsion Energy =             11.9474317808183539
    One-Electron Energy =                 -99.6482974230888345
    Two-Electron Energy =                  35.0831019748610800
    DFT Exchange-Correlation Energy =      -3.8428273269489335
    Empirical Dispersion Energy =           0.0000000000000000
    VV10 Nonlocal Energy =                  0.0000000000000000
    Total Energy =                        -56.4605909943583413

Computation Completed


Properties will be evaluated at   0.000000,   0.000000,   0.000000 [a0]

Properties computed using the SCF density matrix

  Nuclear Dipole Moment: [e a0]
     X:    -0.0037      Y:     0.0000      Z:    -0.8900

  Electronic Dipole Moment: [e a0]
     X:     0.0018      Y:     0.0000      Z:     0.2704

  Dipole Moment: [e a0]
     X:    -0.0020      Y:     0.0000      Z:    -0.6196     Total:     0.6196

  Dipole Moment: [D]
     X:    -0.0050      Y:     0.0000      Z:    -1.5749     Total:     1.5749


*** tstop() called on ca131 at Fri Jul 12 22:18:54 2019
Module time:
        user time   =      30.17 seconds =       0.50 minutes
        system time =       1.14 seconds =       0.02 minutes
        total time  =          3 seconds =       0.05 minutes
Total time:
        user time   =      30.17 seconds =       0.50 minutes
        system time =       1.14 seconds =       0.02 minutes
        total time  =          3 seconds =       0.05 minutes

*** tstart() called on ca131
*** at Fri Jul 12 22:18:54 2019

   => Loading Basis Set <=

    Name: (AUG-CC-PVTZ AUX)
    Role: RIFIT
    Keyword: DF_BASIS_MP2
    atoms 1   entry N          line   206 file /home/lnaden/miniconda3/envs/qca/share/
↪psi4/basis/aug-cc-pvtz-ri.gbs
    atoms 2-4 entry H          line    30 file /home/lnaden/miniconda3/envs/qca/share/
↪psi4/basis/aug-cc-pvtz-ri.gbs

         ---------------------------------------------------------
                                   DF-MP2
```

(continues on next page)

(continued from previous page)

```
             2nd-Order Density-Fitted Moller-Plesset Theory
                     RMP2 Wavefunction,  16 Threads

                Rob Parrish, Justin Turney, Andy Simmonett,
                   Ed Hohenstein, and C. David Sherrill
          ----------------------------------------------------------

   => Auxiliary Basis Set <=

  Basis Set: (AUG-CC-PVTZ AUX)
    Blend: AUG-CC-PVTZ-RI
    Number of shells: 70
    Number of basis function: 244
    Number of Cartesian functions: 301
    Spherical Harmonics?: true
    Max angular momentum: 4

          -------------------------------------------------------
                         NBF =   115, NAUX =   244
          -------------------------------------------------------
            CLASS    FOCC     OCC    AOCC    AVIR     VIR    FVIR
            PAIRS       1       5       4     110     110       0
          -------------------------------------------------------

        ---------------------------------------------------------
         ==================> DF-MP2 Energies <====================
        ---------------------------------------------------------
         Reference Energy          =     -56.4605909943583413 [Eh]
         Singles Energy            =      -0.0000000000000000 [Eh]
         Same-Spin Energy          =      -0.0643396155609966 [Eh]
         Opposite-Spin Energy      =      -0.2226985414819575 [Eh]
         Correlation Energy        =      -0.2870381570429541 [Eh]
         Total Energy              =     -56.7476291514012985 [Eh]
        ---------------------------------------------------------
         ================> DF-SCS-MP2 Energies <==================
        ---------------------------------------------------------
         SCS Same-Spin Scale       =       0.3333333333333333 [-]
         SCS Opposite-Spin Scale   =       1.2000000000000000 [-]
         SCS Same-Spin Energy      =      -0.0214465385203322 [Eh]
         SCS Opposite-Spin Energy  =      -0.2672382497783490 [Eh]
         SCS Correlation Energy    =      -0.2886847882986812 [Eh]
         SCS Total Energy          =     -56.7492757826570227 [Eh]
        ---------------------------------------------------------



    B2PLYP Energy Summary
    ---------------------
    DFT Reference Energy                  =   -56.4605909943583413
    Scaled MP2 Correlation                =    -0.0775003024015976
    @Final double-hybrid DFT total energy =   -56.5380912967599372


*** tstop() called on ca131 at Fri Jul 12 22:18:54 2019
Module time:
        user time   =       6.88 seconds =       0.11 minutes
        system time =       0.01 seconds =       0.00 minutes
```

(continues on next page)

(continued from previous page)

```
        total time  =          0 seconds =       0.00 minutes
Total time:
        user time   =      37.07 seconds =       0.62 minutes
        system time =       1.15 seconds =       0.02 minutes
        total time  =          3 seconds =       0.05 minutes
```

[ ]:

You can run this notebook online in a session or view it on Github.

1.8 Exploring Datasets and ReactionDatasets

When examining a `Dataset` or `ReactionDataset`, it can be helpful to see which fields and associated values are available. That is, this section will discuss how to list what basis sets, methods, programs, etc. are represented in a dataset's results.

[1]:
```
import qcportal as ptl
client = ptl.FractalClient()

ds = client.get_collection("ReactionDataset", "S22")
```

To see available search parameters:

[2]:
```
ds.list_values().reset_index().columns
```

[2]:
```
Index(['native', 'driver', 'program', 'method', 'basis', 'keywords',
       'stoichiometry', 'name'],
      dtype='object')
```

To see available search values (e.g. which basis sets and methods are available in the dataset):

[3]:
```
ds.list_values().reset_index()['method'].unique()
```

[3]:
```
array(['Unknown', 'b2plyp', 'b2plyp-d3', 'b2plyp-d3(bj)', 'b2plyp-d3m',
       'b2plyp-d3m(bj)', 'b3lyp', 'b3lyp-d3', 'b3lyp-d3(bj)', 'b3lyp-d3m',
       'b3lyp-d3m(bj)', 'hf', 'mp2', 'pbe', 'sapt0', 'wb97m-v', 'wb97x-d'],
      dtype=object)
```

[4]:
```
ds.list_values().reset_index()['basis'].unique()
```

[4]:
```
array(['Unknown', 'aug-cc-pvdz', 'aug-cc-pvtz', 'def2-svp', 'def2-tzvp',
       'sto-3g', 'jun-cc-pvdz'], dtype=object)
```

List combinations of method and basis set:

[5]:
```
ds.list_values().reset_index().groupby(['method','basis']).size().reset_index()
```

[5]:

	method	basis	0
0	Unknown	Unknown	3
1	b2plyp	aug-cc-pvdz	2
2	b2plyp	aug-cc-pvtz	2
3	b2plyp	def2-svp	2
4	b2plyp	def2-tzvp	2
5	b2plyp-d3	aug-cc-pvdz	2

(continues on next page)

(continued from previous page)

```
6         b2plyp-d3  aug-cc-pvtz  2
7         b2plyp-d3     def2-svp  2
8         b2plyp-d3    def2-tzvp  2
9     b2plyp-d3(bj)  aug-cc-pvdz  2
10    b2plyp-d3(bj)  aug-cc-pvtz  2
11    b2plyp-d3(bj)     def2-svp  2
12    b2plyp-d3(bj)    def2-tzvp  2
13       b2plyp-d3m  aug-cc-pvdz  2
14       b2plyp-d3m  aug-cc-pvtz  2
15       b2plyp-d3m     def2-svp  2
16       b2plyp-d3m    def2-tzvp  2
17  b2plyp-d3m(bj)  aug-cc-pvdz  2
18  b2plyp-d3m(bj)  aug-cc-pvtz  2
19  b2plyp-d3m(bj)     def2-svp  2
20  b2plyp-d3m(bj)    def2-tzvp  2
21            b3lyp  aug-cc-pvdz  2
22            b3lyp  aug-cc-pvtz  2
23            b3lyp     def2-svp  2
24            b3lyp    def2-tzvp  2
25         b3lyp-d3  aug-cc-pvdz  2
26         b3lyp-d3  aug-cc-pvtz  2
27         b3lyp-d3     def2-svp  2
28         b3lyp-d3    def2-tzvp  2
29     b3lyp-d3(bj)  aug-cc-pvdz  2
30     b3lyp-d3(bj)  aug-cc-pvtz  2
31     b3lyp-d3(bj)     def2-svp  2
32     b3lyp-d3(bj)    def2-tzvp  2
33        b3lyp-d3m  aug-cc-pvdz  2
34        b3lyp-d3m  aug-cc-pvtz  2
35        b3lyp-d3m     def2-svp  2
36        b3lyp-d3m    def2-tzvp  2
37   b3lyp-d3m(bj)  aug-cc-pvdz  2
38   b3lyp-d3m(bj)  aug-cc-pvtz  2
39   b3lyp-d3m(bj)     def2-svp  2
40   b3lyp-d3m(bj)    def2-tzvp  2
41              hf       sto-3g  2
42             mp2  aug-cc-pvdz  2
43             mp2  aug-cc-pvtz  2
44             mp2     def2-svp  2
45             mp2    def2-tzvp  2
46             pbe  aug-cc-pvdz  2
47             pbe  aug-cc-pvtz  2
48             pbe     def2-svp  2
49             pbe    def2-tzvp  2
50           sapt0  aug-cc-pvdz  1
51           sapt0  aug-cc-pvtz  1
52           sapt0  jun-cc-pvdz  1
53         wb97m-v     def2-svp  2
54         wb97m-v    def2-tzvp  2
55         wb97x-d     def2-svp  2
56         wb97x-d    def2-tzvp  2
```

Most datasets support default programs and keywords. For those cases, the above can also be achieved with:

[7]:
```
ds.list_values(native=True).reset_index()['name']
```

[7]:
```
0       cp-B2PLYP/aug-cc-pvdz
1          B2PLYP/aug-cc-pvdz
2       cp-B2PLYP/aug-cc-pvtz
3          B2PLYP/aug-cc-pvtz
4             B2PLYP/def2-svp
                ...
104         WB97M-V/def2-tzvp
105       cp-WB97X-D/def2-svp
106          WB97X-D/def2-svp
107         WB97X-D/def2-tzvp
108      cp-WB97X-D/def2-tzvp
Name: name, Length: 109, dtype: object
```

You can run this notebook online in a session or view it on Github.

1.9 Getting Started

In this example, we will show how to:

- Find existing ML datasets on the QCArchive.
- Extract geometries and quantum chemical results from the datasets.
- Evaluate ML models on existing datasets.
- Compute new data using your own instance of QCArchive.

The demonstration is organized into three ML examples:

- Unsupervised learning: use manifold learning to understand the structure of the QM7b, QM7b-T, and SN2 Reaction datasets.
- Supervised learning: train a kernel model to predict atomization energies with the ANI-1 dataset, and test it on a COMP6 benchmark.
- Dataset creation: Make a new dataset and compute DFT energy labels and neural network predictions.

1.9.1 Connect to the database

The Molecular Sciences Software Institute hosts the Quantum Chemistry Archive (QCArchive) and makes its data available to the entire Computational Molecular Sciences community free of charge. The QCArchive is both a database to view, analyze, and explore existing data as well as a live instance that continuously generates new data as directed by the community.

The primary interface to any QCArchive database is the the `qcportal` Python package which can be downloaded via pip (`pip install -e qcportal`) or conda (`conda install qcportal -c conda-forge`).

(Documentation: http://docs.qcarchive.molssi.org/projects/QCPortal/en/stable/)

The primary interface to a database server is a through a `FractalClient`. We can connect to api.qcarchive.molssi.org to get access to all data contained within the MolSSI server.

[2]:
```
import numpy as np
import pandas as pd

import qcportal as ptl

client = ptl.FractalClient(address="api.qcarchive.molssi.org")
client
```

[2]:
```
FractalClient(server_name='The MolSSI QCArchive Server', address='https://api.
↪qcarchive.molssi.org/', username='None')
```

1.9.2 Exploring collections

We organize datasets into "collections". QCArchive hosts many data collections, including benchmarks for electronic structure development, PES scans for force field fitting, in addition to the ML datasets discussed earlier. The `list_collections` function returns all of the collections on the server:

[3]:
```
client.list_collections()
```

[3]:
```
                                                                                ␣
↪                 tagline
collection          name
Dataset             ANI-1                                    22 million off-
↪equilibrium conformations and e...
                    COMP6 ANI-MD                             Benchmark containing MD␣
↪trajectories from the ...
                    COMP6 DrugBank                           Benchmark containing␣
↪DrugBank off-equilibrium ...
                    COMP6 GDB10to13                          Benchmark containing off-
↪equilibrium molecules...
                    COMP6 GDB7to9                            Benchmark containing off-
↪equilibrium molecules...
...                                                                             ␣
↪                      ...
TorsionDriveDataset OpenFF Primary TorsionDrive Benchmark 1                     ␣
↪                     None
                    OpenFF Substituted Phenyl Set 1                             ␣
↪                     None
                    Pfizer Discrepancy Torsion Dataset 1                        ␣
↪                     None
                    SMIRNOFF Coverage Torsion Set 1                             ␣
↪                     None
                    TorsionDrive Paper                                          ␣
↪                     None

[82 rows x 1 columns]
```

Here we focus on machine learning datasets, searching for collections with the "machine learning" tag:

[4]:
```
client.list_collections(tag="machine learning")
```

[4]:
```
                                                                                ␣
↪tagline
collection name
Dataset    ANI-1                      22 million off-equilibrium conformations and␣
↪e...
           COMP6 ANI-MD               Benchmark containing MD trajectories from the␣
↪...
```

(continues on next page)

(continued from previous page)

```
          COMP6 DrugBank              Benchmark containing DrugBank off-equilibrium␣
↪...
          COMP6 GDB10to13             Benchmark containing off-equilibrium␣
↪molecules...
          COMP6 GDB7to9               Benchmark containing off-equilibrium␣
↪molecules...
          COMP6 S66x8                           Benchmark for noncovalent␣
↪interactions.
          COMP6 Tripeptides           Benchmark containing off-equilibrium␣
↪geometrie...
          G-SchNet Generated           Molecules generated by G-SchNet, trained on␣
↪QM9.
          GDB13-T                     Small organic molecules with up to 13 heavy␣
↪at...
          GDML                        Molecular dynamics trajectories of small␣
↪molec...
          ISO-17                      Molecular dynamics trajectories of isomers of␣
↪...
          QM7                         Small organic molecules with up to 7 heavy␣
↪atoms.
          QM7b                        Small organic molecules with up to 7 heavy␣
↪ato...
          QM7b-T                      Small organic molecules with up to 7 heavy␣
↪ato...
          QM9                         Small organic molecules with up to 9 heavy␣
↪ato...
          SN2 Reactions               Chemical reactions of methyl halides with␣
↪hali...
          Solvated Protein Fragments             Amons derived from proteins, in␣
↪water.
```

These are the same as the ML datasets the QCArchive website

https://qcarchive.molssi.org/apps/ml_datasets

Name	Quality	Data Points	Elements	Sampling	Download
ANI-1	DFT	22,057,374	C H N O	NMS	HDF5
COMP6 ANI-MD	DFT	1,791	C H N O	MD 300K	HDF5 TEXT
COMP6 DrugBank	DFT	13,379	C H N O	DNMS	HDF5 TEXT
COMP6 GDB10to13	DFT	47,670	C H N O	DNMS	HDF5 TEXT
COMP6 GDB7to9	DFT	36,000	C H N O	DNMS	HDF5 TEXT
COMP6 S66x8	DFT	528	C H N O	PES scan	HDF5 TEXT
COMP6 Tripeptides	DFT	1,984	C H N O	DNMS	HDF5 TEXT
G-SchNet Generated	DFT	9,074	C H F N O	Minima	HDF5 TEXT
GDB13-T	HF, MP2	6,000	C H Cl N O ...	MD 350K	HDF5 TEXT
GDML	DFT, CCSD, CCSD(T)	3,875,468	C H N O	MD 500K	HDF5 TEXT
ISO-17	DFT	640,982	C H O	MD ???K	HDF5 TEXT
QM7	DFT	7,165	C H N O S	Minima	HDF5 TEXT

Showing 1 to 12 of 17 entries

Looking at the QM7b dataset

Collections can be obtained from the server with `get_collection`. A collection object is light-weight, initially containing only metadata; extremely large datasets (such as ANI-1) can be pulled in a few seconds. For this example, we will start with the QM7b dataset. To obtain this collection:

[6]:
```
qm7b = client.get_collection("dataset", "qm7b")
print_info(qm7b)
```

```
Name: QM7b

Data Points: 7211
Elements: ['C', 'H', 'Cl', 'N', 'O', 'S']
Labels: ['atomization energy', 'excitation energy', 'lumo', 'ionization potential',
↪'electron affinity', 'polarizability', 'absorption intensity', 'homo']
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

Getting molecules

Datasets contain two types of data:

- Molecules: Representations of molecules, containing atoms, geometry, charge, spin, fragments, etc.
- Values: Properties of those molecules, such as the B3LYP/6-31G* energy

We can access the molecules in a dataset with the `get_molecules` function. This function returns a pandas `DataFrame` of `Molecule` objects.

[7]:
```
qm7b_mols = qm7b.get_molecules()
qm7b_mols
```

[7]:
```
                                                molecule
index
0           Geometry (in Angstrom), charge = 0.0, mult...
1           Geometry (in Angstrom), charge = 0.0, mult...
10          Geometry (in Angstrom), charge = 0.0, mult...
100         Geometry (in Angstrom), charge = 0.0, mult...
1000        Geometry (in Angstrom), charge = 0.0, mult...
...                                                    ...
995         Geometry (in Angstrom), charge = 0.0, mult...
996         Geometry (in Angstrom), charge = 0.0, mult...
997         Geometry (in Angstrom), charge = 0.0, mult...
998         Geometry (in Angstrom), charge = 0.0, mult...
999         Geometry (in Angstrom), charge = 0.0, mult...

[7211 rows x 1 columns]
```

Visualizing molecules

Individual `Molecule` objects can be directly displayed in a Jupyter notebook:

[8]:
```
qm7b_mols["molecule"][100]  # get the element in column "molecule", row 100
```

```
_ColormakerRegistry()
```

```
NGLWidget()
```

Listing values

The `list_values` function shows what data are available in a collection. QM7b is fairly data rich, containing ground and excited state properties at the ZINDO, DFT, and GW levels.

[9]:
```
qm7b.list_values()
```

[9]:
```
                                                                    keywords  \
native driver         program  method                    basis
False  e1             Orca     ZINDO                     Unknown          Unknown
       ea             Orca     ZINDO/s                   Unknown          Unknown
       emax           Orca     ZINDO                     Unknown          Unknown
       energy         FHI-aims pbe0                      Unknown          Unknown
       homo           FHI-aims GW                        Unknown          Unknown
                               pbe0                      Unknown          Unknown
                      Orca     ZINDO/s                   Unknown          Unknown
       imax           Orca     ZINDO                     Unknown          Unknown
       ip             Orca     ZINDO/s                   Unknown          Unknown
       lumo           FHI-aims GW                        Unknown          Unknown
                               pbe0                      Unknown          Unknown
                      Orca     ZINDO/s                   Unknown          Unknown
       polarizability FHI-aims Self-consistent screening Unknown          Unknown
                               pbe0                      Unknown          Unknown
True   energy         psi4     b2plyp                    aug-cc-pvdz  scf_default
                                                         aug-cc-pvtz  scf_default
                                                         def2-svp     scf_default
                                                         def2-tzvp    scf_default
```

(continues on next page)

(continued from previous page)

```
                                                    sto-3g       scf_default
                            b3lyp                   aug-cc-pvdz  scf_default
                                                    aug-cc-pvtz  scf_default
                                                    def2-svp     scf_default
                                                    def2-tzvp    scf_default
                                                    sto-3g       scf_default
                            wb97m-v                 aug-cc-pvdz  scf_default
                                                    aug-cc-pvtz  scf_default
                                                    def2-svp     scf_default
                                                    def2-tzvp    scf_default
                                                    sto-3g       scf_default


↪                          name                                                 ␣
native driver         program  method                basis
False  e1             Orca     ZINDO                 Unknown                     ␣
↪First excitation energy (ZINDO)
       ea             Orca     ZINDO/s               Unknown                     ␣
↪    Electron affinity (ZINDO/s)
       emax           Orca     ZINDO                 Unknown      Excitation␣
↪energy at maximal absorption (ZINDO)
       energy         FHI-aims pbe0                  Unknown                     ␣
↪  Atomization energy (DFT/PBE0)
       homo           FHI-aims GW                    Unknown              Highest␣
↪occupied molecular orbital (GW)
                              pbe0                    Unknown            Highest␣
↪occupied molecular orbital (PBE0)
                      Orca     ZINDO/s               Unknown         Highest␣
↪occupied molecular orbital (ZINDO/s)
       imax           Orca     ZINDO                 Unknown                 ␣
↪Maximal absorption intensity (ZINDO)
       ip             Orca     ZINDO/s               Unknown                     ␣
↪ Ionization potential (ZINDO/s)
       lumo           FHI-aims GW                    Unknown             Lowest␣
↪unoccupied molecular orbital (GW)
                              pbe0                    Unknown           Lowest␣
↪unoccupied molecular orbital (PBE0)
                      Orca     ZINDO/s               Unknown        Lowest␣
↪unoccupied molecular orbital (ZINDO/s)
       polarizability FHI-aims Self-consistent screening Unknown          ␣
↪Polarizability (self-consistent screening)
                              pbe0                    Unknown                     ␣
↪      Polarizability (DFT/PBE0)
True   energy         psi4     b2plyp                aug-cc-pvdz                 ␣
↪             B2PLYP/aug-cc-pvdz
                                                     aug-cc-pvtz                 ␣
↪             B2PLYP/aug-cc-pvtz
                                                     def2-svp                    ␣
↪                B2PLYP/def2-svp
                                                     def2-tzvp                   ␣
↪              B2PLYP/def2-tzvp
                                                     sto-3g                      ␣
↪                 B2PLYP/sto-3g
                              b3lyp                  aug-cc-pvdz                 ␣
↪             B3LYP/aug-cc-pvdz
                                                     aug-cc-pvtz                 ␣
↪             B3LYP/aug-cc-pvtz
```

(continues on next page)

(continued from previous page)

```
                                                    def2-svp                ␣
↪                  B3LYP/def2-svp
                                                    def2-tzvp               ␣
↪                 B3LYP/def2-tzvp
                                                    sto-3g                  ␣
↪                    B3LYP/sto-3g
                              wb97m-v               aug-cc-pvdz             ␣
↪            WB97M-V/aug-cc-pvdz
                                                    aug-cc-pvtz             ␣
↪            WB97M-V/aug-cc-pvtz
                                                    def2-svp                ␣
↪               WB97M-V/def2-svp
                                                    def2-tzvp               ␣
↪              WB97M-V/def2-tzvp
                                                    sto-3g                  ␣
↪                WB97M-V/sto-3g
```

Getting values

The `get_values` function pulls a data column down from the server. Values may be filtered by any of the fields described in `list_values`, including `driver`, `program`, `method`, `basis`, and `name`. Here, we show all calculation performed with the B3LYP functional.

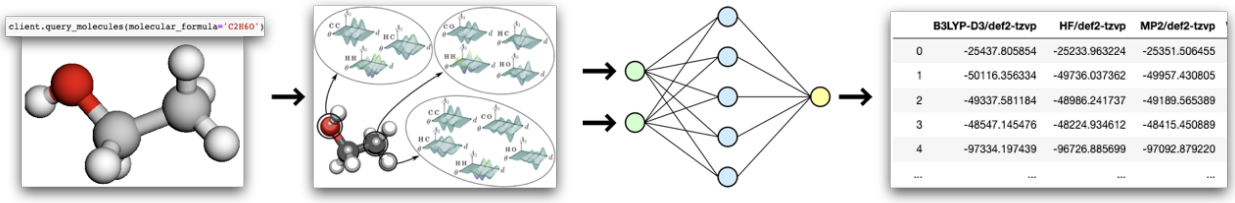
[10]:
```
qm7b.units = "hartree"
qm7b.get_values(method="b3lyp")
```

[10]:
```
     B3LYP/sto-3g B3LYP/aug-cc-pvdz B3LYP/def2-tzvp B3LYP/def2-svp  \
0         -40.0392          -40.5206        -40.5375       -40.4878
1         -78.8861          -79.8361        -79.8638       -79.7717
10        -131.067          -132.771         -132.81       -132.655
100       -207.419           -210.09        -210.147       -209.908
1000      -281.606          -285.326        -285.403       -285.072
...            ...               ...             ...            ...
995        -282.86          -286.575        -286.651        -286.32
996       -282.895          -286.656        -286.734       -286.405
997       -282.861          -286.576        -286.652       -286.321
998       -284.106          -287.805        -287.882       -287.549
999       -284.143          -287.884        -287.963       -287.633

     B3LYP/aug-cc-pvtz
0             -40.5383
1             -79.8645
10            -132.808
100           -210.146
1000            -285.4
...                ...
995           -286.649
996           -286.731
997            -286.65
998           -287.881
999           -287.961

[7211 rows x 5 columns]
```

Summary of QCPortal commands



QCPortal is a Python interface to ML data hosted by MolSSI's QCArchive. Accessing the data requires only five commands.

```
client = ptl.FractalClient()
ds = client.get_collection("dataset", name)
ds.get_molecules()
ds.list_values()
ds.get_values()
```

1.9.3 Extras

[ ]:
```
from IPython.core.display import HTML

def print_info(dataset):
    print(f"Name: {dataset.data.name}")
    print()
    print(f"Data Points: {dataset.data.metadata['data_points']}")
    print(f"Elements: {dataset.data.metadata['elements']}")
    print(f"Labels: {dataset.data.metadata['labels']}")

    display(HTML("<u>Description:</u> " + dataset.data.description))

    for cite in dataset.data.metadata["citations"]:
        display(HTML(cite['acs_citation']))
```

You can run this notebook online in a session or view it on Github.

1.10 (Un)Supervised ML Examples

The notebook provides two ML examples:

- Unsupervised learning: use manifold learning to understand the structure of the QM7b, QM7b-T, and SN2 Reaction datasets.
- Supervised learning: train a kernel model to predict atomization energies with the ANI-1 dataset, and test it on a COMP6 benchmark.

Begin by connecting to the MolSSI server:

[3]:
```
import numpy as np
import pandas as pd
```

(continues on next page)

(continued from previous page)

```
import qcportal as ptl

client = ptl.FractalClient()
```

1.10.1 Unsupervised learning: manifold learning with QM7b, QM7b-T, and SN2 Reactions

Manifold learning is a strategy for exposing the structure of high-dimensional datasets through non-linear dimensionality reduction. In this example, we will look at the similarities and differences within and between three ML datasets. Molecules will be fingerprinted by Coulomb matrix features, and manifold learning will be performed with the Uniform Manifold Approximation and Projection (UMAP) method.

https://umap-learn.readthedocs.io/en/latest/

[4]:
```
qm7b = client.get_collection("dataset", "qm7b")
print_info(qm7b)
```

```
Name: QM7b

Data Points: 7211
Elements: ['C', 'H', 'Cl', 'N', 'O', 'S']
Labels: ['atomization energy', 'excitation energy', 'lumo', 'ionization potential',
↪'electron affinity', 'polarizability', 'absorption intensity', 'homo']
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

Generating coulomb matrix features

`Molecule` objects contain information about atomic symbols/numbers, geometry, charge, spin, fragments, etc. As an example of using this information, the `coulomb_matrix` function below computes the (sorted) Coulomb matrix features corresponding to a `Molecule`. The Coulomb matrix is given by:

Coulomb matrix features are not the most sophisticated, but they are simple to implement:

[5]:
```
import qcelemental
scale = qcelemental.constants.conversion_factor("bohr", "angstrom") * np.sqrt(2)

def coulomb_matrix(mol, size=23):
    """
```

(continues on next page)

(continued from previous page)

```
    Compute the sorted Coulomb matrix features corresponding to a molecules

    Parameters
    ----------
    mol: Molecule
        the molecule whose features are to be computed
    size: int, optional
        maximum dimension of the Coulomb matrix
    """
    natoms = len(mol.atomic_numbers)

    # Create the Coulomb matrix
    M = np.zeros((natoms, natoms))
    for i in range(natoms):
        M[i,i] = 0.5 * mol.atomic_numbers[i]**2.4
        for j in range(0,i):
            M[i,j] = mol.atomic_numbers[i] * mol.atomic_numbers[j] / \
                np.linalg.norm(mol.geometry[i,:] - mol.geometry[j,:] * scale )
            M[j,i] = M[i,j]

    # Sort based on row norm, and zero-pad or truncate to size
    order = np.argsort(np.linalg.norm(M, axis=0))[::-1][:size]
    ret = np.zeros((size, size))
    for i in range(min(natoms, size)):
        for j in range(min(natoms, size)):
            ret[i,j] = M[order[i], order[j]]

    # Flatten upper triangle to 1D
    return ret[np.triu_indices(size)]
```

Compute the Coulomb matrix feature for each molecule. The code below creates a new column in the `DataFrame` containing the features.

[6]:
```
qm7b_mols = qm7b.get_molecules()
qm7b_mols["feature"] = [coulomb_matrix(molecule) for molecule in qm7b_mols["molecule
↪"]]
qm7b_mols
```

[6]:
```
                                                molecule  \
index
0           Geometry (in Angstrom), charge = 0.0, mult...
1           Geometry (in Angstrom), charge = 0.0, mult...
10          Geometry (in Angstrom), charge = 0.0, mult...
100         Geometry (in Angstrom), charge = 0.0, mult...
1000        Geometry (in Angstrom), charge = 0.0, mult...
...                                                    ...
995         Geometry (in Angstrom), charge = 0.0, mult...
996         Geometry (in Angstrom), charge = 0.0, mult...
997         Geometry (in Angstrom), charge = 0.0, mult...
998         Geometry (in Angstrom), charge = 0.0, mult...
999         Geometry (in Angstrom), charge = 0.0, mult...

                                                  feature
index
0       [36.85810519942594, 3.0602011490558505, 3.0553...
1       [36.85810519942594, 10.707116135022035, 3.0751...
10      [53.3587073998281, 12.483520879461704, 7.74603...
```

(continues on next page)

(continued from previous page)

```
100    [53.3587073998281, 7.416502178204354, 10.61293...
1000   [73.51669471981023, 10.524193502952423, 4.7371...
...                                                  ...
995    [73.51669471981023, 12.945983223622989, 6.2342...
996    [73.51669471981023, 17.492583403086307, 17.890...
997    [73.51669471981023, 13.44823763751797, 8.96043...
998    [73.51669471981023, 13.374609104346993, 8.7642...
999    [73.51669471981023, 12.325415318305255, 13.542...

[7211 rows x 2 columns]
```

Manifold learning with UMAP

Often in ML, features live in a high-dimensional space, but may have low dimensional structure. To examine this, we use UMAP to reduce the Coulomb matrix features of QM7b onto two dimensions:

[7]:
```
import umap
import warnings; warnings.simplefilter('ignore')  # supress numba warnings

reducer = umap.UMAP()
X = np.vstack(qm7b_mols["feature"])  # format features into matrix for UMAP
embedding = reducer.fit_transform(X)
print(f"Feature matrix shape: {X.shape}")
print(f"UMAP embedding shape: {embedding.shape}")
```

```
Feature matrix shape: (7211, 276)
UMAP embedding shape: (7211, 2)
```

[8]:
```
import plotly.express as px

qm7b_mols["Embedding Dim. 1"] = embedding[:, 0]
qm7b_mols["Embedding Dim. 2"] = embedding[:, 1]

fig = px.scatter(qm7b_mols, x="Embedding Dim. 1", y="Embedding Dim. 2", title="UMAP␣
↪Embedding of QM7b")
fig.show()
```

Data type cannot be displayed: application/vnd.plotly.v1+json, text/html

Coloring by unique heavy atoms shows that Coulomb matrix features cluster according to chemical element. This leads to a lack of "alchemical transferability", a know deficiency of these features.

[9]:
```
qm7b_mols["Heavy Atoms"] = [frozenset(set(mol.symbols)-set(['H'])) for mol in qm7b_
↪mols["molecule"]]

fig = px.scatter(qm7b_mols, x="Embedding Dim. 1", y="Embedding Dim. 2", color="Heavy␣
↪Atoms",
                 title="UMAP Embedding of QM7b, Colored by Elements")
fig.show()
```

Data type cannot be displayed: application/vnd.plotly.v1+json, text/html

Combining multiple datasets

Because all data in the QCArchive follow the same format, we can easily combine data from multiple datasets. Below, we use UMAP to compare two additional datasets to QM7b.

QM7b-T is chemically similar to QM7b, containing thermalized geometries corresponding to the optimized geometries in QM7b:

[10]:
```
qm7bT = client.get_collection("Dataset", "QM7b-T")
print_info(qm7bT)
```

```
Name: QM7b-T

Data Points: 7211
Elements: ['C', 'H', 'Cl', 'N', 'O', 'S']
Labels: ['energy']
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

The SN2 Reactions dataset is different to QM7b, containing different chemical elements as well as ions:

[11]:
```
sn2 = client.get_collection("Dataset", "SN2 Reactions")
print_info(sn2)
```

```
Name: SN2 Reactions

Data Points: 452709
Elements: ['C', 'H', 'Br', 'Cl', 'F', 'I']
Labels: ['energy', 'gradient', 'dipole']
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

As before, molecules are obtained from the datasets with `get_molecules`, and Coulomb matrix features corresponding to those molecules are generated. For the case of SN2, 2,000 molecules are sampled from a total of 452,709.

[12]:
```
qm7bT_mols = qm7bT.get_molecules()
qm7bT_mols["feature"] = [coulomb_matrix(molecule) for molecule in qm7bT_mols["molecule
↪"]]

index = sn2.get_entries().sample(n=2000, axis=0)
sn2_mols = sn2.get_molecules(subset=list(index["name"]))
sn2_mols["feature"] = [coulomb_matrix(molecule) for molecule in sn2_mols["molecule"]]
```

An example molecule from the SN2 Reactions dataset:

[13]:
```
sn2_mols["molecule"][15]
```

```
_ColormakerRegistry()
```

```
NGLWidget()
```

Comibine all of the data into one `DataFrame`:

[14]:
```
qm7b_mols["Dataset"] = "QM7b"
qm7bT_mols["Dataset"] = "QM7b-T"
sn2_mols["Dataset"] = "SN2"
all_mols = qm7b_mols.append(qm7bT_mols, sort=False).append(sn2_mols, sort=False)
```

And repeat the UMAP process:

[15]:
```
reducer = umap.UMAP()
X = np.vstack(all_mols["feature"])
embedding = reducer.fit_transform(X)

all_mols["Embedding Dim. 1"] = embedding[:, 0]
all_mols["Embedding Dim. 2"] = embedding[:, 1]

fig = px.scatter(all_mols, x="Embedding Dim. 1", y="Embedding Dim. 2", color="Dataset
↪",
                 title="UMAP Embedding of QM7b, QM7b-T, and SN2, Colored by Dataset")
fig.show()
```

Data type cannot be displayed: application/vnd.plotly.v1+json, text/html

We see that the QM7b and QM7b-T datasets overlap in the space of Coulomb features, no suprise given their similarity. The SN2 Reaction dataset, comprised of different chemical elements, has no overlap with the other two datasets.

1.10.2 Supervised learning: training a kernel model to predict DFT energies

ANI-1 is a dataset of over 22 million calculations:

[16]:
```
ani1 = client.get_collection("Dataset", "ANI-1")
print_info(ani1)
```

```
Name: ANI-1

Data Points: 22057374
Elements: ['C', 'H', 'N', 'O']
Labels: ['energy']
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

Creating a test and training set

Because it is so large (10 GB), we only want to pull down a subset of ANI-1's data. We start by getting the `index`, the list of the names of the 22 million entries in ANI-1.

[ ]:
```
ani_index = ani1.get_index()
```

The first 10 names are:

[17]:
```
ani_index[:10]
```

[17]:
```
['gdb11_s01-0-0',
 'gdb11_s01-0-1',
 'gdb11_s01-0-2',
 'gdb11_s01-0-3',
 'gdb11_s01-0-4',
 'gdb11_s01-0-5',
 'gdb11_s01-0-6',
 'gdb11_s01-0-7',
 'gdb11_s01-0-8',
 'gdb11_s01-0-9']
```

We will construct a training set of 1,000 molecules and a test set of 100 molecules.

[18]:
```
import sklearn.model_selection

index_train, index_test = sklearn.model_selection.train_test_split(ani_index,
                                                                    train_size=1000,
                                                                    test_size=100)
```

Obtaining molecules and features

By default, `get_molecules` returns all molecules in a dataset. The `subset` option allows us to only return a subset of the total data.

[19]:
```
mols_train = ani1.get_molecules(subset=index_train)
mols_test  = ani1.get_molecules(subset=index_test)

mols_train
```

[19]:
```
                                                      molecule
index
gdb11_s08-1065-136       Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s04-23-9088        Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s08-37758-164      Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s07-1736-599       Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s07-2195-15        Geometry (in Angstrom), charge = 0.0, mult...
...                                                                ...
gdb11_s08-21374-45       Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s05-193-1649       Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s05-57-5967        Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s07-3769-379       Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s08-27091-83       Geometry (in Angstrom), charge = 0.0, mult...

[1000 rows x 1 columns]
```

[33]:
```
mols_train["molecule"][321]
```

```
NGLWidget()
```

Construct Coulomb matrix features in the same manner as in the previous example:

[21]:
```
mols_train["feature"] = [coulomb_matrix(molecule) for molecule in mols_train["molecule
↪"]]
mols_test["feature"]  = [coulomb_matrix(molecule) for molecule in mols_test["molecule
↪"]]

mols_train
```

[21]:
```
                                                        molecule  \
index
gdb11_s08-1065-136       Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s04-23-9088        Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s08-37758-164      Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s07-1736-599       Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s07-2195-15        Geometry (in Angstrom), charge = 0.0, mult...
...                                                                ...
gdb11_s08-21374-45       Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s05-193-1649       Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s05-57-5967        Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s07-3769-379       Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s08-27091-83       Geometry (in Angstrom), charge = 0.0, mult...

                                                             feature
index
gdb11_s08-1065-136   [53.3587073998281, 8.908692425554532, 8.874621...
gdb11_s04-23-9088    [73.51669471981023, 22.67212602609652, 12.0400...
gdb11_s08-37758-164  [53.3587073998281, 8.082911344753562, 8.080247...
gdb11_s07-1736-599   [73.51669471981023, 13.567652117139643, 8.6530...
gdb11_s07-2195-15    [73.51669471981023, 16.296006930193187, 12.602...
...                                                                ...
gdb11_s08-21374-45   [53.3587073998281, 19.052463779247958, 5.95931...
gdb11_s05-193-1649   [73.51669471981023, 15.580992353082049, 9.3382...
gdb11_s05-57-5967    [73.51669471981023, 22.423817905692996, 24.156...
gdb11_s07-3769-379   [73.51669471981023, 11.152141192626104, 11.212...
gdb11_s08-27091-83   [53.3587073998281, 23.5715179759266, 30.188533...

[1000 rows x 2 columns]
```

Obtaining properties and labels

For supervised learning, we need labels in addition to features. The `list_values` function describes what properties are available. For the case of the ANI-1 dataset, the only property is the B97x/6-31(d) energy.

[22]:
```
ani1.list_values()
```

[22]:
```
                                    keywords                  name
native driver program  method basis
False  energy Gaussian B97x  6-31(d)  Unknown  B97x/6-31(d) Energy
```

The `get_values` function pulls a data column down from the server. Values may be filtered by any of the fields described in `list_values`, including `driver`, `program`, `method`, `basis`, and `name`. Just like in `get_molecules`, the `subset` option allows us to only return a subset of the total data.

[23]:
```
ani1.units = "hartree"
values_train = ani1.get_values(name="B97x/6-31(d) Energy", subset=index_train)
values_test  = ani1.get_values(name="B97x/6-31(d) Energy", subset=index_test)

values_train
```

[23]:
```
                     B97x/6-31(d) Energy
gdb11_s08-1065-136              -330.276
gdb11_s04-23-9088               -193.028
gdb11_s08-37758-164             -326.598
gdb11_s07-1736-599              -377.689
```

(continues on next page)

(continued from previous page)

```
gdb11_s07-2195-15               -341.565
...                                  ...
gdb11_s08-21374-45              -361.028
gdb11_s05-193-1649              -264.398
gdb11_s05-57-5967               -232.342
gdb11_s07-3769-379              -326.965
gdb11_s08-27091-83                -391.9

[1000 rows x 1 columns]
```

Combine the molecules, feautres and energies into a single `DataFrame`:

[24]:
```
data_train = pd.merge(values_train, mols_train, left_index=True, right_index=True)
data_test  = pd.merge(values_test,  mols_test,  left_index=True, right_index=True)

data_train
```

[24]:
```
                     B97x/6-31(d) Energy  \
gdb11_s08-1065-136              -330.276
gdb11_s04-23-9088               -193.028
gdb11_s08-37758-164             -326.598
gdb11_s07-1736-599              -377.689
gdb11_s07-2195-15               -341.565
...                                  ...
gdb11_s08-21374-45              -361.028
gdb11_s05-193-1649              -264.398
gdb11_s05-57-5967               -232.342
gdb11_s07-3769-379              -326.965
gdb11_s08-27091-83                -391.9

                                                    molecule  \
gdb11_s08-1065-136       Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s04-23-9088        Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s08-37758-164      Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s07-1736-599       Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s07-2195-15        Geometry (in Angstrom), charge = 0.0, mult...
...                                                               ...
gdb11_s08-21374-45       Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s05-193-1649       Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s05-57-5967        Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s07-3769-379       Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s08-27091-83       Geometry (in Angstrom), charge = 0.0, mult...

                                                     feature
gdb11_s08-1065-136   [53.3587073998281, 8.908692425554532, 8.874621...
gdb11_s04-23-9088    [73.51669471981023, 22.67212602609652, 12.0400...
gdb11_s08-37758-164  [53.3587073998281, 8.082911344753562, 8.080247...
gdb11_s07-1736-599   [73.51669471981023, 13.567652117139643, 8.6530...
gdb11_s07-2195-15    [73.51669471981023, 16.296006930193187, 12.602...
...                                                                ...
gdb11_s08-21374-45   [53.3587073998281, 19.052463779247958, 5.95931...
gdb11_s05-193-1649   [73.51669471981023, 15.580992353082049, 9.3382...
gdb11_s05-57-5967    [73.51669471981023, 22.423817905692996, 24.156...
gdb11_s07-3769-379   [73.51669471981023, 11.152141192626104, 11.212...
gdb11_s08-27091-83   [53.3587073998281, 23.5715179759266, 30.188533...

[1000 rows x 3 columns]
```

For this example, our labels will be atomization energies:

[25]:
```
def free_atom_energy(molecule):
    """ Returns the energy of the atoms in the molecule separated to infinity."""

    free_atom_ref = {"H": -0.500607632585, "C": -37.8302333826, "N": -54.5680045287,
↪"O": -75.0362229210}
    return sum((free_atom_ref[atom] for atom in molecule.symbols))

data_train["label"] = [data_train.loc[idx, "B97x/6-31(d) Energy"] -
                       free_atom_energy(data_train.loc[idx, "molecule"])
                       for idx in data_train.index]
data_test["label"]  = [data_test.loc[idx, "B97x/6-31(d) Energy"] -
                       free_atom_energy(data_test.loc[idx, "molecule"])
                       for idx in data_test.index]

data_train
```

[25]:
```
                     B97x/6-31(d) Energy  \
gdb11_s08-1065-136              -330.276
gdb11_s04-23-9088               -193.028
gdb11_s08-37758-164             -326.598
gdb11_s07-1736-599              -377.689
gdb11_s07-2195-15               -341.565
...                                  ...
gdb11_s08-21374-45              -361.028
gdb11_s05-193-1649              -264.398
gdb11_s05-57-5967               -232.342
gdb11_s07-3769-379              -326.965
gdb11_s08-27091-83                -391.9

                                                      molecule  \
gdb11_s08-1065-136       Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s04-23-9088        Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s08-37758-164      Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s07-1736-599       Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s07-2195-15        Geometry (in Angstrom), charge = 0.0, mult...
...                                                               ...
gdb11_s08-21374-45       Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s05-193-1649       Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s05-57-5967        Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s07-3769-379       Geometry (in Angstrom), charge = 0.0, mult...
gdb11_s08-27091-83       Geometry (in Angstrom), charge = 0.0, mult...

                                                       feature  \
gdb11_s08-1065-136   [53.3587073998281, 8.908692425554532, 8.874621...
gdb11_s04-23-9088    [73.51669471981023, 22.67212602609652, 12.0400...
gdb11_s08-37758-164  [53.3587073998281, 8.082911344753562, 8.080247...
gdb11_s07-1736-599   [73.51669471981023, 13.567652117139643, 8.6530...
gdb11_s07-2195-15    [73.51669471981023, 16.296006930193187, 12.602...
...                                                               ...
gdb11_s08-21374-45   [53.3587073998281, 19.052463779247958, 5.95931...
gdb11_s05-193-1649   [73.51669471981023, 15.580992353082049, 9.3382...
gdb11_s05-57-5967    [73.51669471981023, 22.423817905692996, 24.156...
gdb11_s07-3769-379   [73.51669471981023, 11.152141192626104, 11.212...
gdb11_s08-27091-83   [53.3587073998281, 23.5715179759266, 30.188533...

                        label
```

(continues on next page)

(continued from previous page)

```
gdb11_s08-1065-136  -3.387656
gdb11_s04-23-9088   -1.497720
gdb11_s08-37758-164 -2.712971
gdb11_s07-1736-599  -1.985845
gdb11_s07-2195-15   -2.066852
...                       ...
gdb11_s08-21374-45  -2.665909
gdb11_s05-193-1649  -1.562118
gdb11_s05-57-5967   -1.980084
gdb11_s07-3769-379  -2.702803
gdb11_s08-27091-83  -2.065380

[1000 rows x 4 columns]
```

Training and evaluating a model

With features and lables in hand, we're ready to train a model. We use the `GPy` package to train a Gaussian Process Regression (`GPRegression`) model using the Radial Basis Function (`RBF`) kernel.

[26]:
```
import GPy
X_train = np.vstack(data_train["feature"])
X_test  = np.vstack(data_test["feature"])
y_train = np.array(data_train["label"])[:,np.newaxis]

kernel = GPy.kern.RBF(
            input_dim=X_train.shape[1])
model = GPy.models.GPRegression(
    X_train,
    y_train,
    kernel=kernel)
model.optimize()

model
```

[26]:
```
<GPy.models.gp_regression.GPRegression at 0x14d91a2e8>
```

Using this model, we make predictions for the test and training sets, and evaluate the errors corresponding to those predictions.

[27]:
```
data_train["prediction"] = model.predict(X_train)[0][:,0]
data_test["prediction"]  = model.predict(X_test)[0][:,0]

data_train["Unsigned Error (Hartree)"] = np.abs(data_train["label"] - data_train[
↪"prediction"])
data_test["Unsigned Error (Hartree)"] = np.abs(data_test["label"] - data_test[
↪"prediction"])
```

Finally, plot the error distributions:

[28]:
```
data_train["Dataset"] = "ANI-1 Training Set"
data_test["Dataset"] = "ANI-1 Test Set"
data_joined = data_train.append(data_test)

fig = px.violin(data_joined, y="Unsigned Error (Hartree)", x="Dataset", box=True,
                title="Error Distribution for our ML Model")
fig.show()
```

Data type cannot be displayed: application/vnd.plotly.v1+json, text/html

Test on the COMP6 S66x8 benchmark

Again taking advantage of the common data format on QCArchive, we can easily test our model on a different dataset. Here, we use the COMP6 S66x8 benchmark set:

[29]:
```
comp6 = client.get_collection("Dataset", "COMP6 S66x8")
print_info(comp6)
```

```
Name: COMP6 S66x8

Data Points: 528
Elements: ['C', 'H', 'N', 'O']
Labels: ['energy', 'gradient', 'charges', 'dipole', 'spin density']
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

Get the molecules and energies from COMP6:

[30]:
```
comp6.units = "hartree"
values = comp6.get_values(name="Energy")
molecules= comp6.get_molecules()
```

An example molecule:

[31]:
```
molecules["molecule"][100]
```

```
NGLWidget()
```

Repeat the same process as done for ANI-1 to get molecules and energies, generate features and labels, and make predictions.

[32]:
```
data = pd.merge(values, molecules, left_index=True, right_index=True)

data["feature"] = [coulomb_matrix(molecule) for molecule in data["molecule"]]
data["label"] = [data.loc[idx, "Energy"] -
                        free_atom_energy(data.loc[idx, "molecule"])
                        for idx in data.index]
X = np.vstack(data["feature"])
data["prediction"] = model.predict(X)[0][:,0]

data["Unsigned Error (Hartree)"] = np.abs(data["label"] - data["prediction"])
data["Dataset"] = "COMP6 S66x8 Test Set"
data_all = data_joined.append(data, sort=False)
data_all["Method"] = "Our ML Model"

fig = px.violin(data_all, y="Unsigned Error (Hartree)", x="Dataset",
                title="Error Distribution for our ML Model")
fig.show()
```

Data type cannot be displayed: application/vnd.plotly.v1+json, text/html

1.10.3 Extras

[2]:
```
from IPython.core.display import HTML

def print_info(dataset):
    print(f"Name: {dataset.data.name}")
    print()
    print(f"Data Points: {dataset.data.metadata['data_points']}")
    print(f"Elements: {dataset.data.metadata['elements']}")
    print(f"Labels: {dataset.data.metadata['labels']}")

    display(HTML("<u>Description:</u> " + dataset.data.description))

    for cite in dataset.data.metadata["citations"]:
        display(HTML(cite['acs_citation']))
```

You can run this notebook online in a session or view it on Github.

1.11 Creating your own datasets on your private server

In addition to querying existing datasets, the QCArchive software can be used to easily generate new ones. In this example, we will create a new dataset of small molecules with up to 3 heavy atoms and compute their DFT and AN1-1x energies.

For this example, we will use a demonstation "Snowflake" server which runs calculations locally in this Jupyter notebook session. In general, QCArchive can be used with thousands of distributed compute nodes at once.

[2]:
```
import numpy as np
import pandas as pd
import qcportal as ptl

from qcfractal import FractalSnowflakeHandler

server = FractalSnowflakeHandler()
local_client = server.client()
```

Our new dataset will be called "QM3":

[3]:
```
qm3 = ptl.collections.Dataset(name="QM3", client=local_client, default_units="hartree
↪")
```

1.11.1 Adding molecules to a dataset

The following function counts heavy atoms in a molecule:

[4]:
```
def count_heavy_atoms(molecule):
    return len(list(filter(lambda a: a != 'H', molecule.symbols)))
```

The `add_entry` function adds a molecule to a dataset. Below, we add all molecules in QM7b with 3 or fewer heavy atoms. The `save` function commits these changes to the server. First, pull QM7b down from the MolSSI server:

[5]:
```
client = ptl.FractalClient()
qm7b = client.get_collection("dataset", "QM7b")
qm7b_mols = qm7b.get_molecules()
```

[6]:
```
for molecule in qm7b_mols["molecule"]:
    if count_heavy_atoms(molecule) <= 3:
        qm3.add_entry(f"{molecule.name}_{molecule.get_hash()[:2]}", molecule)
qm3.save()
```

[6]:
```
'1'
```

We can now query the server for the molecules in our dataset:

[7]:
```
qm3.get_molecules()
```

[7]:

index	molecule
CH4_b3	Geometry (in Angstrom), charge = 0.0, mult...
C2H6_f3	Geometry (in Angstrom), charge = 0.0, mult...
C2H3N_a0	Geometry (in Angstrom), charge = 0.0, mult...
C2H7N_57	Geometry (in Angstrom), charge = 0.0, mult...
C2H4O_07	Geometry (in Angstrom), charge = 0.0, mult...
C2H6O_18	Geometry (in Angstrom), charge = 0.0, mult...
C2H4O_15	Geometry (in Angstrom), charge = 0.0, mult...
C2H6O_69	Geometry (in Angstrom), charge = 0.0, mult...
C2H4_96	Geometry (in Angstrom), charge = 0.0, mult...
C2H2_66	Geometry (in Angstrom), charge = 0.0, mult...
C3H6_d1	Geometry (in Angstrom), charge = 0.0, mult...
C3H8_5c	Geometry (in Angstrom), charge = 0.0, mult...
C3H6_17	Geometry (in Angstrom), charge = 0.0, mult...
C3H4_11	Geometry (in Angstrom), charge = 0.0, mult...
C2H5N_1a	Geometry (in Angstrom), charge = 0.0, mult...
C2H7N_5e	Geometry (in Angstrom), charge = 0.0, mult...

And look at one of them:

[8]:
```
qm3.get_molecules(subset="C2H6O_18")
```

```
_ColormakerRegistry()
```

```
NGLWidget()
```

1.11.2 Running calculations

Our QM3 dataset now has all of its molecules, but no properties have been computed for them.

[9]:
```
qm3.list_values()
```

[9]:
```
Empty DataFrame
Columns: [keywords, name]
Index: []
```

The `compute` function is used to submit calculations for every molecule in a dataset. We will compute the B97x/6-31g(d) energy for each molecule using the Psi4 program, and the ANI-1x energy using the TorchANI program. (Other supported programs include CFOUR, entos, GAMESS, Q-Chem, Molpro, MOPAC, NWChem, RDKit, TeraChem, and Turbomole.)

[10]:
```
qm3.compute(program='psi4', method='wB97x', basis='6-31g(d)')
```

[10]:
```
<ComputeResponse(nsubmitted=16 nexisting=0)>
```

[11]:
```
qm3.compute(program="torchani", method="ANI1x")
```

[11]:
```
<ComputeResponse(nsubmitted=16 nexisting=0)>
```

The calculations are submitted and run asynchronously.

As before, values are described with `list_values` and queried with `get_values`. Incomplete calculations show up as `NaN`, pandas placeholder for missing data.

[12]:
```
qm3.list_values()
```

[12]:
```
                                    keywords                  name
native driver program  method basis
True   energy psi4     wb97x  6-31g(d)     None  WB97X/6-31g(d)-Psi4
              torchani ani1x  None         None       ANI1X-Torchani
```

[16]:
```
dft_data = qm3.get_values(program='psi4')
dft_data
```

[16]:
```
         WB97X/6-31g(d)-Psi4
CH4_b3               -40.4996
C2H6_f3              -79.8015
C2H3N_a0             -132.714
C2H7N_57             -135.122
C2H4O_07             -153.746
C2H6O_18              -154.99
C2H4O_15             -153.786
C2H6O_69             -154.979
C2H4_96              -78.5573
C2H2_66              -77.2971
C3H6_d1              -117.863
C3H8_5c              -119.106
C3H6_17              -117.867
C3H4_11              -116.613
C2H5N_1a             -133.885
C2H7N_5e              -135.13
```

[17]:
```
ml_model_data = qm3.get_values(program='torchani')
ml_model_data
```

[17]:
```
          ANI1X-Torchani
CH4_b3          -40.4997
C2H6_f3         -79.8012
C2H3N_a0        -132.714
C2H7N_57        -135.122
C2H4O_07        -153.746
C2H6O_18         -154.99
C2H4O_15        -153.786
C2H6O_69        -154.978
C2H4_96         -78.5572
C2H2_66         -77.2973
C3H6_d1         -117.863
C3H8_5c         -119.106
C3H6_17         -117.867
C3H4_11         -116.613
C2H5N_1a        -133.885
C2H7N_5e         -135.13
```

We can compare the ANI-1x predictions to the DFT values:

[18]:
```
import plotly.express as px

data = pd.merge(dft_data, ml_model_data, left_index=True, right_index=True)

data["Unsigned Difference (Hartree)"] = np.abs(data["WB97X/6-31g(d)-Psi4"] - data[
↪"ANI1X-Torchani"])

fig = px.violin(data, y="Unsigned Difference (Hartree)", box=True,
                title="Difference Distribution between ANI-1x and B97x/6-31g(d)")
fig.show()
```

Data type cannot be displayed: application/vnd.plotly.v1+json, text/html

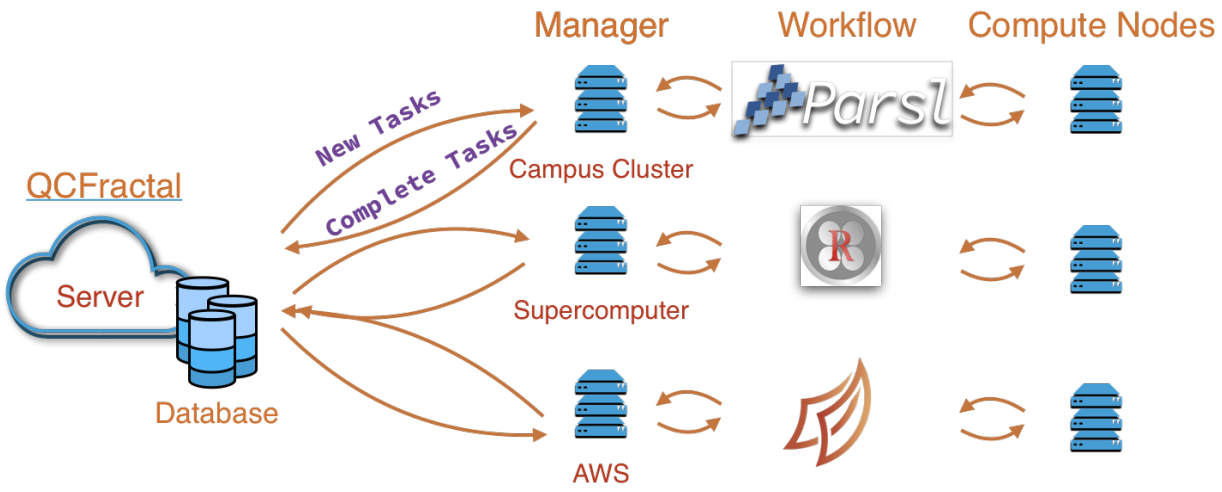
Other calculations you can do

- `Dataset`: Single point, gradients, and freqencies
- `ReactionDataset`: Reactions and interaction energies, with many-body counterpoise corrections
- `OptimizationDataset`: Geometry optimization
- `TorsionDriveDataset`: PES scans over torsion angles, for force field fitting.

Talk to us about adding more!

- (AI)MD trajectories?
- Normal mode sampling?

The QCArchive stack enables large-scale, multi-resource calculations



Ongoing data enrichment efforts

With the QCArchive framework, it is very easy to submit new calculations on existing datasets. In the MolSSI database, we are augmenting existing datasets with a common set of calculations at various levels of DFT:

- HF / Def2-TZVP
- LDA / Def2-TZVP
- PBE-D3M(BJ) / Def2-TZVP
- B3LYP-D3M(BJ) / Def2-TZVP
- B97x-D3(BJ) / Def2-TZVP

and, where feasible:

- MP2 / cc-pVTZ
- CCSD(T) / cc-pVTZ

Let us know if there are properties/calculations that you would like!

Extras

[ ]:
```
from IPython.core.display import HTML

def print_info(dataset):
    print(f"Name: {dataset.data.name}")
    print()
    print(f"Data Points: {dataset.data.metadata['data_points']}")
    print(f"Elements: {dataset.data.metadata['elements']}")
    print(f"Labels: {dataset.data.metadata['labels']}")

    display(HTML("<u>Description:</u> " + dataset.data.description))

    for cite in dataset.data.metadata["citations"]:
        display(HTML(cite['acs_citation']))
```