The MolSSI QCArchive Project: An open-source platform to compute, organize, and share quantum chemistry data

Daniel G. A. Smith,[*,†] Doaa Altarawy,[†,‡] Lori A. Burns,[¶] Matthew Welborn,[†] Levi N. Naden,[†] Logan Ward,[§] Sam Ellis,[†] and T. Daniel Crawford[‖,†]

† *Molecular Sciences Software Institute, Blacksburg, Virginia 24060, USA*

‡ *Department of Computer and Systems Engineering, Alexandria University, Alexandria 21544, Egypt*

¶ *Center for Computational Molecular Science and Technology, School of Chemistry and Biochemistry, Georgia Institute of Technology, Atlanta, Georgia 30332-0400, USA*

§ *Data Science and Learning Division Argonne National Laboratory, Lemont, Illinois 60439, USA*

‖ *Department of Chemistry, Virginia Tech, Blacksburg, Virginia 24061, USA*

E-mail: dgasmith@vt.edu

Abstract

The Molecular Sciences Software Institute's (MolSSI) Quantum Chemistry Archive (QCArchive) project is an umbrella name that covers both a central server hosted by MolSSI for community data and the Python-based software infrastructure that powers automated computation and storage of quantum chemistry results. The MolSSI-hosted central server provides the computational molecular sciences community a location to freely access tens of millions of quantum chemistry computations for machine learning, methodology assessment, force-field fitting, and more through a Python interface.

Facile, user-friendly mining of the centrally archived quantum chemical data also can be achieved through web applications found at https://qcarchive.molssi.org. The software infrastructure can be used as a standalone platform to compute, structure, and distribute hundreds of millions of quantum chemistry computations for individuals or groups of researchers at any scale. The QCArchive Infrastructure is open-source (BSD-3C), code repositories can be found at https://github.com/MolSSI, and releases can be downloaded via PyPI and Conda.

1 Introduction

Data-driven research is uniquely positioned to advance the computational molecular sciences (CMS), and the current rapid movement of the field towards modern data science techniques only reinforces this stance. Successful examples of data-based initiatives in inorganic materials science include the Materials Project[1] with 110,000+ registered users, the NOMAD Laboratory[2] with approximately 100 million stored computations, and AFlow[3] with nearly three million compounds. Collectively, these and other efforts are enabling new science in their fields by allowing access to unprecedented quantities of data.[4,5] Data repositories also naturally enable more reproducible and replicable science.[6,7] Up to this point, however, there has been no similar-scale initiative for quantum chemistry (QC).

Quantum chemistry data currently reside in a dishearteningly wide variety of formats and locations, often being found within supplementary information, on figshare,[8] Zenodo,[9] research group websites, "available upon request" from manuscripts, or lost entirely. The result is a fragmented field where data cannot be easily aggregate data at scale over time which provides additional insights.[10] Any effort to create a quantum chemistry data repository must share data in a manner that is more stable than current methods, is readily accessible, and survives past the data's originator.

Quantum chemistry is naturally suited to archiving its calculations. Although a typical quantum chemical calculation requires substantial computational resources (~1–100 core-

hours and ~1–100 GB of memory), its primary energy, gradient, Hessian, or property output data occupy very little space (~2kB)[1]. Based on a sample of 18+ million results already in the central MolSSI QCArchive server (MQCAS) described by this manuscript, we estimate that 500 million computations can be stored per terabyte (at a cost of ~$200–500 for long-term storage disks and power), representing approximately one billion core-hours of compute time. While academic compute-resource cost is difficult to measure, a compute-optimized AWS cluster costs approximately $0.01 per core hour, yielding a total cost of ~$10 million for one billion core-hours. The general guidance, then, is storing quantum chemistry data is currently 50,000 to 100,000 times cheaper than regenerating it. Put another way, storing quantum chemical results is justified so long as 0.01% of results are reused just once.

The MQCAS offers a central hub for democratizing community-led large-scale data campaigns and commonly used datasets. For example, in a project partially funded by the Open Force Field Consortium (OpenFF),[11] the MQCAS is coordinating a massively distributed queue of torsion drive computations, wherein multiple academic and industry research groups contribute computational resources to create force fields for drug-like molecules through a common best-practices pipeline. Each participating group benefits from the sharing of computational resources for their specific requirements while also adding to the synergistic aggregation of computations of others into massive datasets, queryable on demand. The MQCAS hosts these datasets—which include quantum chemical properties such as geometries, energies, nuclear Hessians, electrostatic potentials, and wavefunctions—as a resource to the community, for whom they can be of direct and immediate use in multiple application domains, including biomolecular simulation, machine learning (ML), instructional courses in physical and quantum chemistry, quantum chemical methodology research, drug discovery, and more. QCArchive makes the ability to perform advanced data-driven research, like that of OpenFF, available to all with our open-source strategy and centralized hosting of quantum chemistry data.

[1] Orbitals and other atomic orbitals quantities radically increase the total size and are stored sparingly.

Beyond force-field fitting, applications such as cheminformatics and training ML models for QC also need to generate large-scale quantum chemical datasets. A common complaint that MolSSI has heard after interviewing over a hundred groups is that producing useful large-scale quantum chemical research datasets requires considerable specialized expertise in multiple domains (cheminformatics, quantum chemistry, high-performance computing, database management), and tools to simplify the management of computation and data would be of high value to enabling research in these communities.

2 Goals

For the QCArchive project, we separate our objectives for the QCArchive into software infrastructure goals usable by the greater CMS community (which powers the MQCAS) and MolSSI's goals for the MQCAS. The MQCAS aspires to organize, curate, and share quantum chemistry data for the benefit of the entire computational molecular sciences community. These data include molecules, computed properties from single-point calculations (energies, gradients, Hessians, dipole moments, orbitals, etc.), and composite data from series of calculations (geometry optimizations, transition-state searches, molecular dynamics trajectories, etc.).

Major MQCAS initiatives are sourced through community engagement and requirements gathering; these initiatives currently fall into two active areas. The first is to curate methodology benchmark sets such as the S22[12] intermolecular dataset of small biomolecules and the YMPJ[13] dataset of natural amino acid conformers. Once ingested, a standard set of DFT and MP2 methods are computed for them using a variety of basis sets, which can then be compared against their literature benchmark values. (See Sec. 4.2 to browse.) Currently, the MQCAS contains 35 benchmark datasets from the literature, and adding additional datasets is an ongoing process. The second major initiative is to curate machine learning datasets from the literature (such as ANI-1[14,15] and QM9[16]) and compute a standard set of DFT

functionals for each of them. Currently, 20 machine learning datasets have been curated, and the computation process is ongoing. (See Sec. 4.1 to browse.) The MQCAS also adheres to Findable, Accessible, Interoperable, and Reusable (FAIR)[17] data practices.

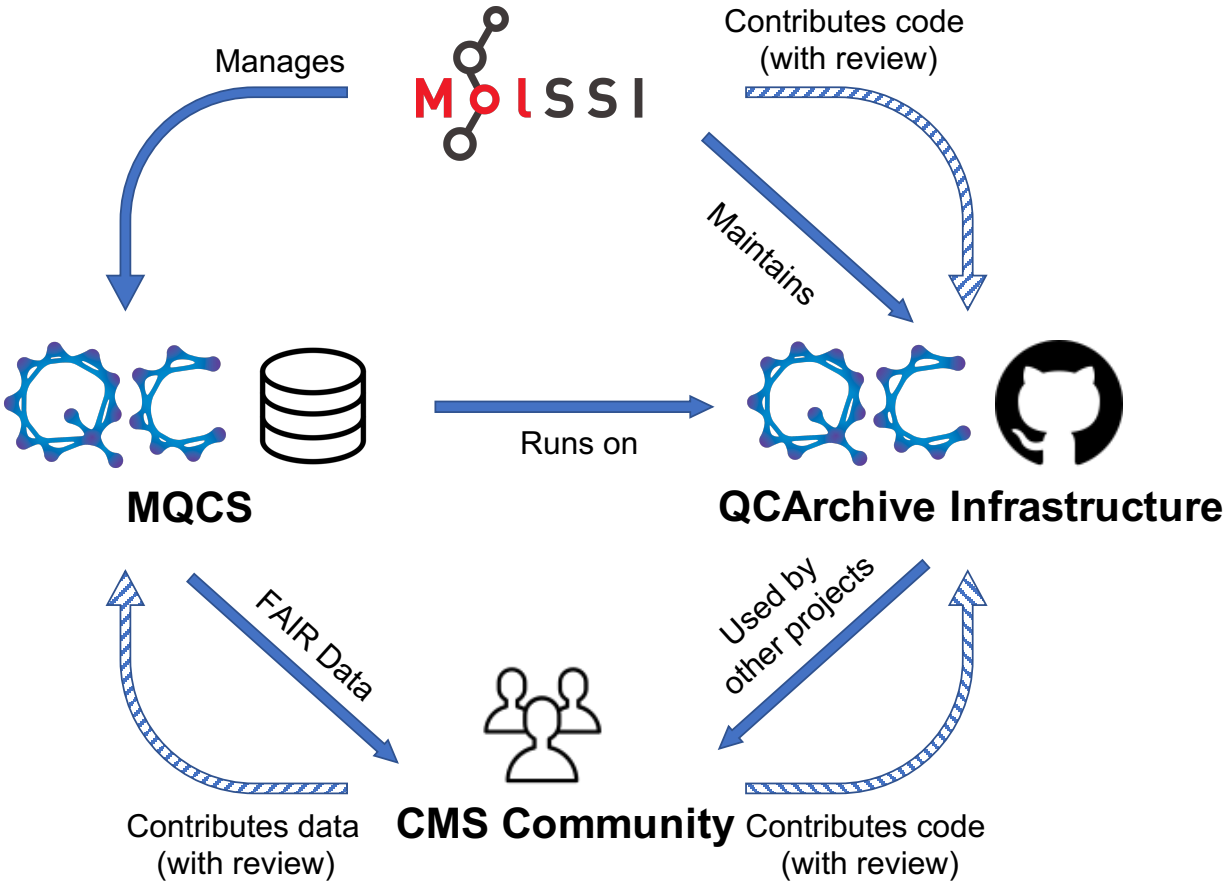

Figure 1: Relationship between the QCARCHIVE project, MolSSI, and the CMS community.

The MQCAS can also be used by the community for those in collaboration with MolSSI and who have target data in mind. In these instances, the community groups can submit new datasets and computations to the MQCAS but are required to supply physical resources to evaluate submitted computational tasks. Several examples are the Open Force Field Initiative to expand data for force field fitting and "A Collection of Chemistry DataBases" (ACCDB)[18] who is assisting the expansion of literature datasets. Expanding the community program is central to the long term goals of the MQCAS, and additional community groups are actively sought.

The software stack supporting the MQCAS, dubbed QCARCHIVE INFRASTRUCTURE, is open-source and modular for reuse by the CMS community. The QCARCHIVE INFRASTRUCTURE can be used by individuals, single research groups, or multiple research groups depending on the data sharing targets of the groups in either public or private settings. Private QCARCHIVE INFRASTRUCTURE instances are fully as capable as MQCAS but are isolated from the public MQCAS and do not communicate until and unless the owner chooses to share their data. The open-source QCARCHIVE INFRASTRUCTURE is already used by a number of research groups that are unaffiliated with MolSSI and have now contributed back dozens of new software features that enhance the capabilities for all researchers using the infrastructure to simplify both their computation and data management.

3 Infrastructure

3.1 Design goals

- *A modular platform.* Though it is often appealing to develop monolithic software to ensure integration between all components, we believe modular software promotes wider adoption in the open-source arena. This loose coupling allows each building block of the core infrastructure to be used in projects beyond the QCARCHIVE scope.

- *Building on community software.* The QCARCHIVE platform exists to promote and link, rather than replace, existing community software. This is evident in several building blocks which collectively wrap dozens of community-built codes and seek to add more through unified interfaces.

- *An open developer community.* All software developed by MolSSI exists on GitHub, where contributions, discussion, criticisms, and bug reports are encouraged and recorded for posterity.

- *Software practices.* The infrastructure follows MolSSI's software "Best Practices"[19,20]

guidelines for contributing, layout, testing, and module distribution.

- *Software distribution.* The monthly release cycle is automatically deployed to PyPI,[21] Conda,[22] and Docker[23] images to ensure ease of use in other projects. Examples of all software projects can also be automatically deployed via Binder[24] and Google Colab.[25]

3.2 Software Components

The QCArchive Infrastructure software components unify to form a platform that can represent computations as structured data, automatically run those computations on a diverse set of physical resources, store computation results, and provide a Python-based front-end to submit and query computations. In addition to forming a complete platform, the four software components were carefully constructed to be usable outside of the QCArchive stack to maximize community utility and reusability. A brief overview of each component can be found below:

- QCSchema- A community built key-value/array description of QC (or QC-like) objects such as molecules and input/output quantities from QC programs. `https://github.com/MolSSI/QCSchema`
- QCElemental- Periodic table information, version-controlled physical constants, molecule parsing, testing infrastructure, and QCSchema models. `https://github.com/MolSSI/QCElemental`
- QCEngine- Quantum chemistry and CMS community program executor with QCSchema enforced input/output models. `https://github.com/MolSSI/QCEngine`
- QCFractal- Distributed task scheduler and executor, database store for chemistry results, and organization of results at scale. `https://github.com/MolSSI/QCFractal`
- QCPortal- Data querying, visualization, organization, and statistical analysis for chemistry-related results and a front-end client for QCFractal. `https://github.`

com/MolSSI/QCPortal

3.2.1 QCElemental

QCElemental encodes data layouts for fundamental CMS entities like Molecule, OptimizationResult, and BasisSet into object-oriented "models" that can be used in Python. In addition to forming a Python reference implementation of the community-built QCSchema, the models impose validation to layout (e.g., $3 * N_{\text{atoms}}$ coordinates) and physics (protons − electrons = charge) beyond what JSON Schema[26] (the implementation of QCSchema) can specify. Models further define convenience functions like Molecule string parsing, coordinate alignment, and structure measurement. The serialization of models is automatic and transparent, including binary/JSON and shaped/flat array data, and recursively extends to all hierarchical models. By embellishing the base QCSchema data layout, QCElemental can avert much of the serialization, translation, resizing, and validation code that inevitably surrounds every downstream QCSchema implementation. This focuses community discussion on the implicit contracts for QCSchema transactions that JSON Schema cannot define and provides modular building blocks for CMS applications.

QCElemental also collects metadata vital to reproducibility in CMS that are difficult to find in structured format (often websites or journal tables) and that are being continuously refined, namely physical constants, periodic table metadata (masses, common isotopes, etc.), covalent radii, etc. These collections are placed behind a light Python API, labeled by context (e.g., CODATA2018), and backed by unit conversion tools. QCElemental can thus facilitate migration of community software toward modern and consistent values, while ensuring that with a context switch, conversions to older data are reproducible. QCSchema values are stored in atomic units to minimize susceptibility to such details and the QCElemental arbitrary unit conversion is used to support the diverse fields and units CMS codes demand. A small QCElemental example can be seen in Fig. 2.

```
In [1]: import qcelemental as qcel

In [2]: mol = qcel.models.Molecule.from_data("""
        O  0.000  0.000 -0.0757
        H  0.000 -0.866  0.6014
        H  0.000  0.866  0.6014
        """)
        mol

Out[2]: "Molecule(name='H2O', formula='H2O', hash='6c6d326')"

In [3]: mol.measure([[0, 1], [1, 0, 2]])

Out[3]: [2.0773414767520113, 103.95838178345787]

In [4]: qcel.constants.conversion_factor('hartree / bohr', 'eV / angstrom')

Out[4]: 51.42206707095895
```

Figure 2: A Jupyter notebook showing a Molecule object created from an XYZ string, the O-H distance (Bohr) and H-O-H angle (degrees) measured, and the computation of a conversion factor for a gradient.

3.2.2 QCEngine

QCEngine is a unified function-based execution engine that provides QCSchema input and output objects for a variety of programs. QCEngine provides this uniform input/output for any program which can be expressed within the restraints of the schema and, as such, covers a variety of quantum chemistry, semi-empirical, force field, and machine learning inference programs. QCEngine works by either interfacing with a program directly at the Python layer or translating the domain-specific input file that a program expects and parsing the subsequent log file or, if available, structured output files. The execution layer automatically discovers the locations and versions of available programs, configures them based on physical resource constraints such as memory and number of physical cores, and isolates and cleans up scratch directories.

The persistent use of QCSchema descriptions in QCEngine simplifies writing software that uses CMS programs. The input to all computations is a QCSchema specification document, such as an `AtomicInput` describing standard input to atomistic calculations (e.g.,

gradient evaluations). Very few aspects of the inputs to QCEngine are tied to the actual code which will fulfill the specified calculation, allowing users to efficiently perform computations with different quantum chemistry codes as scientific demands change. The output from the codes is also returned in a standardized form, which simplifies databases and any software to analyze the outputs of a QC program. Two examples: 1) a Hessian computed with Psi4[27] and NWChem[28] can be analyzed with the same software and 2) the geomeTRIC optimization package[29,30] uses QCEngine as an interface to different quantum chemistry packages so as to be agnostic the backend program.

QCEngine also automatically gathers information needed to reproduce calculations. The input QCSchema completely specifies and documents the calculation that was performed, and the QCEngine executable records all required outputs. All computations that are evaluated with QCEngine return provenance information including program version, hardware specification, physical resources utilized, and QCEngine version. Tracking this information provides both reproducibility and history of every computation undertaken with this software platform. The inputs, provenance, and outputs of a calculation are all stored in the same document, which makes it less likely that metadata will be lost.

Currently the following programs are interfaced to QCEngine:

- *Quantum Chemistry*: CFOUR,[31] Entos,[32] GAMESS,[33] Molpro,[34,35] NWChem,[28] Psi4,[27] Q-Chem,[36] Terachem,[37] Turbomole[38]
- *Semi-empirical*: MOPAC[39]
- *ML Potential*: TorchANI[14,40]
- *Molecular Mechanics*: RDKit,[41] OpenMM[42]
- *Analytical Corrections*: DFTD3,[43,44] MP2D[45,46]
- *Geometry Optimizers*: geomeTRIC,[30,47] Berny[48]

Interfacing to additional programs is an ongoing effort that is open to community contributions and suggestions. Many of the wrappers for specific software has been contributed by non-core QCEngine developers, such as Entos, Molpro and MOPAC. A short example computation can be found in Fig. 3.

```
In [1]: import qcengine as qcng
        import qcelemental as qcel

In [2]: inp = {
            "molecule": qcel.models.Molecule.from_data("pubchem:water"),
            "driver": "energy",
            "model": {"method": "B3LYP", "basis": "6-31g"}
        }

        Searching PubChem database for water (single best match returned)
        Found 1 result(s)

In [3]: ret_psi4 = qcng.compute(inp, "psi4")
        ret_psi4.return_result

Out[3]: -76.38546710382376

In [4]: ret_qchem = qcng.compute(inp, "qchem")
        ret_qchem.return_result

Out[4]: -76.38545273021172
```

Figure 3: A Jupyter notebook demonstrating pulling a water geometry from the PubChem database[49] and computing both Q-Chem and Psi4 with the same input. Note the difference in absolute energies is primarily due to the fact that Q-Chem uses direct SCF algorithms by default while Psi4 uses density-fitting.

3.2.3 QCFractal

QCFractal is the central server of the QCArchive platform, providing distributed computing, data storage, and data querying capabilities. Fundamentally QCFractal is a campaign manager designed for long-duration (years) deployments to passively manage all computation and data storage requirements for an individual researcher or groups of researchers concurrently. For reference, the MQCAS has currently run for eight months without issue, serves data to over a thousand unique users per month, and evaluates 1–5M tasks per month. It is estimated that the MQCAS has the projected capacity to scale by at least 15

fold without optimization.

Storage of computations and associated metadata is accomplished by using a SQL relational database (PostgreSQL[50]). The SQL database is structured so that each type of QCElemental object stored (`Molecule`, `AtomicResult`, etc.) has its own table in the database, and composite objects like `AtomicResult`, which contain a `Molecule` object internally, reference the required table instead of storing the object. For example, a single `Molecule` object can be related to many different energy or gradient calculations so that the molecule is not duplicated, saving space, and all computations related to the molecule can be quickly queried. Each stored object has a carefully constructed unique identifier that can be computed from the object itself. The unique identifier makes it trivial to identify duplicate records and prevent the same calculation from being run twice. If a computation with the same molecule is requested by another computation, each computation links to the same molecule in the database. This linking structure makes tedious composite computations of multiple molecular properties effortless.

To evaluate tasks, QCFractal provides a central task queue, which can be evaluated by a single or many "managers" who interface with a physical resource (campus cluster, supercomputer, cloud resource, or workstation). Each manager can consume either the entire or a fraction of the physical resource, depending on configuration, through a unified cluster interface. Mangers use existing task execution systems like Dask,[51] Parsl,[52] RADICAL,[53] and Fireworks[54] to accomplish high-throughput distributed computing within a given resource, leveraging large amounts of software infrastructure work by the community. In particular, for traditional HPC resources, these task execution systems hook into queuing systems like SLURM[55] to submit "workers" to nodes, which in turn evaluate tasks. Each of these execution systems eliminates the need for humans to submit each computation to the queuing system. The managers stay resident on the system and can automatically remove or request new resources depending on the number of tasks in the central task queue. A diagram of a task execution system is given in Fig 4. The end result is that users submit tasks through a

single interface, and the tasks are automatically evaluated on any physical resource to which the user has access.

QCFractal can also run workflow "services" to orchestrate sophisticated computational campaigns built out of interdependent tasks. For example, a TorsionDrive[56] service will compute the energy surface of a molecule at a frozen set of dihedral coordinates. The TorsionDrive program is a standalone open-source package[57] that implements an iterative procedure for computing smooth dihedral scans. At each step in the iterative TorsionDrive service, new geometry optimizations are supplied by the TorsionDrive package and placed into QCFractal's central task queue. Once all computations for each iteration are complete, QCFractal uses the TorsionDrive package to discover the starting geometries of the next round of geometry optimizations. The TorsionDrive service is iterated until the entire dihedral scan has been completed, and a smooth energy profile has been achieved. The MQCAS routinely iterates thousands of services like TorsionDrive concurrently to completely saturate available computational resources.

In addition to the TorsionDrive service, a GridOptimization service that can do multi dimensional scans of bonds, angles, and dihedrals combined is also implemented. Many future services are in development such as automated conformational searches, reaction pathways, finite difference gradients and Hessians, and more. A core feature of the services is that they can import the logic of different programs and request standard building blocks such as energy and geometry optimizations from the central task queue.

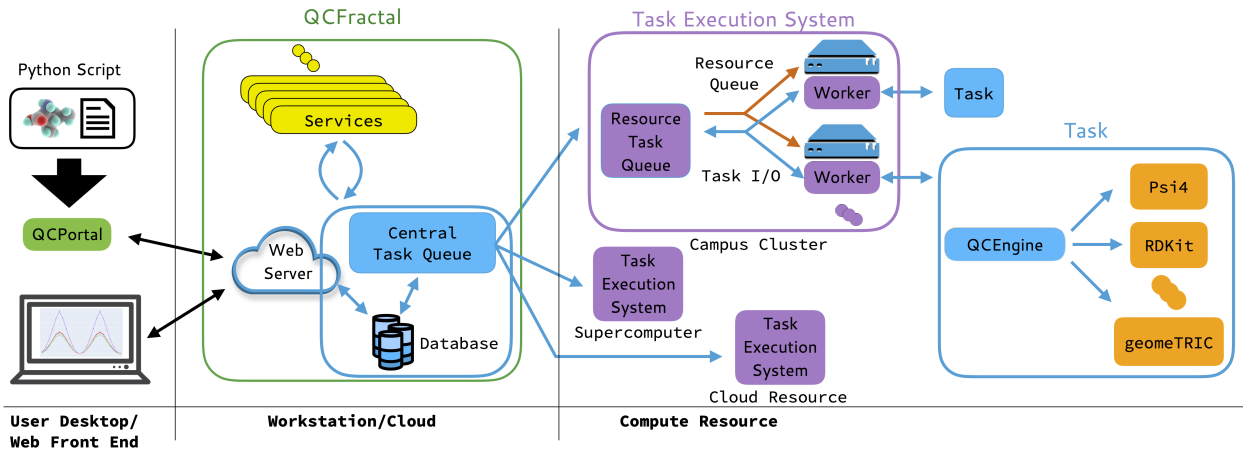

Figure 4: The full QCArchive Infrastructure and how each layer and module communicates. Between different locations, all pieces communicate over TCP/IP protocols; within individual boxes, all pieces are connected through a single Python interpreter; and between a resource task queue and workers, communication can come in a variety of RPC sockets over either MPI or local Ethernet.

3.2.4 QCPortal

QCPortal is a front-end client for QCFractal that is able to access data from the server in a user-friendly fashion and spawn new compute tasks to be evaluated. The basic organizational unit of QCPortal is the collection, a grouping of computations into commonly usable forms. For example, the "Dataset" collection is a dataframe (two-dimensional tabular data) with rows corresponding to named Molecules and columns corresponding to a model chemistry (e.g., B3LYP-D3/def2-svp). The cells in the dataframe reference individual computations in the QCFractal database. New molecules or model chemistries can be added to a Dataset at any time and will be tracked by the Dataset for future use. Data handling is supported by the popular Pandas[58] library for manipulating dataframe objects.

Interactive Python sessions such as Jupyter[59] and Google Colab[25] are core features of QCPortal allowing real-time manipulation of data, visualization of common properties, and built-in charts and properties. This includes visualization capabilities for `Molecule` objects (supported by NGLView[60]), automated performance characteristics of methods computed on Datasets, and graphs that demonstrate the energy profile of a geometry optimiza-

tion per step.

```
In [1]: import qcportal as ptl

In [2]: client = ptl.FractalClient()
        client
```

Out[2]:

FractalClient

- **Server:** The MolSSI QCArchive Server
- **Address:** https://api.qcarchive.molssi.org:443/
- **Username:** None

```
In [3]: s22 = client.get_collection("ReactionDataset", "S22")
        s22.get_values(method="B3LYP").head()
```

Out[3]:

	B3LYP/aug-cc-pvtz	B3LYP/def2-svp	B3LYP/aug-cc-pvdz	B3LYP/def2-tzvp
2-Pyridone-2-Aminopyridine Complex	-14.0236	-18.6168	-14.7904	-14.3219
Adenine-Thymine Complex Stack	0.815241	-2.98381	-0.304599	0.443624
Adenine-Thymine Complex WC	-13.1806	-18.0682	-13.943	-13.327
Ammonia Dimer	-2.20382	-5.12403	-2.38562	-2.92469
Benzene-Ammonia Complex	-0.185656	-0.864532	-0.446438	-0.547386

```
In [4]: s22.get_molecules(subset="Adenine-Thymine Complex Stack").iloc[0, 0]
```

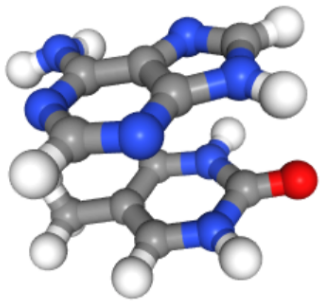

Figure 5: An image showing QCPortal pulling the S22 Dataset from the MQCAS, listing all current B3LYP interaction energies, and rendering an example molecular complex from the dataset.

3.3 Examples

A full software stack example is provided in the form of a TorsionDriveDataset, which computes TorsionDrives for different molecules at multiple levels of theory. Computation with QCFractal Datasets typically happens in three stages: 1) defining the molecules

and relevant metadata (e.g., the dihedral angle to scan) in the Dataset, 2) the addition of computation specifications for evaluation, and 3) querying of the data during or after the computations have completed.

For the first stage, a QCFractal client object is created that connects to a QCFractal server on the internet. In this example, the QCFractal server is on the same computer as the Jupyter Notebook. A new TorsionDriveDataset is then created with the name "TD Demo", and, once created, this Dataset is permanently in the server unless explicitly deleted. Next, hydrogen peroxide and butane geometries are pulled from QCPortal's small library of testing molecules. In general, molecules can be generated and imported from half a dozen common file formats. Molecules are added with an alias for later reference along with the zero-index dihedral specification and the granularity of the torsion angle scan grid in degrees (Fig. 6).

```
import qcportal as ptl
```

```
client = ptl.FractalClient("localhost:7777", verify=False)
```

FractalClient

- **Server:** QCFractal Server
- **Address:** https://localhost:7777/
- **Username:** None

```
ds = ptl.collections.TorsionDriveDataset("TD Demo", client=client)
ds
```

```
<TorsionDriveDataset(name=`TD Demo`, id='local', client='https://localhost:7777/') >
```

```
hooh = ptl.data.get_molecule("hooh.json")
butane = ptl.data.get_molecule("butane.json")

ds.add_entry("hooh", [hooh], [[0, 1, 2, 3]], [15])
ds.add_entry("butane", [butane], [[0, 2, 3, 1]], [15])
```

Figure 6: Setup of a new Dataset object.

The second stage involves the specification of the level of theory for each TorsionDrive computation. To provide a diverse example, a force field (MMFF94[61]), machine-learned po-

tential (ANI-1x[62]), and a quantum chemistry level of theory (B3LYP-D3(BJ)/def2-svp[63–65]) are chosen. For each specification, the geometry optimization program, method, basis set (optionally), and gradient evaluation program are also chosen. Finally, calculations are submitted to the QCFractal server's queue (Fig. 7).

```
optimization_spec = {"program": "geometric", "keywords": {"coordsys": "tric"}}

# The MMFF94 force field with RDKit
atomic_spec = {"driver": "gradient", "method": "mmff94", "program": "rdkit"}
ds.add_specification("MMFF94", optimization_spec, atomic_spec)

# The ANI1x method with the TorchANI program
atomic_spec = {"driver": "gradient", "method": "ANI1x", "program": "torchani"}
ds.add_specification("ANI1x", optimization_spec, atomic_spec)

# A canonical B3LYP-D3/def2-svp quantum chemistry computation with Psi4
atomic_spec = {"driver":" gradient", "method": "B3LYP-D3",
               "basis": "def2-svp", "program": "psi4"}
ds.add_specification("B3LYP-D3", optimization_spec, atomic_spec)
```

```
ds.compute("MMFF94")
ds.compute("ANI1x")
ds.compute("B3LYP-D3")
```

Figure 7: Computation on a Dataset object.

The final stage assumes that sufficient time has passed to evaluate the requested computations. The researcher can then pull the TorsionDriveDataset from the server and begin to explore the results. Shown is a simple example plotting the relative energy as a function of the dihedral angle for butane (Fig. 8). In addition to the plot, every geometry optimization, molecule, gradient evaluation, logfile, and other computational detail can be queried and analyzed.

```
ds = client.get_collection("TorsionDriveDataset", "TD Demo")
```

```
ds.visualize("butane", ["MMFF94", "ANI1x", "B3LYP-D3(BJ)"])
```

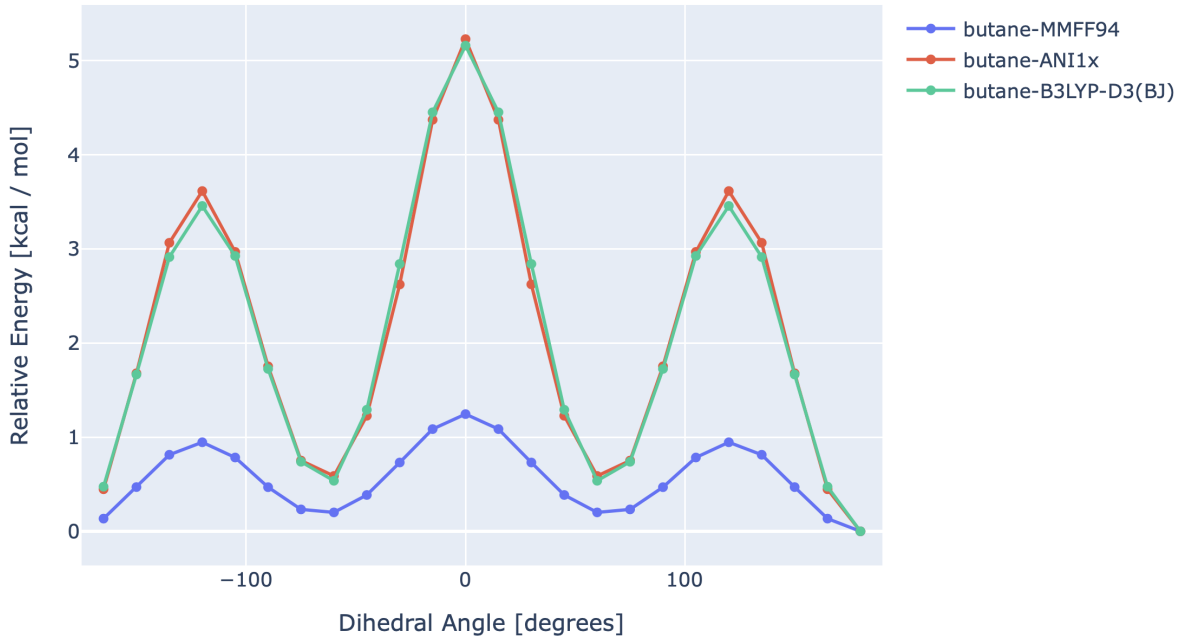

Figure 8: Visualization of a dataset object.

Large datasets in the MQCAS can contain hundreds of thousands of molecules (rows) and hundreds of computational methods (columns).

4 Web Applications

To broaden the accessibility of CMS data beyond Python and Jupyter users, the QCARCHIVE also provides access to its data through Web applications (Web apps). Web apps serve users at all levels, from novice undergraduate students to the most advanced researchers and companies. The goal is for users to view and explore data from the MQCAS in a facile but powerful way, making quantum chemistry available for general computational molecular scientists for a variety of use cases.

Each app is developed as a separate Flask-based[66] micro-service with all apps available through the main QCARCHIVE Web portal (`https://qcarchive.molssi.org`). This ap-

proach enables easy extension and addition of new apps and potential contributions from the community to develop more apps. Plotly Dash[67] is used to compose web apps without requiring the use of Javascript, simplifying the process of hosting and creating new web applications. The downside to Dash is that fine-tune control over the web app is lost and missing features require a more complex implementation making it unsuitable for every web app.

Currently, the QCArchive portal includes two apps, and more are under development with partners like the Science Gateways Community Institute[68] to work on the interactive app needs of the community. A brief description of current Web apps follows.

4.1 Machine Learning Datasets Web App

The MolSSI Quantum Chemistry Machine Learning (ML) datasets repository provides a web app front-end to the curated ML datasets in the mqcas. These currently number 20, including ANI-1,[14,15] PC-9,[69,70] and ISO-17,[71–73] and more are continuously added. Datasets are ingested into a common format of structured metadata and are available for download in text format, or in structured HDF5 format. Figure 9 shows a screenshot of the ML datasets repository. All relevant citation information is shown to ensure the credit to the original authors is correctly assigned.

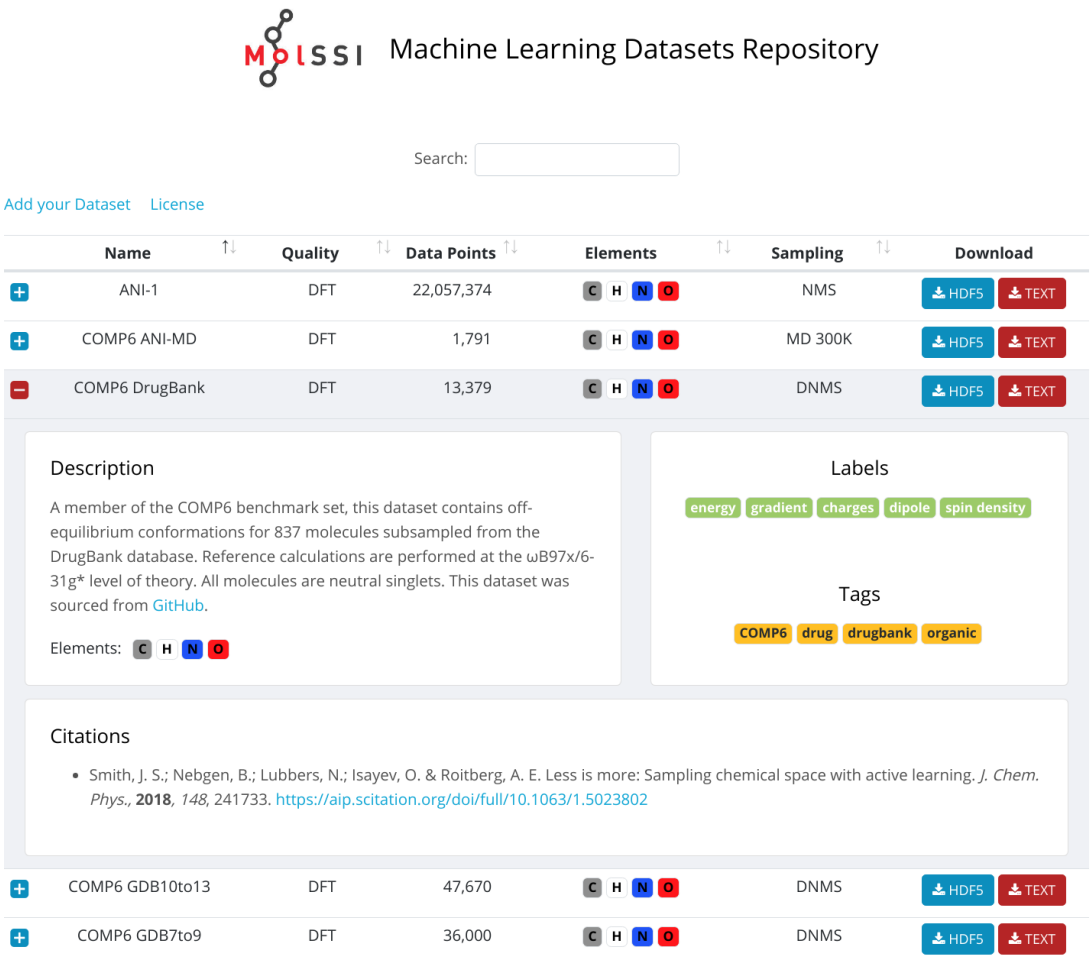

Figure 9: QC Machine Learning Datasets Repository

4.2 Reaction Datasets Web App

The reaction datasets viewer app provides interactive visualization (see Fig. 10) for benchmarking, providing statistics on hundreds of DFT and semiempirical methods for dozens of community benchmark datasets such as S22,[12,74] HSG,[74,75] ACONF,[76] HTBH,[77] and SSI.[78]

Analysis through the app can drive best practices for a given chemical problem while also considering user time and resource constraints. One can imagine pre-defined sets of tautomers, conformations, torsional scans, and dimers which have been pre-evaluated over several kinds and levels of theory, effectively providing a framework with which to quickly evaluate emerging methods

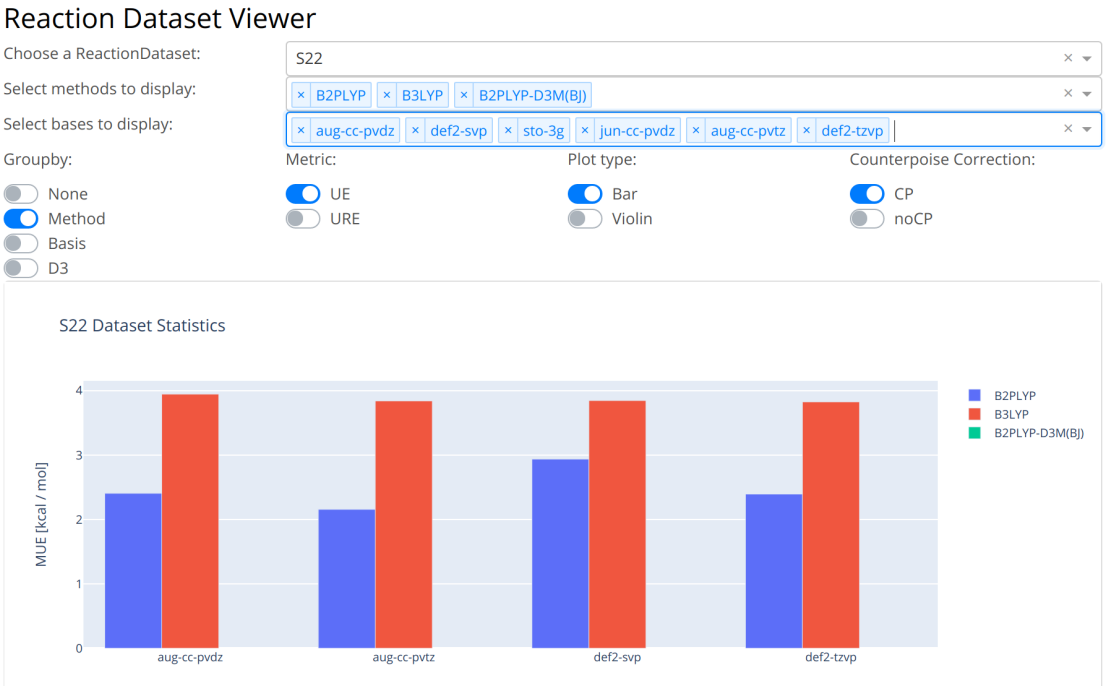

Figure 10: Reaction Datasets Viewer app.

4.3 Other Apps

Additional apps are being developed based on their usefulness and input from the community. For example, a common question that we have heard is the attempt to estimate the time a given quantum chemistry computation will take. This is often hard for a human to accomplish as the steep scaling of quantum chemistry methods can easily mean the difference of one to two orders of magnitude.

The quantum chemistry execution time estimator app predicts the time of a quantum chemistry computation given a molecule (such as SMILES, InChI, or common file formats like XYZ), basis set, and level of theory. In addition, time estimations will vary with respect to the number of threads provided and the physical hardware used. The prediction is based on a machine learning model trained with hundreds of thousands of computations that exist in the MQCAS and is planned to be released shortly.

5 Conclusions

There is an urgent and growing need for data best-practices in the computational molecular sciences as the sheer number of quantum chemistry computations being evaluated across the community continues to grow exponentially. This growth also puts a focus on the need for automation, replicability, and reproducibility.[6] The QCArchive addresses this need in the QC space through a MolSSI-hosted centralized data server (MQCAS) and a Python-based software infrastructure (QCArchive Infrastructure).

The MQCAS represents an ongoing effort to gather, organize, host, and democratize quantum chemistry data for the benefit of the broader molecular sciences community. This goal is achieved first through FAIR data hosted by the MQCAS and through focusing on the areas of force-field fitting, methodological benchmarking, and machine learning. These data are provided in full through either a Python or web interface. To further broaden accessibility and utility of the MQCAS data, the QCArchive project is developing web apps that present chemical insights directly to users without the need for programming ability or data manipulation.

The QCArchive Infrastructure software is composed of a number of building blocks each at a different level of interaction from fully automated workflows on distributed computing to the conversion factor between units. Independently, the software building blocks are in use by the community outside of the QCArchive (such as Psi4, geomeTRIC, and OpenFF) and the hope is these tools will power a larger software ecosystem built off of QCSchema. When these building blocks are used together they create a unified platform for computing, structuring, and distributing computations and procedures at scale.

Taken together, these efforts have produced a substantial resource for the CMS community. At the time of writing, the MQCAS contains 12 million molecules, 18 million calculation results (with full provenance information), and 96 data collections. The QCArchive Infrastructure demonstrates best practices in community-focused software development with 28 external code contributors, and use in 18 external software projects since its beta

launch in August 2019. We hope this kind of software infrastructure and data gathering methods will push CMS towards a more interoperable, data-driven, and computation commoditizing future.

6 Acknowledgements

We are grateful to Shantenu Jha, Matteo Turilli, Andre Merzky, Matt Horton, and John Chodera for helpful discussions.

7 Funding Information

D.G.A.S, D.A., M.W., L.N.N., S.E., and T.D.C. were supported by U. S. National Science Foundation (NSF) grant ACI-1547580. L.A.B. was supported by NSF grant ACI-1449723. L.W. developed capabilities for node-parallel applications and was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. D.G.A.S. also acknowledges the Open Force Field Consortium and Initiative for financial and scientific support.

8 Research Resources

The authors also acknowledge Advanced Research Computing at Virginia Tech, the Pacific Research Platform, the Open Science Grid, the Argonne Leadership Computing Facility, and the Memorial Sloan Kettering Institute for providing computational resources and technical support that have contributed to the results reported within the paper.

References

(1) Jain, A.; Ong, S. P.; Hautier, G.; Chen, W.; Richards, W. D.; Dacek, S.; Cholia, S.; Gunter, D.; Skinner, D.; Ceder, G.; Persson, K. a. The Materials Project: A materials genome approach to accelerating materials innovation. *APL Materials* **2013**, *1*, 011002.

(2) Draxl, C.; Scheffler, M. NOMAD: The FAIR concept for big data-driven materials science. *MRS Bulletin* **2018**, *43*, 676–682.

(3) Toher, C.; Oses, C.; Hicks, D.; Gossett, E.; Rose, F.; Nath, P.; Usanmaz, D.; Ford, D. C.; Perim, E.; Calderon, C. E.; Plata, J. J.; Lederer, Y.; Jahnátek, M.; Setyawan, W.; Wang, S.; Xue, J.; Rasch, K.; Chepulskii, R. V.; Taylor, R. H.; Gomez, G.; Shi, H.; Supka, A. R.; Al Orabi, R. A. R.; Gopal, P.; Cerasoli, F. T.; Liyanage, L.; Wang, H.; Siloi, I.; Agapito, L. A.; Nyshadham, C.; Hart, G. L. W.; Carrete, J.; Legrain, F.; Mingo, N.; Zurek, E.; Isayev, O.; Tropsha, A.; Sanvito, S.; Hanson, R. M.; Takeuchi, I.; Mehl, M. J.; Kolmogorov, A. N.; Yang, K.; D'Amico, P.; Calzolari, A.; Costa, M.; Gennaro, R. D.; Nardelli, M. B.; Fornari, M.; Levy, O.; Curtarolo, S. In *Handbook of Materials Modeling : Methods: Theory and Modeling*; Andreoni, W., Yip, S., Eds.; Springer International Publishing: Cham, 2018; pp 1–28.

(4) Data mining uncovers a treasure trove of topological materials. *Nature* **2019**, *566*, 425–425.

(5) Thygesen, K. S.; Jacobsen, K. W. Making the most of materials computations. *Science* **2016**, *354*, 180–181.

(6) National Academies of Sciences, E.; Medicine, *Reproducibility and Replicability in Science*; The National Academies Press: Washington, DC, 2019.

(7) Widener, A. Scientists must take steps to improve reproducibility and reliability of research results. *Chem. Eng. News* **2019**, *97*.

(8) FIGSHARE: store, discover, research For the current version, see https://figshare.com (accessed January 2020).

(9) ZENODO: Research. Shared. For the current version, see https://zenodo.org (accessed January 2020).

(10) Milham, M. P. Data sharing and the future of science. *Nat. Commun.* **2018**, *9*, 9–10.

(11) Mobley, D. L.; Bannan, C. C.; Rizzi, A.; Bayly, C. I.; Chodera, J. D.; Lim, V. T.; Lim, N. M.; Beauchamp, K. A.; Shirts, M. R.; Gilson, M. K.; Eastman, P. K. Open Force Field Consortium: Escaping atom types using direct chemical perception with SMIRNOFF v0.1. *bioRxiv* **2018**,

(12) Jurečka, P.; Šponer, J.; Černý, J.; Hobza, P. Benchmark Database of Accurate (MP2 and CCSD(T) Complete Basis Set Limit) Interaction Energies of Small Model Complexes, DNA Base Pairs, and Amino Acid Pairs. *Phys. Chem. Chem. Phys.* **2006**, *8*, 1985–1993.

(13) Yuan, Y.; Mills, M. J. L.; Popelier, P. L. A.; Jensen, F. Comprehensive Analysis of Energy Minima of the 20 Natural Amino Acids. *The Journal of Physical Chemistry A* **2014**, *118*, 7876–7891, PMID: 25084473.

(14) Smith, J. S.; Isayev, O.; Roitberg, A. E. ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost. *Chem. Sci.* **2017**, *8*, 3192–3203.

(15) Smith, J. S.; Isayev, O.; Roitberg, A. E. ANI-1, A data set of 20 million calculated off-equilibrium conformations for organic molecules. *Scientific Data* **2017**, *4*, 170193.

(16) Ramakrishnan, R.; Dral, P. O.; Rupp, M.; Von Lilienfeld, O. A. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data* **2014**, *1*, 140022.

(17) Wilkinson, M. D.; Dumontier, M.; Aalbersberg, I. J.; Appleton, G.; Axton, M.; Baak, A.; Blomberg, N.; Boiten, J.-W.; da Silva Santos, L. B.; Bourne, P. E., et al. The

FAIR Guiding Principles for scientific data management and stewardship. *Scientific data* **2016**, *3*.

(18) Morgante, P.; Peverati, R. ACCDB: A collection of chemistry databases for broad computational purposes. *Journal of Computational Chemistry* **2019**, *40*, 839–848.

(19) Best Practices: MolSSI best practices provide a starting point to get into software development operations to ensure that your code is reliable and reproducible while decreasing long-term maintenance requirements, increasing long-term viability, and allowing others to work on your code base to assist your ow efforts. For the current version, see https://molssi.org/education/best-practices/ (accessed February 2020).

(20) Cookiecutter for Computational Molecular Sciences (CMS) Python Packages: A cookiecutter template for those interested in developing computational molecular packages in Python. Skeletal starting repositories can be created from this template to create the file structure semi-autonomously so you can focus on what's important: the science! For the current version, see https://github.com/MolSSI/cookiecutter-cms (accessed February 2020).

(21) PyPI: Find, install and publish Python packages with the Python Package Index For the current version, see https://pypi.org (accessed February 2020).

(22) Conda: Package, dependency and environment management for any language—Python, R, Ruby, Lua, Scala, Java, JavaScript, C/ C++, FORTRAN, and more. For the current version, see https://docs.conda.io/en/latest/ (accessed February 2020).

(23) Docker: Securely build and share any application, anywhere For the current version, see https://www.docker.com (accessed February 2020).

(24) Project Jupyter,; Matthias Bussonnier,; Jessica Forde,; Jeremy Freeman,; Brian Granger,; Tim Head,; Chris Holdgraf,; Kyle Kelley,; Gladys Nalvarte,; Andrew Os-

heroff,; Pacer, M.; Yuvi Panda,; Fernando Perez,; Benjamin Ragan Kelley,; Carol Willing, Binder 2.0 - Reproducible, interactive, sharable environments for science at scale. Proceedings of the 17th Python in Science Conference. 2018; pp 113 – 120.

(25) Google Colaboratory: Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser. For the current version, see `https://colab.research.google.com` (accessed February 2020).

(26) JSON Schema: a vocabulary that allows you to annotate and validate JSON documents. For the current version, see `https://json-schema.org/` (accessed January 2020).

(27) Parrish, R. M.; Burns, L. A.; Smith, D. G. A.; Simmonett, A. C.; DePrince, A. E.; Hohenstein, E. G.; Bozkaya, U.; Sokolov, A. Y.; Di Remigio, R.; Richard, R. M.; Gonthier, J. F.; James, A. M.; McAlexander, H. R.; Kumar, A.; Saitow, M.; Wang, X.; Pritchard, B. P.; Verma, P.; Schaefer, H. F.; Patkowski, K.; King, R. A.; Valeev, E. F.; Evangelista, F. A.; Turney, J. M.; Crawford, T. D.; Sherrill, C. D. Psi4 1.1: An Open-Source Electronic Structure Program Emphasizing Automation, Advanced Libraries, and Interoperability. *J. Chem. Theory Comput.* **2017**, *13*, 3185–3197.

(28) Valiev, M.; Bylaska, E. J.; Govind, N.; Kowalski, K.; Straatsma, T. P.; Dam, H. J. J. V.; Wang, D.; Nieplocha, J.; Apra, E.; Windus, T. L.; de Jong, W. NWChem: A Comprehensive and Scalable Open-source Solution for Large Scale Molecular Simulations. *Comput. Phys. Commun.* **2010**, *181*, 1477–1489.

(29) Wang, L.-P.; Song, C. Geometry optimization made simple with translation and rotation coordinates. *The Journal of Chemical Physics* **2016**, *144*, 214108.

(30) Wang, L.-P.; Smith, D. G. A.; Qiu, Y. geomeTRIC: a geometry optimization code that includes the TRIC coordinate system. For the current version, see `https://github.com/leeping/geomeTRIC` (accessed January 2020).

(31) Stanton, J. F.; Gauss, J.; Cheng, L.; Harding, M. E.; Matthews, D. A.; Szalay, P. G. CFOUR, Coupled-Cluster techniques for Computational Chemistry, a quantum-chemical program package. With contributions from A.A. Auer, R.J. Bartlett, U. Benedikt, C. Berger, D.E. Bernholdt, Y.J. Bomble, O. Christiansen, F. Engel, R. Faber, M. Heckert, O. Heun, M. Hilgenberg, C. Huber, T.-C. Jagau, D. Jonsson, J. Jusélius, T. Kirsch, K. Klein, W.J. Lauderdale, F. Lipparini, T. Metzroth, L.A. Mück, D.P. O'Neill, D.R. Price, E. Prochnow, C. Puzzarini, K. Ruud, F. Schiffmann, W. Schwalbach, C. Simmons, S. Stopkowicz, A. Tajti, J. Vázquez, F. Wang, J.D. Watts and the integral packages MOLECULE (J. Almlöf and P.R. Taylor), PROPS (P.R. Taylor), ABACUS (T. Helgaker, H.J. Aa. Jensen, P. Jørgensen, and J. Olsen), and ECP routines by A. V. Mitin and C. van Wüllen. For the current version, see http://www.cfour.de.

(32) Manby, F.; Miller, T.; Bygrave, P.; Ding, F.; Dresselhaus, T.; Batista-Romero, F.; Buccheri, A.; Bungey, C.; Lee, S.; Meli, R.; et al., entos: A Quantum Molecular Simulation Package. 2019; https://chemrxiv.org/articles/entos_A_Quantum_Molecular_Simulation_Package/7762646/2.

(33) Gordon, M. S.; Schmidt, M. W. In *Theory and Applications of Computational Chemistry: the First Forty Years*; Dykstra, C. E., Frenking, G., Kim, K. S., Scuseria, G. E., Eds.; Elsevier: Amsterdam, 2005; pp 1167–1189.

(34) Werner, H.-J.; Knowles, P. J.; Knizia, G.; Manby, F. R.; Schütz, M.; Celani, P.; Györffy, W.; Kats, D.; Korona, T.; Lindh, R.; Mitrushenkov, A.; Rauhut, G.; Shamasundar, K. R.; Adler, T. B.; Amos, R. D.; Bennie, S. J.; Bernhardsson, A.; Berning, A.; Cooper, D. L.; Deegan, M. J. O.; Dobbyn, A. J.; Eckert, F.; Goll, E.; Hampel, C.; Hesselmann, A.; Hetzer, G.; Hrenar, T.; Jansen, G.; Köppl, C.; Lee, S. J. R.; Liu, Y.; Lloyd, A. W.; Ma, Q.; Mata, R. A.; May, A. J.; McNicholas, S. J.; Meyer, W.; Miller III, T. F.; Mura, M. E.; Nicklass, A.; O'Neill, D. P.; Palmieri, P.; Peng, D.; Pflüger, K.;

Pitzer, R.; Reiher, M.; Shiozaki, T.; Stoll, H.; Stone, A. J.; Tarroni, R.; Thorsteinsson, T.; Wang, M.; Welborn, M. MOLPRO, version 2019.2, a package of ab initio programs. 2019; see "https://www.molpro.net".

(35) Werner, H.-J.; Knowles, P. J.; Knizia, G.; Manby, F. R.; Schütz, M. Molpro: a general-purpose quantum chemistry program package. *WIREs Comput Mol Sci* **2012**, *2*, 242–253.

(36) Shao, Y.; Gan, Z.; Epifanovsky, E.; Gilbert, A. T.; Wormit, M.; Kussmann, J.; Lange, A. W.; Behn, A.; Deng, J.; Feng, X.; Ghosh, D.; Goldey, M.; Horn, P. R.; Jacobson, L. D.; Kaliman, I.; Khaliullin, R. Z.; Kuś, T.; Landau, A.; Liu, J.; Proynov, E. I.; Rhee, Y. M.; Richard, R. M.; Rohrdanz, M. A.; Steele, R. P.; Sundstrom, E. J.; III, H. L. W.; Zimmerman, P. M.; Zuev, D.; Albrecht, B.; Alguire, E.; Austin, B.; Beran, G. J. O.; Bernard, Y. A.; Berquist, E.; Brandhorst, K.; Bravaya, K. B.; Brown, S. T.; Casanova, D.; Chang, C.-M.; Chen, Y.; Chien, S. H.; Closser, K. D.; Crittenden, D. L.; Diedenhofen, M.; Jr., R. A. D.; Do, H.; Dutoi, A. D.; Edgar, R. G.; Fatehi, S.; Fusti-Molnar, L.; Ghysels, A.; Golubeva-Zadorozhnaya, A.; Gomes, J.; Hanson-Heine, M. W.; Harbach, P. H.; Hauser, A. W.; Hohenstein, E. G.; Holden, Z. C.; Jagau, T.-C.; Ji, H.; Kaduk, B.; Khistyaev, K.; Kim, J.; Kim, J.; King, R. A.; Klunzinger, P.; Kosenkov, D.; Kowalczyk, T.; Krauter, C. M.; Lao, K. U.; Laurent, A. D.; Lawler, K. V.; Levchenko, S. V.; Lin, C. Y.; Liu, F.; Livshits, E.; Lochan, R. C.; Luenser, A.; Manohar, P.; Manzer, S. F.; Mao, S.-P.; Mardirossian, N.; Marenich, A. V.; Maurer, S. A.; Mayhall, N. J.; Neuscamman, E.; Oana, C. M.; Olivares-Amaya, R.; O'Neill, D. P.; Parkhill, J. A.; Perrine, T. M.; Peverati, R.; Prociuk, A.; Rehn, D. R.; Rosta, E.; Russ, N. J.; Sharada, S. M.; Sharma, S.; Small, D. W.; Sodt, A.; Stein, T.; Stück, D.; Su, Y.-C.; Thom, A. J.; Tsuchimochi, T.; Vanovschi, V.; Vogt, L.; Vydrov, O.; Wang, T.; Watson, M. A.; Wenzel, J.; White, A.; Williams, C. F.; Yang, J.; Yeganeh, S.; Yost, S. R.; You, Z.-Q.; Zhang, I. Y.; Zhang, X.; Zhao, Y.; Brooks, B. R.;

Chan, G. K.; Chipman, D. M.; Cramer, C. J.; III, W. A. G.; Gordon, M. S.; Hehre, W. J.; Klamt, A.; III, H. F. S.; Schmidt, M. W.; Sherrill, C. D.; Truhlar, D. G.; Warshel, A.; Xu, X.; Aspuru-Guzik, A.; Baer, R.; Bell, A. T.; Besley, N. A.; Chai, J.-D.; Dreuw, A.; Dunietz, B. D.; Furlani, T. R.; Gwaltney, S. R.; Hsu, C.-P.; Jung, Y.; Kong, J.; Lambrecht, D. S.; Liang, W.; Ochsenfeld, C.; Rassolov, V. A.; Slipchenko, L. V.; Subotnik, J. E.; Voorhis, T. V.; Herbert, J. M.; Krylov, A. I.; Gill, P. M.; Head-Gordon, M. Advances in molecular quantum chemistry contained in the Q-Chem 4 program package. *Molecular Physics* **2015**, *113*, 184–215.

(37) Ufimtsev, I. S.; Martinez, T. J. Quantum Chemistry On Graphical Processing Units. 3. Analytical Energy Gradients, Geometry Optimization, and First Principles Molecular Dynamics. *J. Chem. Theory Comput.* **2009**, *5*, 2619–2628.

(38) Furche, F.; Ahlrichs, R.; Hättig, C.; Klopper, W.; Sierka, M.; Weigend, F. Turbomole. *WIREs Computational Molecular Science* **2014**, *4*, 91–100.

(39) Stewart, J. J. P. MOPAC: semiempirical quantum chemistry. For the current version, see `http://OpenMOPAC.net/` (accessed January 2020) for Stewart Computational Chemistry, Colorado Springs, CO, USA.

(40) Gao, X. TorchANI: Accurate Neural Network Potential on PyTorch. For the current version, see `https://github.com/aiqm/torchani` (accessed January 2020).

(41) Landrum, G. RDKit: cheminformatics and machine-learning software in C++ and Python. For the current version, see `10.5281/zenodo.591637` (accessed January 2020).

(42) Eastman, P.; Swails, J.; Chodera, J. D.; McGibbon, R. T.; Zhao, Y.; Beauchamp, K. A.; Wang, L.-P.; Simmonett, A. C.; Harrigan, M. P.; Stern, C. D.; Wiewiora, R. P.; Brooks, B. R.; Pande, V. S. OpenMM 7: Rapid development of high performance algorithms for molecular dynamics. *PLOS Computational Biology* **2017**, *13*, 1–17.

(43) Grimme, S.; Antony, J.; Ehrlich, S.; Krieg, H. DFTD3: dispersion correction for DFT, Hartree–Fock, and semi-empirical quantum chemical methods. For the current version, see https://github.com/loriab/dftd3 (accessed January 2020). For the originating project, see https://www.chemie.uni-bonn.de/pctc/mulliken-center/software/dft-d3.

(44) Grimme, S.; Antony, J.; Ehrlich, S.; Krieg, H. A Consistent and Accurate Ab Initio Parametrization of Density Functional Dispersion Correction (DFT-D) for the 94 Elements H-Pu. *J. Chem. Phys.* **2010**, *132*, 154104.

(45) Greenwell, C. MP2D: a program for calculating the MP2D dispersion energy. For the current version, see https://github.com/Chandemonium/MP2D (accessed January 2020).

(46) Řezáč, J.; Greenwell, C.; Beran, G. J. O. Accurate Noncovalent Interactions via Dispersion-Corrected Second-Order Møller-Plesset Perturbation Theory. *J. Chem. Theory Comput.* **2018**, *14*, 4711–4721.

(47) Wang, L.-P.; Song, C. Geometry Optimization Made Simple with Translation and Rotation Coordinates. *J. Chem. Phys.* **2016**, *144*, 214108.

(48) Hermann, J. BERNY: Molecular structure optimizer. For the current version, see https://github.com/jhrmnn/pyberny (accessed January 2020).

(49) Kim, S.; Chen, J.; Cheng, T.; Gindulyte, A.; He, J.; He, S.; Li, Q.; Shoemaker, B. A.; Thiessen, P. A.; Yu, B.; Zaslavsky, L.; Zhang, J.; Bolton, E. E. PubChem 2019 update: improved access to chemical data. *Nucleic Acids Research* **2018**, *47*, D1102–D1109.

(50) PostgreSQL: The World's Most Advanced Open Source Relational Database For the current version, see https://www.postgresql.org (accessed January 2020).

(51) Rocklin, M. Dask: Parallel Computation with Blocked algorithms and Task Scheduling. Proceedings of the 14th Python in Science Conference. 2015; pp 130 – 136.

(52) Babuji, Y.; Woodard, A.; Li, Z.; Katz, D. S.; Clifford, B.; Kumar, R.; Lacinski, L.; Chard, R.; Wozniak, J.; Foster, I.; Wilde, M.; Chard, K. Parsl: Pervasive Parallel Programming in Python. 28th ACM International Symposium on High-Performance Parallel and Distributed Computing (HPDC). 2019.

(53) Turilli, M.; Merzky, A.; Balasubramanian, V.; Jha, S. Building Blocks for Workflow System Middleware. Proceedings of the 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. 2018; p 348–349.

(54) Jain, A.; Ong, S. P.; Chen, W.; Medasani, B.; Qu, X.; Kocher, M.; Brafman, M.; Petretto, G.; Rignanese, G.-M.; Hautier, G.; Gunter, D.; Persson, K. A. FireWorks: a dynamic workflow system designed for high-throughput applications. *Concurrency and Computation: Practice and Experience* **2015**, *27*, 5037–5059, CPE-14-0307.R2.

(55) Yoo, A. B.; Jette, M. A.; Grondona, M. SLURM: Simple Linux Utility for Resource Management. Job Scheduling Strategies for Parallel Processing. Berlin, Heidelberg, 2003; pp 44–60.

(56) Y. Qiu, D. G. A. Smith, C. D. Stern, M. Feng, and L. Wang, "Driving Torsion Scans with Wavefront Propagation," *in preparation.*

(57) TorsionDrive: Dihedral scanner with wavefront propagation For the current version, see `https://github.com/lpwgroup/torsiondrive` (accessed January 2020).

(58) McKinney, W. Data Structures for Statistical Computing in Python. Proceedings of the 9th Python in Science Conference. 2010; pp 51 – 56.

(59) Kluyver, T.; Ragan-Kelley, B.; Pérez, F.; Granger, B.; Bussonnier, M.; Frederic, J.; Kelley, K.; Hamrick, J.; Grout, J.; Corlay, S.; Ivanov, P.; Avila, D.; Abdalla, S.; Willing, C.

Jupyter Notebooks – a publishing format for reproducible computational workflows. Positioning and Power in Academic Publishing: Players, Agents and Agendas. 2016; pp 87 – 90.

(60) Nguyen, H.; Case, D. A.; Rose, A. S. NGLview–interactive molecular graphics for Jupyter notebooks. *Bioinformatics* **2017**, *34*, 1241–1242.

(61) Halgren, T. A. Merck molecular force field. I. Basis, form, scope, parameterization, and performance of MMFF94. *Journal of computational chemistry* **1996**, *17*, 490–519.

(62) Smith, J. S.; Nebgen, B.; Lubbers, N.; Isayev, O.; Roitberg, A. E. Less is more: Sampling chemical space with active learning. *The Journal of chemical physics* **2018**, *148*, 241733.

(63) Becke, A. D. Density-functional thermochemistry. I. The effect of the exchange-only gradient correction. *The Journal of chemical physics* **1992**, *96*, 2155–2160.

(64) Grimme, S.; Ehrlich, S.; Goerigk, L. Effect of the damping function in dispersion corrected density functional theory. *Journal of computational chemistry* **2011**, *32*, 1456–1465.

(65) Weigend, F.; Ahlrichs, R. Balanced basis sets of split valence, triple zeta valence and quadruple zeta valence quality for H to Rn: Design and assessment of accuracy. *Phys. Chem. Chem. Phys.* **2005**, *7*.

(66) Flask: web development, one drop at a time For the current version, see `https://flask.palletsprojects.com/en/1.1.x/` (accessed January 2020).

(67) Dash: Build beautiful, web-based analytic apps. No JavaScript required. For the current version, see `https://plot.ly/dash/` (accessed January 2020).

(68) Wilkins-Diehr, N.; Zentner, M.; Pierce, M.; Dahan, M.; Lawrence, K.; Hayden, L.; Mullinix, N. The Science Gateways Community Institute at Two Years. Proceedings of

the Practice and Experience on Advanced Research Computing. New York, NY, USA, 2018.

(69) Nakata, M.; Shimazaki, T. PubChemQC Project: A Large-Scale First-Principles Electronic Structure Database for Data-Driven Chemistry. *Journal of Chemical Information and Modeling* **2017**, *57*, 1300–1308.

(70) Glavatskikh, M.; Leguy, J.; Hunault, G.; Cauchy, T.; Da Mota, B. Dataset's chemical diversity limits the generalizability of machine learning predictions. *Journal of Cheminformatics* **2019**, *11*, 69.

(71) Ramakrishnan, R.; Dral, P. O.; Rupp, M.; von Lilienfeld, O. A. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data* **2014**, *1*, 140022.

(72) Schütt, K. T.; Arbabzadah, F.; Chmiela, S.; Müller, K. R.; Tkatchenko, A. Quantum-chemical insights from deep tensor neural networks. *Nature Communications* **2017**, *8*, 13890.

(73) Schütt, K.; Kindermans, P.-J.; Sauceda Felix, H. E.; Chmiela, S.; Tkatchenko, A.; Müller, K.-R. In *Advances in Neural Information Processing Systems 30*; Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc., 2017; pp 991–1001.

(74) Marshall, M. S.; Burns, L. A.; Sherrill, C. D. Basis Set Convergence of the Coupled-cluster Correction, $\delta_{MP2}^{CCSD(T)}$: Best Practices for Benchmarking Non-covalent Interactions and the Attendant Revision of the S22, NBC10, HBC6, and HSG Databases. *J. Chem. Phys.* **2011**, *135*, 194102.

(75) Faver, J. C.; Benson, M. L.; He, X.; Roberts, B. P.; Wang, B.; Marshall, M. S.; Kennedy, M. R.; Sherrill, C. D.; Merz Jr., K. M. Formal Estimation of Errors in Computed Absolute Interaction Energies of Protein-Ligand Complexes. *J. Chem. Theory Comput.* **2011**, *7*, 790–797.

(76) Gruzman, D.; Karton, A.; Martin, J. M. L. Performance of Ab Initio and Density Functional Methods for Conformational Equilibria of C_nH_{2n+2} Alkane Isomers ($n = 4 - 8$). *J. Phys. Chem. A* **2009**, *113*, 11974–11983.

(77) Zhao, Y.; González-García, N.; Truhlar, D. G. Benchmark Database of Barrier Heights for Heavy Atom Transfer, Nucleophilic Substitution, Association, and Unimolecular Reactions and Its Use to Test Theoretical Methods. *J. Phys. Chem. A* **2005**, *109*, 2012–2018.

(78) Burns, L. A.; Faver, J. C.; Zheng, Z.; Marshall, M. S.; Smith, D. G. A.; Vanommeslaeghe, K.; MacKerell, A. D.; Merz, K. M.; Sherrill, C. D. The BioFragment Database (BFDb): An Open-Data Platform for Computational Chemistry Analysis of Noncovalent Interactions. *J. Chem. Phys.* **2017**, *147*, 161727.