JOURNAL OF CHEMICAL EDUCATION

pubs.acs.org/jchemeduc

Technology Report

Cookiecutter for Computational Molecular Sciences: A Best Practices Ready Python Project Generator

Levi N. Naden,* Jessica Nash, T. Daniel Crawford, and Ashley Ringer McDonald

Cite This: *J. Chem. Educ.* 2024, 101, 5105−5109

Read Online

ACCESS | Metrics & More | Article Recommendations

ABSTRACT: Scientific software development takes far more than good programming abilities and scientific reasoning. Concepts such as version control, continuous integration, packaging, deployment, automatic documentation compiling, licensing, and even file structure are not traditionally taught to scientific programmers. The skill gap leads to inconsistent code quality and difficulty deploying products to the broader audience. Most of the implementation of these skills however can be constructed at project inception. The Cookiecutter for Computational Molecular Sciences generates ready-to-go Python projects that incorporate all of the concepts above from a single command. The final product is then a software project which lets developers focus on the science and minimizes worries about nonscientific and nonprogramming concepts because the best practices, as established by the Molecular Sciences Software Institute, have already been incorporated for them. This is a community driven project with widespread adoption across the computational molecular sciences. The Molecular Sciences Software Institute and Computational Molecular Sciences community also continually contribute and update the Cookiecutter for Computational Molecular Science, ensuring that the project is responsive to community needs and tool updates. All are welcome to suggest changes and contribute to making this the best starting point for Python-based scientific code.

KEYWORDS: *Python, Cookiecutter, Software Development, GitHub, CI, Deployment, Graduate Education/Research, Continuing Education, Interdisciplinary/Multidisciplinary, Computer-Based Learning, Distance Learning/Self Instruction, Computational Chemistry*

1. INTRODUCTION

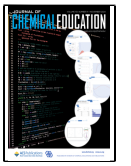
Scientific enterprise in the modern day requires software for everything from data acquisition and analysis to modeling and data generation, and that software has to be written by someone. Much of this code is so-called *end-user* software, written by scientists (as opposed to professional software developers) to be used by other researchers in their field. These end-user developers vastly outnumber professional software engineers[1] and produce important software projects in the chemistry discipline.

Creating useful software is nontrivial. For users other than the original developer to be able to use a scientific software package, there are many considerations, including documentation and user guides beyond in-line code and comments, version control, testing to ensure not only programmatic accuracy but also scientific regression accuracy, correct code formatting to allow future development of the software, and packaging with formal deployment that makes the code easy to install and use. This set of skills comprises what the Molecular Sciences Software Institute has identified as our "best practices" for molecular sciences software development.

These concepts are not typically taught to students in the undergraduate or graduate chemistry curriculum. Rather, students develop these skills informally, many times by learning from other members of their research group, which unfortunately often means adoption of best practices is inconsistent. We find that new software projects often simply copy the configuration and files from older, existing projects. This results in many deprecated methods and ideas, as well as bad ones, perpetuating while new and better approaches are never discovered.

To address this knowledge gap, the Molecular Sciences Software Institute (MolSSI) has created a Python package template, called the MolSSI Cookiecutter for Computational Molecular Sciences (CC-CMS) (available at https://github.

Received: June 27, 2024
Revised: October 18, 2024
Accepted: October 21, 2024
Published: October 29, 2024



https://doi.org/10.1021/acs.jchemed.4c00793
J. Chem. Educ. 2024, 101, 5105−5109

com/MolSSI/cookiecutter-cms/), to succinctly and efficiently teach chemistry students and professionals the most critical best practices in software development that will help them write more useful, usable, maintainable, and sustainable chemistry software. This can help ease the burden of the nuances in software development that can otherwise be a distraction from the primary scientific purpose of the software. The MolSSI CC-CMS identifies a formalized set of best practices, provides a curriculum to teach them consistently, and creates a structure to build these best practices into a software project *from the start*. This drastically simplifies the cognitive overhead to developing scientific software for not only our students, but also internally with our own projects. Software that is more sustainable will ultimately benefit the chemistry enterprise, enabling new science and reducing duplication of effort, just as it does in most fields that deploy software.[2]

The Cookiecutter for Computational Molecular Sciences (CC-CMS) Python Package created by MolSSI generates skeletal starting repositories for new Python-based scientific software projects with all of MolSSI's best practices built in. This tool will aid users of all skill levels in jump-starting their software development by incorporating and setting up nonscientific assets and tools that MolSSI teaches as best practices.[3] This project is constantly updated to reflect modern tools and skills, and we encourage and have received active contributions from the larger computational molecular sciences (CMS) community as well. We have extensively applied this at MolSSI ourselves and all of our new Python-based projects start from this tool.

2. SOFTWARE DESCRIPTION AND FEATURES

At its core, the Cookiecutter for Computational Molecular Sciences (CC-CMS) generates a directory on a user's computer with all of the configurations and file structures needed to implement the best practices in a Python software project. The folder and file structure are generated with the powerful Cookiecutter program, a generalized software template creation tool.[4] One of the most important requirements of any scientific software is reproducibility, and that includes with the software itself. This is accomplished by using version control software, such as Git. To that end, the CC-CMS automatically creates a Git directory of its output that is ready to be deployed to GitHub, GitLab, or any other service supporting Git for version control. The CC-CMS combines numerous preconfigured tools and as many MolSSI best practices as possible together into a new file tree, version controlled through Git, that can serve as the foundation, or roots, of a new project, which we have represented in Figure 1.

The CC-CMS provides features and tools for software development that most computational molecular scientists, chemists, cheminformatics specialists, and others will likely not encounter in their formal scientific training. We developed this software as a response to a combination of the MolSSI staff's own personal frustrations with software development and from the struggles we were hearing from external collaborators, contacts, principal investigators, and their students in public forums. It was clear there was a need for better software development practices to be more readily available to employ without needing to scour the internet, parse through an oversaturated amount of tutorials and suggestions, and then figure out how to apply those ideas locally. That was when we found the Cookiecutter program[4] which provides the framework for creating and deploying project structures. Cookiecutter itself does not have any of the content for the structures it asks users to design, that falls on us. So, we leveraged this framework, along with our internal expertise, to directly address the pain points we had been seeing in the community and create a starting template Python directory which also has the following features:

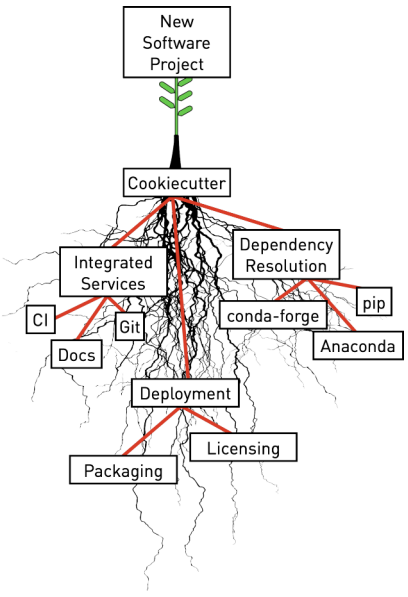


Figure 1. Pictorial representation of key tools and elements of a new project generated by the Cookiecutter for Computational Molecular Sciences. Every tool—from documentation, to dependency resolution, to packaging—works to create a foundation for new software projects to flourish while automating tedious, often difficult-to-understand nonscientific concepts in software development. Base image licensed under the CC0 1.0 Public Domain License.[5]

2.1. Preconfigured, Autoversioned, Python Installation Setup File `pyproject.toml`

The Python installation configuration file is not a simple object to create from scratch or even to copy from existing projects. This is particularly true now that it is recommended by the Python Package Authority to use the `pyproject.toml` instead of the legacy, but much more commonly seen, `setup.py`.[6] We have included a fully documented setup file with the conditions to include or exclude specific settings. Further, we have included integration with Git's tagging system to automatically assign version numbers to the software as it is released so users do not accidentally forget to update their version numbers between releases.

2.2. GitHub Actions Ready Continuous Integration (CI) Files

Automatic testing of code, both syntactically and scientifically, is critical to ensuring accurate and bug-free software. One of the best times to execute such tests is when code is changed and committed to the version control system, triggering a CI event. We have chosen to provide the configuration files for GitHub Actions, although many other quality CI services exist. This way, once a user of the CC-CMS uploads their new project to GitHub, the CI event triggers and GitHub will automatically run preconfigured tests on Linux, OSX, and Windows environments with multiple Python versions.

https://doi.org/10.1021/acs.jchemed.4c00793
J. Chem. Educ. 2024, 101, 5105−5109

2.3. Automatic Dependency sourcing

The Python ecosystem of packages is distributed over a large number of providers. Even to expert Python users, sourcing dependencies for your project can be a daunting and nuanced task. We have integrated the CC-CMS to be able to pull from `pip`,[7] `conda-forge`[8] or Anaconda's `main` channel.[9] The user is given the choice of which to apply to their project at creation time, and this can be changed later if they so choose. These configurations are also applied to the sample documentation for the package, and to the CI configuration automatically.

2.4. Ready-to-Write Documentation Configurations

Documentation is often as important as the code itself. Configuring documentation, however, can be difficult, even when copying from previous code bases. The CC-CMS provides a documentation structure that requires no user input to be ready to build docs in static web pages through the Sphinx build system.[10] This allows users to simply start writing instead of configuring. There are future plans to allow deployment to GitHub Pages as well.

2.5. Licensing from the Beginning

Scientific software projects must have a legal license provided to end-users just like commercial software. Doing so protects the developers from malicious actors as well as frivolous lawsuits to a certain extent. The CC-CMS creates the license file based on user input at the outset, before the first line of code is written. Although this may seem like a trivial mater, it is often an overlooked one, especially for users who have never created a product with the intent for it to be deployed.

3. IMPACT ON THE COMMUNITY AND EDUCATIONAL CURRICULA

The MolSSI CC-CMS is taught to approximately 200 students per year directly through MolSSI educational programming, particularly through the MolSSI Best Practices in Software Development workshop.[11] In this short course, students use the CC-CMS to set up a Python package and learn a collaborative code development workflow using Git and GitHub. The workshop is offered each summer for our new MolSSI Software Fellows,[12] as well as one or two other times during the year at community-organized workshops, where a university or group invites MolSSI to present the course on their campus, or at events held in conjunction with existing computational chemistry conferences.

In our MolSSI short courses where we teach the CC-CMS, students learn the proper application of each CC-CMS feature. A one to two hour lesson is provided for each one of the features and concepts, including version control, code testing, python code style, dependency sourcing, documentation writing, and continuous integration. These lessons can be found on our MolSSI Education Web site at https://education.molssi.org/python-package-best-practices/. The lessons are taught in an interactive style where the CC-CMS provides the initial content and the highly regularized structures, and the students populate the repository with specific information for their particular research project using these features. In this way, we attempt to maximize the CC-CMS's effectiveness and instill both an appreciation and competency for the underlying features within our tool.

Many more students find the CC-CMS through our online resources.[11] In total, the CC-CMS appears in over 1600 GitHub repositories as of April 29, 2024. Furthermore, the CC-CMS has been actively accessed continuously since its release in March 2018. We have tracked the number of times the CC-CMS has been both viewed and downloaded since February 2019 and plotted these results in Figure 2. The project has over 10,000 downloads from unique IP addresses and is a highly successful educational project from MolSSI.

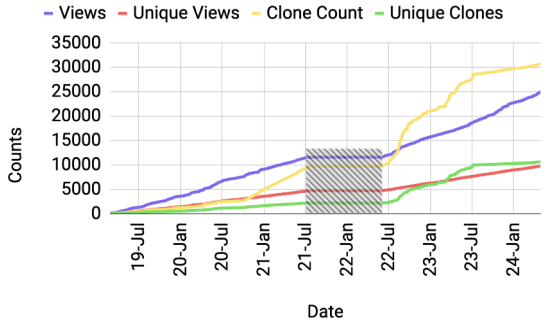


Figure 2. Cumulative GitHub activity of the MolSSI CMS Cookiecutter from February 2019 to April 2024. Data tracked are the number of visits to the base repository through GitHub's Web site (Views) and the number of times the CMS Cookiecutter has been downloaded (Clones). Also tracked are the number of unique views and clones, based on distinct IP addresses rather than multiple single-source accesses such as those arising from repeated services such as automated CI and classroom-related downloads. Tracking data were lost (failed hard drive) for the hatch-marked time frame and thus do not contribute to the totals.

The CC-CMS has also been used by faculty to train their research teams. Though this is a slightly different context than using the tool in a classroom course, this is nonetheless an important educational application of the CC-CMS. For example, one of the authors (ARM) has used the CC-CMS with her team of undergraduate research students to develop a Python workflow tool that enables analysis of molecular interactions in structures from molecular dynamics simulations. The package is published on GitHub and a manuscript is currently under review at the Journal of Open Source Software. In another case, Prof. Jay Foley, a faculty member at University of North Carolina at Charlotte, had their research team of two undergraduate chemistry majors and two masters students participate in MolSSI best practices workshop, learning to use the CC-CMS to build a Python package utilizing modern software development practices and tools including documentation generation in Sphinx, unit testing integrated with version control tools using GitHub Actions, and a standard protocol for features updates and fixes using pull requests. After completing the workshop, the research team mapped out how to apply these practices to an existing software package the group develops, called `WPTherml`.[13] This led to a rewrite of the majority of the code in that package starting from the CC-CMS. The faculty member now regularly uses the lessons from the Best Practices in Python Package Development workshop as one of the onboarding activities for the research team, where each student must build an example Python package from the CC-CMS.[14]

The CC-CMS has also been used as the basis for programming projects assigned in chemistry courses. For example, in a quantum software course at Virginia Tech the entire class uses the CC-CMS for a class project writing a package to compute thermodynamics properties of the Ising

https://doi.org/10.1021/acs.jchemed.4c00793
J. Chem. Educ. 2024, 101, 5105−5109

model. In the second course in the series, the students used the CC-CMS to create new software for a quantum computing research project that they create throughout the semester.[15]

In the wider CMS community, the CC-CMS serves as the basis for a growing number of widely used open-source projects and tutorials. For example, MDAKits from MDAnalysis[16] have their own Cookiecutter template based on the CC-CMS. The Riniker lab has used the project as a starting point for their package Ensembler[17] - a molecular dynamics prototyping package. Another notable example is from Dr. Patrick Walters for a package of useful RDKit utilities.

We have also collected internal survey information to directly measure the impact of the CC-CMS on student learning. For every workshop taught at MolSSI, we post voluntary exit surveys which ask for both numerical ratings and free responses from participants. As of September 6, 2024; there have been 350 numerical ratings given to the question "Rate the usefulness of [the CC-CMS]" on a scale of 0−10 with 10 being "extremely useful." The CC-CMS has an average of an 8.8 rating to this question. When the CC-CMS appears in the open questions, its most commonly in response to our question "What is something you hadn't expected to learn about but found useful/enjoyable?" Overall, the responses are overwhelmingly positive.

We want to note that these specific examples are ones we are aware of or are from faculty and researchers who contacted us to let us know they were using the CC-CMS; they are representative examples of how the CC-CMS has and could be used, not a comprehensive list.

4. ACTIVE DEVELOPMENT

The CC-CMS is in constant active development to keep up with advancements in development, and has been for years. The current version of the CC-CMS is 1.10.0, at the time of writing. The versions represent years of third-party tool advancement and updated best practices according to both MolSSI[11] and the broader CMS community. For example, the original version of the CC-CMS supported Python 2.7, 3.5, and 3.6; whereas now Python 3.12 is supported along with its two predecessors. As another example, The CI back-end has also evolved with time to transition from Travis-CI[18] and AppVeyor[19] to the single GitHub Actions platform as features and ease of access were considered. The CC-CMS cannot and will not ever enforce certain tools or versions, only recommendations and initial setups are provided because us developers want to leave all final decisions up to the scientists using this tool.

Best practices are difficult to keep up to date on, due to regular updates of the backend tools, but the CC-CMS attempts not to overwhelm users by having a slower update cycle. The development cycle for this tool releases major versions only once a year, or for any large bug fixes as updating the best practices take additional time to have community buy in. MolSSI does not seek to be the definitive authority on such things, so we invite those from outside our institute to help steer the feature development for the CC-CMS.

Every year, the decisions as to what are the best practices to include in the CC-CMS are determined with input from the CMS community. Although MolSSI is the primary developers of this tool, MolSSI is not the absolute authority on what are the best practices. We seek constant input from the CMS community as to what features, and what concepts, they want to see incorporated into the tool through an annual development sprint. These sprints include both MolSSI Software Scientists, along with specifically invited third party developers as part of an open invite to the CMS community. In each sprint, the best practices educational material is debated, decided, and updated;[11] then the relevant concepts are transferred to the CC-CMS. Some additions explicitly from the community have included: better Integrated Development Environment (IDE) connections, addition of GitHub Actions as a supported CI service, and implementation of PEP 621[20] and full migration to `pyproject.toml`;[6] just to name a few. The most recent development sprint from November 2023 updated the majority of third-party tools, documentation themes for more visually pleasing outputs, and fixed a number of depreciated features so our users do not have to think about all the upstream changes that get in the way of their work.

5. A CALL TO THE COMMUNITY TO CONTRIBUTE

Software development never happens in a vacuum. Today's scientific software ecosystem relies on countless upstream tools, third-parties, and decades of services and infrastructure changes to be successful. Yes, a developer could write their code on their own and distribute it purely by themselves; but with the widespread adoption of open source software, distribution platforms such as PyPI (through `pip`[7]) or Conda (through `conda-forge`[8,9]), and the services to handle testing and CI; more of that developer's time can be spent on the science, and less on everything else which makes software usable.

The CC-CMS likewise is not developed in a vacuum and we at MolSSI call on the community to help improve this package. We fully acknowledge we will not solve all the distractions which impede scientific software development, but we can try to alleviate as many as we can. The best way for us to serve the community with this package is for the community, and those reading this paper, to tell us what features they would like to see and/or pain points they have with scientific software development we as MolSSI can investigate solutions for. Direct contributions and development are also welcome at any time for discussion. Opening an issue or pull request on the CC-CMS GitHub[21] is the most effective way to communicate on this issue and we encourage anyone to comment and contribute. We are especially interested in discussing developing the CC-CMS for other languages beyond Python such as C++, Fortran, and Julia. Any discussions and suggestions for additional languages is highly encouraged and we openly welcome all discourse though the CC-CMS GitHub project or contacting MolSSI directly.

AUTHOR INFORMATION

Corresponding Author

Levi N. Naden − *Molecular Sciences Software Institute, Virginia Tech, Blacksburg, Virginia 24060, United States;* orcid.org/0000-0002-3692-5027; Email: info@molssi.org

Authors

Jessica Nash − *Molecular Sciences Software Institute, Virginia Tech, Blacksburg, Virginia 24060, United States;* orcid.org/0000-0003-1967-5094

T. Daniel Crawford − *Molecular Sciences Software Institute, Virginia Tech, Blacksburg, Virginia 24060, United States;* orcid.org/0000-0002-7961-7016

https://doi.org/10.1021/acs.jchemed.4c00793
J. Chem. Educ. 2024, 101, 5105−5109

Ashley Ringer McDonald − *Department of Chemistry and Biochemistry, Bailey College of Science and Mathematics, California Polytechnic State University, San Luis Obispo, California 93407, United States;* orcid.org/0000-0002-4381-1239

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jchemed.4c00793

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

The Molecular Sciences Software Institute is supported by the National Science Foundation under Grant CHE-2136142. The Cookiecutter for Computational Molecular Sciences has also received significant development by Daniel G. A. Smith, Jaime Rodríguez-Guerra Pedregal, M. Eric Irrgang, The John Chodera Lab, and the wider computational molecular sciences community.

■ REFERENCES

(1) Ko, A. J.; Abraham, R.; Beckwith, L.; Blackwell, A.; Burnett, M.; Erwig, M.; Scaffidi, C.; Lawrance, J.; Lieberman, H.; Myers, B.; Rosson, M. B.; Rothermel, G.; Shaw, M.; Wiedenbeck, S. The State of the Art in End-User Software Engineering. *ACM Comput. Surv.* **2011**, *43*, 1−44.

(2) Kan, S. H. *Metrics and Models in Software Quality Engineering*, 2nd ed.; Addison-Wesley Longman, 2002.

(3) Nash, J. A.; Mostafanejad, M.; Crawford, T. D.; McDonald, A. R. MolSSI Education: Empowering the Next Generation of Computational Molecular Scientists. *Computing in Science & Engineering* **2022**, *24*, 72−76.

(4) Roy, A.; Cookiecutter Community *Cookiecutter: Better Project Templates*. https://cookiecutter.readthedocs.io/ (accessed 2023-10-20).

(5) GDJ What Lies Beneath. *OpenClipArt* 2018. (Accessed 2023-10-23.) (This work is licensed under the Creative Commons Zero 1.0 Public Domain License).

(6) Python Software Foundation. *Python Packaging User Guide*, 2020. https://packaging.python.org/ (accessed 2023-10-20).

(7) Python Software Foundation. *PyPI—The Python Package Index*. https://pypi.org/ (accessed 2023-10-20).

(8) conda-forge community *The conda-forge Project: Community-Based Software Distribution Built on the conda Package Format and Ecosystem*, 2015. https://zenodo.org/records/4774217.

(9) *Anaconda*, 2016. https://anaconda.com. (accessed October 23, 2023).

(10) Sphinx Developers *Sphinx*. https://www.sphinx-doc.org/ (accessed 2023-10-20).

(11) The Molecular Sciences Software Institute *Best Practices in Python Package Development*, 2023. https://education.molssi.org/python-package-best-practices/. (accessed October 19, 2023).

(12) The Molecular Sciences Software Institute The MolSSI Fellowship Program. https://molssi.org/fellowship/. (accessed June 25, 2024).

(13) Varner, J. F.; Eldabagh, N.; Volta, D.; Eldabagh, R.; Foley, J. J. WPTherml: A Python Package for the Design of Materials for Harnessing Heat. *J. Open Res. Software* **2019**, *7*, 28.

(14) Foley, J. Personal Communication, November 2024. Printed with permission. (October 22, 2023).

(15) Mayhal, N. Personal Communication, November 2024. Printed with permission. (October 24, 2023).

(16) Alibay, I.; Barnoud, J.; Beckstein, O.; Gowers, R. J.; Loche, P. R.; MacDermott-Opeskin, H.; Matta, M.; Naughton, F. B.; Reddy, T.; Wang, L. Building a community-driven ecosystem for fast, reproducible, and reusable molecular simulation analysis using mdanalysis. *Biophysical journal* **2023**, *122*, 420a.

(17) Ries, B.; Linker, S. M.; Hahn, D. F.; König, G.; Riniker, S. Ensembler: A Simple Package for Fast Prototyping and Teaching Molecular Simulations. *J. Chem. Inf. Model.* **2021**, *61*, 560−564.

(18) Idera, Inc. *Travis-CI*. https://www.travis-ci.com/ (accessed October-2023-10-20).

(19) AppVeyor Systems Inc. *AppVeyor*. https://www.appveyor.com/, (accessed 2023-10-20).

(20) Cannon, B.; Ingram, D.; Ganssle, P.; Gedam, P.; Eustace, S.; Kluyver, T.; Chung, T. *PEP 621—Storing project metadata in pyproject.toml*, 2020. https://peps.python.org/pep-0621/

(21) The Molecular Sciences Software Institute *Cookiecutter for Computational Molecular Sciences (CMS) Python Packages*. https://github.com/MolSSI/cookiecutter-cms. (accessed April 29, 2024).