TeachOpenCADD 2021: Open Source and FAIR Python Pipelines to Assist in Structural Bioinformatics and Cheminformatics Research

Dominique Sydow[1, †], **Jaime Rodríguez-Guerra**[1, †], **Talia B. Kimber**[1], **David Schaller**[1], **Corey J. Taylor**[1], **Yonghui Chen**[1], **Mareike Leja**[1], **Sakshi Misra**[1], **Michele Wichmann**[1], **Armin Ariamajd**[1], **Andrea Volkamer**[1]

[1] *In Silico* Toxicology and Structural Bioinformatics, Institute of Physiology, Charité – Universitätsmedizin Berlin, corporate member of Freie Universität Berlin and Humboldt-Universität zu Berlin, Augustenburger Platz 1, 13353 Berlin, Germany

***For correspondence:**
andrea.volkamer@charite.de (AV)

[†]These authors contributed equally to this work

Abstract
Computational pipelines have become a crucial part of modern drug discovery campaigns. Setting up and maintaining such pipelines, however, can be challenging and time-consuming — especially for novice scientists in this domain. TeachOpenCADD is a platform that aims to teach domain-specific skills and to provide pipeline templates as starting points for research projects. We offer Python-based solutions for common tasks in cheminformatics and structural bioinformatics in the form of Jupyter notebooks and based on open source resources only. Including the 12 newly released additions, TeachOpenCADD now contains 22 notebooks that each cover both theoretical background as well as hands-on programming. To promote reproducible and reusable research, we apply software best practices to our notebooks such as testing with an automated continuous integration and adhering to a more idiomatic Python style. The new TeachOpenCADD website is available at https://projects.volkamerlab.org/teachopencadd and all code is deposited on GitHub.

Introduction
Computational methods play an integral role in the design-make-test-analyze (DMTA) cycle that drives real-world drug design projects [1]. To address questions raised during this cycle, a single method does not suffice to deliver an answer; instead, a pipeline combining different methods can produce complementary and useful insights. Setting up such complex pipelines, however, can be difficult and time-consuming for many reasons: the scientist may not have had the training necessary to tackle these tasks [2], tools and their usage are constantly evolving (or deprecating), and feeding the output from one tool into another is often not straightforward. On top of these considerations, sustainable pipelines need to be findable, accessible, interoperable, and reusable (FAIR principles [3]) — not only today but in many years from now — to drive reproducible research.

In 2019, we launched the teaching platform TeachOpenCADD [4] on GitHub to help face these challenges (https://github.com/volkamerlab/teachopencadd). TeachOpenCADD teaches by example how to build pipelines with open source resources used in the fields of cheminformatics and structural bioinformatics to answer central questions in computer-aided drug design (CADD). The code is written in Python, a popular programming language in many fields including computational life sciences. With these ready-to-use pipelines, we target students and teachers who need training material to CADD-related topics, as well as researchers who need a template or an inspiration to tackle their research questions. Usage can range from applying the whole pipeline on another molecular system to extracting selected pieces from the pipeline to be included in another context. The theoretical and practical aspects of each topic are covered in an interactive Jupyter notebook [5]. This setup makes it easy for users from different fields to understand the computational concepts and to get started with hands-on Python programming. We call these Jupyter notebooks *talktorials* (talk + tutorial) because their format is suited for presentations as well.

The initial stack of talktorials T001–T010 covers common CADD tasks involving webserver queries, cheminformatics, and structural bioinformatics [4]. We show how to fetch chemical and structural data from the ChEMBL [6] and PDB [7, 8] online databases and how to encode, filter, cluster, and screen such datasets to find novel drug candidates and off-targets [4] (Figure 1, T001–T010). The TeachOpenCADD platform enjoys wide usage in the community, as exemplified by frequently posted GitHub issues, about $13,000$ article views [9], over 250 GitHub repository stars and about 90 repository forks (as of 2021-10-14) [10] as well as teaching feedback [11]. The cheminformatics-focused talktorials are inspired by a variety of online resources, which we recommend for further reading; e.g. the virtual screening tutorials in context of the Teach-Discover-Treat initiative [12], and several blogs such as Practical Cheminformatics [13], the RDKit blog [14], Is life worth living? [15], and Cheminformania [16]. Another prominent example for a cheminformatics code collection is the Chemistry Development Kit (CDK) [17].

Over the last two years, the TeachOpenCADD GitHub repository underwent many additions and changes: we now have more than doubled our content and extended the application of software best practices rigorously. Twelve new Python talktorials in addition to the previous ten are available (Figure 1). The talktorials cover data retrieval via webserver queries from major structural, chemical and pharmacological databases including the PDB [7, 8], ChEMBL [6], PubChem [18], and KLIFS [19] databases. Important structural modeling tasks include protein pocket detection and comparison, protein-ligand docking and interaction analysis, as well as molecular dynamics simulations using tools such as ProteinsPlus [20, 21], Smina [22], PLIP [23], and OpenMM [24]. Throughout all talktorials, the 3D information is interactively visualized with NGLView [25, 26]. Finally, besides the previous examples for cheminformatics tasks such as molecular filtering, clustering, and similarity search using RDKit functionalities [27], we introduce deep learning (DL) for activity prediction and show how to train a model using the deep learning library Keras [28].

The full collection of talktorials is easily accessible on the new TeachOpenCADD website (https://projects.volkamerlab.org/teachopencadd). We comply with software best practices regarding the code style as well as maintenance and facilitate installation with a dedicated conda package [29]. In the following, we will discuss each of these additions in more detail.

New talktorials

The new stack of talktorials showcases data acquisition from additional CADD-relevant databases, adds many examples for structure-based tasks, and extends the cheminformatics side with straightforward DL applications (see Figure 1, T011–T022). Our example use case is the EGFR kinase [30] but the talktorials are easily adaptable to other targets as long as sufficient data is available. Besides the domain-specific resources described below, we rely in all talktorials on established Python packages for data science and data visualization such as NumPy [31], pandas [32], scikit-learn [33], matplotlib [34], and seaborn [35].

Webservices queries

Over the last decades, the scientific community has produced an incredible amount of data and analysis software, and adapted modern technologies to make these resources easily available via online webservices [36]. However, it might not always be obvious to the beginner how to use a web application programming interface (API) to access such data and how to integrate them into larger pipelines. TeachOpenCADD dedicates several talktorials to the usage of different webservers relevant for the life sciences.

In the first TeachOpenCADD release from 2019, we already showed how to query the ChEMBL [6] and PDB [7, 8] databases. From the ChEMBL webservice, compounds and bioactivities are fetched for the EGFR kinase using the ChEMBL webresource client [37] (T001). This dataset is used in many downstream talktorials for common cheminformatics tasks using primarily RDKit [27]. From the PDB webservice, we fetch a set of EGFR kinase structures based on criteria such as "ligand-bound structures from X-ray experiments

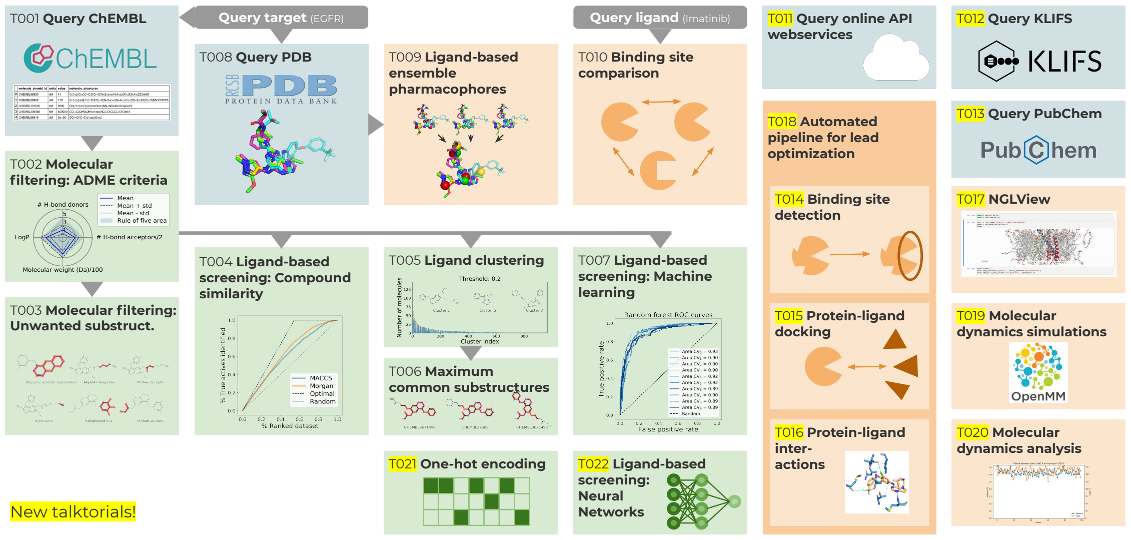
T001 Query ChEMBL
Query target (EGFR)
Query ligand (Imatinib)
T008 Query PDB
T009 Ligand-based ensemble pharmacophores
T010 Binding site comparison
T011 Query online API webservices
T012 Query KLIFS
T013 Query PubChem
T018 Automated pipeline for lead optimization
T014 Binding site detection
T017 NGLView
T002 Molecular filtering: ADME criteria
T003 Molecular filtering: Unwanted substruct.
T004 Ligand-based screening: Compound similarity
T005 Ligand clustering
T006 Maximum common substructures
T007 Ligand-based screening: Machine learning
T015 Protein-ligand docking
T019 Molecular dynamics simulations
T016 Protein-ligand inter-actions
T020 Molecular dynamics analysis
T021 One-hot encoding
T022 Ligand-based screening: Neural Networks
New talktorials!

Figure 1. Overview of all TeachOpenCADD topics from the 2019 and 2021 releases covering webserver queries (blue) and tasks from both cheminformatics (green) and structural bioinformatics (orange). New talktorials from the 2021 release are highlighted in yellow.

with a resolution below 3.0 Å" using the biotite [38] and PyPDB [39] (T008) packages. The structures in this dataset are aligned invoking OpenCADD's superposition module [40] and used to create a shared ligand-based pharmacophore model (T009). In T010, the PDB is queried to fetch all imatinib-bound structures and assess their pocket similarities — using the root-mean-square deviation (RMSD) of pocket residues — to study differences between the on-target ABL1 and its off-targets. In the 2021 release, we now have added three more notebooks covering the usage of online API webservices (Figure 2).

T011: Querying online API webservices. We give a broad introduction on how to programmatically use online webservices from Python with a focus on REST services and web scraping. The usage of several libraries is demonstrated; e.g. we use requests [41] to retrieve content from UniProt [42], bravado [43] to generate a Python client for any API — exemplified for the KLIFS database and its RESTful API [19, 44] —, and Beautiful Soup [45] to scrape (parse) HTML content from the web. Next, this newly gained knowledge is applied to interact with the KLIFS [19] and PubChem [18] databases.

T012: Data acquisition from KLIFS. KLIFS [19] is a kinase database gathering information about experimental kinase structures and interacting inhibitors. The talktorial shows how to quickly fetch data from KLIFS given a query kinase or ligand. For example, we spot frequent key interactions in EGFR based on KLIFS interaction fingerprints and we assess kinome-wide bioactivity values for gefitinib. These queries are demonstrated by using the KLIFS OpenAPI [44] directly with bravado [43], or by using the KLIFS-dedicated wrapper OpenCADD-KLIFS implemented in the Python package OpenCADD [40].

T013: Data acquisition from PubChem. PubChem [18] is a database holding chemical information for over 100 million compounds. We demonstrate how to fetch data from PubChem's PUG-REST API [46], given the name or SMILES [47] of a query ligand. For example, we show how to fetch molecular properties for a ligand of interest by name (aspirin) and how to query PubChem for the most similar compounds given a query SMILES (gefitinib).

A summary of the information that can be acquired automatically for a target of interest using these web services is exemplified in Figure 3 for the EGFR kinase, using talktorials T001, T008, T012, and T013.

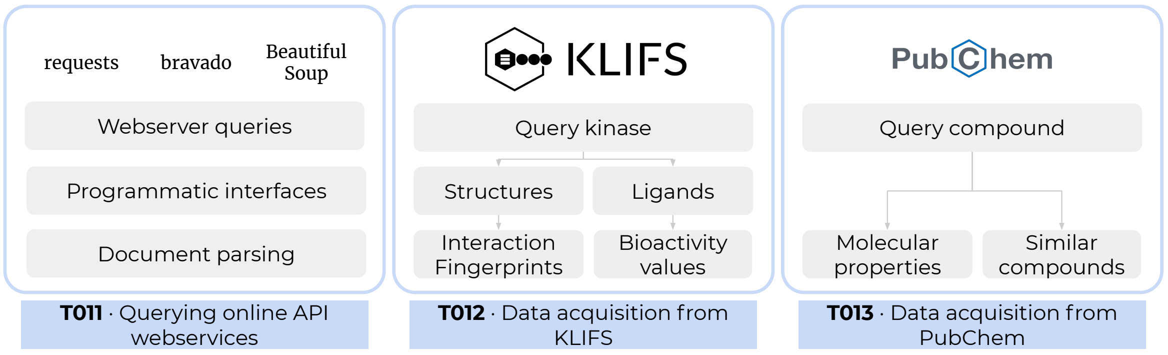

Figure 2. New talktorials on querying webservices. T011 gives a broad introduction on programmatic access to webservices from Python, T012 and T013 demonstrate how to query the KLIFS [19] and PubChem [18] databases for kinase and compound data, respectively.

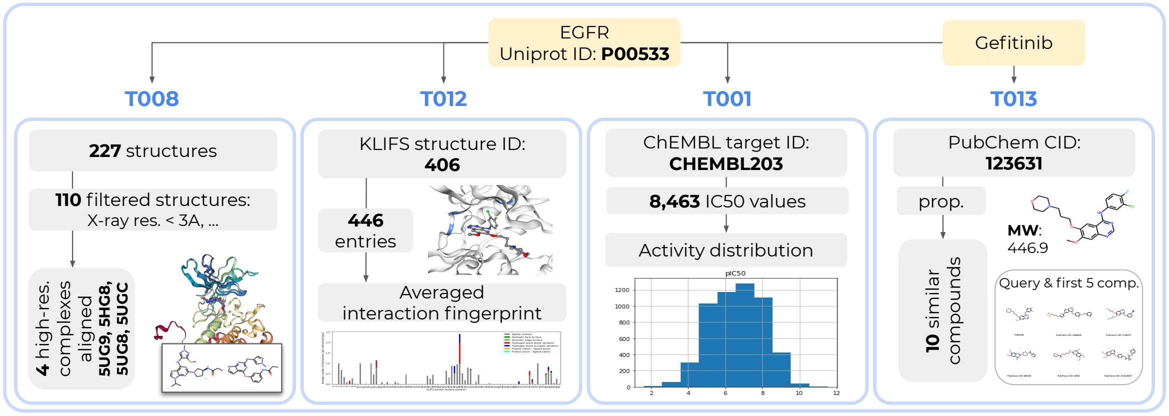

Figure 3. Data and information that can be automatically gathered for the EGFR kinase using the different web query talktorials as of September 2021, created based on ChEMBL v.27 [6] (T001), PDB [8] (T008), PubChem [18] (T013), and KLIFS [19] (T012). Input: yellow box, output: grey boxes, plots, and molecule visualizations (using NGLView [25] and RDKit [27]).

Pocket detection, ligand-protein docking and interactions

During a drug discovery campaign, frequent questions from a medicinal chemist to a cheminformatician are: What should I test next? Can you suggest a diverse set of small molecules likely to bind to this protein? Which parts of the lead compound should I start modifying with what kind of functional groups to increase the binding affinity? Answering these kinds of questions involves multiple scientific observations, and thus, multiple computational steps. Potentially fruitful steps are individually addressed in talktorials T014–T017. Finally, in T018, all steps are combined in an automated pipeline that — starting with a protein structure and a lead or hit compound — proposes several similar ligands with optimized estimated affinities and interactions based on the docked protein-ligand structures.

T014: Binding site detection. First, we need to know where ligands may bind to a protein of interest. Sometimes the binding site is known from experimental protein-ligand structures. If only experimental apo structures are available, putative binding sites can be predicted with computational methods. We demonstrate how to use the REST API of the ProteinsPlus webserver [20, 21] to detect the main pocket of an EGFR structure using the DoGSiteScorer [48] pocket detection algorithm. To validate our results, the predicted pocket is compared with the KLIFS-defined kinase pocket, which encompasses 85 residues in contact with ligands in over 2000 kinase-ligand structures [49].

T015: Protein-ligand docking. Next, we introduce molecular docking to predict the binding mode of a ligand to its protein target by explaining several sampling algorithms and scoring functions, as well as commenting on limitations and interpretation of docking result. The theoretical background is then applied in a re-docking experiment aiming to reproduce the binding mode observed in a published X-ray structure of EGFR. Protein and ligand are prepared using Pybel [50], the ligand is docked into the protein using Smina [22], and finally, the docking poses can be visually inspected using NGLView [25]. We refer to JupyterDock [51] for further reading on different docking protocols run from Jupyter notebooks.

T016: Protein-ligand interactions. Understanding which forces and interactions drive molecular recognition is important for drug design [52]. In this talktorial, we give an introduction to relevant protein-ligand interactions and their programmatic detection using the protein-ligand interaction profiler PLIP [23]. To this end, all interactions in an EGFR-ligand complex fetched from the PDB are detected and visualized in 3D using NGLView.

T017: Advanced NGLView usage. Since the molecular visualization package NGLView is invoked in many talktorials, we give a dedicated overview of its usage and show some advanced cases on how to customize residue coloring, and how to create interactive interfaces with IPyWidgets [53]. In addition, access to the JavaScript layer NGL [26, 54] is exemplified to perform operations that are not exposed in the interface of the Python wrapper NGLView.

T018: Automated pipeline for lead optimization. All previous talktorials are composed of stand-alone tasks that can be completed independently. Proposing ligand modifications that will improve interaction patterns with target proteins in a complete end-to-end process, however, necessitates orchestration of code and concepts implemented in the previously discussed talktorials T014–T017. A docking pipeline is constructed in T018 that is comprised of both a step-by-step demonstration and a fully automated procedure. Given a query protein and a hit or lead compound, similar ligands fetched from PubChem are suggested based on generated docking poses, which show optimized affinity estimates and interaction profiles.

Molecular dynamics

Experimentally resolved structures offer immense insights for drug design but can only provide a static snapshot of the full conformational space that represents the flexible nature of biological systems. Molecular dynamics (MD) simulations approximate such flexibility *in silico* with a trajectory of atom positions over a series of time steps (frames). These trajectories thereby reveal a more detailed — albeit still incomplete — picture of drug-target recognition and binding by providing access to protein-ligand interaction patterns over time [55–57]. These insights can for example help in lead discovery to examine the stability and validity of a predicted ligand docking pose, and in lead optimization phases to estimate the effect of a chemical modification on binding affinity.

T019: MD simulations. In this talktorial, we explain the key concepts behind MD simulations and provide the code to run a short MD simulation of EGFR in complex with a ligand on a local machine or on Google Colab [58, 59], which allows for GPU-accelerated simulations. The protein and ligand are thereby separately prepared with pdbfixer [60] and RDKit [27], and subsequently combined using MDTraj [61] and openff-toolkit [62]. The simulation is run with OpenMM [24], a high-performance toolkit for molecular simulation including language bindings for Python. The talktorial produces a 100 ps trajectory if run on Google Colab. On a local machine, only 20 fs are generated by default to keep computational efforts reasonable. However, the simulation time can be increased by the user in case a local GPU is available. We refer to the work by Arantes et al. [63] for further reading on different MD protocols run with OpenMM using Jupyter notebooks on Google Colab.

T020: Analyzing MD simulations. We analyze and visualize the results of the trajectory generated with the previous talktorial using the Python packages MDAnalysis [64, 65] and NGLView. First, the protein is structurally aligned across all trajectory frames, followed by calculating the root-mean-square deviation (RMSD) for different system components; i.e. protein, backbone, and ligand. Then, we take a closer look at a selected interaction between ligand and protein atoms, showcasing the contribution of distance and angle to the hydrogen bond strengths.

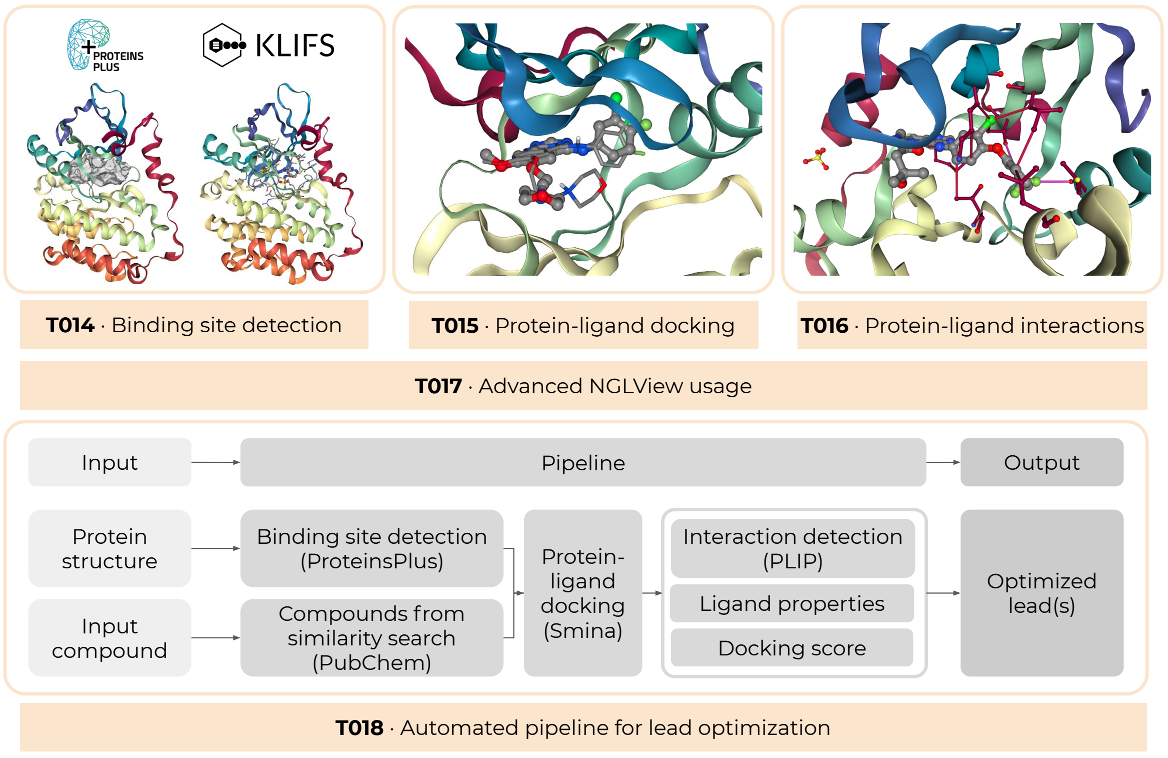
PROTEINS PLUS
KLIFS
T014 · Binding site detection
T015 · Protein-ligand docking
T016 · Protein-ligand interactions
T017 · Advanced NGLView usage
Input
Pipeline
Output
Protein structure
Binding site detection (ProteinsPlus)
Input compound
Compounds from similarity search (PubChem)
Protein-ligand docking (Smina)
Interaction detection (PLIP)
Ligand properties
Docking score
Optimized lead(s)
T018 · Automated pipeline for lead optimization

Figure 4. New talktorials on common tasks in structural bioinformatics to assess interaction patterns. T014 detects the binding site in an EGFR kinase structure and compares the prediction to the binding site defined by KLIFS [19]. T015 performs a re-docking for an EGFR-ligand complex with Smina [22]. T016 detects protein-ligand interactions in an EGFR-ligand complex structure with PLIP [23]. T017 introduces basic and advanced usages of the molecular visualization tool NGLView [25], which is used throughout most of TeachOpenCADD's talktorials. Finally, T018 outlines a fully automated lead optimization pipeline: Based on an input structure, the pocket is detected and a set of compounds similar to a selected ligand are fetched from PubChem [18]. These compounds are docked into the selected binding site. The most promising compounds w.r.t. docking scores and interaction profiles are proposed as optimized compounds.

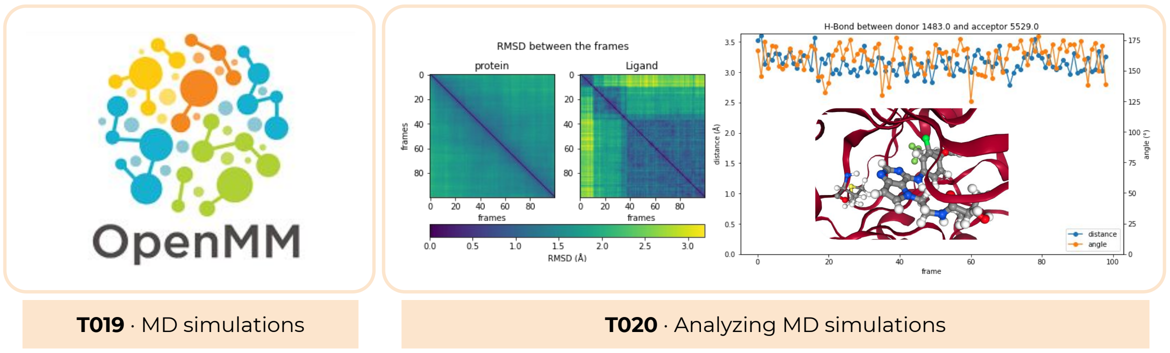

Figure 5. New talktorials on common tasks in structural bioinformatics to simulate dynamics. T019 demonstrates how to set up and run a molecular dynamics (MD) simulation on Google Colab [58, 59] with OpenMM [24]. T020 analyzes the resulting MD trajectory with a focus on the root-mean-square deviation (RMSD) between trajectory frames and the dynamics of protein-ligand interactions using MDAnalysis [64, 65].

Deep learning

Machine learning and more specifically deep learning have gained in popularity over the last few decades thanks to powerful computational resources such as GPUs, the research in the algorithms, and the growing amount of data [66]. Applications to CADD are diverse, ranging from molecular property prediction [67] to *de novo* molecular design [68]. Here, the focus is the featurization of molecular entities (T021) and ligand-based screening (T022).

T021: One-hot encoding. In CADD, machine learning algorithms require as input a numerical representation of small molecules. Besides molecular fingerprints (see T004), a popular featurization is the SMILES notation [47]. However, these representations are composed of strings and therefore cannot simply be input to an algorithm. One-hot encoding provides such a solution and the details are explained in this talktorial.

T022: Ligand-based screening: neural networks. We introduce the basics of neural networks and build a simple two-layer neural network. A model is trained on a subset of ChEMBL data to predict the pIC50 values of compounds against EGFR using MACCS [69] fingerprints as input. This talktorial is meant as groundwork for the understanding of neural networks. More complex architectures such as convolutional and recurrent neural networks will be explored in future notebooks. Such models may use the one-hot encoding of SMILES as input [70].

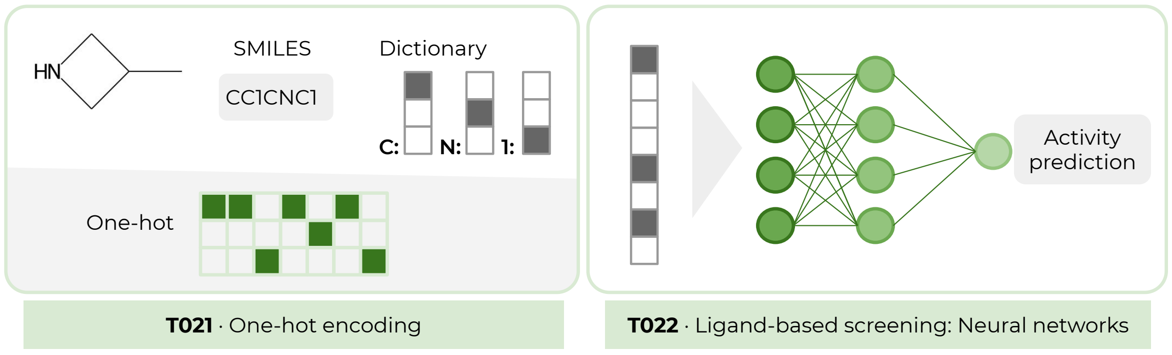

Figure 6. New talktorials on molecular encoding and deep learning. T021 exhibits the steps to numerically encode a small molecule from its SMILES representation. T022 lays the groundwork for deep learning and focuses on a simple feed-forward neural network for activity prediction using molecular fingerprints.

Best practices

In this enhanced version of TeachOpenCADD, we provide reliable and reproducible pipelines, periodically checked via automated testing mechanisms, and a streamlined and easy-to-understand code style across all talktorials. Everyone benefits from this work: the maintainers of the project, the contributors (to easily participate), as well as users (to learn setting up FAIR-compliant pipelines).

Testing. Reproducibility is ensured by testing if the notebooks can run without errors and whether the output of specific operations can be reproduced. For this purpose, we use the tools pytest [71] and nbval [72].

Continuous integration. We are testing the talktorials regularly for Linux, OSX, and Windows and different Python versions on GitHub Actions [73]. This ensures identical behavior across different operating systems and Python versions and also spots issues like conflicting dependency updates or changing outputs.

Repository structure. The repository structure is based on the cookiecutter-cms template [74], which provides a Python-focused project scaffold with pre-configured settings for packaging, continuous integration, Sphinx-based documentation [75], and much more. We have adapted the template to our notebook-focused needs.

Code style. We aim to adhere to the PEP8 [76] style guide for Python code, which defines how to write idiomatic Python (Pythonic) code. Such rules are important so that new developers — or in our case talktorial users — can quickly read and understand the code. Furthermore, we use black-nb [77, 78] to format the Python notebooks compliant with PEP8.

Dependency updates. Since we published the first ten talktorials in 2019, webservices and tools have changed or easier solutions have become available. This led to many updates in the existing notebooks, of which two are highlighted here. First, we are now using a combination of biotite [38] and PyPDB [39] to select structures and fetch structure files from the PDB to offer the same functionalities after the major update of the PDB webservices API [79]. Second, in the initial release, PyMol [80] was used for molecular visualization in 3D and structural alignment. To facilitate and expand operations in 3D, we now implemented a combination of NGLView [25] to visualize molecules in 3D and OpenCADD [40] to align structures in 3D.

TeachOpenCADD usage

There are many ways to use the talktorials. If users simply want to go through the material, they can use the read-only version on our newly launched website at https://projects.volkamerlab.org/teachopencadd. If users would rather like to execute and modify the notebooks, two options are available. First, the Jupyter notebooks can be executed online thanks to the Binder integrations [81]. Second, the Jupyter notebooks can be executed locally using the new conda package.

New website. Firing up Jupyter notebooks can entail unexpected complications if one wants to simply read through a talktorial or wants to look something up quickly. To make the access easy and fast, we launched a new TeachOpenCADD website (https://projects.volkamerlab.org/teachopencadd). The website statically renders the talktorials for immediate online reading using sphinx-nb [82] and provides detailed documentation for local usage, contributions, and external resources. This access option is recommended if the user plans on reading the material only.

New Binder support. The Binder project offers a place to share computing environments [83] via a single link. The environment setup of TeachOpenCADD can take a couple of minutes but does not require any kind of action on the user's end. This access option is recommended if the user plans on executing the material but does not need to save the changes.

New conda package. To make the local installation of TeachOpenCADD as easy as possible, we offer a conda package that ships all Jupyter notebooks with all necessary dependencies. The installation instructions are lined out in the TeachOpenCADD documentation [84]. This access option is recommended if the user plans on adapting the material for individual use cases.

Conclusion

The increasing amount of data and the focus on data-driven methods call for reproducible and reliable pipelines for computer-aided drug design (CADD). Knowing how to access and use these resources programmatically, however, requires domain-specific training and inspiration. The TeachOpenCADD platform showcases webserver-based data acquisition and common tasks in the fields of cheminformatics and structural bioinformatics. The theoretical and programmatic aspects of each topic are outlined side-by-side in Jupyter notebooks (talktorials) using open source resources only. To foster FAIR research, we apply software best practices such as testing, continuous integration, and idiomatic coding throughout the whole project. The talktorials are accessible via our website, Binder, and conda package to accommodate different use cases such as reading, executing, and modifying, respectively. We believe that TeachOpenCADD is not only a rich resource for CADD pipelines and teaching material on computational concepts and programming but as well a good example of how to set up websites, automated testing, and packaging for notebook-centric repositories. TeachOpenCADD is a living resource; problems can be voiced via GitHub issues and contributions can be made in the form of pull requests on GitHub. TeachOpenCADD is meant to grow; everyone is welcome to add new topics. Whenever you explore a new topic for your work, we invite you to fill our talktorial template [85] with what one learns along the way and to submit it to TeachOpenCADD.

Code and data availability

- TeachOpenCADD website: https://projects.volkamerlab.org/teachopencadd/
- TeachOpenCADD GitHub repository: https://github.com/volkamerlab/teachopencadd

Funding

Note that the TeachOpenCADD project has been a group effort and has received no explicit funding, while the positions of individual authors were supported by diverse funding agencies. The Volkamer Lab received funding from the Bundesministerium für Bildung und Forschung (AV: grant number 031A262C), Deutsche Forschungsgemeinschaft (DS: grant VO 2353/1-1), the Stiftung Charité in the context of the Einstein BIH Visiting Fellow Project (TBK, JRG and CT), Bayer in the context of the MIAME project (DaS), and the China Scholarship Council Project (YC: grant number 201906210079).

Author Contributions

Conceptualization: DS, JRG, AV; Data Curation, Formal Analysis, Investigation, Software, Validation and Visualization: DS, JRG, TBK, DaS, CT, YC, ML, SM, MW, AA, AV; Funding Acquisition: AV; Methodology and Maintenance: DS, JRG, TBK, DaS, AV; Project Administration: DS, JRG, AV; Resources: AV; Supervision: DS, JRG, DaS, TBK, AV; Writing-Original Draft: DS, TK, DaS, JRG, AV; Writing - Review and Editing: DS, JRG, TBK, DaS, CT, YC, ML, SM, MW, AA, AV.

Disclosures

None.

Acknowledgements

The authors thank Piedro Gerletti (T019), Ahmed Atta (T022), Melanie Vogel (T018), Abishek Laxmanan Ravi Shankar (T014), Maria Trofimova (T015), and Jeffrey R. Wagner (T019) for the initial drafts of or contributions to the above-mentioned talktorials.

The authors are grateful to the PyPDB and biotite maintainers for their work on updating their packages according to the new RCSB PDB API (special thanks to Patrick Kunzmann), and Albert Kooistra for helping with questions regarding the KLIFS database. The authors thank Hai Nguyen for dedicated and helpful NGLView support. The authors are thankful for a fruitful hackathon at the RDKit UGM in 2019, where we started to tackle a few TeachOpenCADD enhancements: structural superimposition and visualization without PyMol with Richard Gowers and testing Jupyter notebooks with Floriane Montanari. Finally, the authors thank Pat Walters and Hai Nguyen for endorsing the TeachOpenCADD platform on their websites, and we thank Greg Landrum for giving TeachOpenCADD a spot at the RDKit UGMs 2019, 2020, and 2021!

References

[1] **Schneider P**, Walters WP, Plowright AT, Sieroka N, Listgarten J, Goodnow RA, Fisher J, Jansen JM, Duca JS, Rush TS, Zentgraf M, Hill JE, Krutoholow E, Kohler M, Blaney J, Funatsu K, Luebkemann C, Schneider G. Rethinking drug design in the artificial intelligence era. Nat Rev Drug Discovery. 2020; 19(5):353–364. doi: 10.1038/s41573-019-0050-3.

[2] **Ringer McDonald A**. Teaching Programming across the Chemistry Curriculum: A Revolution or a Revival? In: Teaching Programming across the Chemistry Curriculum American Chemical Society; 2021. p. 1–11. doi: 10.1021/bk-2021-1387.ch001.

[3] **Wilkinson MD**, Dumontier M, Aalbersberg IJ, Appleton G, Axton M, Baak A, Blomberg N, Boiten JW, da Silva Santos LB, Bourne PE, Bouwman J, Brookes AJ, Clark T, Crosas M, Dillo I, Dumon O, Edmunds S, Evelo CT, Finkers R, Gonzalez-Beltran A, et al. The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data. 2016; 3(1):160018. doi: 10.1038/sdata.2016.18.

[4] **Sydow D**, Morger A, Driller M, Volkamer A. TeachOpenCADD: A Teaching Platform For Computer-Aided Drug Design Using Open Source Packages And Data. J Cheminform. 2019; 11(1):29. doi: 10.1186/s13321-019-0351-x.

[5] **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, development team J. Jupyter Notebooks - a publishing format for reproducible computational workflows. In: Loizides F, Scmidt B, editors. *Positioning and Power in Academic Publishing: Players, Agents and Agendas* IOS Press; 2016. p. 87–90. doi: 10.3233/978-1-61499-649-1-87.

[6] **Mendez D**, Gaulton A, Bento AP, Chambers J, De Veij M, Félix E, Magariños M, Mosquera J, Mutowo P, Nowotka M, Gordillo-Marañón M, Hunter F, Junco L, Mugumbate G, Rodriguez-Lopez M, Atkinson F, Bosc N, Radoux C, Segura-Cabrera A, Hersey A, et al. ChEMBL: towards direct deposition of bioassay data. Nucleic Acids Research. 2018; 47(D1):D930–D940. doi: 10.1093/nar/gky1075.

[7] **Berman HM**, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, Shindyalov IN, Bourne PE. The Protein Data Bank. Nucleic Acids Research. 2000; 28(1):235–242. doi: 10.1093/nar/28.1.235.

[8] **Burley SK**, Bhikadiya C, Bi C, Bittrich S, Chen L, Crichlow GV, Christie CH, Dalenberg K, Di Costanzo L, Duarte JM, Dutta S, Feng Z, Ganesan S, Goodsell DS, Ghosh S, Green RK, Guranović V, Guzenko D, Hudson BP, Lawson C, et al. RCSB Protein Data Bank: powerful new tools for exploring 3D structures of biological macromolecules for basic and applied research and education in fundamental biology, biomedicine, biotechnology, bioengineering and energy sciences. Nucleic Acids Research. 2020; 49(D1):D437–D451. doi: 10.1093/nar/gkaa1038.

[9] **Journal of Cheminformatics**, TeachOpenCADD paper's access and citations. https://jcheminf.biomedcentral.com/articles/10.1186/s13321-019-0351-x/metrics, [Online; accessed 2021-10-05].

[10] **GitHub**, TeachOpenCADD's GitHub stars. https://github.com/volkamerlab/teachopencadd/stargazers, [Online; accessed 2021-10-05].

[11] **Sydow D**, Rodríguez-Guerra J, Volkamer A. Teaching Computer-Aided Drug Design Using TeachOpenCADD. In: Teaching Programming across the Chemistry Curriculum American Chemical Society; 2021. p. 135–158. doi: 10.1021/bk-2021-1387.ch010.

[12] **Riniker S**, Landrum G, Montanari F, Villalba S, Maier J, Jansen J, Walters W, Shelat A. Virtual-screening workflow tutorials and prospective results from the Teach-Discover-Treat competition 2014 against malaria [version 2; peer review: 3 approved]. F1000Research. 2018; 6(1136). doi: 10.12688/f1000research.11905.2.

[13] **Patrick Walters**, Practical Cheminformatics. https://patwalters.github.io/practicalcheminformatics/, [Online; accessed 2021-10-12].

[14] **Gregory Landrum**, RDKit blog. https://greglandrum.github.io/rdkit-blog/, [Online; accessed 2021-10-12].

[15] **iwatobipen**, Is life worth living? https://iwatobipen.wordpress.com/, [Online; accessed 2021-10-12].

[16] **Esben Jannik Bjerrum**, Cheminformania. https://www.cheminformania.com/, [Online; accessed 2021-10-12].

[17] **Willighagen EL**, Mayfield JW, Alvarsson J, Berg A, Carlsson L, Jeliazkova N, Kuhn S, Pluskal T, Rojas-Chertó M, Spjuth O, Torrance G, Evelo CT, Guha R, Steinbeck C. The Chemistry Development Kit (CDK) v2.0: atom typing, depiction, molecular formulas, and substructure searching. Journal of Cheminformatics. 2017; 9(1):33. doi: 10.1186/s13321-017-0220-4.

[18] **Kim S**, Chen J, Cheng T, Gindulyte A, He J, He S, Li Q, Shoemaker BA, Thiessen PA, Yu B, Zaslavsky L, Zhang J, Bolton EE. PubChem in 2021: new data content and improved web interfaces. Nucleic Acids Research. 2020; 49(D1):D1388–D1395. doi: 10.1093/nar/gkaa971.

[19] **Kanev GK**, de Graaf C, Westerman BA, de Esch IJP, Kooistra AJ. KLIFS: an overhaul after the first 5 years of supporting kinase research. Nucleic Acids Research. 2020; 49(D1):D562–D569. doi: 10.1093/nar/gkaa895.

[20] **Fährrolfes R**, Bietz S, Flachsenberg F, Meyder A, Nittinger E, Otto T, Volkamer A, Rarey M. ProteinsPlus: a web portal for structure analysis of macromolecules. Nucleic Acids Research. 2017; 45(W1):W337–W343. doi: 10.1093/nar/gkx333.

[21] **Computational Molecular Design Group headed by Matthias Rarey**, ProteinsPlus. https://proteins.plus/, [Online; accessed 2021-10-15].

[22] **Koes DR**, Baumgartner MP, Camacho CJ. Lessons Learned in Empirical Scoring with smina from the CSAR 2011 Benchmarking Exercise. Journal of Chemical Information and Modeling. 2013; 53(8):1893–1904. doi: 10.1021/ci300604z.

[23] **Salentin S**, Schreiber S, Haupt VJ, Adasme MF, Schroeder M. PLIP: fully automated protein–ligand interaction profiler. Nucleic Acids Research. 2015; 43(W1):W443–W447. doi: 10.1093/nar/gkv315.

[24] **Eastman P**, Swails J, Chodera JD, McGibbon RT, Zhao Y, Beauchamp KA, Wang LP, Simmonett AC, Harrigan MP, Stern CD, Wiewiora RP, Brooks BR, Pande VS. OpenMM 7: Rapid development of high performance algorithms for molecular dynamics. PLOS Computational Biology. 2017; 13(7):1–17. doi: 10.1371/journal.pcbi.1005659.

[25] **Nguyen H**, Case DA, Rose AS. NGLView - Interactive Molecular Graphics For Jupyter Notebooks. Bioinformatics. 2017; 34(7):1241–1242. doi: 10.1093/bioinformatics/btx789.

[26] **Rose AS**, Hildebrand PW. NGL Viewer: a web application for molecular visualization. Nucleic Acids Research. 2015; 43(W1):W576–W579. doi: 10.1093/nar/gkv402.

[27] **RDKit**, RDKit: Open-Source Cheminformatics. http://www.rdkit.org, [Online; accessed 2021-10-05].

[28] **Chollet F**, et al., Keras. https://keras.io, [Online; accessed 2021-10-26].

[29] **conda-forge community**, The conda-forge Project: Community-based Software Distribution Built on the conda Package Format and Ecosystem. doi: 10.5281/zenodo.4774216.

[30] **Herbst RS**. Review of epidermal growth factor receptor biology. International Journal of Radiation Oncology*Biology*Physics. 2004; 59(2, Supplement):S21–S26. doi: 10.1016/j.ijrobp.2003.11.041.

[31] **Harris CR**, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, Wieser E, Taylor J, Berg S, Smith NJ, Kern R, Picus M, Hoyer S, van Kerkwijk MH, Brett M, Haldane A, del Río JF, Wiebe M, Peterson P, Gérard-Marchant P, et al. Array programming with NumPy. Nature. 2020 Sep; 585(7825):357–362. doi: 10.1038/s41586-020-2649-2.

[32] **The pandas development team**, pandas-dev/pandas: Pandas. doi: 10.5281/zenodo.3509134.

[33] **Pedregosa F**, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel P M a Prettenhofer, Weiss R, Dubourg V, Vanderplas A J a Passos, Cournapeau D, Brucher M, Perrot E M a Duchesnay. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011; 12:2825–2830. http://jmlr.org/papers/v12/pedregosa11a.html.

[34] **Hunter JD**. Matplotlib: A 2D graphics environment. Computing in Science & Engineering. 2007; 9(3):90–95. doi: 10.1109/MCSE.2007.55.

[35] **Waskom ML**. seaborn: statistical data visualization. Journal of Open Source Software. 2021; 6(60):3021. doi: 10.21105/joss.03021.

[36] **Ireland SM**, Martin ACR. GraphQL for the Delivery of Bioinformatics Web APIs and Application to ZincBind. Bioinformatics Advances. 2021; doi: 10.1093/bioadv/vbab023.

[37] **Davies M**, Nowotka M, Papadatos G, Dedman N, Gaulton A, Atkinson F, Bellis L, Overington JP. Chembl Web Services: Streamlining Access To Drug Discovery Data And Utilities. Nucleic Acids Research. 2015; 43:W612–W620. doi: 10.1093/nar/gkv352.

[38] **Kunzmann P**, Hamacher K. Biotite: a unifying open source computational biology framework in Python. BMC Bioinformatics. 2018; 19(1):346. doi: 10.1186/s12859-018-2367-z.

[39] **Gilpin W**. PyPDB: A Python API For The Protein Data Bank. Bioinformatics. 2015 9; 32:159–60. doi: 10.1093/bioinformatics/btv543.

[40] **Volkamer Lab**, OpenCADD: Python package for structural cheminformatics. https://opencadd.readthedocs.io/en/latest/, [Online; accessed 2021-10-05].

[41] **requests**, requests. https://docs.python-requests.org/, [Online; accessed 2021-10-05].

[42] **Consortium TU**. UniProt: the universal protein knowledgebase in 2021. Nucleic Acids Research. 2020; 49(D1):D480–D489. doi: 10.1093/nar/gkaa1100.

[43] **bravado**, bravado. https://github.com/Yelp/bravado, [Online; accessed 2021-10-05].

[44] **KLIFS**, KLIFS OpenAPI. https://klifs.net/swagger/, [Online; accessed 2021-10-05].

[45] **Beautiful Soup**, Beautiful Soup Documentation. https://www.crummy.com/software/BeautifulSoup/bs4/doc/, [Online; accessed 2021-10-05].

[46] **Kim S**, Thiessen PA, Cheng T, Yu B, Bolton EE. An update on PUG-REST: RESTful interface for programmatic access to PubChem. Nucleic Acids Research. 2018; 46(W1). doi: 10.1093/nar/gky294.

[47] **Weininger D**. SMILES, A Chemical Language And Information System. 1. Introduction To Methodology And Encoding Rules. Journal of Chemical Information and Modeling. 1988; 28(1):31–36. doi: 10.1021/ci00057a005.

[48] **Volkamer A**, Kuhn D, Grombacher T, Rippmann F, Rarey M. Combining Global and Local Measures for Structure-Based Druggability Predictions. Journal of Chemical Information and Modeling. 2012; 52(2):360–372. doi: 10.1021/ci200454v.

[49] **van Linden OPJ**, Kooistra AJ, Leurs R, de Esch IJP, de Graaf C. KLIFS: A Knowledge-Based Structural Database To Navigate Kinase–Ligand Interaction Space. Journal of Medicinal Chemistry. 2014; 57(2):249–277. doi: 10.1021/jm400378w.

[50] **O'Boyle NM**, Banck M, James CA, Morley C, Vandermeersch T, Hutchison GR. Open Babel: An open chemical toolbox. Journal of Cheminformatics. 2011; 3(1):33. doi: 10.1186/1758-2946-3-33.

[51] **Moreno AJR**, Jupyter_Dock: v0.2.5. doi: 10.5281/zenodo.5514956.

[52] **Klebe G**. Protein–Ligand Interactions as the Basis for Drug Action. In: Drug Design: Methodology, Concepts, and Mode-of-Action Springer Berlin Heidelberg; 2013. p. 61–88. doi: 10.1007/978-3-642-17907-5_4.

[53] **IPyWidgets**, IPyWidgets Documentation. https://ipywidgets.readthedocs.io/en/latest/, [Online; accessed 2021-10-05].

[54] **Rose AS**, Bradley AR, Valasatava Y, Duarte JM, Prlić A, Rose PW. NGL viewer: web-based molecular graphics for large complexes. Bioinformatics. 2018; 34(21):3755–3758. doi: 10.1093/bioinformatics/bty419.

[55] **Mortier J**, Rakers C, Bermudez M, Murgueitio MS, Riniker S, Wolber G. The impact of molecular dynamics on drug design: applications for the characterization of ligand–macromolecule complexes. Drug Discovery Today. 2015; 20(6):686–702. doi: https://doi.org/10.1016/j.drudis.2015.01.003.

[56] **De Vivo M**, Masetti M, Bottegoni G, Cavalli A. Role of Molecular Dynamics and Related Methods in Drug Discovery. Journal of Medicinal Chemistry. 2016; 59(9):4035–4061. doi: 10.1021/acs.jmedchem.5b01684.

[57] **Salmaso V**, Moro S. Bridging Molecular Docking to Molecular Dynamics in Exploring Ligand-Protein Recognition Process: An Overview. Frontiers in Pharmacology. 2018; 9:923. doi: 10.3389/fphar.2018.00923.

[58] **Google Research**, Google Colab. https://colab.research.google.com/, [Online; accessed 2021-03-17].

[59] **Rodríguez-Guerra J**, condacolab. https://github.com/conda-incubator/condacolab.

[60] **pdbfixer**, pdbfixer. https://github.com/openmm/pdbfixer, [Online; accessed 2021-10-06].

[61] **McGibbon RT**, Beauchamp KA, Harrigan MP, Klein C, Swails JM, Hernández CX, Schwantes CR, Wang LP, Lane TJ, Pande VS. MDTraj: A Modern Open Library for the Analysis of Molecular Dynamics Trajectories. Biophysical Journal. 2015; 109(8):1528 – 1532. doi: 10.1016/j.bpj.2015.08.015.

[62] **Wagner J**, Mobley DL, Thompson M, Chodera J, Bannan C, Rizzi A, trevorgokey, Dotson D, Rodríguez-Guerra J, Camila, Behara P, Bayly C, Mitchell JA, JoshHorton, Lim NM, Lim V, Sasmal S, Wang L, Dalke A, SimonBoothroyd, et al. openforcefield/openff-toolkit: 0.10.0 Improvements for force field fitting; 2021, doi: 10.5281/zenodo.5153946.

[63] **Arantes PR**, Polêto MD, Pedebos C, Ligabue-Braun R. Making it Rain: Cloud-Based Molecular Simulations for Everyone. Journal of Chemical Information and Modeling. 0; 0(0):null. doi: 10.1021/acs.jcim.1c00998.

[64] **Michaud-Agrawal N**, Denning EJ, Woolf TB, Beckstein O. MDAnalysis: A toolkit for the analysis of molecular dynamics simulations. J Comput Chem. 2011; 32(10):2319–2327. doi: 10.1002/JCC.21787.

[65] **Richard J Gowers**, Max Linke, Jonathan Barnoud, Tyler J E Reddy, Manuel N Melo, Sean L Seyler, Jan Domański, David L Dotson, Sébastien Buchoux, Ian M Kenney, Oliver Beckstein. MDAnalysis: A Python Package for the Rapid Analysis of Molecular Dynamics Simulations. In: Sebastian Benthall, Scott Rostrup, editors. *Proceedings of the 15th Python in Science Conference*; 2016. p. 98 – 105. doi: 10.25080/Majora-629e541a-00e.

[66] **Goodfellow I**, Bengio Y, Courville A. Deep Learning. MIT Press; 2016. http://www.deeplearningbook.org.

[67] **Wu Z**, Ramsundar B, Feinberg EN, Gomes J, Geniesse C, Pappu AS, Leswing K, Pande V. MoleculeNet: a benchmark for molecular machine learning. Chem Sci. 2018; 9:513–530. doi: 10.1039/C7SC02664A.

[68] **Brown N**, Fiscato M, Segler MHS, Vaucher AC. GuacaMol: Benchmarking Models for de Novo Molecular Design. Journal of Chemical Information and Modeling. 2019; 59(3):1096–1108. doi: 10.1021/acs.jcim.8b00839.

[69] **Accelrys Inc , San Diego, CA, USA**, MACCS Structural Keys.

[70] **Kimber TB**, Engelke S, Tetko IV, Bruno E, Godin G. Synergy effect between convolutional neural networks and the multiplicity of SMILES for improvement of molecular prediction. arXiv preprint arXiv:181204439. 2018; https://arxiv.org/abs/1812.04439.

[71] **pytest**, pytest. https://docs.pytest.org/, [Online; accessed 2021-10-06].

[72] **nbval**, nbval. https://nbval.readthedocs.io/en/latest/, [Online; accessed 2021-10-06].

[73] **GitHub**, GitHub Actions. https://docs.github.com/en/actions, [Online; accessed 2021-10-06].

[74] **MolSSI**, cookiecutter-cms. https://github.com/MolSSI/cookiecutter-cms, [Online; accessed 2021-10-06].

[75] **sphinx**, sphinx - Python Documentation Generator. https://www.sphinx-doc.org/, [Online; accessed 2021-10-06].

[76] **Python Software Foundation**, Python Enhancement Proposal 8. https://www.python.org/dev/peps/pep-0008/, [Online; accessed 2021-10-06].

[77] **Python Software Foundation**, Black: The Uncompromising Python Code Formatter. https://github.com/psf/black, [Online; accessed 2021-10-06].

[78] **Black-nb**, Black-nb: The Uncompromising Code Formatter, for Jupyter Notebooks. https://github.com/tomcatling/black-nb, [Online; accessed 2021-10-06].

[79] **PDB R**, Legacy Fetch API Web Service. https://data.rcsb.org/migration-guide.html#legacy-fetch-api, [Online; accessed 2021-10-26].

[80] **Schrödinger L**, The PyMOL Molecular Graphics System, Version 1.8. https://pymol.org/.

[81] **Binder**, TeachOpenCADD on Binder. https://mybinder.org/v2/gh/volkamerlab/TeachOpenCADD/master, [Online; accessed 2021-10-06].

[82] **Sphinx-nb**, Sphinx-nb: Jupyter Notebook Tools for Sphinx. https://nbsphinx.readthedocs.io/, [Online; accessed 2021-10-06].

[83] **Binder**, Binder. https://mybinder.org/, [Online; accessed 2021-10-06].

[84] **TeachOpenCADD**, TeachOpenCADD installation instructions. https://projects.volkamerlab.org/teachopencadd/installing.html, [Online; accessed 2021-10-06].

[85] **Volkamer Lab**, TeachOpenCADD Talktorial Template. https://github.com/volkamerlab/teachopencadd/blob/master/teachopencadd/talktorials/T000_template/talktorial.ipynb, [Online; accessed 2021-03-11].