Available online at www.sciencedirect.com

ScienceDirect

Current Opinion in Structural Biology

Undergraduate structural biology education: A shift from users to developers of computation and simulation tools

Ashley Ringer McDonald[1], Rebecca Roberts[2], Julia R. Koeppe[3] and Bonnie L. Hall[4]

Abstract

The use of theory and simulation in undergraduate education in biochemistry, molecular biology, and structural biology is now common, but the skills students need and the curriculum instructors have to train their students are evolving. The global pandemic and the immediate switch to remote instruction forced instructors to reconsider how they can use computation to teach concepts previously approached with other instructional methods. In this review, we survey some of the curricula, materials, and resources for instructors who want to include theory, simulation, and computation in the undergraduate curriculum. There has been a notable progression from teaching students to use discipline-specific computational tools to developing interactive computational tools that promote active learning to having students write code themselves, such that they view computation as another tool for solving problems. We are moving toward a future where computational skills, including programming, data analysis, visualization, and simulation, will no longer be considered an optional bonus for students but a required skill for the 21st century STEM (Science, Technology, Engineering, and Mathematics) workforce; therefore, all physical and life science students should learn to program in the undergraduate curriculum.

Addresses

[1] Department of Chemistry and Biochemistry, California Polytechnic State University, San Luis Obispo, CA 93401, USA
[2] Department of Biology, Ursinus College, Collegeville, PA 19426, USA
[3] Department of Chemistry, SUNY at Oswego, Oswego, NY 13126, USA
[4] Department of Chemistry and Physics, Grand View University, Des Moines, IA 50316, USA

Corresponding author: McDonald, Ashley Ringer (armcdona@calpoly.edu)

Current Opinion in Structural Biology 2022, **72**:39–45

This review comes from a themed issue on **Theory and Simulation/ Computational Methods**

Edited by **Qiang Cui** and **Michelle McCully**

For a complete overview see the Issue and the Editorial

https://doi.org/10.1016/j.sbi.2021.07.012

Introduction

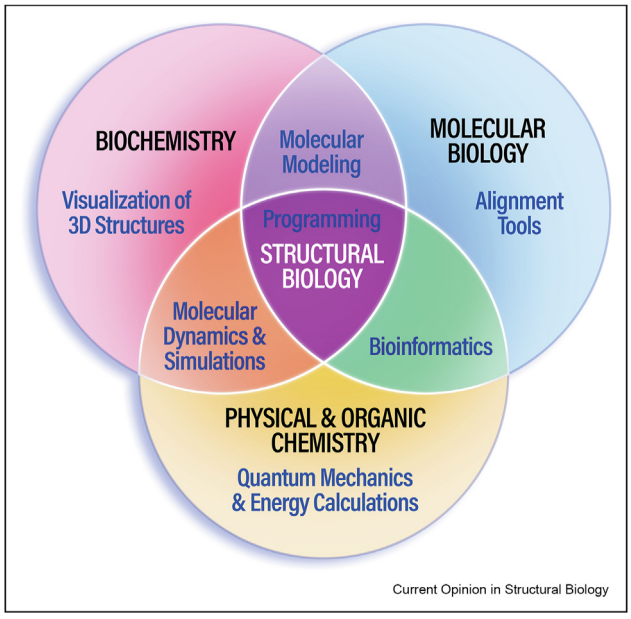
The use of theory and simulation in undergraduate education in chemistry, biology, and biophysics has changed dramatically over the last decade and even over the last several years. There is a growing recognition that the modern technical workforce requires virtually all students to have competency using computational tools to generate and analyze data, especially with the continually growing emphasis on data science and machine learning in all aspects of computational molecular science. We use the term computational molecular science (CMS) to refer to the wide range of disciplines that use computational tools to solve problems related to chemistry, biochemistry, molecular biology, structural biology, and even materials science, though the latter will not be the focus of this review. Figure 1 illustrates the intersection of these disciplines and methodologies.

The expansion of computational science in the undergraduate curriculum was certainly hastened by the SARS-CoV-2 global pandemic and the instantaneous pivot to emergency remote instruction for most universities around the world [1]. Educators took the opportunity to teach the computational tools of their discipline and to create new computational tools to teach other concepts, including some that were previously taught with wet laboratory techniques. In this review, we highlight some of the recent advances and exemplars that use theory, simulation, and computation in the undergraduate chemistry, biochemistry, molecular biology, structural biology, and bioinformatics curricula. We delineate the progression of these educational resources from those that are more 'plug and play' and teach students to use existing computational tools to those that teach students programming and have students write their own code to address chemical and biophysical problems.

Teaching computation in structural biology and biochemistry

A few decades ago, the thought of teaching an undergraduate student the nuances of structural biology was rare. Training in the required computational analyses

Figure 1



Intersection of CMS disciplines and methodologies that contribute to structural biology curricula.

and visualization was reserved for graduate students and postdoctoral fellows. However, modern research and industries began calling for these skills in the workforce [2,3]. For example, the pharmaceutical industry was shifting to identification of lead compounds using computational modeling and molecular docking, yet undergraduates were not receiving the requisite training in these fields.

As the call for improved CMS training became louder, undergraduate educators were faced with several barriers. First, this arena is inherently interdisciplinary and dynamic [4]. An individual educator rarely has all of the skills needed to develop (and continuously redesign) a robust curriculum. Second, course materials were scarce. A field in rapid development results in textbooks, and even web-based resources, becoming obsolete quickly. Luckily, these obstacles are being overcome and undergraduate computational structural biology is at the start of being a norm in training the workforce for 21st century STEM careers.

The interdisciplinary nature of structural bioinformatics has nurtured the formation of collaborative endeavors to support faculty development and curriculum innovation. The historical and current influence of four organizations spearheading the reform for inclusion of computation in biology curricula [BioQuest, MathBench Biology Modules, Quantitative Undergraduate Biology Education and Synthesis, and Intercollegiate Biomathematics Alliance] is presented by Akman et al. [5]. Quantitative Undergraduate Biology Education and Synthesis and the Network for the Integration of Bioinformatics into Life Science Education groups embraced an incubator model. This model facilitates production of freely available resources by bringing together groups of educators with varying skill sets in short, online collaborations [6]. The Cheminformatics Online Chemistry Course approached the problem by bringing together educators from various institutions and external experts from industry and/or government. Students enrolled in the course interact with many different people who, combined, provide the expertise required to run the course [7]. Other cross-disciplinary groups have formed organically. A team led by Vater at the University of California, Davis, encompassing educators from 10 institutions, developed a robust Course-based Undergraduate Research Experience (CURE) that challenges students to investigate protein structure modifications using Foldit [8]. The Biochemistry Authentic Scientific Inquiry Lab (BASIL) is another CURE developed by a multidisciplinary team of educators with expertise ranging from chemistry and biology to computer programing and educational assessment [9,10]. The BASIL CURE introduces students to computational science in structural biology using molecular visualization tools, molecular docking software, and sequence and structure alignment tools.

Through these groups and individual educators, the availability of pedagogical resources is increasing. For the majority, students are using software to investigate structural biology problems, in essence, a 'plug and play' format. Recent publications describe individual modules or courses, whereas others illuminate complete overhauls of a major curriculum [11–18]. Hall et al. [13] provide a helpful overview of resources including textbooks, databases, and computer-based applications. Augmented reality (AR) technology is a novel pedagogical approach; Hoog et al. [14] describe using smartphone-based AR technology in educational applications, and a preprint posted on ChemRxiv from Cortés Rodríguez describes interactive AR experiments for structural biology available on the MoleculARweb website [15]. Two recent articles highlight the use of computational techniques to support further understanding of biochemistry and/or bioinformatics. Sharp et al. [16] designed online computational modules aimed at introducing and reinforcing structure–function relationships. Acknowledging that third-year students were struggling in a biotechnology course, Zhang et al. [17] developed an elective course on preliminary structural biology and showed that students enrolled were more successful in the subsequent biotechnology course. Gatherer reflected on a holistic incorporation of bioinformatics into the Lancaster University undergraduate program [18]. No matter the form, the need to educate undergraduates in CMS is

being heeded, and myriad resources are now available to any educator who wishes to increase the training levels in their programs.

Using computational tools to incorporate algorithmic thinking

Most curricula described in the previous section use established software to teach biophysical principles, focusing less on how the software works and certainly much less on adapting or adding code to address new problems. The next step in the evolution of computational structural biology education is to incorporate more problem solving and algorithmic understanding in the use of computational tools. Activities where students change computational parameters and iterate through a discovery process help students understand the algorithm behind the calculation, think about the steps to solving a problem, and analyze how they could change the parameters to affect the calculation. Such activities help students appreciate what kinds of problems a given computational tool can address. These types of computational tools and curricula are currently less prominent in the structural biology classroom, yet their pedagogical strengths are numerous. Some recent examples from the chemistry curriculum illustrate how computational tools can be used in this way. Computation in the chemistry curriculum often has its start in the physical or organic chemistry course, but the shift to remote instruction compelled more educators to think about how computation and algorithmic thinking could be applied to other courses and disciplines. We believe that structural biology educators can and should consider these examples in considering how to better include computation and algorithmic thinking in their own curricula.

Computation can be particularly valuable to help students conceptualize and visualize concepts that are otherwise hard to illustrate. For example, in the past, many students would encounter programs that could simulate spectra (including NMR and MS) of small organic molecules, and they would be introduced to the use of spreadsheets for data analysis. These tools and new ones were used for remote learning during the pandemic, and there was particular emphasis on computation to simulate wet laboratory experiments, including titration and HPLC [19–22]. Even with the return to traditional labs after the pandemic, many of these simulation tools can be useful for demonstrating concepts before students carry out experiments in the laboratory. There has also been a concerted effort to use computational tools to assist students in their understanding of more complex physical chemistry concepts such as NMR and the Boltzmann distribution using interactive Python programs [23,24].

A key insight from these examples is the importance of using algorithmic thinking to enhance understanding of chemical principles; that the computations are not performed for the sake of doing calculations, but because the calculations can provide unique insight and solve unique problems about chemical phenomena. We note several common characteristics among activities that facilitate student learning in this way: make simulations interactive with several decision points; use visualization to help conceptualize concepts that are hard to otherwise understand; and recognize that more than just computational concepts can be taught with computational tools. For example, it is relatively straightforward to use the Mol* [25] viewing tool in the Protein Data Bank to explore structures highlighted in biochemistry and molecular biology textbooks, creating a much more interactive experience for students. Concepts such as structure–function relationships and the role of intermolecular forces can be demonstrated with specific examples, bringing static images to life and empowering students to explore these concepts on their own.

The availability of several new software packages will facilitate these kinds of activities in more specialized contexts. Examples include RNA-Puzzles [26], a tool kit for predicting and comparing RNA structures, and ProDy [27] which models protein dynamics. There are also several Python libraries that provide tools for sequence and structure analysis and the ability to work with large data sets, including iFeature [28], Biopython [29], and Graphein [30; preprint on bioRxiv]. Importantly, these libraries can be used to introduce concepts in bioinformatics and the study of large data sets, which are routinely used in understanding evolution, structure–function relationships, and drug design.

Table 1 summarizes some of the computational tools and resources that are used in the curriculum and activities discussed previously. To aid the reader in accessing the resources, we have included the URL of any relevant websites or GitHub repositories for each resource.

Teaching programming in the undergraduate CMS curriculum

Beyond using computation in a more interactive way to help students develop algorithmic thinking skills, the final step in giving students the theory and simulation skills required for the 21st century STEM workforce is teaching students to program. Only then can students identify new problems and develop their own computational tools to solve these problems. Students themselves recognize this benefit; in a recent article from the Earlham Institute, undergraduate student Georgia Whitton describes how teaching herself to code made her recognize that not knowing how to program had been a barrier and that her new skill opened up new research opportunities (Article available at https://www.earlham.ac.uk/articles/students-need-their-

Table 1

Selected resources and websites for computational tools related to each of the focus areas discussed in the review. The references listed for each computational tool are not necessarily the primary references for that computational tool, but rather are articles discussed in this review which use the tool. All URLs were accessed on July 13, 2021.

Area	Computational Tools and Resources	References
Visualization of 3D structures	Visual molecular dynamics (VMD): https://www.ks.uiuc.edu/Research/vmd/	[43]
	MoleculARweb: https://molecularweb.epfl.ch/	[15]
	Swiss PDB viewer: https://spdbv.vital-it.ch/	[12]
	Mol*: https://molstar.org/, https://www.rcsb.org/3d-view/	[25]
	BioMolViz: https://biomolviz.org/	[46, 47]
Molecular dynamics	GROMACS: https://www.gromacs.org/	[43]
	Visual molecular dynamics (VMD): https://www.ks.uiuc.edu/Research/vmd/	[43]
Alignment tools	ProMOL: http://www.promol.org/	[9]
	BLAST: https://blast.ncbi.nlm.nih.gov/Blast.cgi	[9]
	PFam: http://pfam.xfam.org/	[9]
	Dali: http://ekhidna2.biocenter.helsinki.fi/dali/	[9]
Bioinformatics	Tutorials on Structural Bioinformatics: https://github.com/pb3lab/ibm3202	[45]
	Cheminformatics OLCC: https://chem.libretexts.org/Courses/Intercollegiate_Courses/Cheminformatics_OLCC_(2019)	[7]
Molecular modeling	RNApuzzles: https://www.rnapuzzles.org/	[26]
	ProDy: http://prody.csb.pitt.edu/	[27]
	Design2Data: https://d2d.ucdavis.edu/	[8]

bioinformatics-game-why-i-am-learning-code-python; accessed July 13, 2021). Although many people associate mathematical skills as the precursor to learning to program, language aptitude is a better predictor of the learning rate when approaching Python [31]. Thus, learning to program can actually help students see themselves as a person who has the aptitude for STEM, which will make STEM jobs more accessible to a wider range of individuals. Programming has become mainstream and important enough in life sciences that WIRED magazine recommended in 2017 that all biologists learn to program, to keep up with 'the era of big data.' (Article available at https://www.wired.com/2017/03/biologists-teaching-code-survive/; accessed July 13, 2021).

As we approach a future where programming is not just a bonus or ancillary skill for scientists but a required competency, all physical and life scientists need to learn to program as part of the undergraduate curriculum. This requires current university faculty to embrace some level of programming proficiency or to introduce programming even earlier in students' education. Many countries teach some form of programming in secondary school or even as early as primary school. For this to happen widely, many more future teachers need exposure to computational thinking to gain some level of proficiency with programming. Teaching programming in undergraduate science courses designed for future educators could provide the necessary exposure, in turn, preparing their future students for programming as they reach the undergraduate curriculum (Report from the Brookings Institute available at https://www.brookings.edu/research/what-do-we-know-about-the-expansion-of-k-12-computer-science-education/; accessed July 13, 2021).

Efforts to include programming in the undergraduate curriculum have been growing over the last five years. A significant support for this effort has been the work of the Molecular Sciences Software Institute (the MolSSI), which was founded in 2016 [32]. This National Science Foundation funded institute has served as a leader in improving software, education, and training in CMS through their Software Fellows program, community workshops, and the development of numerous educational resources that introduce best practices in programming and software development. A recent review of the MolSSI Education program highlights the resources they offer to support teaching students Python programming, data analysis, and visualization techniques, as well as best practices in software engineering [33]. Another thorough discussion of teaching general programming principles comes from Weiss [34], who has developed a Creative Commons textbook that teaches scientific computing principles using Python and Jupyter notebooks. This open-source textbook covers everything from basic Python programming to data processing, data visualization, and signal processing.

In a special issue of the Journal of Chemical Education highlighting insights gained while teaching during the COVID-19 pandemic, Stewart et al. [35] describe a

curriculum to teach programming in a physical chemistry course. Examples include plotting wave functions and probability distributions and evaluating expectation values and uncertainties. Numerous examples of teaching programming in the physical chemistry curriculum can be found in the 2019 American Chemical Society Symposium series book Using Computational Methods to Teach Chemical Principles [36], especially chapters 12, 13, and 14 [37–39]. Of particular note, all these chapters highlight the importance of discipline-specific context to illustrate the importance of using programming to solve chemical problems. This approach helps students retain and apply the programming skills they learn to new contexts and new problems. Teaching programming in chemistry is not limited to physical or computational chemistry courses; Menke [40] presents a set of Jupyter notebooks that introduce Python programming in an analytical chemistry course to perform statistical analysis.

Examples of teaching programming in biochemistry and structural biology are less numerous than in chemistry but are increasing. Mariano et al. [41] discuss the importance of teaching programming to life science students and give several examples of scaffolded assignments designed for students with no background in algorithmic thinking, starting with teaching basic algorithm structure and mathematical operations, advancing to strings, detecting patterns, and, ultimately, to aligning sequences and manipulating large amounts of data such as a PDB file. Critically, they also discuss the challenges of including programming lessons and how these challenges can be addressed. A report from David [42] in The American Biology Teacher discusses teaching Python programming in the undergraduate biology curriculum. The curriculum introduced was used in a parasitology course, so perhaps not directly applicable to structural biology, but the article does discuss the importance of and techniques to include programming lessons in a biological context.

More specifically applicable to biochemistry and structural biology, Justino et al. [43] introduce programming to the biochemistry curriculum through molecular dynamics simulations and visualizations. Students do learn to use existing molecular dynamics software tools but also write their own code to identify noncovalent interactions. In a preprint posted on bioRxiv, Santana de Araújo [44] proposes teaching Python to bioinformatics students and discusses how to move students from being bioinformatics users to software developers. Finally, Engelberger et al. [45] present a comprehensive curriculum that combines using bioinformatics and visualization software as well as coding in Python to teach structural bioinformatics.

Table 2 summarizes some of the tools and resources that are used to teach programming in the curriculum and activities discussed previously. To aid the reader in accessing the resources, we have included the URL of any relevant websites or GitHub repositories for each resource.

Table 2

Selected resources for teaching programming. All URLs were accessed on July 13, 2021.

Resource	References
MolSSI education resources: http://education.molssi.org/resources.html	[33]
iFeature: https://ifeature.erc.monash.edu/	[28]
Biopython: https://biopython.org/	[29]
Graphein: https://github.com/a-r-j/graphein	[30]
Free textbook for teaching scientific computing: https://github.com/weisscharlesj/SciCompforChemists	[34]
Jupyter notebooks for analytical chemistry statistical analyses: https://github.com/erik-menke/AnalyticalProjects	[40]

MolSSI, Molecular Sciences Software Institute.

Summary and outlook

The future of biochemistry, molecular biology, and structural biology will include theory, computation, and simulation. To prepare students to contribute in these disciplines, students must be able to not just use software but also develop it. Innovation, discovery, and success in the field rely on students not being limited in the tools they can use to tackle problems.

The lofty goal of teaching all physical and life science students to program introduces challenges to educators. Many feel their curriculum is already crowded and worry that taking time to introduce programming will take away coverage from other critical concepts. Faculty may lack sufficient computational skills themselves to introduce these concepts in their classes. Convincing colleagues and administrators that this is a required skill for biologists may be difficult. Even with a growing body of curricular resources such as those presented in this review, there is the significant matter of validating these resources in the classroom. Educational assessment is yet another skill set necessary to complement the push for improved undergraduate CMS curricula. Several standalone studies evaluate newly developed resources, the majority through Likert-scale questionnaires on student experience [8,12,14,19,41,45]. Direct assessment of student learning is less frequent [10]. BioMolViz, a group of educators focused on promoting biomolecular visualization literacy, provides a framework for assessing biomolecular visualization, offers faculty development workshops on assessment design, and is developing an open-access repository of assessment [46,47]. Two broadly useful collections of resources available to assess

learning were curated in 2020: one from CBE Life Science Education [48] and the other from the American Society for Biochemistry and Molecular Biology (https://www.asbmb.org/education/online-teaching).

Addressing these challenges will require an interdisciplinary solution. Fortunately, educators in structural biology are up to the challenge, as this field inherently sits at the intersection of multiple disciplines, and so it is a natural fit to expand to include training in computation, simulation, and programming.

Conflict of interest statement

Nothing declared.

Acknowledgements

ARM acknowledges the support of NSF award CHE-2018427.

References

Papers of particular interest, published within the period of review, have been highlighted as:

* of special interest
** of outstanding interest

1. Jili NN, Ede CI, Masuku MM: **Emergency remote teaching in higher education during covid-19: challenges and opportunities**. *Int J High Educ* 2021, **10**:1.

2. Brewer CA, Smith D: *Vision and change in undergraduate biology education*. 2009.

3. Black PN: **A revolution in biochemistry and molecular biology education informed by basic research to meet the demands of 21st century career paths**. *J Biol Chem* 2020, **295**:10653–10661.

4. Nussinov R, Tsai CJ, Shehu A, Jang H: **Computational structural biology: successes, future directions, and challenges**. *Molecules* 2019, **24**.

5. Akman O, Eaton CD, Hrozencik D, Jenkins KP, Thompson KV: **Building community-based approaches to systemic reform in mathematical biology education**. *Bull Math Biol* 2020, **82**:109.

6. Ryder EF, Morgan WR, Sierk M, Donovan SS, Robertson SD, Orndorf HC, Rosenwald AG, Triplett EW, Dinsdale E, Pauley MA, *et al.*: **Incubators: building community networks and developing open educational resources to integrate bioinformatics into life science education**. *Biochem Mol Biol Educ* 2020, **48**:381–390.

7. Kim S, Bucholtz EC, Briney K, Cornell AP, Cuadros J, Fulfer KD, Gupta T, Hepler-Smith E, Johnston DH, Lang ASID, *et al.*: **Teaching cheminformatics through a collaborative intercollegiate online chemistry course (OLCC)**. *J Chem Educ* 2020, **98**:416–425.

8. * Vater A, Mayoral J, Nunez-Castilla J, Labonte JW, Briggs LA, Gray JJ, Makarevitch I, Rumjahn SM, Siegel JB: **Development of a broadly accessible, computationally guided biochemistry course-based undergraduate research experience**. *J Chem Educ* 2021, **98**:400–409.
A CURE with a Design2Data workflow is described that involves computational modeling for students to design novel enzyme variants. Students express and purify the mutants for further biophysical characterization.

9. Roberts R, Hall B, Daubner C, Goodman A, Pikaart M, Sikora A, Craig P: **Flexible implementation of the BASIL CURE**. *Biochem Mol Biol Educ* 2019, **47**:498–505.

10. Sikora A, Irby SM, Hall BL, Mills SA, Koeppe JR, Pikaart MJ, Wilner SE, Craig PA, Roberts R: **Responses to the COVID-19 pandemic by the biochemistry authentic scientific Inquiry lab (BASIL) CURE consortium: reflections and a case study on the switch to remote learning**. *J Chem Educ* 2020, **97**:3455–3462.

11. Marquioni V, Morais Franco Nunes F, Teresa Marques Novo-Mansur M: **Protein identification by database searching of mass spectrometry data in the teaching of proteomics**. *J Chem Educ* 2021, **98**:812–823.

12. * Abreu PA, Carvalho K de L, Rabelo VWH, Castro HC: **Computational strategy for visualizing structures and teaching biochemistry**. *Biochem Mol Biol Educ* 2019, **47**:76–84.
Describes computational activities that use 3D structure visualization of proteins and DNA to teach concepts including structure–function relationships, substrate–inhibitor interactions, and DNA stuctural features.

13. Covey T, Watson KD, Hall BL: *Resources for teaching project-based undergraduate medicinal chemistry courses*. 2019.

14. Hoog TG, Aufdembrink LM, Gaut NJ, Sung RJ, Adamala KP, Engelhart AE: **Rapid deployment of smartphone-based augmented reality tools for field and online education in structural biology**. *Biochem Mol Biol Educ* 2020, **48**:448–451.

15. Cortés Rodríguez F, Frattini G, Krapp L, Martinez-Huang H, Moreno D, Salomón J, Stemkoski L, Träger S, Dal Peraro M, Abriata L: *MoleculARweb: a website for chemistry and structural biology education through interactive augmented reality out of the box in commodity devices*. 2020, https://doi.org/10.26434/CHEMRXIV.13012463.V1.

16. Sharp AK, Gottschalk CJ, Brown AM: **Utilization of computational techniques and tools to introduce or reinforce knowledge of biochemistry and protein structure–function relationships**. *Biochem Mol Biol Educ* 2020, **48**:662–664.

17. * Zhang J, Jing H, Luo P, Zhang X, Zou Q: **Design, implementation, and outcomes of an elective course on preliminary structural biology for undergraduate students majoring in biotechnology**. *Biochem Mol Biol Educ* 2020, **48**:168–174.
Describes the development of a structural biology elective course for undergraduates. Topics included cloning, expression, purification, and crystal screening of a protein.

18. Gatherer D: **Reflections on integrating bioinformatics into the undergraduate curriculum: the Lancaster experience**. *Biochem Mol Biol Educ* 2020, **48**:118–127.

19. Buchberger AR, Evans T, Doolittle P: **Analytical chemistry online? Lessons learned from transitioning a project lab online due to COVID-19**. *J Chem Educ* 2020, **97**:2976–2980.

20. Tan SWB, Naraharisetti PK, Chin SK, Lee LY: **Simple visual-aided automated titration using the Python programming language**. *J Chem Educ* 2020, **97**:850–854.

21. Perri MJ: **Online data generation in quantitative analysis: excel spreadsheets and an online HPLC simulator using a Jupyter notebook on the chem compute web site**. *J Chem Educ* 2020, **97**:2950–2954.

22. Hoover GC, Dicks AP, Seferos DS: **Upper-year materials chemistry computational modeling module for organic display technologies**. *J Chem Educ* 2021, **98**:805–811.

23. Sengupta I: **Illustrating elementary NMR concepts through simple interactive Python programs**. *J Chem Educ* 2021, **98**:1673–1680.

24. Jameson G, Brüschweiler R: **Active learning approach for an intuitive understanding of the Boltzmann distribution by basic computer simulations**. *J Chem Educ* 2020, **97**:3910–3913.

25. * Sehnal D, Bittrich S, Deshpande M, Svobodová R, Berka K, Bazgier V, Velankar S, Burley SK, Koča J, Rose AS: **Mol* Viewer: modern web app for 3D visualization and analysis of large biomolecular structures**. *Nucleic Acids Res* 2021, https://doi.org/10.1093/nar/gkab314.
The Mol* viewer is a web-based tool that enables very fast, interactive visualization of 3D biomolecular structures in a web-browser. Since it requires no software installtion on the user's computer, it is well-suited for classroom demonstrations and activities.

26. Magnus M, Antczak M, Zok T, Wiedemann J, Lukasiak P, Cao Y, Bujnicki JM, Westhof E, Szachniuk M, Miao Z: **RNA-Puzzles**

toolkit: a computational resource of RNA 3D structure benchmark datasets, structure manipulation, and evaluation tools. *Nucleic Acids Res* 2020, **48**:576–588.

27. Zhang S, Krieger JM, Zhang Y, Kaya C, Kaynak B, Mikulska-Ruminska K, Doruker P, Li H, Bahar I: **ProDy 2.0: increased scale and scope after 10 Years of protein dynamics modelling with Python**. *Bioinformatics* 2021, https://doi.org/10.1093/bioinformatics/btab187.

28. Chen Z, Zhao P, Li F, Leier A, Marquez-Lago TT, Wang Y, Webb GI, Smith AI, Daly RJ, Chou KC, *et al.*: **IFeature: a Python package and web server for features extraction and selection from protein and peptide sequences**. *Bioinformatics* 2018, **34**:2499–2502.

29. Cock PJA, Antao T, Chang JT, Chapman BA, Cox CJ, Dalke A, Friedberg I, Hamelryck T, Kauff F, Wilczynski B, *et al.*: **Biopython: freely available Python tools for computational molecular biology and bioinformatics**. *Bioinformatics* 2009, **25**:1422–1423.

30. * Jamasb A, Lió P, Blundell T: **Graphein - a Python library for geometric deep learning and Network analysis on protein structures**. *bioRxiv* 2020, https://doi.org/10.1101/2020.07.15.204701.
Graphein is a python library for constructing graph and surface-mesh representations of protein structures that interfaces with multiple geometric deep learning libraries. An example workflow is provided that presents two new protein structure-related datasets.

31. Prat CS, Madhyastha TM, Mottarella MJ, Kuo CH: **Relating natural language aptitude to individual differences in learning programming languages**. *Sci Rep* 2020, **10**:1–10.

32. Wilkins-Diehr N, Crawford TD: **Nsf's inaugural software institutes: the science gateways community institute and the molecular sciences software institute**. *Comput Sci Eng* 2018, **20**:26–38.

33. ** McDonald AR, Nash JA, Nerenberg PS, Ball KA, Sode O, Foley IV JJ, Windus TL, Crawford TD: **Building capacity for undergraduate education and training in computational molecular science: a collaboration between the MERCURY consortium and the Molecular Sciences Software Institute**. *Int J Quant Chem* 2020, **120**, e26359.
Review of the Molecular Sciences Software Institute (the MolSSI) Education program. Highlighted educational resources include: the Python Data and Scripting workshop, which teaches fundmental python syntax, file parsing, data analysis using numpy, and plotting using matplotlib; the QM Tools Workshop, which discusses fundamental calculation types in quantum mechanics, intepreting and visualizing QM results; the MM Tools workshop, which teachs the fundamentals of molecular mechanics and molecular dynamics simluations, interpreting, and analyzing MD results; and the Best Practices workshop, which teaches best practices in software development including testing, documentation, version control, and continuous integration.

34. ** Weiss CJ: **A creative commons textbook for teaching scientific computing to chemistry students with Python and Jupyter notebooks**. *J Chem Educ* 2021, **98**:489–494.
Describes an open-access textbook that teaching scientific computing principles using Python and Jupyter notebooks. Topics included cover everything from basic Python programming to data processing, data visualization, and signal processing.

35. Cahill ST, Bergstrom Mann PE, Worrall AF, Stewart MI: **Remote teaching of programming in mathematica: lessons learned**. *J Chem Educ* 2020, **97**:3085–3089.

36. Grushow A, Reeves M S. *Using computational methods to teach chemical principles*; 2019.

37. Singleton SM: **Computational narrative activities: combining computing, context, and communication to teach chemical concepts**. In *ACS Symposium series: using computational methods to teach chemical principles*; 2019:163–181.

38. Tribe L: **Computational chemistry as a course for students majoring in the sciences**. In *ACS Symposium series: using computational methods to teach chemical principles*; 2019:183–194.

39. McDonald AR, Hagen JP: **Beyond the analytical solution: using mathematical software to enhance understanding of physical chemistry**. In *ACS Symposium series: using computational methods to teach chemical principles*. American Chemical Society; 2019:195–210.

40. Menke EJ: **Series of Jupyter notebooks using Python for an analytical chemistry course**. *J Chem Educ* 2020, **97**:3899–3903.

41. ** Mariano D, Martins P, Helene Santos L, de Melo- Minardi RC: **Introducing programming skills for life science students**. *Biochem Mol Biol Educ* 2019, **47**:288–295.
Discusses how to introduce programming to life science students with no background in programming. Begins with basic algorithm structure and mathematical operations, advancing to strings, detecting patterns and ultimately to aligning sequences and manipulating large amounts of data such as a PDB file.

42. * David AA: **Introducing Python programming into undergraduate biology**. *Am Biol Teach* 2021, **83**:33–41.
Discusses teaching python programming in an undergraduate parisotology course. Highlights the types of computational skills most useful for biologists to learn and how programming can be taught in the context of biology.

43. * Justino GC, Nascimento CP, Justino MC: **Molecular dynamics simulations and analysis for bioinformatics undergraduate students**. *Biochem Mol Biol Educ* 2021, https://doi.org/10.1002/bmb.21512.
Activites that use GROMACS and VMD to teach concepts in bioinformatics. The activities also teach how to use python libraries and require students to write their own code to identify noncovalent interactions. Discusses how to integrate programming into a biochemistry course and how learning programming enhances learning biochemistry.

44. Santana De Araújo G: **From bioinformatics user to bioinformatics engineer: a report**. *bioRxiv* 2020, https://doi.org/10.1101/2020.08.03.225979.

45. ** Engelberger F, Galaz-Davison P, Bravo G, Rivera M, Ramírez-Sarmiento CA: **Developing and implementing cloud-based tutorials that combine bioinformatics software, interactive coding, and visualization exercises for distance learning on structural bioinformatics**. *J Chem Educ* 2021, **98**:1801–1807.
Presents twelve lab activities that introduce students to topics in bioinformatics including phylogenetic analysis, molecular modeling, molecular docking, molecular dynamics, and coevolutionary analysis. Activities are performed using Google Colaboratory notebooks, such that students do not have to install software on their own computers. Each of these notebooks includes a brief introduction to the topic, explanation of required tools, and analysis of results.

46. Procko K, Engelman S, Jakubowski H, Beckham JT, Dean DM, Franzen MA, Novak WRP, Roberts R, Roca AI, Shor AC, *et al.*: **Meeting report: BioMolViz workshops for developing assessments of biomolecular visual literacy**. *Biochem Mol Biol Educ* 2021, **49**:278–286.

47. Dries DR, Dean DM, Listenberger LL, Novak WRP, Franzen MA, Craig PA: **An expanded framework for biomolecular visualization in the classroom: learning goals and competencies**. *Biochem Mol Biol Educ* 2017, **45**:69–75.

48. Branchaw JL, Pape-Lindstrom PA, Tanner KD, Bissonnette SA, Cary TL, Couch BA, Crowe AJ, Knight JK, Semsar K, Smith JI, *et al.*: **Resources for teaching and assessing the vision and change biology core concepts**. *CBE-Life Sci Educ* 2020, **19**.