PyRosetta Jupyter Notebooks Teach Biomolecular Structure Prediction and Design

Kathy H. Le[1], Jared Adolf-Bryfogle[2], Jason C. Klima[3,4,5], Sergey Lyskov[6], Jason Labonte[6,7], Steven Bertolani[8], Shourya S. Roy Burman[6], Andrew Leaver-Fay[9], Brian Weitzner[3,4,5], Jack Maguire[10], Ramya Rangan[11], Matt A. Adrianowycz[11], Rebecca F. Alford[6], Aleexsan Adal[6], Morgan L. Nance[12], Rhiju Das[11], Roland L. Dunbrack, Jr.[13], William Schief[2], Brian Kuhlman[9,10], Justin B. Siegel[8], Jeffrey J. Gray[6*]

1. T. C. Jenkins Department of Biophysics, Johns Hopkins University, Baltimore, Maryland, United States.
2. Department of Immunology and Microbiology, The Scripps Research Institute, La Jolla, California, United States.
3. Institute for Protein Design, University of Washington, Seattle, Washington, United States.
4. Department of Biochemistry, University of Washington, Seattle, Washington, United States.
5. Lyell Immunopharma, Inc., Seattle, Washington.
6. Department of Chemical and Biomolecular Engineering, Johns Hopkins University, Baltimore, Maryland, United States.
7. Department of Chemistry, Franklin & Marshall College, Lancaster, PA, USA
8. Department of Chemistry, Department of Biochemistry and Molecular Medicine, Genome Center, University of California, Davis, Davis, California 95616, United States
9. Department of Biochemistry, University of North Carolina at Chapel Hill, Chapel Hill, North Carolina, United States
10. Program in Bioinformatics and Computational Biology, University of North Carolina at Chapel Hill, Chapel Hill, North Carolina, United States
11. Program in Biophysics, Stanford University, Stanford, California, United States.
12. Program in Molecular Biophysics, Johns Hopkins University, Baltimore, Maryland, United States
13. Fox Chase Cancer Center, Philadelphia, PA, United States

[*] Corresponding author: jgray@jhu.edu, 410-516-5313

Keywords: Protein structure and dynamics, Molecular structure and modeling, Protein and macromolecules, Computational methods and bioinformatics, Computer-based teaching tools, Learning materials and teaching tools, Multimedia teaching tools

PyRosetta Jupyter Notebooks Teach Biomolecular Structure Prediction and Design

Abstract

Biomolecular structure drives function, and computational capabilities have progressed such that the prediction and computational design of biomolecular structures is increasingly feasible. Because computational biophysics attracts students from many different backgrounds and with different levels of resources, teaching the subject can be challenging. One strategy to teach diverse learners is with interactive multimedia material that promotes self-paced, active learning. We have created a hands-on education strategy with a set of fifteen modules that teach topics in biomolecular structure and design, from fundamentals of conformational sampling and energy evaluation to applications like protein docking, antibody design, and RNA structure prediction. Our modules are based on *PyRosetta*, a Python library that encapsulates all computational modules and methods in the Rosetta software package. The workshop-style modules are implemented as Jupyter Notebooks that can be executed in the Google Colaboratory, allowing learners access with just a web browser. The digital format of Jupyter Notebooks allows us to embed images, molecular visualization movies, and interactive coding exercises. This multimodal approach may better reach students from different disciplines and experience levels as well as attract more researchers from smaller labs and cognate backgrounds to leverage PyRosetta in their science and engineering research. All materials are freely available at https://github.com/RosettaCommons/PyRosetta.notebooks.

Introduction

Structural models of proteins and other biomolecules help explain their functions and properties. Methods for computational structure prediction (protein folding and docking, as well as interactions with nucleic acids, carbohydrates, and other biomolecules) have been successful in many cases and certainly useful to drive structural and functional research hypotheses (1). Design of biomolecules (protein design, prediction of mutational effects, and complex design) has also exhibited many successes, with potential impacts in medicine, biology, biotechnology, materials, and chemistry (2). Thus, there is a need to disseminate these interdisciplinary methods to a broader audience. Here, we present a set of workshops for teaching or self-study of biomolecular structure prediction and design.

Scientific and Pedagogical Background

Computational methods are a relatively inexpensive way to predict and manipulate biomolecular structures, especially when experimental methods prove difficult. There is a long history in biophysics of using computational modeling to better understand structure, dynamics, and function. In fact, the 2013 Nobel Prize in Chemistry was awarded for the pioneering contributions in quantum and molecular mechanics of complex chemical systems (3). There are now many available dynamic simulation tools for observing the behavior of biomolecules over time and predicting thermodynamic and kinetic properties from estimates of the system's partition function. Some of these tools include CHARMM, Schrödinger software suite, MOE, NAMD, Amber, and Gromacs (4–9). A complementary approach to model biomolecules is with so-called *structure prediction* approaches. Instead of seeking a full description of all the states and kinetic rates of the system, these approaches seek the dominant, low-energy conformational state(s) that is (are) most relevant at biological conditions (10). These methods often speed calculations with approximations, such as constant bond lengths and angles, implicit solvent models, and empirically tuned energy functions. In exchange for these approximations, structure prediction approaches can capture the behavior of large biomolecules in equilibrium without necessitating simulations over long timescales (11). These approaches are fundamentally based on optimization of an energy function in a very large conformational space. The same algorithmic components can then be used in reverse to *design* biomolecules by optimizing the energy function across different biomolecular sequences.

One leading structure prediction and design software suite is Rosetta, a collection of algorithms for protein structure prediction, docking, and design (10, 12–14) as well as protein interactions with small molecules (15), nucleic acids (16), carbohydrates, or in a lipid bilayer (17). Rosetta has been a scientific leader in several blind structure prediction challenges (18–21) and has shown proof-of-principle for many design goals, including *de novo* folds (22–24), loop design, interface design (25–28), symmetric assembly (29, 30), and mineral binding (31, 32). In addition to its success in science and engineering, Rosetta is suited for teaching structure prediction and design for several reasons. The Rosetta methods are available as a Python library called PyRosetta (33), which makes them easier to learn and combine with other scientific code libraries. PyRosetta allows access to low-level data and has a range of pre-built protocols for many tasks in biophysical research. Students can measure and manipulate protein conformations, dock proteins, run folding algorithms, and explore other emerging topics in biomolecular structure prediction and design, such as RNA modeling and non-canonical amino acids. Furthermore, students can learn how to use these tools by creating and testing their own algorithms.

For about a decade now, structure prediction and design has been taught with PyRosetta, primarily through the use of a set of workshops that are available both as a printed book (34) and as downloadable PDF files (35). These workshops have been used to teach a course for undergraduate and graduate students at Johns Hopkins University for over ten years and intermittently at other schools, including the Massachusetts Institute of Technology, Stanford University, the University of Kansas, and the University of North Carolina. Workshops have been downloaded over 120,000 times (several tutorials over 1,000 times per year), and a complementary set of lecture videos have registered over 14,000 views, reflecting a growing interest in biomolecular structure prediction and design. In addition, these workshops have been an important resource for the Rosetta community, with the workshops being the primary learning tool for many now-senior core developers.

Despite the strong demand for educational resources, there have been several challenges in teaching with these materials. One problem in this interdisciplinary field has been how to train students from all levels and different skill sets. To address this challenge, the Rosetta Commons has established several programs and resources for students and researchers who are interested in Rosetta and PyRosetta, such as the PyRosetta and C++ Code Academies and the Rosetta REU for undergraduate students (36). Other resources from the Rosetta community include: XML documentation (37), the Rosetta User Guide (38), Code Manual (39), and the active, managed user forum (40). While these resources have helped expose the field to a broad audience, hurdles still remain. Learning new software can be challenging for beginners, and the available beginner academies have limits on their annual cohort size. In addition, most of the resources currently available are in the form of code documentation or static text, which lack the interactive components that would enable active learning. Multimodal environments (e.g. including visualizations in addition to text) enhance students' mental representations of fundamental concepts (41, 42), and in at least one coding class, interactive web-based content increased student time engaging with material and improved their quiz scores (43).

In addition, there are technological challenges that pose problems for new learners. Because the capabilities of Rosetta are constantly changing and expanding as functions are modified and new algorithms added, educational resources need to evolve in parallel with the main Rosetta software. Since the original PyRosetta workshops, some commands and protocols have been deprecated or replaced, and many new frameworks have become standard. Static text workshops have been difficult to maintain because they require manual testing and updating. A related challenge is that PyRosetta is difficult to configure on Windows, making the barrier of entry for some beginners and self-learners prohibitively high.

In this work, we describe our latest contribution to address the pedagogical and technological limitations of previous educational resources by creating an accessible, multimedia platform for teaching structure prediction and design methods with PyRosetta. Our solution combines the accessibility of Jupyter Notebooks, a shareable web application that supports live code, equations, visualization, and text (44), with the free computing power of Google Colaboratory (45) to develop a way for students of all experience levels to use PyRosetta on the cloud. Starting with our existing static workshops, we created a new, expanded set of interactive, multimedia Jupyter Notebooks with coding examples and conceptual questions that engage students with the material and let them test their understanding. We discuss below how this approach may improve engagement and retention, and how the technical implementation removes barriers to entry and enables the materials to stay current with emerging Rosetta methods.

Results

A. PyRosetta notebooks cover a broad range of basic and advanced topics

To make a broad range of topics in the field accessible to the public, we have created a diverse set of PyRosetta notebooks and shared them in a public GitHub repository (https://github.com/RosettaCommons/PyRosetta.notebooks). These notebooks aim to teach both the fundamentals as well as the applications of biomolecular structure prediction and design. The set currently includes 14 modules, which are split into two groups. Part I introduces the basics of PyRosetta (Chapters 2-8), and Part II explores advanced applications (Chapters 9-15), such as antibody design and membrane protein modeling (**Fig. 1**). Chapter 1 walks students through the process of setting up PyRosetta in Google Colaboratory with step-by-step instructions and guiding screenshots.

Part I focuses on the two main scientific capabilities of Rosetta: sampling and scoring biomolecular conformations. The notebooks explain the technical basics of PyRosetta, starting with how to create a `Pose`, which is the container object that holds all molecules, coordinates, energies, and other details about the system. Next, students learn how to make a

`ScoreFunction` to approximate energies and how to combine `Mover` objects to manipulate `Pose` conformations. In addition to these technical skills, Part I builds the fundamental theoretical concepts that frame the challenges of sampling and scoring conformations. Students are introduced to *Levinthal's paradox*, the idea that the conformational space available to proteins is exponentially large and thus impossible to search comprehensively (46). They also learn about *Anfinsen's Dogma*, the idea that a folded protein is at a thermodynamic minimum free energy state (47). The workshops teach students how to employ various potential functions, which can be physics-based (van der Waals, Coulomb) or knowledge-based (hydrogen bonding, side-chain energies). Students learn that these functions can be empirically optimized for protein-scale phenomena, such as folding and design. Most exercises in Part I are short and provide detailed guidance. Moving GIF animations, schematics, and images of biomolecules (created in PyMOL) are used to illustrate general concepts. For example, students are guided through the application of a `TrialMover`, which tests a conformational change, evaluates the new energy, and uses the Metropolis Monte Carlo criterion to either accept or reject the change. In addition to general concepts, some visualizations also depict expected outcomes, such as a PyMOL movie of a basic folding algorithm. The learning objectives for the workshops in Part I can be found in **Table 1.**

Part II guides learners through advanced applications of PyRosetta, relying on the basic skills and concepts introduced in Part I. Chapter 11, for example, explores how PyRosetta can also be used to model and design antibodies, which is an important challenge faced by pharmaceutical companies (48). In Chapter 13, students learn how to apply the same approaches to predict RNA structures, which are increasingly recognized for critical roles in catalysis and regulation (49). Chapter 14 explores the tools for investigating membrane proteins, which comprise approximately 60% of drug targets (50). A larger emphasis is placed on workshop exercises to introduce learners to a variety of questions and methods that are currently used in the field. For advanced students, Chapter 15 reviews more intensive tasks that can be executed outside of Google Colaboratory, such as parallelization with GNU and `dask` libraries (51). The learning objectives for the workshops in Part II can be found in **Table 2.**

B. Students can access the multimedia PyRosetta workshops on the Google Colaboratory platform

Google Colaboratory is an online web environment for Jupyter Notebooks on a cloud-based virtual machine accessible with any browser. Google Colaboratory provides students with powerful computational resources, including 13 GB of RAM, 33 GB of disk space, and continuous sessions of up to 12 hours (45). While Jupyter Notebooks have been used for engineering education (52, 53), Google Colaboratory offers a few advantages for studying biomolecular modeling, starting with the free in-the-cloud computing power. Students can complete most of the PyRosetta Notebooks in the Google Colaboratory environment (**Fig. 2**). They can open notebook files and store different versions directly in their Google Drive. The initial configuration of the PyRosetta software package in Google Colaboratory is automated and takes approximately 10 minutes. Afterwards, students simply import the supporting pip package `pyrosettacolabsetup` (54) and the configured PyRosetta package. Students can complete the provided exercises to build their own solutions and modify any line of code in the workshops, which pair introductory passages, concepts, and exercises with supporting PyMOL images, movies, and diagrams (**Fig. 3**).

C. Notebooks enable features for students and instructors

To create both student and instructor versions of assignments in the notebooks, we incorporated `nbgrader` (55). `Nbgrader` enables developers and instructors to create and maintain a single master copy of each workshop. The master copy includes solutions to all exercises, and the student version of the workshop is automatically generated without selected solutions (**Fig. 5**). Thus, developers can write PyRosetta coding examples and problems for students to attempt on their own. To help students locate examples of specific concepts and commands, we also incorporated `nbpages` (56), which enables the automatic generation of the Table of Contents and a searchable Keyword Index in notebook and markdown form (**Fig. 4**). These tools are activated by the `make-student-nb` script, which developers can use to update the student notebooks, Table of Contents, and Keywords Index with a single command.

Further, in Chapter 2 we showcase the ability to visualize macromolecules directly in the notebooks using `py3Dmol`, a web-based Jupyter widget encompassing an interactive `3Dmol.js` molecular viewer (57). The `py3Dmol` bindings (in the `pyrosetta.distributed.viewer` namespace) can also allow on-the-fly, interactive visualization of PyRosetta `ResidueSelector` objects, which allow students to choose subsets of residues based on sequence, chemistry, or structural properties (**Fig. 3C**). For those who install PyRosetta on their local computer, the motions of a protein in a protocol can be watched in an external PyMOL window using the PyRosetta `PyMOLObserver` (58).

Chapter 15 demonstrates how to scale up simulations to high performance computing resources using the SLURM scheduler with GNU `parallel`, `dask` and `distributed` modules (51, 59). We additionally introduce the `pyrosetta.distributed.dask` namespace for PyRosetta integration with the `dask-jobqueue` module, providing a user-friendly interface for PyRosetta pre-initialization of worker machines allowing options-based configuration of macromolecular modeling tasks in distributed computing and cloud computing environments. These developments will enable future pedagogical programs to encompass advanced macromolecular modeling exercises and allow for additional educational content to be added with ease.

D. Learning outcomes

We piloted early versions of the notebooks in four separate teaching contexts. They were used in a formal university course in spring 2019 for a combined graduate/undergraduate elective course, in a Code Academy for new graduate students and postdocs in Rosetta Commons labs, and for a one-week code school for a class of undergraduate summer interns. Finally, we have shared the GitHub link with several individuals learning PyRosetta on their own. The courses all received high reviews (quality of course 4.56/5.00 and teaching effectiveness 4.69/5.00). For the university course, the skills gained by students were evidenced by a range of successful course projects on topics including “Structure-based prediction of peptide-MHC binding,” “Finding the relationship between epistasis and score in sequentially mutated TEM-1 β-lactamase,” and, on the methodological side, “Comparison of optimization methods used in protein structure prediction.” The Code Academy trainees similarly completed mini-projects such as “Antibody design for Ebola” and “Modeling intrinsically disordered proteins for cell signaling.” The summer interns continued to complete successful research projects in ten different academic labs and one industry research site (36). Finally, some self-paced learners who tested the complete multimedia workshops shared comments including: “these notebooks make PyRosetta more approachable to non-experts”, “you can install PyRosetta in your Google Drive and use it from many different machines”, and “attempting problems myself allowed me to pinpoint gaps in my understanding.”

Discussion

Protein structure prediction and design tools are powerful and have the potential to impact biophysics and many cognate disciplines, but there are several challenges for students including access to the tools and the varied backgrounds of students. Here we have described a set of interactive notebooks for learning structure prediction and design that can be used in a classroom context or for individual self-study. One of the advantages of this platform is that it is free and publicly available on GitHub (https://github.com/RosettaCommons/PyRosetta.notebooks). Another advantage is that PyRosetta can be accessed through the Google Colaboratory online web browser, which requires no local computer installation and can quickly integrate open-source packages. In addition to its accessibility, Google Colaboratory provides students with free and powerful computational resources (45). These advantages address the current technological challenges with the current resources for PyRosetta. This online platform also provides an environment for multimodal learning material, such as molecular visualization movies and coding examples. With a broad scope of topics from contributors of different areas of expertise, students are also able to gain exposure to the different applications of PyRosetta and develop the skills to pursue more in-depth applications. For instructors, this set of modules can be easily adapted to a course syllabus by modifying workshops or adding relevant examples.

In addition to the advantages, the platform has some limits. The Google Colaboratory platform complicates communication with the visualization software PyMOL, and we have so far been unable to make this connection simple. While the `PyMOLObserver` is the archetypal tool for real-time visualization of PyRosetta modeling trajectories (58), students with a local installation of PyRosetta are nevertheless able to view their algorithms in real-time on their local computer’s PyMOL. Within the Colaboratory, the `pyrosetta.distributed.viewer` currently supports dynamic visualization updates upon macromolecular conformational changes, which is convenient for viewing intermediate steps of macromolecular modeling tasks, such as between PyRosetta Movers, within Jupyter Notebooks. While the `pyrosetta.distributed.viewer` mimics only a subset of PyMOL functionalities, it accepts `ResidueSelector`-based user inputs, thus allowing a more streamlined interface to interactive molecular modeling and design.

Our notebooks may be compared with other educational materials for computational molecular biophysics. There are several textbook style software resources, such as the beginner’s guide to CHARMM (60) and the web-based lessons on CHARMM (61). Recently, MOE has been used for an integrated engineering curriculum (62). Additionally, FoldIt (63) and EteRNA (64) have been used in an interdisciplinary week-long program for undergraduate and high school students (65). Our contribution of PyRosetta Notebooks is complementary to these and has advantages with its active involvement of students, multimedia integration, and engagement on a viable and leading tool that can be used flexibly in new, innovative research.

Overall, the PyRosetta Notebooks are designed to be a gateway tool to introduce students to the fundamentals of biomolecular structure prediction and design. This platform could potentially be employed in high school lab courses, STEM summer programs, advanced undergraduate courses, as well as graduate courses. In addition, we encourage students to seek additional support from the distributed, collaborative network by posting on the Rosetta Commons forum (https://www.rosettacommons.org/forum). In addition, Google Colaboratory is compatible with TensorFlow and is heavily used for developing machine learning models because of the free GPU resources. In the future, machine learning could be incorporated with the capabilities of PyRosetta to explore new areas of research in the field (66). Furthermore, the GitHub archive is a living collection that will continually expand to include other applications of PyRosetta and aspects of macromolecular modeling to introduce a broader audience to the field.

Author Contributions

All authors contributed to one or more of the PyRosetta notebooks, and each notebook online includes author credits. SSRB, ALF, and JJG designed original curricular materials. KHL, JCK, and JJG wrote the manuscript. All authors reviewed and provided critical feedback on the manuscript.

Acknowledgements

Sid Chaudhury co-wrote the original set of (paper) workshops that grew over the years into this collection. Evan Baugh expanded these with a set of code-level tools into the first set of workshops posted online. JJG thanks the students of ChemBE 414/614 at Johns Hopkins University over the past decade who have completed the workshops as they have evolved and provided feedback each year. Funding provided from a Rosetta Commons mini-grant, a Johns Hopkins Center for Educational Resources Technology Fellowship to KHL, a Stanford School of Medicine TRS Teaching Innovation Grant to MAA, a Hertz Foundation Fellowship to RFA, NSF Graduate Research Fellowships to JCK, RR, and RFA, and grants NSF/DBI 1659649, NIH R01-GM127578, NIH 5R01-GM078221, NIH R01-GM73151, NSF/CCI 1740549, and NSF CHE-1740549.

References

1. D. Guzenko, A. Lafita, B. Monastyrskyy, A. Kryshtafovych, J. M. Duarte, Assessment of protein assembly prediction in CASP13. *Proteins Struct. Funct. Bioinforma.* (2019) https:/doi.org/10.1002/prot.25795.
2. P. S. Huang, S. E. Boyken, D. Baker, The coming of age of de novo protein design. *Nature* (2016) https:/doi.org/10.1038/nature19946.
3. NobelPrize.org, The Nobel Prize in Chemistry 2013. *Nobel Media AB 2019*, [accessed December 15, 2019]. https://www.nobelprize.org/prizes/chemistry/2013/summary/.
4. B. R. Brooks, *et al.*, CHARMM: The biomolecular simulation program. *J. Comput. Chem.* (2009) https:/doi.org/10.1002/jcc.21287.
5. LLC, *Schrodinger Software Suite*, New York, NY, USA, 2011.
6. *Molecular Operating Environment (MOE)*, 2013.08; Chemical Computing Group Inc.: Montreal, QC, Canada, 2013.

7. J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kalé, K. Schulten, Scalable molecular dynamics with NAMD. *J. Comput. Chem.* (2005) https:/doi.org/10.1002/jcc.20289.
8. D. A. Pearlman, D. A. Case, J. W. Caldwell, W. S. Ross, T. E. Cheatham, S. DeBolt, D. Ferguson, G. Seibel, P. Kollman, AMBER, a package of computer programs for applying molecular mechanics, normal mode analysis, molecular dynamics and free energy calculations to simulate the structural and energetic properties of molecules. *Comput. Phys. Commun.* (1995) https:/doi.org/10.1016/0010-4655(95)00041-D.
9. H. J. C. Berendsen, D. van der Spoel, R. van Drunen, GROMACS: A message-passing parallel molecular dynamics implementation. *Comput. Phys. Commun.* (1995) https:/doi.org/10.1016/0010-4655(95)00042-E.
10. R. Das, D. Baker, Macromolecular Modeling with Rosetta. *Annu. Rev. Biochem.* (2008) https:/doi.org/10.1146/annurev.biochem.77.062906.171838.
11. N. A. Marze, S. S. Roy Burman, W. Sheffler, J. J. Gray, Efficient flexible backbone protein-protein docking for challenging targets. *Bioinformatics* (2018) https:/doi.org/10.1093/bioinformatics/bty355.
12. J. K. Leman, B. D. Weitzner, S. M. Lewis, R. Consortium, R. Bonneau, Macromolecular modeling and design in Rosetta: new methods and frameworks. *Preprints*, 1–14 (2019).
13. A. Leaver-Fay, *et al.*, "Rosetta3: An object-oriented software suite for the simulation and design of macromolecules" in *Methods in Enzymology*, (2011) https:/doi.org/10.1016/B978-0-12-381270-4.00019-6.
14. K. W. Kaufmann, G. H. Lemmon, S. L. Deluca, J. H. Sheehan, J. Meiler, Practically useful: What the R osetta protein modeling suite can do for you. *Biochemistry* (2010) https:/doi.org/10.1021/bi902153g.
15. S. A. Combs, S. L. Deluca, S. H. Deluca, G. H. Lemmon, D. P. Nannemann, E. D. Nguyen, J. R. Willis, J. H. Sheehan, J. Meiler, Small-molecule ligand docking into comparative models with Rosetta. *Nat. Protoc.* (2013) https:/doi.org/10.1038/nprot.2013.074.
16. C. Y. Cheng, F. C. Chou, R. Das, "Modeling complex RNA tertiary folds with Rosetta" in *Methods in Enzymology*, (2015) https:/doi.org/10.1016/bs.mie.2014.10.051.
17. R. F. Alford, J. Koehler Leman, B. D. Weitzner, A. M. Duran, D. C. Tilley, A. Elazar, J. J. Gray, An Integrated Framework Advancing Membrane Protein Modeling and Design. *PLoS Comput. Biol.* (2015) https:/doi.org/10.1371/journal.pcbi.1004398.
18. S. Ovchinnikov, H. Park, D. E. Kim, F. DiMaio, D. Baker, Protein structure prediction using Rosetta in CASP12. *Proteins Struct. Funct. Bioinforma.* (2018) https:/doi.org/10.1002/prot.25390.
19. B. D. Weitzner, D. Kuroda, N. Marze, J. Xu, J. J. Gray, Blind prediction performance of RosettaAntibody 3.0: Grafting, relaxation, kinematic loop modeling, and full CDR optimization. *Proteins Struct. Funct. Bioinforma.* (2014) https:/doi.org/10.1002/prot.24534.
20. S. S. R. Burman, M. L. Nance, J. R. Jeliazkov, J. W. Labonte, J. H. Lubin, N. Biswas, J. J. Gray, Novel sampling strategies and a coarse-grained score function for docking homomers, flexible heteromers, and oligosaccharides using Rosetta in CAPRI Rounds 37-45. *bioRxiv Biophys.*, 1–13 (2019).
21. O. Marcu, E. J. Dodson, N. Alam, M. Sperber, D. Kozakov, M. F. Lensink, O. Schueler-Furman, FlexPepDock lessons from CAPRI peptide–protein rounds and suggested new

criteria for assessment of model quality and utility. *Proteins Struct. Funct. Bioinforma.* (2017) https:/doi.org/10.1002/prot.25230.
22. B. Kuhlman, G. Dantas, G. C. Ireton, G. Varani, B. L. Stoddard, D. Baker, Design of a Novel Globular Protein Fold with Atomic-Level Accuracy. *Science (80-.).* **302**, 1364–1368 (2003).
23. N. Koga, R. Tatsumi-Koga, G. Liu, R. Xiao, T. B. Acton, G. T. Montelione, D. Baker, Principles for designing ideal protein structures. *Nature* **491**, 222–227 (2012).
24. T. M. Jacobs, B. Williams, T. Williams, X. Xu, A. Eletsky, J. F. Federizon, T. Szyperski, B. Kuhlman, Design of structurally distinct proteins using strategies inspired by evolution. *Science* **352**, 687–90 (2016).
25. E. L. Humphris, T. Kortemme, Prediction of Protein-Protein Interface Sequence Diversity Using Flexible Backbone Computational Protein Design. *Structure* **16**, 1777–1788 (2008).
26. G. Guntas, C. Purbeck, B. Kuhlman, Engineering a protein–protein interface using a computationally designed library. *Proc. Natl. Acad. Sci.* **107**, 19296–19301 (2010).
27. S. J. Fleishman, T. A. Whitehead, D. C. Ekiert, C. Dreyfus, J. E. Corn, E.-M. Strauch, I. A. Wilson, D. Baker, Computational Design of Proteins Targeting the Conserved Stem Region of Influenza Hemagglutinin. *Science (80-.).* **332**, 816–821 (2011).
28. E.-M. Strauch, *et al.*, Computational design of trimeric influenza-neutralizing proteins targeting the hemagglutinin receptor binding site. *Nat. Biotechnol.* **35**, 667–671 (2017).
29. N. P. King, J. B. Bale, W. Sheffler, D. E. McNamara, S. Gonen, T. Gonen, T. O. Yeates, D. Baker, Accurate design of co-assembling multi-component protein nanomaterials. *Nature* **510**, 103–8 (2014).
30. N. P. King, W. Sheffler, M. R. Sawaya, B. S. Vollmar, J. P. Sumida, I. André, T. Gonen, T. O. Yeates, D. Baker, Computational Design of Self-Assembling Protein Nanomaterials with Atomic Level Accuracy. *Science (80-.).* **336**, 1171–1174 (2012).
31. D. L. Masica, S. B. Schrier, E. A. Specht, J. J. Gray, De novo design of peptide-calcite biomineralization systems. *J. Am. Chem. Soc.* **132**, 12252–12262 (2010).
32. H. Pyles, S. Zhang, J. J. De Yoreo, D. Baker, Controlling protein assembly on inorganic crystals through designed protein interfaces. *Nature* **571**, 251–256 (2019).
33. S. Chaudhury, S. Lyskov, J. J. Gray, PyRosetta: A script-based interface for implementing molecular modeling algorithms using Rosetta. *Bioinformatics* (2010) https:/doi.org/10.1093/bioinformatics/btq007.
34. J. J. Gray, S. Chaudhury, S. Lyskov, J. Labonte, *The PyRosetta interactive platform for protein structure prediction and design: PyRosetta4 Update Edition*, CreateSpace Independent Publishing Platform, Baltimore (2017).
35. PyRosetta Tutorials, [accessed December 11, 2019]. http://www.pyrosetta.org/tutorials.
36. R. F. Alford, A. Leaver-Fay, L. Gonzales, E. L. Dolan, J. J. Gray, A cyber-linked undergraduate research experience in computational biomolecular structure prediction and design. *PLoS Comput. Biol.* (2017) https:/doi.org/10.1371/journal.pcbi.1005837.
37. S. J. Fleishman, A. Leaver-Fay, J. E. Corn, E. M. Strauch, S. D. Khare, N. Koga, J. Ashworth, P. Murphy, F. Richter, G. Lemmon, J. Meiler, D. Baker, Rosettascripts: A scripting language interface to the Rosetta Macromolecular modeling suite. *PLoS One* **6**, 1–10 (2011).
38. RosettaCommons, What is Rosetta?, [accessed December 11, 2019]. https://www.rosettacommons.org/docs/latest/Home.
39. RosettaCommons, Rosetta Manuals, [accessed December 11, 2019].

https://www.rosettacommons.org/manuals/latest/.
40. RosettaCommons, Rosetta Forums, [accessed December 11, 2019]. https://www.rosettacommons.org/forum.
41. S. Freeman, S. L. Eddy, M. McDonough, M. K. Smith, N. Okoroafor, H. Jordt, M. P. Wenderoth, Active learning increases student performance in science, engineering, and mathematics. *Proc. Natl. Acad. Sci. U. S. A.* (2014) https:/doi.org/10.1073/pnas.1319030111.
42. M. Sankey, D. Birch, M. Gardiner, Engaging students through multimodal learning environments: The journey continues in *ASCILITE 2010 - The Australasian Society for Computers in Learning in Tertiary Education*, (2010).
43. A. D. Edgcomb, F. Vahid, Effectiveness of online textbooks vs. Interactive web-native content. *ASEE Annu. Conf. Expo. Conf. Proc.* (2014).
44. T. Kluyver, B. Ragan-kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, C. Willing, *Jupyter Notebooks—a publishing format for reproducible computational workflows* (2016) https:/doi.org/10.3233/978-1-61499-649-1-87.
45. T. Carneiro, R. V. M. Da Nobrega, T. Nepomuceno, G. Bin Bian, V. H. C. De Albuquerque, P. P. R. Filho, Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications. *IEEE Access* (2018) https:/doi.org/10.1109/ACCESS.2018.2874767.
46. M. Karplus, The Levinthal paradox: Yesterday and today. *Fold. Des.* **2**, 69–75 (1997).
47. Anfinsen, Folding of Protein Chains. *Science (80-.).* **181**, 223–230 (1973).
48. B. D. Weitzner, J. R. Jeliazkov, S. Lyskov, N. Marze, D. Kuroda, R. Frick, J. Adolf-Bryfogle, N. Biswas, R. L. Dunbrack, J. J. Gray, Modeling and docking of antibody structures with Rosetta. *Nat. Protoc.* (2017) https:/doi.org/10.1038/nprot.2016.180.
49. R. Das, J. Karanicolas, D. Baker, Atomic accuracy in predicting and designing noncanonical RNA structure. *Nat. Methods* (2010) https:/doi.org/10.1038/nmeth.1433.
50. J. P. Overington, B. Al-Lazikani, A. L. Hopkins, How many drug targets are there? *Nat. Rev. Drug Discov.* (2006) https:/doi.org/10.1038/nrd2199.
51. A. S. Ford, B. D. Weitzner, C. D. Bahl, Integration of the Rosetta Suite with the Python Software Stack via reproducible packaging and core programming interfaces for distributed simulation. *Protein Sci.* (2019) https:/doi.org/10.1002/pro.3721.
52. J. Kantor, Introduction to Chemical Engineering Analysis, [accessed December 15, 2019]. http://jckantor.github.io/CBE20255/.
53. A. Marrugo, Sensors and Actuators, [accessed December 15, 2019]. https://github.com/agmarrugo/sensors-actuators.
54. K. H. Le, pyrosettacolabsetup, https://pypi.org/project/pyrosettacolabsetup/.
55. P. Jupyter, D. Blank, D. Bourgin, A. Brown, M. Bussonnier, J. Frederic, B. Granger, T. Griffiths, J. Hamrick, K. Kelley, M. Pacer, L. Page, F. Pérez, B. Ragan-Kelley, J. Suchow, C. Willing, nbgrader: A Tool for Creating and Grading Assignments in the Jupyter Notebook. *J. Open Source Educ.* (2019) https:/doi.org/10.21105/jose.00032.
56. J. Kantor, nbpages, https://pypi.org/project/nbpages/.
57. N. Rego, D. Koes, 3Dmol.js: Molecular visualization with WebGL. *Bioinformatics* **31**, 1322–1324 (2015).
58. E. H. Baugh, S. Lyskov, B. D. Weitzner, J. J. Gray, Real-time PyMOL visualization for Rosetta and PyRosetta. *PLoS One* (2011) https:/doi.org/10.1371/journal.pone.0021931.

59. M. Rocklin, Dask: Parallel Computation with Blocked algorithms and Task Scheduling in *Proceedings of the 14th Python in Science Conference*, (2015) https:/doi.org/10.25080/majora-7b98e3ed-013.
60. R. Schleif, A Concise Guide to CHARMM and the Analysis of Protein Structure and Function (2013).
61. B. T. Miller, R. P. Singh, V. Schalk, Y. Pevzner, J. Sun, C. S. Miller, S. Boresch, T. Ichiye, B. R. Brooks, H. L. Woodcock, Web-Based Computational Chemistry Education with CHARMMing I: Lessons and Tutorial. *PLoS Comput. Biol.* **10** (2014).
62. P. Ludovice, D. MacNair, Integrated Molecular Modeling Education in the Chemical Engineering Curriculum in (2019).
63. F. Khatib, A. Desfosses, B. Koepnick, J. Flatten, Z. Popović, D. Baker, S. Cooper, I. Gutsche, S. Horowitz, Building de novo cryo-electron microscopy structures collaboratively with citizen scientists. *PLoS Biol.* (2019) https:/doi.org/10.1371/journal.pbio.3000472.
64. J. Lee, W. Kladwang, M. Lee, D. Cantu, M. Azizyan, H. Kim, A. Limpaecher, S. Yoon, A. Treuille, R. Das, RNA design rules from a massive open laboratory. *Proc. Natl. Acad. Sci. U. S. A.* (2014) https:/doi.org/10.1073/pnas.1313039111.
65. A. Taly, F. Nitti, M. Baaden, S. Pasquali, Molecular modelling as the spark for active learning approaches for interdisciplinary biology teaching. *Interface Focus* **9** (2019).
66. S. M. Kandathil, J. G. Greener, D. T. Jones, Recent developments in deep learning applied to protein structure prediction. *Proteins Struct. Funct. Bioinforma.* (2019) https:/doi.org/10.1002/prot.25824.

Figures

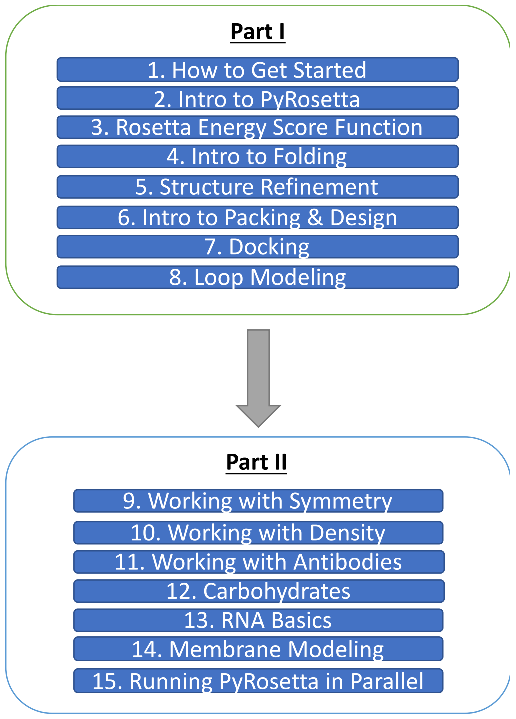
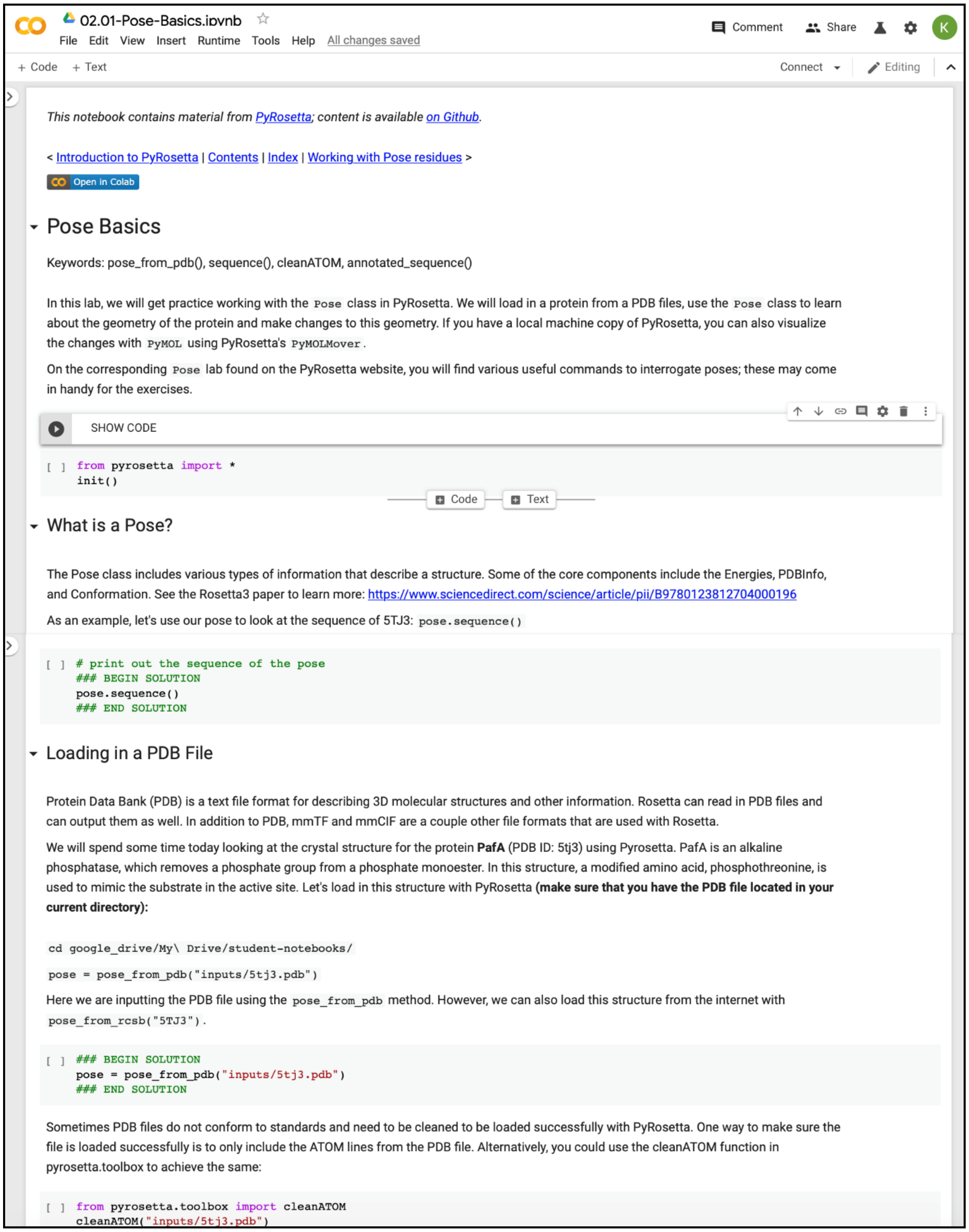


Figure 1: Map of general topics covered in the PyRosetta Notebooks (as of December 2019).



This notebook contains material from PyRosetta; content is available on Github.

< Introduction to PyRosetta | Contents | Index | Working with Pose residues >

Pose Basics

Keywords: pose_from_pdb(), sequence(), cleanATOM, annotated_sequence()

In this lab, we will get practice working with the `Pose` class in PyRosetta. We will load in a protein from a PDB files, use the `Pose` class to learn about the geometry of the protein and make changes to this geometry. If you have a local machine copy of PyRosetta, you can also visualize the changes with `PyMOL` using PyRosetta's `PyMOLMover`.

On the corresponding `Pose` lab found on the PyRosetta website, you will find various useful commands to interrogate poses; these may come in handy for the exercises.

SHOW CODE

```
[ ] from pyrosetta import *
    init()
```

What is a Pose?

The Pose class includes various types of information that describe a structure. Some of the core components include the Energies, PDBInfo, and Conformation. See the Rosetta3 paper to learn more: https://www.sciencedirect.com/science/article/pii/B9780123812704000196

As an example, let's use our pose to look at the sequence of 5TJ3: `pose.sequence()`

```
[ ] # print out the sequence of the pose
    ### BEGIN SOLUTION
    pose.sequence()
    ### END SOLUTION
```

Loading in a PDB File

Protein Data Bank (PDB) is a text file format for describing 3D molecular structures and other information. Rosetta can read in PDB files and can output them as well. In addition to PDB, mmTF and mmCIF are a couple other file formats that are used with Rosetta.

We will spend some time today looking at the crystal structure for the protein **PafA** (PDB ID: 5tj3) using Pyrosetta. PafA is an alkaline phosphatase, which removes a phosphate group from a phosphate monoester. In this structure, a modified amino acid, phosphothreonine, is used to mimic the substrate in the active site. Let's load in this structure with PyRosetta **(make sure that you have the PDB file located in your current directory):**

```
cd google_drive/My\ Drive/student-notebooks/
```

```
pose = pose_from_pdb("inputs/5tj3.pdb")
```

Here we are inputting the PDB file using the `pose_from_pdb` method. However, we can also load this structure from the internet with `pose_from_rcsb("5TJ3")`.

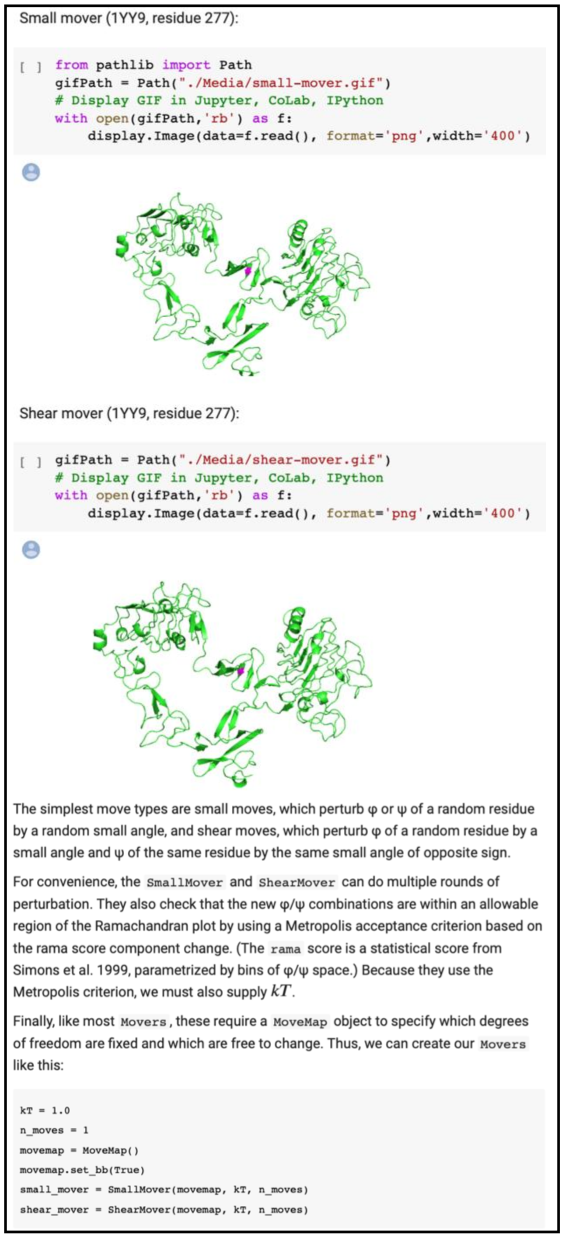
```
[ ] ### BEGIN SOLUTION
    pose = pose_from_pdb("inputs/5tj3.pdb")
    ### END SOLUTION
```

Sometimes PDB files do not conform to standards and need to be cleaned to be loaded successfully with PyRosetta. One way to make sure the file is loaded successfully is to only include the ATOM lines from the PDB file. Alternatively, you could use the cleanATOM function in pyrosetta.toolbox to achieve the same:

```
[ ] from pyrosetta.toolbox import cleanATOM
    cleanATOM("inputs/5tj3.pdb")
```

Figure 2: Screenshot of the Pose Basics workshop module in Google Colaboratory.

A



Small mover (1YY9, residue 277):

```
[ ] from pathlib import Path
    gifPath = Path("./Media/small-mover.gif")
    # Display GIF in Jupyter, CoLab, IPython
    with open(gifPath,'rb') as f:
        display.Image(data=f.read(), format='png',width='400')
```

Shear mover (1YY9, residue 277):

```
[ ] gifPath = Path("./Media/shear-mover.gif")
    # Display GIF in Jupyter, CoLab, IPython
    with open(gifPath,'rb') as f:
        display.Image(data=f.read(), format='png',width='400')
```

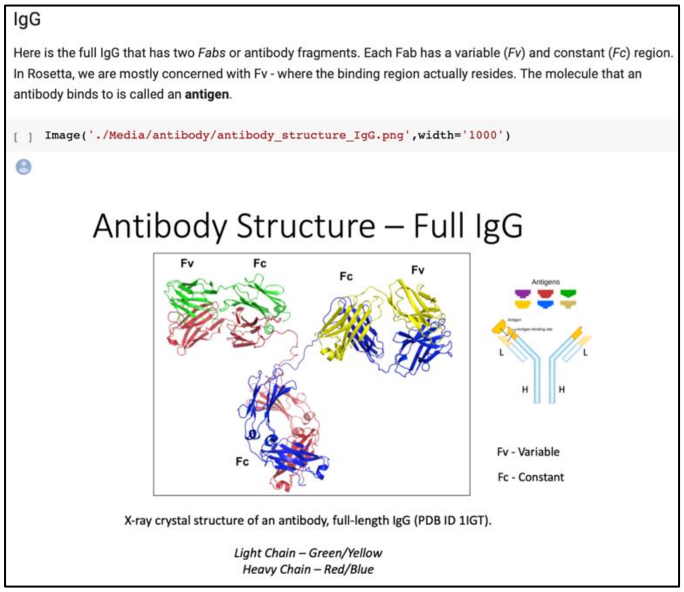
The simplest move types are small moves, which perturb φ or ψ of a random residue by a random small angle, and shear moves, which perturb φ of a random residue by a small angle and ψ of the same residue by the same small angle of opposite sign.

For convenience, the `SmallMover` and `ShearMover` can do multiple rounds of perturbation. They also check that the new φ/ψ combinations are within an allowable region of the Ramachandran plot by using a Metropolis acceptance criterion based on the rama score component change. (The `rama` score is a statistical score from Simons et al. 1999, parametrized by bins of φ/ψ space.) Because they use the Metropolis criterion, we must also supply kT.

Finally, like most `Movers`, these require a `MoveMap` object to specify which degrees of freedom are fixed and which are free to change. Thus, we can create our `Movers` like this:

```
kT = 1.0
n_moves = 1
movemap = MoveMap()
movemap.set_bb(True)
small_mover = SmallMover(movemap, kT, n_moves)
shear_mover = ShearMover(movemap, kT, n_moves)
```

B



IgG

Here is the full IgG that has two *Fabs* or antibody fragments. Each Fab has a variable (*Fv*) and constant (*Fc*) region. In Rosetta, we are mostly concerned with Fv - where the binding region actually resides. The molecule that an antibody binds to is called an **antigen**.

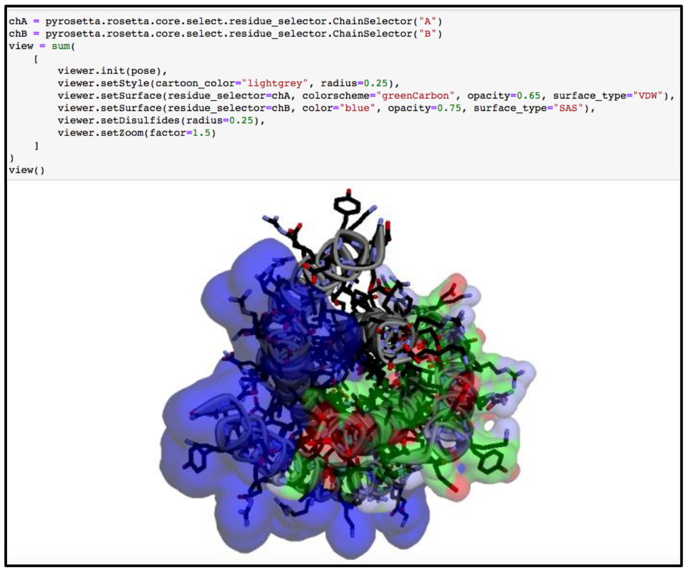
```
[ ] Image('./Media/antibody/antibody_structure_IgG.png',width='1000')
```

Antibody Structure – Full IgG

X-ray crystal structure of an antibody, full-length IgG (PDB ID 1IGT).

Light Chain – Green/Yellow
Heavy Chain – Red/Blue

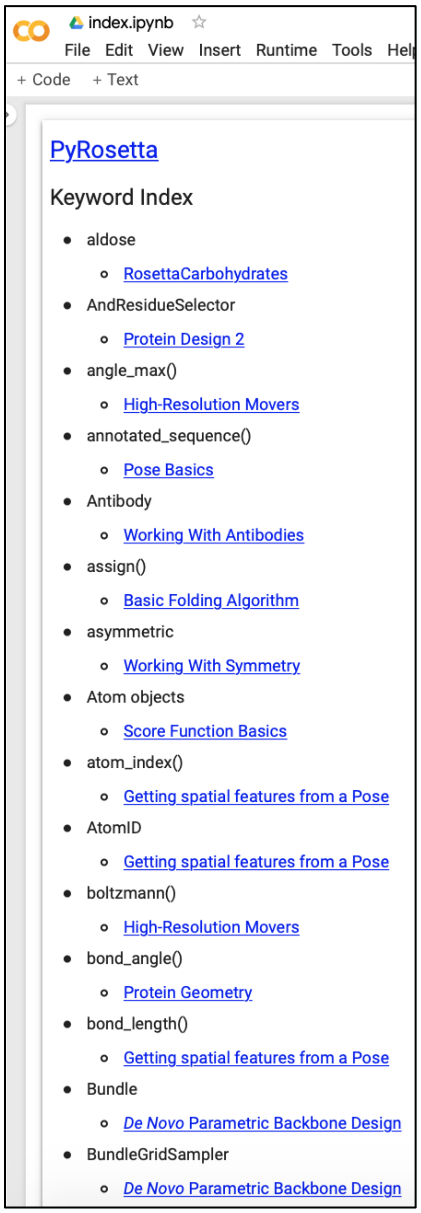
C



```
chA = pyrosetta.rosetta.core.select.residue_selector.ChainSelector("A")
chB = pyrosetta.rosetta.core.select.residue_selector.ChainSelector("B")
view = sum(
    [
        viewer.init(pose),
        viewer.setStyle(cartoon_color="lightgrey", radius=0.25),
        viewer.setSurface(residue_selector=chA, colorscheme="greenCarbon", opacity=0.65, surface_type="VDW"),
        viewer.setSurface(residue_selector=chB, color="blue", opacity=0.75, surface_type="SAS"),
        viewer.setDisulfides(radius=0.25),
        viewer.setZoom(factor=1.5)
    ]
)
view()
```

Figure 3: Notebook multimedia examples. (**A**) Moving GIFs of small and shear movers from “05.01 High Res Movers” (**B**) Images and diagrams of antibody structures from “11.00 Working with Antibodies” (**C**) Integration of PyRosetta with py3dmol for interactive macromolecular visualization in Jupyter Notebooks.

A

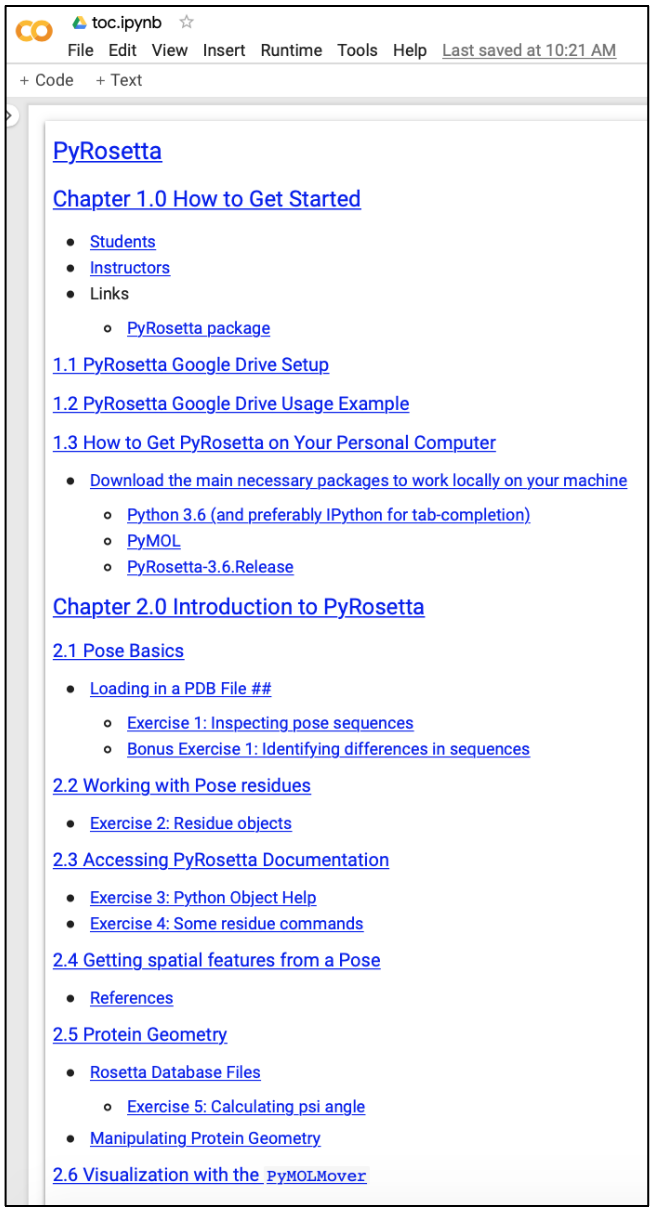


PyRosetta

Keyword Index

- aldose
  - RosettaCarbohydrates
- AndResidueSelector
  - Protein Design 2
- angle_max()
  - High-Resolution Movers
- annotated_sequence()
  - Pose Basics
- Antibody
  - Working With Antibodies
- assign()
  - Basic Folding Algorithm
- asymmetric
  - Working With Symmetry
- Atom objects
  - Score Function Basics
- atom_index()
  - Getting spatial features from a Pose
- AtomID
  - Getting spatial features from a Pose
- boltzmann()
  - High-Resolution Movers
- bond_angle()
  - Protein Geometry
- bond_length()
  - Getting spatial features from a Pose
- Bundle
  - *De Novo* Parametric Backbone Design
- BundleGridSampler
  - *De Novo* Parametric Backbone Design

B


toc.ipynb
File Edit View Insert Runtime Tools Help Last saved at 10:21 AM
+ Code + Text

PyRosetta

Chapter 1.0 How to Get Started

- Students
- Instructors
- Links
  - PyRosetta package

1.1 PyRosetta Google Drive Setup

1.2 PyRosetta Google Drive Usage Example

1.3 How to Get PyRosetta on Your Personal Computer

- Download the main necessary packages to work locally on your machine
  - Python 3.6 (and preferably IPython for tab-completion)
  - PyMOL
  - PyRosetta-3.6.Release

Chapter 2.0 Introduction to PyRosetta

2.1 Pose Basics

- Loading in a PDB File ##
  - Exercise 1: Inspecting pose sequences
  - Bonus Exercise 1: Identifying differences in sequences

2.2 Working with Pose residues

- Exercise 2: Residue objects

2.3 Accessing PyRosetta Documentation

- Exercise 3: Python Object Help
- Exercise 4: Some residue commands

2.4 Getting spatial features from a Pose

- References

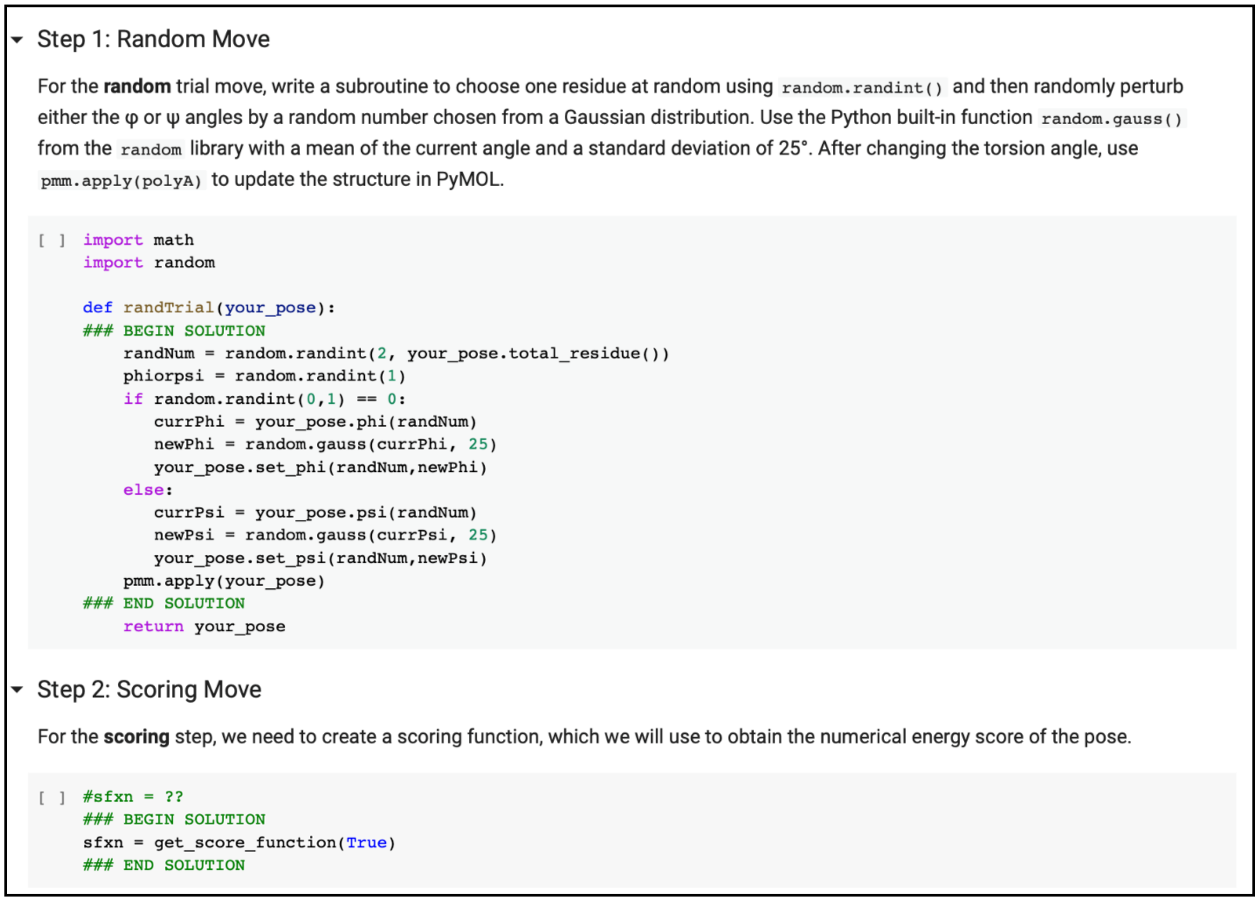
2.5 Protein Geometry

- Rosetta Database Files
  - Exercise 5: Calculating psi angle
- Manipulating Protein Geometry

2.6 Visualization with the `PyMOLMover`

Figure 4: Automatically-generated supporting material using `nbpages`. (**A**) Keyword index notebook with links. (**B**) Table of contents notebook with links.

A



Step 1: Random Move

For the **random** trial move, write a subroutine to choose one residue at random using `random.randint()` and then randomly perturb either the φ or ψ angles by a random number chosen from a Gaussian distribution. Use the Python built-in function `random.gauss()` from the `random` library with a mean of the current angle and a standard deviation of 25°. After changing the torsion angle, use `pmm.apply(polyA)` to update the structure in PyMOL.

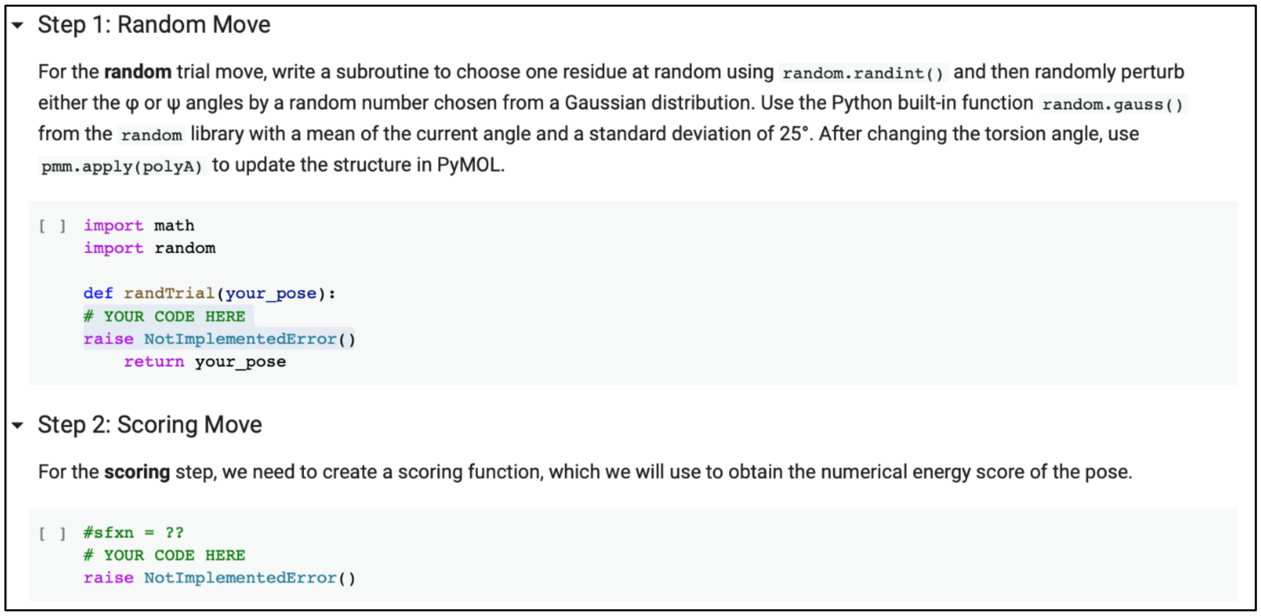
```
[ ] import math
    import random

    def randTrial(your_pose):
    ### BEGIN SOLUTION
        randNum = random.randint(2, your_pose.total_residue())
        phiorpsi = random.randint(1)
        if random.randint(0,1) == 0:
           currPhi = your_pose.phi(randNum)
           newPhi = random.gauss(currPhi, 25)
           your_pose.set_phi(randNum,newPhi)
        else:
           currPsi = your_pose.psi(randNum)
           newPsi = random.gauss(currPsi, 25)
           your_pose.set_psi(randNum,newPsi)
        pmm.apply(your_pose)
    ### END SOLUTION
        return your_pose
```

Step 2: Scoring Move

For the **scoring** step, we need to create a scoring function, which we will use to obtain the numerical energy score of the pose.

```
[ ] #sfxn = ??
    ### BEGIN SOLUTION
    sfxn = get_score_function(True)
    ### END SOLUTION
```

B



Step 1: Random Move

For the **random** trial move, write a subroutine to choose one residue at random using `random.randint()` and then randomly perturb either the φ or ψ angles by a random number chosen from a Gaussian distribution. Use the Python built-in function `random.gauss()` from the `random` library with a mean of the current angle and a standard deviation of 25°. After changing the torsion angle, use `pmm.apply(polyA)` to update the structure in PyMOL.

```
[ ] import math
    import random

    def randTrial(your_pose):
    # YOUR CODE HERE
    raise NotImplementedError()
        return your_pose
```

Step 2: Scoring Move

For the **scoring** step, we need to create a scoring function, which we will use to obtain the numerical energy score of the pose.

```
[ ] #sfxn = ??
    # YOUR CODE HERE
    raise NotImplementedError()
```

Figure 5: Example of student exercises. (**A**) Instructor version of "04.01 Basic Folding Algorithm" workshop with written solutions and (**B**) student version of the workshop with omitted solutions.

Tables

Table 1: List of workshop topics and learning objectives in Part I.

Current Topics	**Students Will Be Able To**
1.00 How to Get Started 1.01 PyRosetta Google Drive Setup 1.02 PyRosetta Google Drive Usage Example 1.03 How to Install Local PyRosetta	• Set up PyRosetta in Google Colab. • Set up PyRosetta on local computer (optional).
2.00 Intro to PyRosetta 2.01 Pose Basics 2.02 Working with Pose Residues 2.03 Accessing PyRosetta Documentation 2.04 Getting Spatial Features from Pose 2.05 Protein Geometry 2.06 Visualization and PyMOL Mover 2.07 RosettaScripts in PyRosetta 2.08 Visualization and pyrosetta.distributed.viewer	• Load a PDB structure. • Measure and alter protein structure (in internal or Cartesian coordinates). • Visualize macromolecules and PyRosetta ResidueSelectors within Jupyter Notebooks and through the PyMol-PyRosetta interface. • Run a RosettaScript from Python. • Instantiate and use individual configured components (objects) from a RosettaScript
3.00 Rosetta Energy Score Functions 3.01 Score Function Basics 3.02 Analyzing Energy Between Residues 3.03 Energies and the PyMOL Mover	• Test different score function components or weighted combinations.
4.00 Intro to Folding 4.01 Basic Folding Algorithm 4.02 Low-Res Folding and Fragments 4.03 De novo Protein Design	• Explain the fundamental challenges of protein structure prediction. • Describe the use of protein fragments for building protein backbones. • Implement a Metropolis Monte Carlo search strategy. • Use standard PyRosetta protocols to optimize protein structure.
5.00 Structure Refinement 5.01 High-Res Movers 5.02 Refinement Protocol	• Implement a Monte Carlo-plus-minimization algorithm. • Use various standard PyRosetta Movers to manipulate protein structure.
6.00 Intro to Packing and Design 6.01 Side-Chain Conformations and Dunbrack Energies 6.02 Packing Design Regional Relax 6.03 Design with a Resfile and Relax 6.04 Protein Design 2 6.05 HBnet Before Design 6.06 Intro to Parametric Backbone Design 6.07 Intro to de novo Protein Design	• Optimize side-chain conformations for a set of specified residues using PyRosetta. • Write custom PyRosetta protocols to simultaneously optimize protein structure and sequence. • Integrate sidechain packing with small and shear moves and minimization in PyRosetta refinement protocols. • Precede sidechain packing with hydrogen bond network design • Design proteins using custom scorefunctions with non-pairwise decomposable scoreterms in PyRosetta. • Design symmetrical proteins using Parametric backbone design • Design families of proteins with regular arrangements of secondary-structure elements
7.00 Docking 7.01 Fast Fourier Transform Docking 7.03 Ligand Docking PyRosetta 7.04 Ligand Docking XMLObjects 7.05 Ligand Docking pyrosetta.distributed	• Describe the major approaches to docking (grid based, FFT, Monte Carlo) and their advantages and disadvantages. • Use the PyJobDistributor for job distribution • Perform high-resolution protein-ligand refinement using the DockMCMProtocol mover. • Perform global ligand docking using XMLObjects. • Perform ligand docking with a genetic algorithm using pyrosetta.distributed.
8.00 Loop Modeling	• Describe the loop modeling and loop closure problems. • Describe the cyclic coordinate descent and kinematic closure approach and identify their advantages and limitations.

Table 2: List of workshop topics and learning objectives in Part II.

9.00 Working with Symmetry	• Create crystallographic symmetry files • Load proteins with symmetric components • Convert a monomer into a symmetric assembly. • Learn how to use common Rosetta protocols with symmetry enabled
10.00 Working with Density	• Convert PDB density files into Rosetta-readable files • Load density files into Rosetta • Use RosettaDensity to score a structure and use density to guide modeling
11.00 Working with Antibodies 11.01 Rosetta Antibody Framework and Simple Metrics 11.02 Rosetta Antibody Design	• Load antibody structures into the RosettaAntibody framework • Retrieve antibody specific information such as CDR loop regions, clusters, etc. for use in custom protocols • Set antibody-specific residue selectors and configure task operations for use in modeling and design • Design new antibodies with the RosettaAntibodyDesign protocol
12.00 Carbohydrates 12.01 Glycan Trees, Selectors and Movers 12.02 Glycan Modeling and Design	• Load an oligosaccharide or a glycoprotein. • Use *RosettaCarbohydrates* to add glycans conjugated to proteins. • Evaluate sugar-sugar linkage energies. • Select carbohydrates and get carbohydrate chemical and connectivity information • Optimize carbohydrate structure through linkage torsions, ring conformers, and sidechain conformers. Design carbohydrate recognition motifs (sequons) for designing glycans into proteins
13.00 RNA Basics	• Load nucleic acids and identify nucleic acid residues in poses. • Identify canonical and non-canonical base pairs in RNA structures. • View and manipulate nucleic acid torsion angles. • Evaluate nucleic acid energies using RNA-specific low-resolution and high-resolution score functions. Isolate RNA-specific score terms (e.g. stacking energies, base pairing potential). • Decompose RNA structures into 3D RNA motifs. • Use idealized torsion angles for RNA residues to generate an idealized A-form helix. • Replace RNA residues with a new sequence for homology modeling. • Use RNA fragments when building an RNA backbone and use minimization to refine resulting structures. Build a Monte Carlo search strategy using these approaches. • Use the FARFAR protocol for sampling RNA structures, which combines fragment assembly and high-resolution minimization moves.
14.00 Membrane Protein Modeling	• Use membrane tools to orient a protein in the lipid bilayer. • Calculate the lowest energy orientation for a membrane protein. • Identify membrane protein pores and cavities. • Interpret model quality using terms from *franklin2019*, the membrane energy function.
15.00 Running PyRosetta in Parallel 15.01 PyData, DDGs, and PSSMs 15.02 PyData Miniprotein Design 15.03 GNU parallel 15.04 dask.delayed via SLURM 15.05 Ligand Docking dask	• Parallelize macromolecule modeling tasks using distributed computing, elastic cloud computing, and high-performance computing infrastructures. • Parallelize PyRosetta jobs using GNU parallel and the SLURM job scheduling system. • Visualize and execute PyRosetta job parallelization with the dask module. • Analyze outputs from parallelized PyRosetta jobs in real-time as completed.