

PyRosetta Jupyter Notebooks Teach Biomolecular Structure Prediction and Design

Kathy H. Le¹, Jared Adolf-Bryfogle², Jason C. Klima^{3,4,5}, Sergey Lyskov⁶, Jason W. Labonte^{6,7}, Steven Bertolani⁸, Shourya S. Roy Burman⁶, Andrew Leaver-Fay⁹, Brian D. Weitzner^{3,4,5}, Jack Maguire¹⁰, Ramya Rangan¹¹, Matt A. Adrianowycz¹¹, Rebecca F. Alford⁶, Aleexsan Adal⁶, Morgan L. Nance¹², Yuanhan Wu¹³, Jordan Willis¹⁴, Daniel W. Kulp¹³, Rhiju Das¹¹, Roland L. Dunbrack, Jr.¹⁵, William Schief², Brian Kuhlman^{9,10}, Justin B. Siegel⁸, Jeffrey J. Gray^{6,12,*}

¹T. C. Jenkins Department of Biophysics, Johns Hopkins University, Baltimore, MD 21218, USA

²Department of Immunology and Microbiology, The Scripps Research Institute, La Jolla, CA, 92037, USA

³Institute for Protein Design, University of Washington, Seattle, WA 98195, USA

⁴Department of Biochemistry, University of Washington, Seattle, WA 98195, USA

⁵Lyell Immunopharma, Inc., Seattle, WA 98109, USA

⁶Department of Chemical and Biomolecular Engineering, Johns Hopkins University, Baltimore, MA 21218, USA

⁷Department of Chemistry, Franklin & Marshall College, Lancaster, PA 17604, USA

⁸Department of Chemistry, Department of Biochemistry and Molecular Medicine, Genome Center, University of California, Davis, Davis, CA 95621, USA

⁹Department of Biochemistry, University of North Carolina at Chapel Hill, Chapel Hill, NC 27599, USA

¹⁰Program in Bioinformatics and Computational Biology, University of North Carolina at Chapel Hill, Chapel Hill, NC 27599, USA

¹¹Program in Biophysics, Stanford University, Stanford, CA 94305, USA

¹²Program in Molecular Biophysics, Johns Hopkins University, Baltimore, MD 21218, USA

¹³Vaccine and Immunotherapy Center, Wistar Institute, Philadelphia, PA 19104, USA

¹⁴RubrYc Therapeutics, San Ramon, CA 94070, USA

¹⁵Fox Chase Cancer Center, Philadelphia, PA 19111, USA

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 16 modules that teach topics in biomolecular structure and design, from fundamentals of conformational sampling and energy evaluation to applications, such as 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

“*” corresponding author

Received: 31 December 2019

Accepted: 30 September 2020

Published: 14 April 2021

© 2021 Biophysical Society.

from different disciplines and experience levels, as well as attract more researchers from smaller labs and cognate backgrounds to leverage PyRosetta in science and engineering research. All materials are freely available at <https://github.com/RosettaCommons/PyRosetta.notebooks>.

KEY WORDS protein structure and dynamics; molecular structure and modeling; proteins and macromolecules; computer-based teaching tools; learning materials and teaching tools; multimedia teaching tools; teachers and students of graduate and upper level undergraduate courses in the biophysics-related sciences; researchers in the biophysics-related sciences

I. INTRODUCTION

Structural models of proteins and other biomolecules help explain the functions and properties. Methods for computational structure prediction (i.e., 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 (i.e., protein design, prediction of mutational effects, and molecular 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. The use of student feedback and course evaluations in this study was reviewed and approved by the Homewood Institutional Review Board at Johns Hopkins University (HIRB 11185). Here, we present a set of workshops for teaching or self-study of biomolecular structure prediction and design.

II. SCIENTIFIC AND PEDAGOGIC 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, Molecular Operating Environment (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 that is most relevant in biologic conditions (10). These methods often accelerate 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 structure of large biomolecules in equilibrium without necessitating simulations over long timescales. 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 (10–13), as well as protein interactions with small molecules (14), nucleic acids (15), and carbohydrates in solution or in a lipid bilayer (16). Rosetta has been a scientific leader in several blind structure prediction challenges (17–21) 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 meth-

ods 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 prebuilt protocols for many tasks in biophysical research. Students can measure and manipulate protein conformations, dock proteins and small molecules, run folding algorithms, and explore other emerging topics in biomolecular structure prediction and design, such as RNA modeling and noncanonical 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 Portable Document Format files (35). These workshops have been used to teach a course for undergraduate and graduate students at Johns Hopkins University for over 10 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 online lecture videos has registered over 14,000 views, reflecting a fast-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 research experience for undergraduates (36). Other salient resources from the Rosetta community include Extensible Markup

Language (XML) documentation (37), the Rosetta user guide (38), code manual (39), and the active, managed user forum (40). Although 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 the 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 quiz scores (43).

In addition, there are technologic challenges that pose problems for new learners. Because the capabilities of Rosetta are constantly changing and expanding, as methods 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 pedagogic and technologic limitations of previous educational resources by creating an accessible, multimedia platform for teaching biomolecular 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

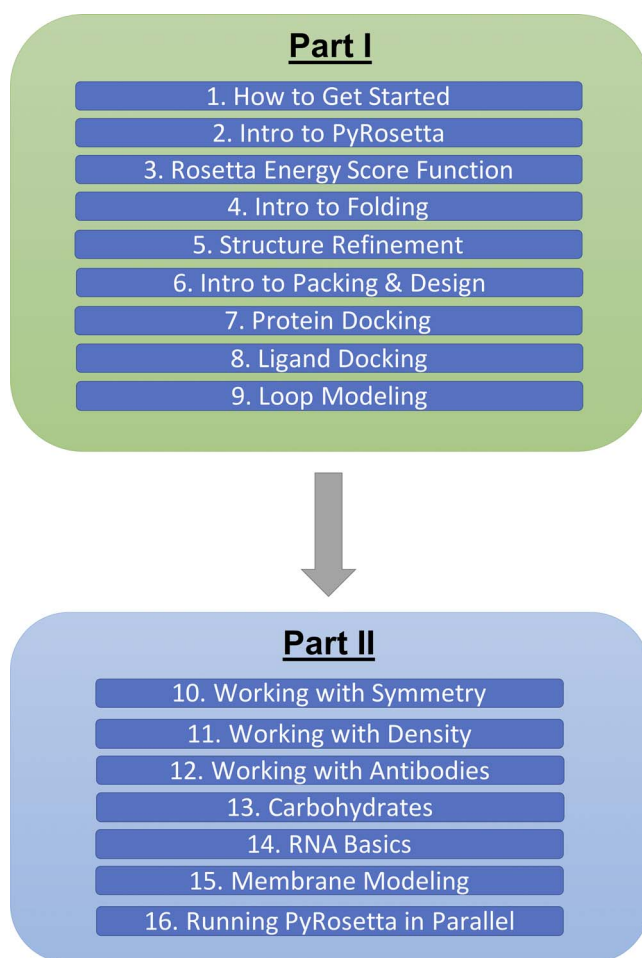


Fig 1. Map of general topics covered in the PyRosetta notebooks (as of October 2020).

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 in the following 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.

III. RESULTS

A. PyRosetta workshops 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 workshops within Jupyter Notebooks (i.e., PyRosetta notebooks) and shared them in a public, open-source 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 of notebooks currently includes 16 modules and is split into two parts. Part I introduces the basics of PyRosetta (Chapters 2 to 9), and Part II explores advanced applications (Chapters 10 to 16), 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` object, which is the container that holds all atoms, molecules, coordinates, energies, and other details about the system. Next, students learn how to make a `ScoreFunction` object to approximate free energies and how to combine `Mover` objects to manipulate the `Pose` conformations. In addition to these technical skills, Part I builds the fundamental theoretic concepts that frame the challenges of sampling and scoring conformations. Students are introduced to the Levinthal 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 dogma, the idea that a folded protein is at a thermodynamic minimum free energy state (47). The workshops teach students how to use 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 software) are used to illustrate general concepts (48). 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 (49). 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 12, for example, explores how PyRosetta can also be used to model and design antibodies, which is an important challenge faced by pharmaceutical companies (50). In Chapter 14, students learn how to apply the same approaches to predict RNA structures, which are increasingly recognized for critical roles in catalysis and regulation (51). Chapter 15 explores the tools for investigating membrane proteins, which include approximately 60% of drug targets (52). 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 16 reviews more intensive tasks that can be executed outside of Google Colaboratory, such as parallelization with GNU (www.gnu.org) and `dask` libraries (53). 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 random-access memory, 33 GB of disk space, 2.30 GHz of a central processing unit, and continuous sessions of up to 12 h (45).

Although Jupyter Notebooks have been used for engineering education (54, 55), 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 Google Drive. The initial configuration of the PyRosetta software package in Google Colaboratory is automated and takes approximately 10 min. Afterwards, students simply import the supporting pip package `pyrosettacolabsetup` (56) 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. Jupyter Notebooks enable features for students and instructors

To create both student and instructor versions of assignments in the notebooks, we incorporated `nbgrader` (57). The `nbgrader` module 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 4). 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` (58), which enables the automatic generation of the table of contents and a searchable keyword index in notebook and markdown form (Fig 5). These tools are activated by the provided `make-student-nb.bash` script, which developers can use to update the student notebooks, table of contents, and keywords index with a single command.

Instructors can make changes to the original set of Jupyter Notebook workshops by forking the main public repository (59). This allows instructors to tailor the workshops for specific

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 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 3.00 Rosetta Energy Score Functions 3.01 Score Function Basics 3.02 Analyzing Energy between Residues 3.03 Energies and the PyMOL Mover 4.00 Intro to Folding 4.01 Basic Folding Algorithm 4.02 Low-Res Folding and Fragments 5.00 Structure Refinement 5.01 High-Res Movers 5.02 Refinement Protocol 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 6.08 Point Mutation Scan 7.00 Protein Docking 7.01 Fast Fourier Transform Docking 7.02 Docking Moves in Rosetta 8.00 Ligand Docking PyRosetta 8.01 Ligand Docking XMLObjects 8.02 Ligand Docking pyrosetta.distributed 9.00 Loop Modeling 9.01 Using Gen KIC	<ul style="list-style-type: none"> • Set up PyRosetta in Google Colaboratory • Set up PyRosetta on a local computer (optional) • 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 • Test different score function components or weighted combinations • 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 • Implement a Monte Carlo plus minimization algorithm • Use various standard PyRosetta movers to manipulate a protein structure • Optimize side-chain conformations for a set of specified residues by using PyRosetta • Write custom PyRosetta protocols to simultaneously optimize a protein structure and sequence • Integrate side-chain packing with small and shear moves and minimization in PyRosetta refinement protocols • Precede side-chain packing with hydrogen bond network design • Design proteins by using custom score functions with nonpairwise decomposable score terms in PyRosetta • Design symmetric proteins by using parametric backbone design • Design families of proteins with regular arrangements of secondary structure elements • Generate mutagenesis library for antigen–antibody binding with PyRosetta • Compare mutant and wild-type binding energy • Visualize mutagenesis results by using a Python heatmap • Describe the major approaches to docking (grid based, FFT, Monte Carlo) and the advantages and disadvantages • Use the PyJobDistributor for job distribution • Perform high-resolution protein–ligand refinement by using the DockMCMProtocol mover • Perform global ligand docking by using XMLObjects • Perform ligand docking with a genetic algorithm by using pyrosetta.distributed • Describe the loop modeling and loop closure problems • Describe the cyclic coordinate descent and kinematic closure approach and identify the advantages and limitations • Close loops by using Gen KIC protocol

PDB, Protein Data Bank; FFT, Fast Fourier Transformation; Gen KIC, Generalized Kinematic Closure.

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

Current topics	Students will be able to
10.00 Working with Symmetry	<ul style="list-style-type: none"> • 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
11.00 Working with Density	<ul style="list-style-type: none"> • 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
12.00 Working with Antibodies 12.01 Rosetta Antibody Framework and Simple Metrics 12.02 Rosetta Antibody Design	<ul style="list-style-type: none"> • Load antibody structures into the RosettaAntibody framework • Retrieve antibody-specific information, such as CDR loop regions and clusters, 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
13.00 Carbohydrates 13.01 Glycan Trees, Selectors, and Movers 13.02 Glycan Modeling and Design	<ul style="list-style-type: none"> • 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 a carbohydrate structure through linkage torsions, ring conformers, and side-chain conformers; design carbohydrate recognition motifs (sequons) for designing glycans into proteins
14.00 RNA Basics	<ul style="list-style-type: none"> • Load nucleic acids and identify nucleic acid residues in poses • Identify canonical and noncanonical base pairs in RNA structures • View and manipulate nucleic acid torsion angles • Evaluate nucleic acid energies by using RNA-specific low- and high-resolution score functions; isolate RNA-specific score terms (e.g., stacking energies, base pairing potential) • Decompose RNA structures into three-dimensional 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, combining fragment assembly and high-resolution minimization moves
15.00 Modeling Membrane Proteins 15.01 Accounting for the Lipid Bilayer 15.02 Membrane Protein $\Delta\Delta G$ of mutation	<ul style="list-style-type: none"> • 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 by using terms from franklin2019, the membrane energy function • Compute the $\Delta\Delta G$ of mutation
16.00 Running PyRosetta in Parallel 16.01 PyData, $\Delta\Delta G$ s, and PSSMs 16.02 PyData Miniprotein Design 16.03 GNU Parallel 16.04 dask.delayed via SLURM 16.05 Ligand Docking Dask	<ul style="list-style-type: none"> • Parallelize macromolecule modeling tasks by using distributed computing, elastic cloud computing, and high-performance computing infrastructures • Parallelize PyRosetta jobs by 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

CDR, Complementarity-determining regions; FARFAR, Fragment Assembly of RNA with Full Atom Refinement; PSSMs, Position-Specific Scoring Matrices.

This notebook contains material from [PyRosetta](#); content is available [on Github](#).

< [Introduction to PyRosetta](#) | [Contents](#) | [Index](#) | [Working with Pose residues](#) >

[Open in Colab](#)

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, make changes to this geometry, and visualize the changes easily with `PyMOL` and 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.

PyRosetta Installation: The following lines will load in the PyRosetta library and load in database files.

```
[3] 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>

Loading in a Pose from 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/ OR cd google_drive/My\ Drive/notebooks/
pose = pose_from_pdb("inputs/5tj3.pdb")
```

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

    'NAVPRPKLVVGLVVDQMRWDLYRYYSKYEGGGFKRMLNTGYSLNNVHIDYVPTVTAIGHTSIFTGSVPSIHGIAGNDWYDKELGKSVICTSDETVQFVGTTSNSVQGHSPRLNLSWT'
```

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")
```

This method will create a cleaned `5tj3.clean.pdb` file for you. Lets load this into PyRosetta as well:

```
[ ] pose_clean = pose_from_pdb("inputs/5tj3.clean.pdb")
```

In our case, we could load in the PDB file for 5tj3 without cleaning it. In fact, we've lost some residues when cleaning the PDB file with `cleanATOM`. What is the difference in the sequence of the `pose_clean` now, compared to before?

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

A

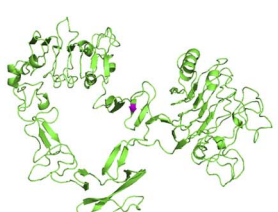
Small and Shear Moves

↑ ↓ 🔗 🗨 ⚙ 📄 🗑 ⋮

SHOW CODE

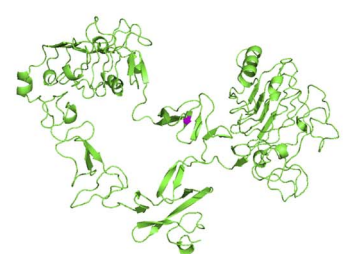
Small mover (1YY9, residue 277):

```
[ ] 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)
```

Fig 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 “12.00 Working with Antibodies.” (C) Integration of PyRosetta with py3Dmol for interactive macromolecular visualization in Jupyter Notebooks.

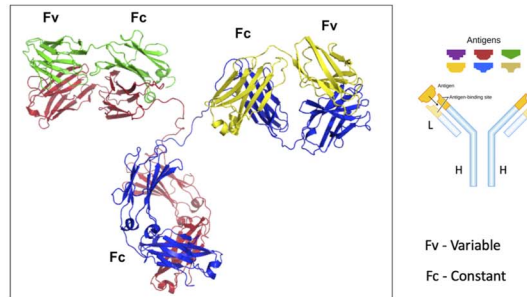
B

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()
```

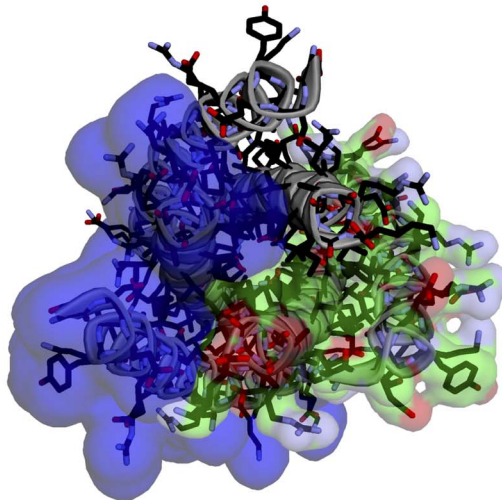


Fig 3. Continued.

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.

```
[5] import math
import random

def randTrial(your_pose):
    ### BEGIN SOLUTION
    randNum = random.randint(2, your_pose.total_residue())
    phiOrpsi = random.randint(0,1)
    if phiOrpsi == 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.

```
[5] 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()
```

Fig 4. 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.

A

index.ipynb ☆
File Edit View Insert Runtime Tools Help *Last edited on October 4*

+ Code + Text

PyRosetta

Keyword Index

- AddMembraneMover
 - [Setting up a membrane protein in the bilayer](#)
- 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](#)
- bilayer
 - [Setting up a membrane protein in the bilayer](#)
- boltzmann()
 - [High-Resolution Movers](#)
- bond_angle()
 - [Protein Geometry](#)
- bond_length()

B

toc.ipynb ☆
File Edit View Insert Runtime Tools Help *All changes saved*

+ 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](#)
- [1.4 Jupyter Notebooks, Python, and Google Colaboratory](#)
 - [Links](#)
 - [tutorial](#)
- [1.5 Frequently Asked Questions/Troubleshooting Tips](#)

Chapter 2.0 Introduction to PyRosetta

- [Links](#)
 - [PyRosetta book](#)
- [2.1 Pose Basics](#)
 - [Loading in a PDB File ##](#)
 - [What is a Pose?](#)
 - [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](#)

Fig 5. Automatically generated supporting material by using `nbpages`. (A) Keyword index notebook with links. (B) Table of contents notebook with links.

curriculums. In addition, any changes to the repository files that would benefit the public can be incorporated directly into the main repository via GitHub pull requests, which can be reviewed and approved by a RosettaCommons member. Figure 6A shows a workflow for instructors who simply want to use the workshops in courses, and Figure 6B illustrates a workflow for instructors who wish to make changes to the material.

Further, in Chapter 2, we showcase the ability to visualize macromolecules directly within the Jupyter notebooks by using `py3Dmol`, a Web-based Jupyter widget encompassing an interactive `3Dmol.js` molecular viewer (60). The `py3Dmol` bindings (in the `pyrosetta.distributed.viewer` namespace) facilitate on-the-fly, interactive visualization of PyRosetta `ResidueSelector` objects, which allow students to choose subsets of residues on the

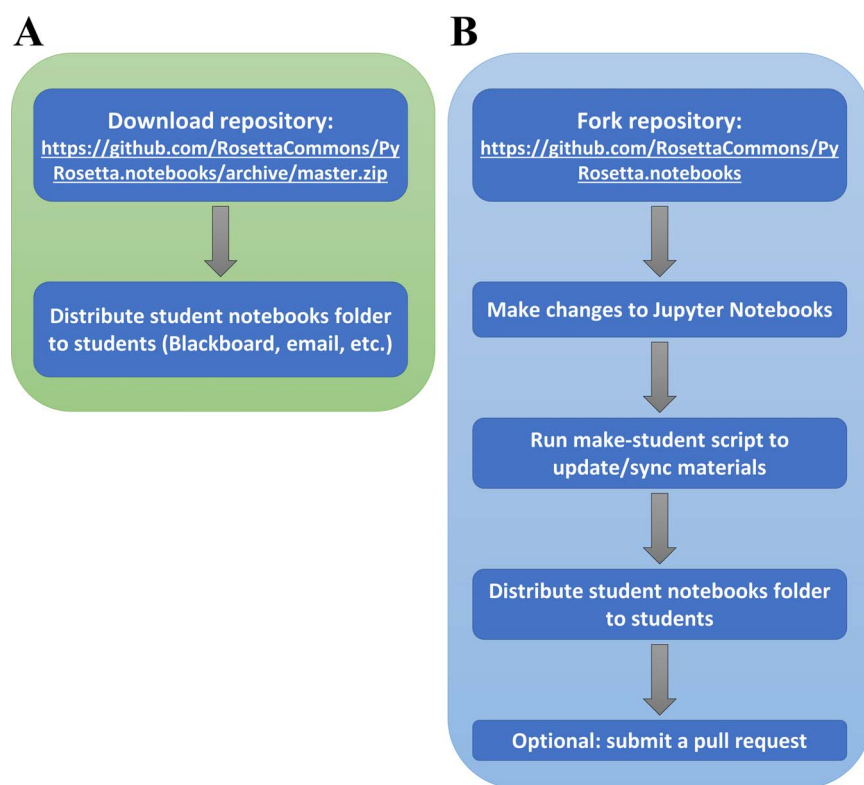


Fig 6. Instructor workflows. (A) Workflow without making changes. (B) Workflow with changes to share with others via pull request to the RosettaCommons GitHub repository.

basis of the sequence, chemistry, or structural properties (Fig 3C). For those who install PyRosetta on a local computer, the motions of a protein in a protocol can be watched in an external PyMOL window by using the PyRosetta PyMOLObserver (61).

Chapter 16 demonstrates how to scale up simulations to high-performance computing resources by using the Slurm workload manager (<https://slurm.schedmd.com/>) with GNU parallel, dask and distributed modules (53, 62). We additionally introduce the `pyrosetta.distributed.dask` namespace for PyRosetta integration with the `dask-jobqueue` module, providing a user-friendly interface for PyRosetta preinitialization of worker machines allowing options-based configuration of macromolecular modeling tasks in distributed computing and cloud computing environments. These developments will enable future pedagogic 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 4 separate teaching contexts. They were used in a formal university course in spring 2019 for a combined graduate and undergraduate elective course, in a code academy for new graduate and postdoctoral students and in RosettaCommons labs and for a 1-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.

In the spring of 2019, the formal university course, ChemBE414/614: Protein Structure Prediction and Design, enrolled 9 undergraduate students and 11 master's or PhD students. Over the years, students who have taken this course have come from departments of chemical and biomolecular engineering, biomedical engineering, biophysics, chemistry, computer science, and applied math. In the precourse survey, roughly half (45%) of the spring 2019 class indicated that they had “good” or “expert-level” familiarity with Python. Nearly

all of these students had programming experience in some language prior to the course. However, experience levels varied greatly in programming, biology, chemistry, and math. Following the course, students gave high reviews (quality of course 4.65 of 5.00 and teaching effectiveness 4.53 of 5.00). The biomolecular computation 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 methodologic side, “Comparison of Optimization Methods Used in Protein Structure Prediction.” This course used the notebooks on local Linux (<https://www.linux.org>) or Mac (Apple, Inc., Cupertino, CA) installations, without the use of the Google Colaboratory platform. In the pilot, multiple students mentioned the technical challenges of using PyRosetta on their computer in course evaluation.

Similar to ChemBE414/614, code academy trainees varied in programming and scientific experience. In the precourse survey, 7 of 19 of trainees (37%) indicated that they had “very little to no programming experience.” After the course, trainees were asked to respond to a postcourse survey. When asked about whether Jupyter Notebooks were effective teaching tools, 13 of 15 respondents selected “agree” or “strongly agree.” Furthermore, all respondents agreed or strongly agreed that the course gave them confidence to write more advanced protocols for research. Code academy trainees completed miniprojects such as “Antibody Design for Ebola” and “Modeling Intrinsically Disordered Proteins for Cell Signaling.”

The summer interns continued to complete successful research projects in 10 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.”

IV. 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 biomolecular structure prediction and design that can be used in a classroom context or for individual self-study. Educators within the Rosetta community have already used these notebooks extensively, and we hope that educators teaching high school, undergraduate, and graduate courses will also benefit from using and adapting these notebooks. A good starting point for new instructors to develop the necessary background to teach these workshops is a recent review article by Kuhlman and Bradley (63). New instructors can complete the workshops themselves and read the associated primary literature linked within each notebook.

Students are advised to have some familiarity with basic Python capabilities, including creating and calling variables, functions, and classes. For a classroom setting, reviewing these skills prior to attempting the workshops may be beneficial. In ChemBE414/614 at Johns Hopkins, the instructor spent 1 week reviewing the necessary skills and assigned 1 homework assignment practicing Python.

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 in a 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 technologic 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 (48), and we have so far been unable to make this connection simple. Although the `PyMOLObserver` is the archetypal tool for real-time visualization of PyRosetta modeling trajectories (61), students with a local installation of PyRosetta are, nevertheless, able to view the algorithms in real-time on the local computer's PyMOL. Within the Google Colaboratory, the `pyrosetta.distributed.viewer` (with `py3Dmol` bindings) currently supports dynamic visualization updates upon biomolecular conformational changes, which is convenient for viewing intermediate steps of biomolecular modeling tasks, such as between PyRosetta movers, directly within Jupyter Notebooks. Although 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 biomolecular modeling and design. However, because `py3Dmol` does not have multithreading or communication between threads, Google Colaboratory users cannot continuously update an instance of the `3Dmol.js` molecular viewer as a PyRosetta protocol trajectory is calculated.

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 (64) and the Web-based lessons on CHARMM (65). Recently, MOE has been used for an integrated engineering curriculum (66). Additionally, Foldit (67) and EteRNA (68) have been used in an interdisciplinary week-long program for un-

dergraduate and high school students (69). Our contribution of PyRosetta notebooks is complementary to these and has advantages with its active involvement of students, multimedia integration, and engagement with viable and leading tools that can be used flexibly in new, innovative research. In addition, unlike many other electronic workshop materials, the PyRosetta notebooks are used to automatically test new versions of the PyRosetta software. Each Jupyter Notebook is converted into a simple Python script that is run continually on the community's testing servers, and any malfunctions from new or modified code must be fixed before accepted into the main Rosetta repository. The automated notebook testing and GitHub pull request practices ensure that the workshops always remain functional for users.

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 used in high school lab courses, science, technology, engineering, and mathematics 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 RosettaCommons forum (<https://www.rosettacommons.org/forum>). In addition, Google Colaboratory is compatible with TensorFlow (<https://www.tensorflow.org/>) and is heavily used for developing machine learning models because of the free graphics processing unit resources. In the future, machine learning could be incorporated with the capabilities of PyRosetta to explore new areas of research in the field (70). 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.

ACKNOWLEDGMENTS

Sidhartha Chaudhury cowrote the original set of (paper) workshops that grew over the years into this collection. Evan H. 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 RosettaCommons mini-grant, a Johns Hopkins Center for Educational Resources Technology Fellowship to KHL, a Stanford School of Medicine Tuition Revenue Sharing Teaching Innovation Grant to MAA, a Hertz Foundation Fellowship to RFA, National Science Foundation (NSF) Graduate Research Fellowships to JCK, RR, and RFA, and grants NSF/DBI 1659649, National Institutes of Health (NIH) R01-GM127578, NIH R01-GM078221, NIH R01-GM73151, NIH R35 GM122517, NSF/CCI 1740549, and NSF CHE-1740549.

Declaration of Competing Interests: The teaching tools described in this article relate to Rosetta software, and some of the funding for this study was provided by RosettaCommons. JJG and JBS are unpaid board members of the RosettaCommons. Under institutional participation agreements between the University of Washington, acting on behalf of the RosettaCommons, Johns Hopkins University and University of California, Davis (as well as other RosettaCommons institutions where work was performed on this article) may be entitled to a portion of revenue received on commercial licensing of Rosetta software. As a member of the scientific advisory board, JJG has a financial interest in Cyrus Biotechnology. Cyrus Biotechnology distributes the Rosetta software, which may include methods described in this article. These arrangements have been reviewed and approved by Johns Hopkins University in accordance with its conflict of interest policies.

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.

REFERENCES

- Guzenko, D., A. Lafita, B. Monastyrskyy, A. Kryshtafovych, and J. M. Duarte. 2019. Assessment of protein assembly prediction in CASP13. *Proteins* 87(12):1190–1199. <https://doi.org/10.1002/prot.25795>.
- Huang, P. S., S. E. Boyken, and D. Baker. 2016. The coming of age of de novo protein design. *Nature* 537(7620):320–327. <https://doi.org/10.1038/nature19946>.
- Nobel Media AB. 2021. The Nobel Prize in Chemistry 2013. Accessed 11 February 2021. <https://www.nobelprize.org/prizes/chemistry/2013/summary/>.
- Brooks, B. R., C. L. Brooks, III, A. D. Mackerell, Jr., L. Nilsson, R. J. Petrella, B. Roux, Y. Won, G. Archontis, C. Bartels, S. Boresch, A. Caffisch, L. Caves, Q. Cui, A. R. Dinner, M. Feig, S. Fischer, J. Gao, M. Hodoscek, W. Im, K. Kuczera, T. Lazaridis, J. Ma, V. Ovchinnikov, E. Paci, R. W. Pastor, C. B. Post, J. Z. Pu, M. Schaefer, B. Tidor, R. M. Venable, H. L. Woodcock, X. Wu, W. Yang, D. M. York, and M. Karplus. 2009. CHARMM: the biomolecular simulation program. *J Comput Chem* 30(10):1545–1614. <https://doi.org/10.1002/jcc.21287>.
- Schrödinger, Inc. 2011. Schrodinger Software Suite, version 2021-1. <https://www.schrodinger.com/>.
- Chemical Computing Group, Inc. 2013. Molecular Operating Environment (MOE), version 2019.01. <https://www.chemcomp.com/index.htm>.
- Phillips, J. C., R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kalé, and K. Schulten. 2005. Scalable molecular dynamics with NAMD. *J Comput Chem* 26(16):1781–1802. <https://doi.org/10.1002/jcc.20289>.
- Pearlman, D. A., D. A. Case, J. W. Caldwell, W. S. Ross, T. E. Cheatham, S. DeBolt, D. Ferguson, G. Seibel, and P. Kollman. 1995. 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* 91(1–3):1–41. [https://doi.org/10.1016/0010-4655\(95\)00041-D](https://doi.org/10.1016/0010-4655(95)00041-D).
- Berendsen, H. J. C., D. van der Spoel, and R. van Drunen. 1995. GROMACS: a message-passing parallel molecular dynamics implementation. *Comput Phys Commun* 91(1–3):43–56. [https://doi.org/10.1016/0010-4655\(95\)00042-E](https://doi.org/10.1016/0010-4655(95)00042-E).
- Das, R., and D. Baker. 2008. Macromolecular modeling with Rosetta. *Annu Rev Biochem* 77:363–382. <https://doi.org/10.1146/annurev.biochem.77.062906.171838>.
- Leman, J. K., B. D. Weitzner, S. M. Lewis, Jared Adolf-Bryfogle, N. Alam, R. F. Alford, M. Aprahamian, D. Baker, K. A. Barlow, P. Barth, B. Basanta, B. J. Bender, K. Blacklock, J. Bonet, S. E. Boyken, P. Bradley, C. Bystruff, P. Conway, S. Cooper, B. E. Correia, B. Coventry, R. Das, R. M. De Jong, F. DiMaio, L. Dsilva, R. Dunbrack, A. S. Ford, B. Frenz, D. Y. Fu, C. Geniesse, L. Goldschmidt, R. Gowthaman, J. J. Gray, D. Gront, S. Guffy, S. Horowitz, P.-S. Huang, T. Huber, T. M. Jacobs, J. R. Jeliaskov, O. K. Johnson, K. Kappel, J. Karanickolas, H. Khakzad, K. R. Khar, S. D. Khare, F. Khatib, A. Khrumushin, I. C. King, R. Kleffner, B. Koepnick, T. Kortemme, G. Kuenze, B. Kuhlman, D. Kuroda, J. W. Labonte, J. K. Lai, G. Lapidoth, A. Leaver-Fay, S. Lindert, T. Linsky, N. London, J. H. Lubin, S. Lyskov, J. Maguire, L. Malmström, E. Marcos, O. Marcu, N. A. Marze, J. Meiler, R. Moretti, V. Khipple Mulligan, S. Nerli, C. Norm, S. Ó'Conchúir, N. Ollikainen, S. Ovchinnikov, M. S. Pacella, X. Pan, H. Park, R. E. Pavlovic, M. Pethe, B. G. Pierce, K. Bharath Pilla, B. Raveh, P. D. Renfrew, S. S. Roy Burman, A. Rubenstein, M. F. Sauer, A. Scheck, W. Schief, O. Schueler-Furman, Y. Sedan, A. M. Sevy, N. G. Sgourakis, L. Shi, J. B. Siegel, D.-A. Silva, S. Smith, Y. Song, A. Stein, M. Szegegy, F. K. D. Teets, S. B. Thyme, R. Yu-Ruei Wang, A. Watkins, L. Zimmerman, and R. Bonneau. 2020. Macromolecular modeling and design in Rosetta: new methods and frameworks. *Nat Methods* 17:665–680.
- Leaver-Fay, A., M. Tyka, S. M. Lewis, O. F. Lange, J. Thompson, R. Jacak, K. W. Kaufman, P. D. Renfrew, C. A. Smith, W. Sheffler, I. W. Davis, S. Cooper, A. Kreuille, D. J. Mandell, F. Richter, Y.-E. A. Ban, S. J. Fleishman, J. E. Corn, D. E. Kim, S. Lyskov, M. Berrondo, S. Mentzer, Z. Popović, J. J. Havranek, J. Karanickolas, R. Das, J. Meiler, T. Kortemme, J. J. Gray, B. Kuhlman, D. Baker, and P. Bradley. 2011. Rosetta3: an object-oriented software suite for the simulation and design of macromolecules. In *Methods in Enzymology*, vol 487. M. L. Johnson and L. Brand, editors. Academic Press, Cambridge, MA, pp. 545–574. <https://doi.org/10.1016/B978-0-12-381270-4.00019-6>.
- Kaufmann, K. W., G. H. Lemmon, S. L. Deluca, J. H. Sheehan, and J. Meiler. 2010. Practically useful: what the Rosetta protein modeling suite can do for you. *Biochemistry* 49(14):2987–2998. <https://doi.org/10.1021/bi902153g>.
- Combs, S. A., S. L. Deluca, S. H. Deluca, G. H. Lemmon, D. P. Nannemann, E. D. Nguyen, J. R. Willis, J. H. Sheehan, and J. Meiler. 2013. Small-molecule ligand docking into comparative models with Rosetta. *Nat Protoc* 8(7):1277–1298. <https://doi.org/10.1038/nprot.2013.074>.
- Cheng, C. Y., F. C. Chou, and R. Das. 2015. Modeling complex RNA tertiary folds with Rosetta. In *Methods in Enzymology*, vol 553. S.-J. Chen and D. H. Burke-Aguero, editors. Academic Press, Cambridge, MA, pp. 35–64. <https://doi.org/10.1016/bs.mie.2014.10.051>.
- Alford, R. F., J. Koehler Leman, B. D. Weitzner, A. M. Duran, D. C. Tilley, A. Elazar, J. and J. Gray. 2015. An integrated framework advancing membrane protein modeling and design. *PLoS Comput Biol* 11(9):e1004398. <https://doi.org/10.1371/journal.pcbi.1004398>.
- Ovchinnikov, S., H. Park, D. E. Kim, F. DiMaio, and D. Baker. 2018. Protein structure prediction using Rosetta in CASP12. *Proteins* 86(S1):113–121. <https://doi.org/10.1002/prot.25390>.
- Weitzner, B. D., D. Kuroda, N. Marze, J. Xu, and J. J. Gray. 2014. Blind prediction performance of RosettaAntibody 3.0: grafting, relaxation, kinematic loop modeling, and full CDR optimization. *Proteins* 82(8):1611–1623. <https://doi.org/10.1002/prot.24534>.
- Burman, S. S. R., M. L. Nance, J. R. Jeliaskov, J. W. Labonte, J. H. Lubin, N. Biswas, J. and J. Gray. 2020. Novel sampling strategies and a coarse-grained score function for docking homomers, flexible heteromers, and oligosaccharides using Rosetta in CAPRI Rounds 37–45. *Proteins* 88(8):973–985. <https://doi.org/10.1002/prot.25855>.
- Marcu, O., E. J. Dodson, N. Alam, M. Sperber, D. Kozakov, M. F. Lensink, and O. Schueler-Furman. 2017. FlexPepDock lessons from CAPRI peptide–protein rounds and suggested new criteria for assessment of model quality and utility. *Proteins* 85(3):445–462. <https://doi.org/10.1002/prot.25230>.
- Miao, Z., R. W. Adamiak, M. Antczak, R. T. Batey, A. J. Becka, M. Biesiada, M. J. Boniecki, J. M. Bujnicki, S.-J. Chen, C. Y. Cheng, F.-C. Chou, A. R. Ferré-D'Amaré, R. Das, W. K. Dawson, F. Ding, N. V. Dokholyan, S. Dunin-Horkawicz, C. Geniesse, K. Kappel, W. Kladwang, A. Krokhotin, G. E. Läch, F. Major, T. H. Mann, M. Magnus, K. Pachulska-Wieczorek, D. J. Patel, J. A. Piccirilli, M. Popena, K. J. Purzycka, A.

- Ren, G. M. Rice, J. Santalucia, Jr., J. Sarzynska, M. Szachniuk, A. Tandon, J. J. Trausch, S. Tian, J. Wang, K. M. Weeks, B. Williams, II, Y. Xiao, X. Xu, D. Zhang, T. Zok, and E. Westhof. 2017. RNA-puzzles round III: 3D RNA structure prediction of five riboswitches and one ribozyme. *RNA* 23(5):655–672. <https://doi.org/10.1261/ma.060368.116>.
22. Kuhlman, B., G. Dantas, G. C. Ireton, G. Varani, B. L. Stoddard, and D. Baker. 2003. Design of a novel globular protein fold with atomic-level accuracy. *Science* 302(5649):1364–1368.
23. Koga, N., R. Tatsumi-Koga, G. Liu, R. Xiao, T. B. Acton, G. T. Montelione, and D. Baker. 2012. Principles for designing ideal protein structures. *Nature* 491(7423):222–227.
24. Jacobs, T. M., B. Williams, T. Williams, X. Xu, A. Eletsy, J. F. Federizon, T. Szyperski, and B. Kuhlman. 2016. Design of structurally distinct proteins using strategies inspired by evolution. *Science* 352(6286):687–690.
25. Humphris, E. L., and T. Kortemme. 2008. Prediction of protein-protein interface sequence diversity using flexible backbone computational protein design. *Structure* 16(12):1777–1788.
26. Guntas, G., C. Purbeck, and B. Kuhlman. 2010. Engineering a protein–protein interface using a computationally designed library. *Proc Natl Acad Sci U S A* 107(45):19296–19301.
27. Fleishman, S. J., T. A. Whitehead, D. C. Ekiert, C. Dreyfus, J. E. Corn, E.-M. Strauch, I. A. Wilson, and D. Baker. 2011. Computational design of proteins targeting the conserved stem region of influenza hemagglutinin. *Science* 332(6031):816–821.
28. Strauch, E.-M., S. M. Bernard, D. La, A. J. Bohn, P. S. Lee, C. E. Anderson, T. Nieuwsma, C. A. Holstein, N. K. Garcia, K. A. Hooper, R. Ravichandran, J. W. Nelson, W. Sheffler, J. D. Bloom, K. K. Lee, A. B. Ward, P. Yager, D. H. Fuller, I. A. Wilson, and D. Baker. 2017. Computational design of trimeric influenza-neutralizing proteins targeting the hemagglutinin receptor binding site. *Nat Biotechnol* 35(7):667–671.
29. King, N. P., J. B. Bale, W. Sheffler, D. E. McNamara, S. Gonen, T. Gonen, T. O. Yeates, and D. Baker. 2014. Accurate design of co-assembling multi-component protein nanomaterials. *Nature* 510(7503):103–108.
30. King, N. P., W. Sheffler, M. R. Sawaya, B. S. Vollmar, J. P. Sumida, I. André, T. Gonen, T. O. Yeates, and D. Baker. 2012. Computational design of self-assembling protein nanomaterials with atomic level accuracy. *Science* 336(6085):1171–1174.
31. Masica, D. L., S. B. Schrier, E. A. Specht, and J. J. Gray. 2010. De novo design of peptide-calcite biomineralization systems. *J Am Chem Soc* 132(35):12252–12262.
32. Pyles, H., S. Zhang, J. J. De Yoreo, and D. Baker. 2019. Controlling protein assembly on inorganic crystals through designed protein interfaces. *Nature* 571(7764):251–256.
33. Chaudhury, S., S. Lyskov, and J. J. Gray. 2010. PyRosetta: a script-based interface for implementing molecular modeling algorithms using Rosetta. *Bioinformatics* 26(5):689–691. <https://doi.org/10.1093/bioinformatics/btq007>.
34. Gray, J. J., S. Chaudhury, S. Lyskov, and J. W. Labonte. 2017. The PyRosetta Interactive Platform for Protein Structure Prediction and Design: A Set of Educational Modules. 3rd edition. Createspace Independent Publishing Platform, Scotts Valley, CA.
35. Gray, J. J., S. Chaudhury, S. Lyskov, and J. W. Labonte. 2019. PyRosetta tutorials. Accessed 11 December 2019. <http://www.pyrosetta.org/tutorials>.
36. Alford, R. F., A. Leaver-Fay, L. Gonzales, E. L. Dolan, and J. J. Gray. 2017. A cyber-linked undergraduate research experience in computational biomolecular structure prediction and design. *PLOS Comput Biol* 13(12):e1005837. <https://doi.org/10.1371/journal.pcbi.1005837>.
37. Fleishman, S. J., A. Leaver-Fay, J. E. Corn, E. M. Strauch, S. D. Khare, N. Koga, J. Ashworth, P. Murphy, F. Richter, G. Lemmon, J. Meiler, and D. Baker. 2011. RosettaScripts: a scripting language interface to the Rosetta macromolecular modeling suite. *PLOS ONE* 6(6):e20161. <https://doi.org/10.1371/journal.pone.0020161>.
38. RosettaCommons. 2019. What is Rosetta? Accessed 11 December 2019. <https://www.rosettacommons.org/docs/latest/Home>.
39. RosettaCommons. 2019. Rosetta manuals. Accessed 11 December 2019. <https://www.rosettacommons.org/manuals/latest/>.
40. RosettaCommons. 2019. Rosetta forums. Accessed 11 December 2019. <https://www.rosettacommons.org/forum>.
41. Freeman, S., S. L. Eddy, M. McDonough, M. K. Smith, N. Okoroafo, H. Jordt, and M. P. Wenderoth. 2014. Active learning increases student performance in science, engineering, and mathematics. *Proc Natl Acad Sci U S A* 111(23):8410–8415. <https://doi.org/10.1073/pnas.1319030111>.
42. Sankey, M., D. Birch, and M. Gardiner. 2010. Engaging students through multimodal learning environments: the journey continues. Paper presented at the Australasian Society for Computers in Learning in Tertiary Education 2010. Sydney, Australia, 5–8 December 2010.
43. Edgcomb, A. D., and F. Vahid. 2014. Effectiveness of online textbooks vs. interactive web-native content. Paper presented at the American Society for Engineering Education Annual Conference and Exposition. Indianapolis, IN, 15–18 June 2014.
44. Kluyver, T., B. Ragan-Lelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing. 2016. Jupyter Notebooks—a publishing format for reproducible computational workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. F. Lorizides and B. Schmidt, editors. IOS Press, Clifton, VA, pp. 87–90. <https://doi.org/10.3233/978-1-61499-649-1-87>.
45. Carneiro, T., R. V. M. Da Nobrega, T. Nepomuceno, G. Bin Bian, V. H. C. De Albuquerque, and P. P. R. Filho. 2018. Performance analysis of Google Colaboratory as a tool for accelerating deep learning applications. *IEEE Access* 6:61677–61685. <https://doi.org/10.1109/ACCESS.2018.2874767>.
46. Karplus, M. 1997. The Levinthal paradox: yesterday and today. *Fold Des* 2:69–75. [https://doi.org/10.1016/S1359-0278\(97\)00067-9](https://doi.org/10.1016/S1359-0278(97)00067-9).
47. Anfinsen, C. B. 1973. Folding of protein chains. *Science*. 181(4096):223–230.
48. Schrödinger, Inc. 2017. The PyMOL Molecular Graphics System, version 2.0. Accessed X XXX XXXX. XXXXX.
49. Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. 1953. Equation of state calculations by fast computing machines. *J Chem Phys* 21(6):1087–1092. <https://doi.org/10.1063/1.1699114>.
50. Weitzner, B. D., J. R. Jeliakzov, S. Lyskov, N. Marze, D. Kuroda, R. Frick, J. Adolf-Bryfogle, N. Biswas, R. L. Dunbrack, and J. J. Gray. 2017. Modeling and docking of antibody structures with Rosetta. *Nat Protoc* 12(2):401–416. <https://doi.org/10.1038/nprot.2016.180>.
51. Das, R., J. Karanicolos, and D. Baker. 2010. Atomic accuracy in predicting and designing noncanonical RNA structure. *Nat Methods* 7(4):291–294. <https://doi.org/10.1038/nmeth.1433>.
52. Overington, J. P., B. Al-Lazikani, and A. L. Hopkins. 2006. How many drug targets are there? *Nat Rev Drug Discov* 5(12):993–996. <https://doi.org/10.1038/nrd2199>.
53. Ford, A. S., B. D. Weitzner, and C. D. Bahl. 2019. Integration of the Rosetta Suite with the Python Software Stack via reproducible packaging and core programming interfaces for distributed simulation. *Protein Sci* 29(1):43–51. <https://doi.org/10.1002/pro.3721>.
54. Kantor, J. 2019. CBE20255 Introduction to chemical engineering analysis. Accessed 15 December 2019. <http://jckantor.github.io/CBE20255/>.
55. Marrugo, A. 2019. Sensors and actuators. Accessed 15 December 2019. <https://github.com/agmarrugo/sensors-actuators>.
56. Le, K. H. 2021. Pyrosettaacolabsetup 0.2. Accessed 12 February 2021. <https://pypi.org/project/pyrosettaacolabsetup/>.
57. Jupyter, P., 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, and C. Willing. 2019. nbgrader: a tool for creating and grading assignments in the Jupyter Notebook. *J Open Source Educ* 2(11):32. <https://doi.org/10.21105/jose.00032>.
58. Kantor, J. 2021. nbpages. Accessed 12 February 2021. <https://pypi.org/project/nbpages/>.
59. GitHub. 2020. About collaborative development models. Accessed 9 April 2020. <https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/about-collaborative-development-models>.
60. Rego, N., and D. Koes. 2015. 3Dmol.js: molecular visualization with WebGL. *Bioinformatics* 31(8):1322–1324.
61. Baugh, E. H., S. Lyskov, B. D. Weitzner, and J. J. Gray. 2011. Real-time PyMOL visualization for Rosetta and PyRosetta. *PLOS ONE* 6(8):e21931. <https://doi.org/10.1371/journal.pone.0021931>.
62. Rocklin, M. 2015. Dask: parallel computation with blocked algorithms and task scheduling. In *Proceedings of the 14th Python in Science Conference*. K. Huff and J. Bergstra, editors. Austin, TX, 6–12 July 2015. pp. 126–132. <https://doi.org/10.25080/majora-7b98e3ed-013>.
63. Kuhlman, B., and P. Bradley. 2019. Advances in protein structure prediction and design. *Nat Rev Mol Cell Biol* 20(11):681–697. <https://doi.org/10.1038/s41580-019-0163-x>.
64. Schleich, R. 2013. A concise guide to CHARMM and the analysis of protein structure and function. Accessed 14 October 2020. https://pages.jh.edu/~rschlei1/Random_stuff/publications/charmmbook.pdf.
65. Miller, B. T., R. P. Singh, V. Schalk, Y. Pevzner, J. Sun, C. S. Miller, S. Boresch, T. Ichiye, B. R. Brooks, and H. L. Woodcock. 2014. Web-based computational

- chemistry education with CHARMMing I: lessons and tutorial. *PLOS Comput Biol* 10(7):e1003719. <https://doi.org/10.1371/journal.pcbi.1003719>.
66. Ludovice, P., and D. MacNair. 2019. Integrated molecular modeling education in the chemical engineering curriculum. Paper presented at American Institute of Chemical Engineers 2019 Annual Meeting. Orlando, FL, 12 November 2019.
67. Khatib, F., A. Desfosses, B. Koepnick, J. Flatten, Z. Popović, D. Baker, S. Cooper, I. Gutsche, and S. Horowitz. 2019. Building de novo cryo-electron microscopy structures collaboratively with citizen scientists. *PLOS Biol* 17(11):e3000472. <https://doi.org/10.1371/journal.pbio.3000472>.
68. Lee, J., W. Kladwang, M. Lee, D. Cantu, M. Azizyan, H. Kim, A. Limpaecher, S. Yoon, A. Treuille, and R. Das. 2014. RNA design rules from a massive open laboratory. *Proc Natl Acad Sci U S A* 111(6):2122–2127. <https://doi.org/10.1073/pnas.1313039111>.
69. Taly, A., F. Nitti, M. Baaden, and S. Pasquali. 2019. Molecular modelling as the spark for active learning approaches for interdisciplinary biology teaching. *Interface Focus* 9(3):20180065.
70. Kandathil, S. M., J. G. Greener, and D. T. Jones. 2019. Recent developments in deep learning applied to protein structure prediction. *Proteins* 87(12):1179–1189. <https://doi.org/10.1002/prot.25824>.