Project Guide

CS 279 / BIOE 279 / BIOPHYS 279 / BMDS 245 / CME 279
2026

The CS279 project is an open-ended opportunity for you to explore a topic of your choice in more depth. While this freedom is exciting, it can also feel overwhelming at first. This optional handout walks you through choosing and scoping a project step by step.

This handout is here to guide you in:

- Narrowing down your interests into a feasible project.
- Choosing a focus area for your project, a biological system or dataset (if applicable), and a question.
- Checking that your project is the right size and scope.
- Identifying useful resources, tools, and datasets.

Suggested Project Roadmap

Step 1: What Interests You?

Start by thinking about all the topics covered in lectures, including biomolecular structure prediction, molecular dynamics simulation, docking, virtual screening, protein design, cryo-EM, X-ray crystallography, fluorescence microscopy, cell image analysis, and reaction-diffusion simulation. Other topics are also acceptable, but your project must involve 3D structures of biomolecules and/or the spatial organization of molecules within a cell. Then ask yourself:

- Would you rather run experiments with existing software, write code of your own, or a mix of both?
- Is there a computational method or piece of software you want to understand more deeply, test, or improve?
- Is there a protein, other biomolecule, cellular process, or cell type you are curious about?

Any of these interests can make a good project.

Step 2: What Type of Project Suits You?

Your project can fall into one (or more) of these categories:

- **Using existing tools**: Apply existing software to analyze a biological system of interest.
- **Comparing tools**: Evaluate multiple computational tools (e.g., different structure prediction methods, different docking methods, or different MD simulation packages) to see which performs best under specific conditions.

- **Building new tools**: Develop a new algorithm or software, or extend existing algorithms or software (including the codes you worked with in the Assignments), for a computational biology task.

You can combine these types, and many good projects do.

Step 3: Choose a Biological System or Dataset

Most projects involve either a biological system (e.g., a particular biomolecule, a set of related biomolecules, a particular cell type, or a particular biological process) or a dataset. Some ways to find a biological system or dataset of interest to you:

- **From class**: a number of important biological systems have been mentioned as examples during lectures.
- **Databases on important protein families and drugs**: for example, GPCRdb provides information on G protein-coupled receptors (GPCRs); KLIFS provides information on kinases; and the IUPHAR/BPS Guide to Pharmacology provides information on almost all drug targets.
- **From published papers**: review papers, in particular, often describe multiple case studies of computational tools being applied to a specific system, which can point you to systems that work well. Note that Google Scholar and other search tools allow you to search specifically for review papers. The Optional Readings on the course website include a number of papers that may be helpful.
- **From other databases and code repositories**: browse the databases and curated GitHub lists under Tools and Datasets below. Looking at what previous software and research projects have done and which libraries they use can be a quick way to find a starting point.
- For some projects, you can generate synthetic data yourself. For example, you could generate synthetic cell images or simulate noisy cryo-EM images of a known structure.
- Some projects require minimal data (e.g., a single protein structure from the PDB, or a couple reactions to add to a reaction-diffusion simulation).
- If you are doing research in a lab at Stanford or elsewhere, you are welcome to use data you have access to through that research.

Step 4: Frame Your Research Question

Craft a guiding question for your project based on your chosen system or dataset. Examples include:

- How does a common mutation in protein X affect its ability to bind its key ligands?
- Which structure prediction method most accurately predicts structures of protein type Z, and why?
- Given images of healthy and diseased cells, how can an algorithm tell them apart?
- How does adding a reaction between two molecule types change how they spread through a cell?

- How much noise can a cryo-EM reconstruction method tolerate before it no longer recovers a known structure?

Your question will shape your analysis plan, so make it specific and actionable.

Step 5: Plan Your Analysis

How will you analyze results to address the research question? For example:

- If comparing computational tools, identify metrics (e.g., ways to measure accuracy or speed) for evaluation.
- If studying a particular cellular process, what will your results tell you about how that process works?
- If performing an MD simulation, how will you analyze the results to learn about the function or behavior of the molecules you're simulating?
- If designing an algorithm or building a machine learning model, how will you test its accuracy and/or performance?

For many projects, it is helpful to break your research question into smaller sub-questions or tasks and then consider how you will address each.

Step 6: Check Your Project's Feasibility

Your project should:

- **Be relevant to the course** (i.e., involve 3D biomolecular structures or spatial organization).
- **Involve real computation and/or analysis**. Putting a few sequences into AlphaFold and looking at the output is too small, but predicting structures for a carefully chosen set of proteins and analyzing what the predictions get right and wrong could work well. Writing code is not required.
- **Be achievable within the quarter**, at about the amount of work per person of Assignment 2 or 3. Building "AlphaFold 4" is too ambitious, but improving one specific part of structure prediction might not be. Start with a focused idea and expand it if you finish early.

If your project doesn't fully "work"—i.e., if you don't achieve your original goals—that's completely fine. You can get full credit for the project by explaining what you tried, why it didn't work, and what you would do next.

Resources and Practical Tips

Computing Resources

Most projects can be completed using your own laptop computer and/or webservers such as those listed in the sections below. If you wish to do a more computationally intensive project, we have listed here a number of other computing resources available to all Stanford students. Students who have access to other computing resources through individual research labs may also use those.

1. General CPU Computing

These resources are intended for students whose laptops have insufficient computational power but whose projects do not require specialized high-performance computing infrastructure. This includes applications like PyMOL, AutoDock Vina (small docking experiments), ImageJ/Fiji, CellProfiler, ProteinMPNN, small OpenMM or GROMACS simulations, and reaction-diffusion simulations.

- **Stanford LTS computers:** Refer to the course's LTS handout for instructions on accessing these machines. LTS computers come with many commonly used scientific and engineering software packages preinstalled. Remote LTS machines are limited in number, so use in-person machines when possible.
- **Stanford FarmShare:** Free for eligible Stanford students. Provides remote Linux machines for running Python scripts, simulations, and other computational workloads.

2. More Intensive CPU Computing

These resources are intended for students working with larger datasets, longer simulations, or parallel workloads. Suitable applications include molecular dynamics simulations with GROMACS or OpenMM, large-scale molecular docking with AutoDock Vina, image processing with CellProfiler, and reaction-diffusion simulations with MCell or Smoldyn.

- **Stanford FarmShare:** Provides multicore CPU nodes (wheat), high-memory nodes (rye), and a Slurm job scheduler for running computational jobs remotely.
- **Google Cloud Compute Engine:** Eligible new users can receive $300 in trial credits for 90 days to run larger CPU virtual machines. Note that the Always Free e2-micro instance is generally insufficient for intensive scientific computing.

Note: FarmShare is a shared resource, so CPU and memory availability, job durations, and concurrent jobs are subject to limits.

3. GPU Computing

These resources are intended for students working on computationally intensive projects that benefit from GPU acceleration. Suitable applications include biomolecular structure prediction with ColabFold or Boltz, protein design with RFdiffusion, molecular dynamics simulations with OpenMM or GROMACS, and image segmentation or machine learning with Cellpose or PyTorch.

- **Stanford FarmShare:** Provides GPU-equipped oat nodes accessible through the Slurm job scheduler.

    - FarmShare has limited shared GPU capacity, so jobs may need to wait in a queue. Some models require more GPU memory than the available hardware provides.
- **Google Colab:** Offers limited free GPU access through Jupyter notebooks, with paid tiers for additional computing resources. Free GPUs may not always be available due to high demand or usage limits and sessions have runtime limits.

General Software Resources

These software packages may be useful in a variety of projects, but this is by no means an exhaustive list.

- A software suite from Schrödinger—which includes Glide for docking, Desmond for MD simulation, and much more)—is free for current Stanford students through the Science Library's site license. To install it on your own computer: (1) create an account at schrodinger.com with your Stanford email; (2) request a license key with the ServiceNow form; (3) download the software from the releases page and install it. The license only works on the Stanford network, so off campus connect to the Stanford VPN with full-traffic (not split-tunnel) mode. If the key doesn't work, email sciencelibrary@stanford.edu.

- MATLAB (scientific computing, simulation, and data analysis): Available free to Stanford students through the Stanford MATLAB Portal. Sign in with your Stanford email address, create a MathWorks account if needed, and follow the instructions to download and install MATLAB. Alternatively, you can use MATLAB Online directly in your browser without installing anything. For additional help, visit Stanford IT's MATLAB licensing page.

- Tamarind Bio (browser-based computational biology tools): Run protein structure prediction and design tools without local installation. The free plan includes 10 jobs per month, with additional access available through an academic plan with your university email. Also started by former students from this class!

Topic-specific Software, Tools and Datasets

Below are examples of possible project directions and useful resources. These lists are not exhaustive, and you are welcome to propose other topics or tools.

We recommend starting with a project idea that interests you, then choosing the tools and datasets needed to investigate it. Browser-based tools and Google Colab are often the easiest starting points, while some projects may require local installation or more powerful computing resources.

Biomolecular Structure Prediction:

- **Examples of project ideas:** Compare results of different structure prediction methods; compare structure predictions against experimental data; investigate where prediction

methods struggle (e.g., flexible regions, certain complexes, or certain protein families). You could also test whether confidence scores (pLDDT, PAE, pTM, etc.) reliably reflect structural accuracy.

- **Structure prediction tools:**
  - **General protein structure prediction:** ColabFold (Google Colab; no local GPU) and ESMFold (protein structure prediction; GPU or hosted notebook).
  - **AlphaFold 3–style models:** AlphaFold Server (web-based; free with usage limits), Protenix-Mini, Chai-1, and Boltz (local GPU). AlphaFold 3 itself is more difficult to install and requires substantial computing resources.
  - **Template-based modeling:** SWISS-MODEL (web-based), MODELLER (local CPU).
  - **Antibody and nanobody modeling:** IgFold and ImmuneBuilder (local CPU/GPU).
- **Structure evaluation:** Foldseek (web-based structural similarity search), US-align (TM-score), DockQ (protein complex accuracy).
- **Datasets:** PDB (experimental structures), AlphaFold Protein Structure Database (predicted structures), CASP and CAMEO (structure prediction benchmarks).
- **Evaluation guidance:** When comparing methods, report the top-ranked prediction rather than selecting whichever prediction turns out to be most accurate. Consider whether test structures were available during a model's training, and whether confidence scores correlate with actual accuracy.

Molecular Dynamics:

- **Examples of project ideas:** Compare molecular dynamics (MD) simulations using different force fields; investigate how a ligand or mutation affects protein flexibility; compare simulated conformations with experimentally determined structures; analyze how protein dynamics change under different conditions.
- **Tools:**
  - **Simulation:** OpenMM (Python-based; CPU/GPU), GROMACS (CPU/GPU), Desmond (Stanford Schrödinger license; includes a graphical user interface for simulation setup), Tinker (CPU; slower but easier to modify and extend).
  - **Analysis:** MDAnalysis (trajectory analysis), VMD (molecular visualization and simple analysis).
  - **Web-based:** CHARMM-GUI (simulation setup).
- **Datasets:** PDB (experimental structures), GPCRdb (receptors with different ligands), mdCATH and ATLAS (precomputed MD trajectories).

Small-Molecule Docking and Virtual Screening:

- **Examples of project ideas:** Compare docking methods on experimentally solved protein–ligand complexes; evaluate how well docking scores distinguish active compounds from decoys; investigate whether docking programs reproduce experimentally observed binding poses; compare virtual screening performance across protein targets.
- **Tools:**

    - **Docking tools:** Glide (Stanford Schrödinger license), AutoDock Vina (local CPU), GNINA (deep-learning docking; GPU recommended).
    - **Web-based tools:** SwissDock (protein–ligand docking).
    - **Evaluation tools:** PoseBusters (physical plausibility of predicted poses), PLIP (protein–ligand interactions).
- **Datasets and benchmarks:** PDBbind (experimental protein–ligand complexes and affinities; ; older data available free), ZINC (compound library), DUD-E and DUDE-Z (virtual screening benchmarks with active compounds and decoys), PLINDER (protein–ligand benchmark with data splits).
- **Evaluation guidance:** For virtual screening projects, evaluate how well methods distinguish known binders from decoys (e.g., enrichment factors or ROC-AUC), rather than relying only on docking scores. Be mindful of benchmark biases and potential data leakage.

Cellular Image Analysis:

- **Examples of project ideas:** Build a model that tells healthy from diseased cells in an image dataset based on differences in cell morphology; apply the segmentation methods from Assignment 3 to a new dataset; measure how a drug or mutation changes cell or organelle shape or protein distribution within a cell.
- **Tools:**
    - **Image analysis tools:** Fiji/ImageJ (image processing), CellProfiler (automated image analysis), scikit-image and MATLAB (image processing and analysis; MATLAB is free through Stanford's license), PyCytoMiner (single-cell feature analysis).
    - **Segmentation tools:** Cellpose (deep-learning cell segmentation; also available through an official Google Colab notebook), ilastik (interactive pixel classification).
    - **Visualization:** napari (Python image viewer).
- **Datasets:** Image Data Resource (IDR), Broad Bioimage Benchmark Collection, Allen Cell Collection, Human Protein Atlas (protein localization), OpenCell (tagged human proteins).

Protein Design:

- **Examples of project ideas:** Design a protein or peptide to bind a target (e.g., another protein); compare protein design methods on the same target; investigate how design parameters affect structural quality or predicted binding; evaluate whether designed sequences are predicted to fold into their intended structures.
- **Protein design tools:**
    - **Accessible design tools:** ColabDesign and RFdiffusion (Colab notebook) (Google Colab).
    - **Sequence design for a given structure:** ProteinMPNN (protein backbone-based sequence design), LigandMPNN (sequence design around ligands).

    - **Advanced structure and binder design:** RFdiffusion, Chroma, BindCraft, RFdiffusion3, and BoltzGen (GPU required; newer models may need substantial GPU memory).
- **Evaluation guidance:** Assess whether designed sequences are predicted to adopt their intended structures or binding interfaces. Consider structural confidence, predicted interactions, and diversity of designs. Computational predictions alone do not establish experimental success.

Cryo-Electron Microscopy (Cryo-EM):

- **Examples of project ideas:** Build an atomic model from a deposited cryo-EM map and evaluate improvements from refinement; implement a 2D projection, back-projection, or pose estimation algorithm; compare particle-picking methods; benchmark reconstruction methods on CryoBench; investigate whether a model's learned latent space captures known conformational changes.
- **Tools:**
    - **Model building:** ModelAngelo (automatic atomic model building from density maps; GPU with at least 8 GB memory).
    - **Reconstruction and analysis:** CryoSPARC (free for academic use), RELION (CPU/GPU), cryoDRGN (deep-learning analysis of structural heterogeneity; GPU).
    - **Web-based tools:** COSMIC2 (cloud-based cryo-EM processing).
- **Datasets and benchmarks:** EMDB (deposited density maps), CryoBench (five benchmark datasets with ground truth for structural heterogeneity), EMPIAR (raw cryo-EM images), CryoET Data Portal (cryo-electron tomography datasets), CryoPPP (particle-picking benchmark; use a manageable subset).
- **Computational considerations:** Full 3D reconstructions from raw data are GPU-intensive and time-consuming. More manageable projects include implementing a 2D algorithm, using CryoBench datasets with provided poses, or building and refining an atomic model from a deposited density map.

X-ray Crystallography:

- **Examples of project ideas:** Re-refine experimental crystal structures and compare changes in R-free and model geometry; compare deposited structures with re-refined versions from PDB-REDO; simulate diffraction patterns using FFTs and investigate how resolution or missing data affect electron density maps; evaluate AlphaFold predictions as molecular-replacement search models.
- **Tools:**
    - **Structure determination and refinement:** Phenix and CCP4 (free for academic use), Coot (interactive model building).
    - **Structure analysis and validation:** MolProbity (structure quality), gemmi and reciprocalspaceship (Python crystallography tools).

    - **Small-molecule crystallography:** Cambridge Crystallographic Data Centre (CCDC) (free online tools for exploring and analyzing crystal structures; additional software available through Stanford's license).
- **Datasets:** PDB (experimental structures and structure factors), PDB-REDO (re-refined structures), Cambridge Structural Database (CSD) (small-molecule crystal structures), SBGrid Data Bank (raw diffraction images).
- **Computational considerations:** Re-refinement, structure analysis, and diffraction simulations are generally more manageable than processing raw diffraction images. Students can start with deposited PDB structures and structure factors rather than collecting or processing experimental data from scratch.

Reaction-Diffusion Simulation:

- **Examples of project ideas:** Extend the Assignment 3 diffusion simulator to include chemical reactions, molecular crowding, or more complex cell geometries; investigate how spatial organization affects reaction rates or molecular distributions; model diffusion in a realistic cellular environment using experimentally measured parameters.
- **Simulation tools:** Assignment 3 diffusion code (Python; easy to modify), Smoldyn (particle-based reaction-diffusion simulations), MCell (spatial simulations in complex cell geometries), ReaDDy (Python-based particle simulations), Lattice Microbes (hybrid approach; pyLM Python environment avaialble)
- **Computational considerations:** Most small-scale simulations can run on a local CPU. Extending the Assignment 3 code or comparing simple reaction-diffusion models is a good starting point; detailed cellular geometries and large particle counts may require more computational resources.

Other Machine Learning Projects:

- **Examples of project ideas:** Develop a machine learning model to predict protein residue contacts, 3D geometry, or structural properties of molecular interfaces; train a graph neural network to predict protein–ligand interactions; compare different neural network architectures on a structure-based prediction task; investigate whether incorporating 3D structural information improves prediction accuracy.
- **Tools:** PyTorch Geometric (graph neural networks), e3nn (3D equivariant neural networks), PyTorch (general deep learning).
- **Datasets and benchmarks:** ATOM3D (machine learning benchmarks involving 3D molecular structures), PDB (experimental protein structures), PLINDER (protein–ligand interaction benchmarks), CryoBench (cryo-EM reconstruction benchmarks).
- **Evaluation guidance:** Projects should directly involve biomolecular structure, rather than sequence-only tasks such as predicting mutation effects with a protein language model. Use appropriate train/test splits to avoid structural similarity or data leakage between training and evaluation sets.

Other Structure-Related Topics:

- Examples of project ideas: Simulate a disordered protein and test how mutations change how compact it is; compare predicted RNA structures with solved ones; find distant relatives of a protein that a structure search catches but a sequence search misses; analyze how chromosomes fold inside the nucleus using Hi-C data.
- **Tools:**
    - **Disordered proteins:** CALVADOS (protein simulations; Google Colab).
    - **RNA structure:** ViennaRNA (RNA secondary structure), RhoFold+ (RNA 3D structure prediction; GPU).
    - **Structural analysis:** Foldseek (web-based protein structure search).
    - **Chromosome organization:** cooltools (Hi-C data analysis; Python).
- **Datasets:** DisProt (disordered proteins), RNA 3D Hub (experimental RNA structures), ClinVar (disease-associated variants), ENCODE (Hi-C datasets).

Grading and Final Thoughts

Projects are graded holistically on the amount of work per person (i.e., we expect that a three-person project will involve three times as much total work as an individual project), clarity as a piece of scientific writing, and relevance to the course. See the Project Instructions handout for the full requirements. Successful projects are almost never perfect; learning and progress matter more than polished results. Don't hesitate to ask for help from TAs or use resources like GitHub forums and research papers. Most importantly, choose a topic you find exciting and engaging. This is your chance to explore and contribute to the fascinating field of computational biology!