Modeling biological complexes using *Integrative Modeling Platform*

Daniel Saltzberg[1], Charles H. Greenberg[1], Shruthi Viswanath[1], Ilan Chemmama[1], Ben Webb[1], Riccardo Pellarin[2], Ignacia Echeverria[1], Andrej Sali[1]

[1] California Institute for Quantitative Biosciences, University of California, San Francisco, CA 94158, USA.
[2] Structural Bioinformatics Unit, Institut Pasteur, CNRS UMR 3528, 75015 Paris, France.

Summary

Integrative structure modeling provides 3D models of macromolecular systems that are based on information from multiple types of experiments, physical principles, statistical inferences, and prior structural models. Here, we provide a hands-on realistic example of integrative structure modeling of the quaternary structure of the actin, tropomyosin, and gelsolin protein assembly, based on electron microscopy, solution X-ray scattering, and chemical crosslinking data for the complex as well as excluded volume, sequence connectivity, and rigid atomic X-ray structures of the individual subunits. We follow the general four-stage process for integrative modeling, including gathering the input information, converting the input information into a representation of the system and a scoring function, sampling alternative model configurations guided by the scoring function, and analyzing the results. The computational aspects of this approach are implemented in our open-source *Integrative Modeling Platform* (IMP), a comprehensive and extensible software package for integrative modeling (https://integrativemodeling.org). In particular, we rely on the *Python Modeling Interface* (PMI) module of IMP that provides facile mixing and matching of macromolecular representations, restraints based on different types of information, sampling algorithms, and analysis including validations of the input data and output models. Finally, we also outline how to deposit an integrative structure and corresponding experimental data into PDB-Dev, the nascent worldwide Protein Data Bank (wwPDB) resource for archiving and disseminating integrative structures (https://pdb-dev.wwpdb.org). The example application provides a starting point for a user interested in using IMP for integrative modeling of other biomolecular systems.

Key Words

Integrative Modeling. Biomolecular simulation. Biophysical data. Structural modeling

1. Introduction

To understand the function of a macromolecular assembly, we must know the structure and dynamics of its components and the interactions between them. [1–4] However, direct experimental determination of such a structure is generally rather difficult, as no experimental method is universally applicable. For example, crystals suitable for X-ray crystallography cannot always be produced, especially for large assemblies of multiple components. [5] Cryo-electron microscopy, on the other hand, can be used to study large assemblies, but is often limited to worse than atomic resolution. [6–8] Finally, molecular biology, biochemistry, and proteomics techniques, such as yeast two-hybrid, [9] affinity purification, [10] and mass spectrometry, [11] yield information about the interactions between proteins, but not the positions of these proteins within the assembly or the structures of the proteins themselves.

One approach to solve this problem is integrative modeling, [12] that is used to characterize the structures of single proteins or their complexes by relying on multiple types of input information, including varied experiments, physical theories, statistical inferences, and prior structural models. By simultaneously considering all information, the method maximizes the accuracy, precision, completeness, and efficiency of structure determination. Numerous structures have already been solved using this approach, including the 26S ribosome, [13] the bacterial type II pilus, [14] the structure of chromatin around the alpha-globin gene, [15] the molecular architecture of the yeast spindle pole body core, [16] and the architecture of the yeast nuclear pore complex. [17] The method can also compute multi-state models of conformationally heterogenous systems, as demonstrated by the two-state model of the PhoQ sensor histidine kinase. [18]

The Integrative Modeling Platform (IMP) is a comprehensive and extensible software package for performing integrative modeling. The flexibility of the core software allows for constructing customized representations of structure and data as well as sampling and analysis protocols. The tools to complete the entire integrative modeling workflow [Fig. 1] are contained within IMP. Here, we describe the Python Modeling Interface (PMI) to IMP that significantly simplifies encoding the modeling process. [19]

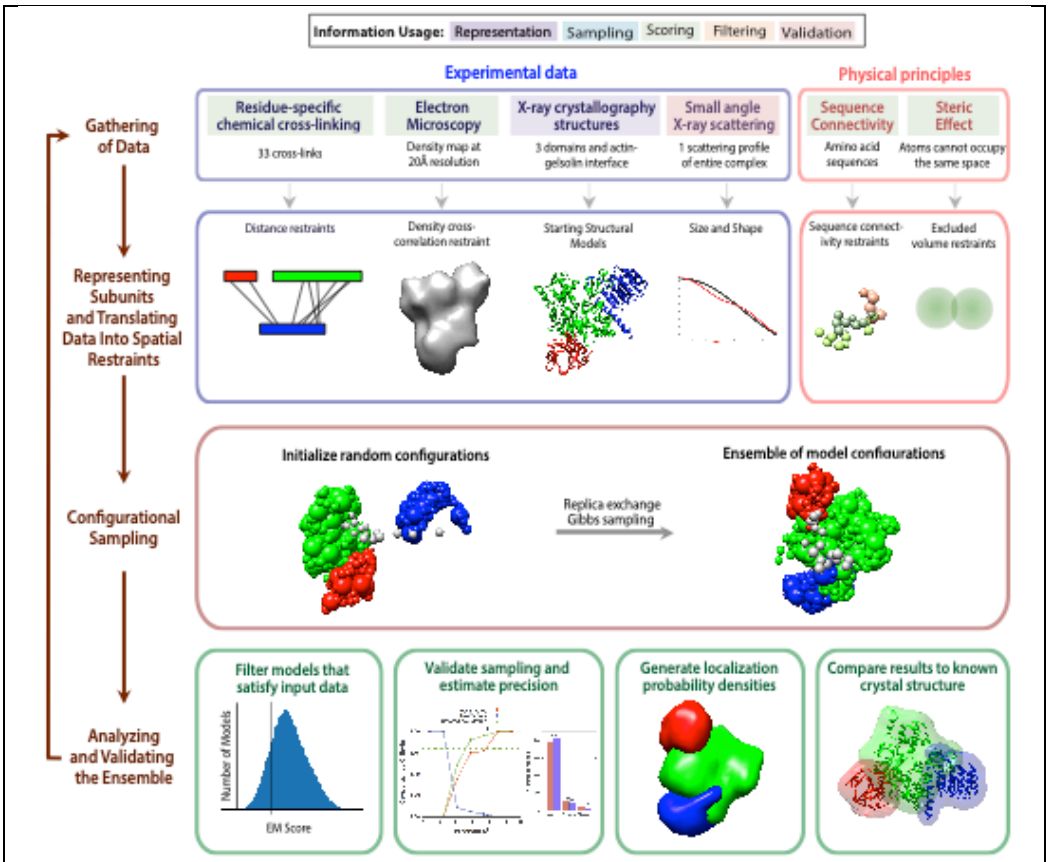

Figure 1: The four stages of integrative modeling. This schematic describes the integrative structure modeling procedures used in this tutorial. The first row details the information to be used in modeling. The background color of each information source indicates where the information is applied in modeling, as detailed in the key at the top. The second row describes how each information source is converted into spatial restraints. The third row details the sampling protocol. The last row details the analysis and validation steps of the modeling. The modifiable Adobe Illustrator file for this figure can be found in the tutorial repository: `figures/Figure1/actin_tutorial_4stage.ai`.

2. Methods

The goal of PMI is to allow structural biologists with limited programming expertise to determine the structures of large protein complexes. PMI is a top-down modeling system that relies on a series of macros and classes to simplify encoding of the modeling protocol, including designing the system representation, specifying scoring function, sampling alternative structures, analyzing the results, facilitating the creation of publication-ready figures, and depositing into PDB-Dev (see below). PMI exchanges the high flexibility of IMP for ease-of-use, all within one short Python script (<100 lines). Despite its simplicity in creating standard modeling workflows, PMI is powerful and extensible - it is built on IMP and creates native IMP objects, which means that the advanced user can customize many aspects of the modeling protocol. Below, we outline each stage of the modeling process as performed in PMI. [Fig. 2]

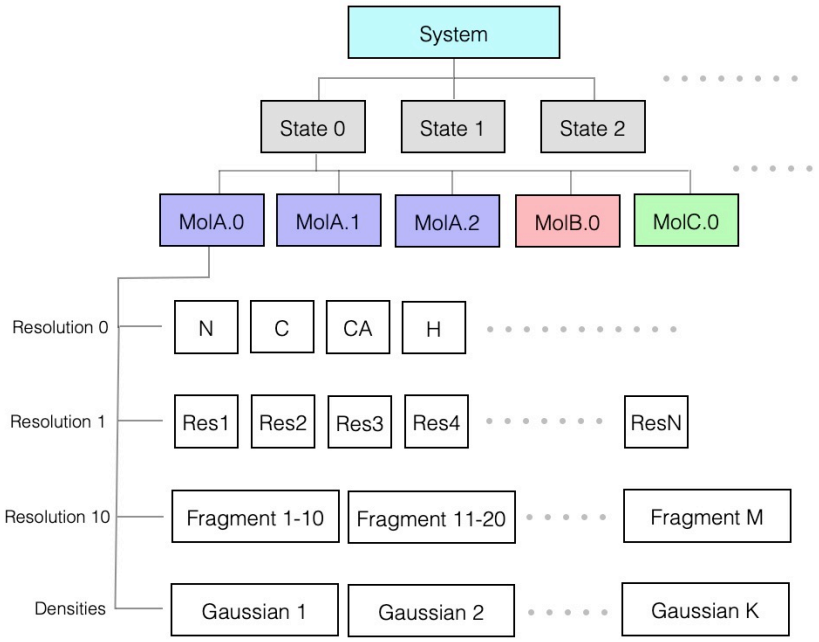

Figure 2: PMI Hierarchy. PMI is constructed as a top-down hierarchy beginning with a System. A system can contain one or more states with each state being a different conformation, composition or time-ordered step in the system. Each state is comprised of one or more molecules that may have one or more copies per molecule. At the final level, each molecule is represented at one or more resolutions.

2.1. Gathering Information

Information about a system that we wish to model includes everything that we directly observe, can infer through comparison to other systems, and fundamental physical principles. Experimental data that are commonly utilized in integrative modeling include X-ray crystal structures, EM density maps, NMR data, chemical crosslinks, yeast two-hybrid data, and Förster resonance energy transfer (FRET) measurements. Atomic resolution information may be applied directly as structural restraints from atomic statistical potentials [20, 21] and molecular mechanics force fields [22, 23] or derived from comparative modeling programs such as MODELLER [24] and PHYRE2. [25]

Each piece of information can be utilized within the modeling procedure in one or more of five distinct ways: defining model **representation**, defining **sampling** space/degrees of freedom, **scoring** models during sampling, **filtering** models post-sampling, and **validating** completed models.

2.2. System and data representation

The representation of the system defines the structural degrees of freedom that will be sampled and is designed based on the information at hand. We can utilize a multi-scale representation, where model components can be modeled at one or more different resolutions commensurate with the information content at that site [Fig 3]. For example, a domain described by a crystal structure can be represented at atomic resolution and a disordered segment can be represented as a string of spherical beads of 10 residues each. In addition, non-particle based representations, such as Gaussian mixture models (GMMs), can also be used; for example, in EM density fitting. [26, 27] Choosing a representation reflects a compromise between the need

for details required by the biological application of the model and the need for coarseness required by limited computing power.

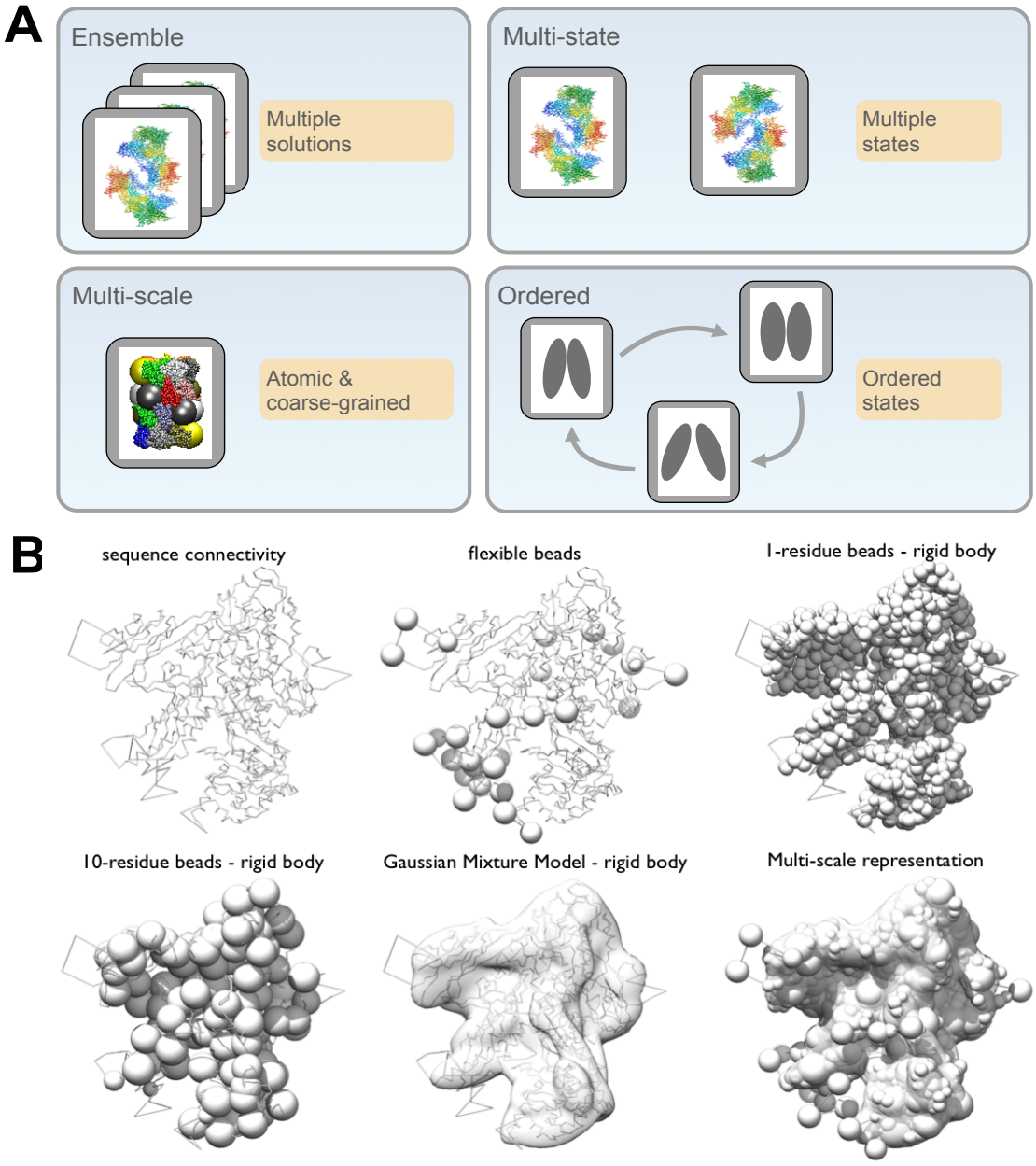

Figure 3: Ways of representing a single biomolecule. A: The complexity of a molecular system can be represented in four ways. An ensemble of states describes the structural heterogeneity around a single solution. Multiple states are used to describe systems that exist in multiple thermodynamic wells. The system can be modeled at a multitude of scales commensurate with the different types of information known about it. Finally, individual states can be time-ordered, allowing for the modeling of the transition rates between them. B: Multiple representations can be simultaneously applied to the same biomolecule so that information of various types can be applied at the proper scale and form. The molecule is first defined by its sequence connectivity. Flexible beads comprising one or more residues are commonly applied to loops where no high-resolution structure is available. Areas that have high resolution structure can be modeled by spherical beads of 1 residue for the evaluation of residue-specific information such as chemical crosslinks or NMR distance restraints. 10-residue beads are generally used to model lower resolution information such as SAXS data and the excluded volume restraint. The molecule can be represented as a Gaussian mixture model for comparisons to EM densities. IMP and PMI can utilize all of these representations simultaneously in a multi-scale model. Panel A adapted from [34].

Some of the input information is translated into restraints on the structure of the model. These spatial restraints are combined into a single scoring function that ranks alternative model configurations (models) based on their agreement with the information. The scoring function defines a multi-dimensional landscape spanned by the model degrees of freedom; the good-scoring models on this landscape satisfy the input restraints.

2.3. Sampling

In most cases, all possible models cannot be generated. Thus, we utilize sampling methods to search for models that agree with the input data according to the scoring function defined above (good-scoring models). One approach for sampling models in IMP is a Monte Carlo algorithm, [28] guided by our scoring function and accelerated *via* replica exchange. [29] Other sampling methods can be utilized for specific cases. [see Note 1]

2.4. Analysis

The results of stochastic sampling (i.e., an ensemble of output structures and their respective scores) must be analyzed to estimate the sampling precision and accuracy, detect inconsistencies with respect to the input information, and suggest future experiments. [Fig. 4]

We wish to analyze only models that are sufficiently consistent with the input information (good-scoring models). A good-scoring model must sufficiently satisfy every single piece of information used to compute it; therefore one needs a threshold for every data point or set of data. Sampling may produce zero such models, which can result from inconsistent data or an unconsidered multiplicity of conformational states. [see Note 2]

Given a set of good-scoring models, we must first estimate the precision at which sampling found these most good-scoring solutions (sampling precision). [Fig. 4, step 1] [16, 17, 30] This estimate relies on splitting the set of good-scoring models into two independent samples, followed by comparing them to each other using four independent tests: 1) convergence of the model score, 2) whether model scores for the two samples were drawn from the same parent distribution, 3) whether each structural cluster includes models from each sample proportionally to its size 4) sufficient similarity between the localization densities [see Note 3] for the entire system, from each sample. After threshold clustering of models, the sampling precision is defined as the largest RMSD value between a pair of structures within any cluster, in the finest clustering for which the structures from the two independent runs contribute proportionally to their size. [Fig. 6D] In other words, the sampling precision is defined as the precision at which the two independent samples are statistically indistinguishable. The individual clusters for each sample are also compared visually (Section 4.5.3) to confirm similarity.

At this step, the model precision (uncertainty), which is represented by the variability among the good-scoring models, is also reported. This uncertainty can be quantified by measures such as root-mean-square deviation (RMSD) of model components for models within each cluster or between clusters determined above. The lower bound on model precision is provided by the sampling precision; the model precision cannot be higher than the sampling precision.

An accurate model must satisfy all information about the system, and this is evaluated in a number of steps. First, the consistency of the model with input information is assessed by independently assessing the clusters determined above against the input data [Fig. 4, Step 2]. In the next step, the models are assessed by random or systematic cross-validation [Fig. 4, Step 3]. The next and most robust validation is the consistency of the model with data not used to compute it [Fig. 4, Step 4], similar to a crystallographic R_{free}.

A final validation is the presence of features in the model that are unlikely to occur by chance and/or are consistent with the biological context of the system [Fig. 4, step 5]. For example, a 16 fold symmetry was found in the model of the Nuclear Pore Complex when only 8-fold symmetry had been enforced [31] and the displacement of the aspartate sensor domain in a two state model of the histidine kinase PhoQ transmembrane signaling agreed with previous analysis. [18]

A key feature of the four-step procedure for integrative modeling [Fig. 1] is that it is iterative. Assessment may reveal a need to collect more input data, or suggest future experiments, both by the researchers that constructed the initial model and by others.

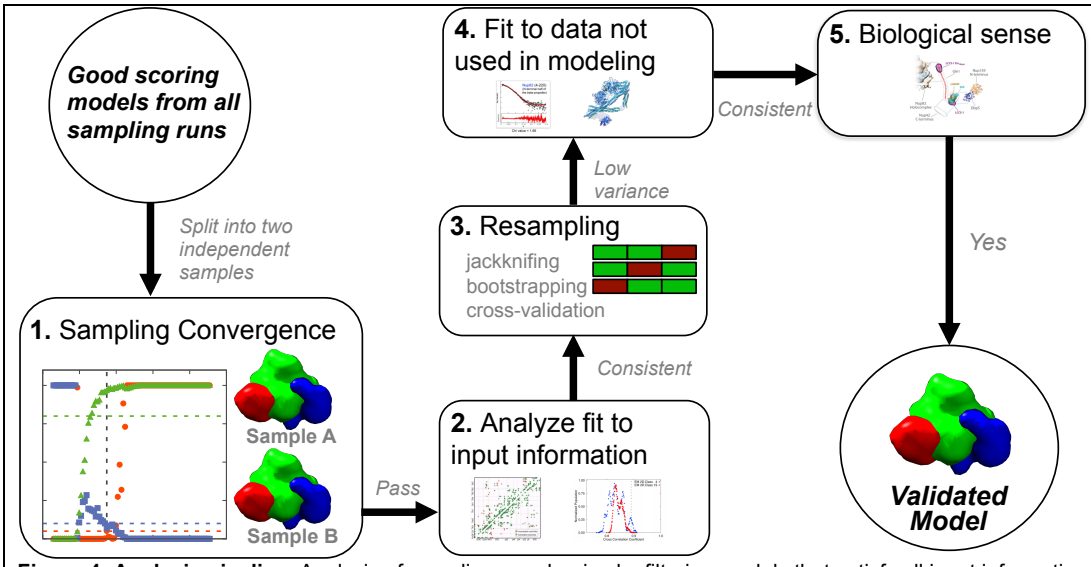

Figure 4: Analysis pipeline. Analysis of sampling runs begins by filtering models that satisfy all input information. In step one, this set is split into two independent samples to assess the precision at which sampling is converged. If sampling has converged at a high enough precision, the resulting models can be assessed against the input information to identify potential multiple states. Resampling can be performed by either systematically or randomly excluding data sets and rerunning the simulation and sampling convergence algorithms. The models can then be assessed against data that was not used in modeling. Finally, the models are assessed for logical sense in answering the original biological question.

2.5. Deposition

For the models, data, and modeling protocols to be generally useful, they must be reproducible and available to everyone in a publicly accessible database. This availability allows any scientist to use a deposited model to plan experiments by

simulating potential benefits gained from new data. Computational groups can more easily experiment with new scoring, sampling, and analysis methods, without having to reimplement the existing methods from scratch. Finally, the authors themselves will maximize the impact of their work, increasing the odds that their results are incorporated into future modeling. Following the recommendations of the wwPDB Hybrid/Integrative Methods Task Force in 2015, [32] a prototype archive, PDB-Development (PDB-Dev, https://pdb-dev.wwpdb.org/) [33, 34] was recently established to store integrative models and corresponding data. The mmCIF file format used to archive regular atomic PDB structures was extended to support the description of integrative models, including information on the input data used, the modeling protocol, and the final output models. As of July 2018, PDB-Dev contains 14 depositions, including 9 generated by IMP.

3. Materials

3.1. IMP

IMP binaries for most platforms can be downloaded and installed from: https://integrativemodeling.org/download.html.
The tutorial has been built to work with the latest stable release of IMP at time of writing, 2.9.0.

3.2. Chimera

Modeling results can be visualized using Chimera version 1.13 or later, which can be downloaded from https://www.cgl.ucsf.edu/chimera/download.

3.3. Actin tutorial code and data

The data and code used in the tutorial below can be downloaded from https://github.com/salilab/actin_tutorial. The home directory of the repository, `actin_tutorial`, will be used to reference all other paths in the tutorial below. Analysis scripts are located in `./analysis/scripts`. These are slightly modified from the stand-alone script library for performing sampling exhaustiveness found at https://github.com/salilab/IMP-sampcon. These analyses rely on pyRMSD. [35]

3.4. Computer skills requirements

PMI stands for Python Modeling Interface. Interaction with PMI requires Python scripts. The tutorial scripts for PMI are written to be interpretable by even those with minimal or no Python experience. However, performing advanced tasks and/or designing novel workflows benefits from a working knowledge of Python.

3.5. Computational resources and time

The full tutorial simulation can be run in a few hours on a modern desktop or laptop computer. A multi-core system is preferred to utilize replica exchange.

4. Integrative modeling of ADP-actin, gelsolin and C-terminal actin-binding domain of tropomodulin

Here, we demonstrate integrative modeling using the PMI interface by modeling the complex of actin and tropomodulin-gelsolin chimera using SAXS, EM, crosslinking, crystal structures of the individual domains, and physical principles. This complex was solved via X-ray crystallography at 2.3 Å resolution (PDB: 4PKI). [36] We use this structure to simulate biophysical data and assess the accuracy of the modeled

complexes. In this simple exercise, we assume that we have a crystal structure of only the actin-gelsolin interface and would like to find the tropomyosin-actin binding interface. The entire modeling protocol is summarized in the four-stage diagram [Fig. 1].

4.1. Gathering and preparing information

All data is contained in subfolders of the `./data` directory of the tutorial.

4.1.1. Structural data from the PDB

The crystal structure 4PKI is used to set the atomic coordinates for each of the domains in the FASTA sequence that determines the composition of each biomolecule, as well as the coordinates for tropomyosin and the actin-gelsolin complex. [Fig. 5]

4.1.2. Chemical Crosslinks

Thirty-three simulated crosslinks were generated from a random subset of lysine residue pairs whose CA-CA distances are under 25 Å.

4.1.3. Electron Microscopy

A simulated EM density of the entire complex was created at 20 Å resolution using IMP [see Note 4]. The simulated map is approximated as a Gaussian Mixture Model (GMM). [27]

4.1.4. SAXS

A simulated SAXS profile of the entire 4pki.pdb complex was created using FoXS. [37]

4.1.5. Other information

We also define restraints such as excluded volume and sequence connectivity to add chemical and physical knowledge to the modeling protocol.

4.2. Defining System Representation and Degrees of Freedom in the Topology File

The model representation (e.g., bead size and rigid bodies) can be set within the topology file. The topology file is a pipe-delimited format with each line specifying a separate domain and keyword values determining how the domain is represented. A definition of each keyword is given in Table 1.

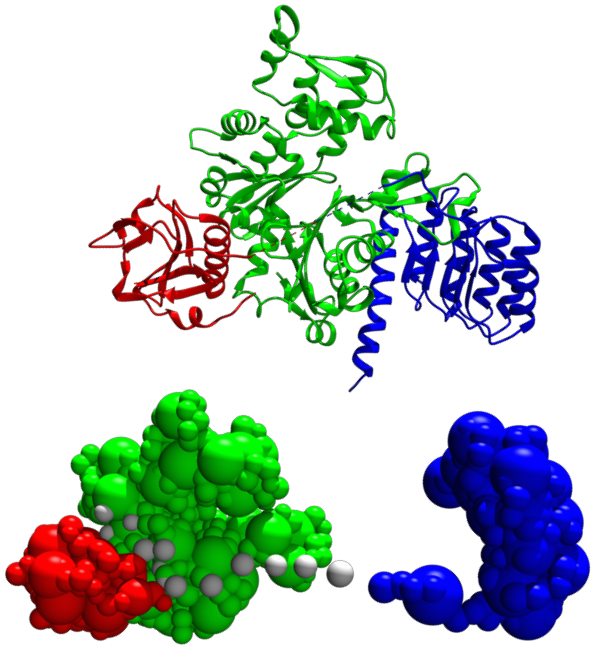

Figure 5: Actin-gelsolin-tropomyosin complex. Top: Reference crystal structure 4PKI showing actin in green, gelsolin in red and tropomyosin in blue. Bottom: Multi-scale representation and position of the system after domain shuffling and bead relaxation. Structured domains are represented by spherical beads of 1 and 10 residues. Unstructured residues from the linker between the gelsolin and tropomyosin domains are represented as gray beads.

The topology file for this tutorial, shown below, is found at `./modeling/topology.txt`. Here, the system is subdivided into four distinct domains: one each for the three structured domains (actin, gelsolin, and tropomyosin) and one consisting of the 18-residue engineered linker between gelsolin and tropomyosin. The first domain, the entire actin molecule, is colored green and contains the entirety of `chain` A from 4pki.pdb. A `bead_size` of 1 residue per bead is assigned to any unmodeled section (i.e., not present in the PDB file) [see Note 5]. A GMM is approximated using 10 residues per Gaussian. This domain is assigned to `rigid_body` 1. The second domain, the gelsolin portion of the chimera, is constructed by selecting the `residue_range` 52-177 of `chain` G. These residues, however, are numbered 1-126 in the FASTA file, therefore a `pdb_offset` of -51 must be added. This domain is also assigned to `rigid_body` 1 to preserve the actin/gelsolin interface. The third domain is the linker, whose residues have no structure associated with them; thus, they are given a `pdb_fn` of BEADS with a `bead_size` of 1 [see Note 6]. The final domain, tropomyosin, is built similarly to gelsolin and assigned to `rigid_body` 2, since we would like to sample its position separate of the rest of the complex.

```
|molecule_name | color | fasta_fn | fasta_id | pdb_fn | chain | residue_range | pdb_offset |
bead_size | em_residues_per_gaussian | rigid_body | super_rigid_body |
chain_of_super_rigid_bodies |
|actin    |green |4pki.fasta.txt|actin               |4pki.pdb|A|1,END    |0    |1|10|1|1||
|geltrop |red    |4pki.fasta.txt|gelsolin-tropomyosin|4pki.pdb|G|52,177   |-51  |1|10|1|1||
|geltrop |gray   |4pki.fasta.txt|gelsolin-tropomyosin|BEADS   |G|178,195  |-51  |1|10|1|1||
|geltrop |blue   |4pki.fasta.txt|gelsolin-tropomyosin|4pki.pdb|G|1170,1349|-1025|1|10|2|1||
```

This topology file also places all domains in a single `super_rigid_body`. This definition allows the entire complex to move as a single unit, which is useful for fitting to the EM map.

`molecule_name`	Name of the molecule that this domain is a part of
`color`	The color used in the output RMF file for this component. Uses Chimera defined names [see Note 23] or RGB values (e.g. 155,35,0)
`fasta_fn`	Name of FASTA file containing this component.
`fasta_id`	String found in FASTA sequence header line.
`pdb_fn`	Name of PDB file with coordinates (if available). If left empty, will set up as `BEADS`. Using `IDEAL_HELIX` will build a helix. [see Note 25]
`chain`	Chain ID of this domain in the PDB file.
`residue_range`	Comma delimited pair defining range of residues. Can leave empty or put `all` to use entire sequence from FASTA file.
`pdb_offset`	Offset to sync PDB residue numbering with FASTA numbering.
`bead_size`	The size (in residues) of beads used to model areas not covered by PDB coordinates.
`em_residues_per_gaussian`	The number of residues per Gaussian used to model the electron density of this domain. Set this to zero if no EM fitting will be done.
`rigid_body`	The ID number of the rigid body that contains this component.
`super_rigid_body`	The ID number(s) of the super rigid body(ies) containing this component.
`chain_of_super_rigid_bodies`	Automatically group overlapping segments of beads into super rigid bodies. The number here, as for `rigid_body`, specifies the member of the chain to which this domain belongs.

Table 1: Topology file keywords and descriptions

4.3. Constructing the modeling script

The modeling script contains the entire workflow from defining the system representation through execution of sampling. The system representation and sampling degrees of freedom can be built manually [see Note 7] or, as here, read from a topology file. Restraints are added, and the sampling protocol defined and executed.

4.3.1. Importing and building system representation

First, we create an IMP `Model` object, which stores all components of the model. Second, we create a `BuildSystem` object and define the `resolutions` at which residues in the structured sections will be modeled. Here, we set resolutions of 1 and 10 residues per bead so that crosslinking restraints can be evaluated at residue resolution and the expensive excluded volume restraint (below) can be evaluated at the lower resolution. Third, the topology file is read using a `TopologyReader` object, followed by generating a useful list of component `molecules`. To this `BuildSystem` object, we add a state corresponding to the representation defined in the topology file using `bs.add_state()`. [see Note 8]

```
mdl = IMP.Model()
bs = IMP.pmi.macros.BuildSystem(mdl, resolutions=[1,10])
t = IMP.pmi.topology.TopologyReader(topology.txt)
molecules = t.get_components()
bs.add_state(t)
```

We then execute the macro, which returns the `root_hier` root hierarchy and `dof` degrees of freedom objects, which will be used later. Within the macro, we set the movement parameters of individual beads and rigid bodies. Translations (`trans`) are defined in angstroms and rotations (`rot`) in radians.

```
root_hier, dof = bs.execute_macro(max_rb_trans=1.0,
                                  max_rb_rot=0.5,
                                  max_bead_trans=2.0,
                                  max_srb_trans=1.0,
                                  max_srb_rot=0.5)
```

4.3.2. Adding restraints to the model

PMI contains simple interfaces for a number of IMP restraints that model various types of chemical and physical data and knowledge. All of these restraints produce output, which we will collect in an `output_objects` list. Each restraint also needs to be explicitly added to the scoring function for sampling, using the `add_to_model()` command. We will add the restraints to the scoring function in a specific order, discussed below.

First, we define the restraints that enforce physical and chemical principles. [see Note 9] The `ConnectivityRestraint` adds a bond between each pair of consecutive residues in each molecule. The `ExcludedVolumeSphere` restraint is applied to the entire system and enforced at the lowest resolution possible (indicated by resolution=1000), because this restraint is costly to evaluate.

```
output_objects=[]

for m in molecules:
    cr = IMP.pmi.restraints.stereochemistry.ConnectivityRestraint(m)
    cr.add_to_model()
    output_objects.append(cr)

evr = IMP.pmi.restraints.stereochemistry.ExcludedVolumeSphere(
                                      included_objects=[root_hier],
                                      resolution=1000)
```

Second, we build a `SAXSRestraint` based on the comparison of SAXS data to the model. Since our model is calculated at residue resolution, we calculate the SAXS profile using residue form factors. For residue-based calculations, we compare curves out to a q of 0.15 [see Note 10].

```
sr = IMP.pmi.restraints.saxs.SAXSRestraint(input_objects=[root_hier],
                saxs_datafile=saxs_data,
                weight=0.01,
                ff_type=IMP.saxs.RESIDUES,
                maxq=0.15)
```

To set up a crosslinking restraint, we first build a PMI `CrossLinkDataBase` that uses a `CrossLinkDataBaseKeywordsConverter` to interpret a crosslink data file. At a minimum, the crosslink data file needs four columns labeled with a key: one for each protein name and one for each residue number of the crosslink. The standard keys are `Protein1`, `Residue1`, `Protein2`, `Residue2`. [see Note 11].

```
xl_data = "./derived_data/xl/derived_xls.dat

xldbkc = IMP.pmi.io.crosslink.CrossLinkDataBaseKeywordsConverter()
xldbkc.set_standard_keys()
xldb = IMP.pmi.io.crosslink.CrossLinkDataBase()

xldb.create_set_from_file(file_name=xl_data,
                          converter=xldbkc)
```

Using this database, we can construct the crosslinking restraint. We input the root hierarchy of the system and the database, and specify the length of the crosslinker. The restraint can be evaluated at any resolution, though is generally most informative at resolution = 1. The length determines the inflection point of the scoring function sigmoid [18] and is generally set to 10 Å + the crosslinker length for Lys-Lys crosslinkers.

```
xlr = IMP.pmi.restraints.crosslinking.CrossLinkingMassSpectrometryRestraint(
                root_hier=root_hier,    # Must pass the system root hierarchy
                CrossLinkDataBase=xldb, # The crosslink database.
                length=25,              # The crosslinker plus side chain length
                resolution=1,           # The resolution to evaluate the crosslink
                slope=0.0001,           # This adds a linear term to the score
                                        #   to bias crosslinks towards each other
                weight=10)           # Scaling factor for the restraint score.

output_objects.append(xlr)
```

The EM restraint is determined by calculating the overlap (cross-correlation) between the system GMM density particles and the map GMM particles. First, we must collect the density particles using an IMP `Selection`. We then invoke the restraint using these particles and the gmm file generated from the EM map.

```
densities = IMP.atom.Selection(root_hier,
                     representation_type=IMP.atom.DENSITIES).get_selected_particles()

em_map = "./derived_data/em/4pki_20a_50.gmm"

emr = IMP.pmi.restraints.em.GaussianEMRestraint(
        densities,         # Evaluate the restraint using these model densities
        target_fn=em_map,  # The EM map approximated as a Gaussian mixture model (GMM)
        slope=0.00000001,  # a small force to pull objects towards the EM map
        scale_target_to_mass=True, # Normalizes the mass of the model wrs: EM map
        weight=100)          # the scaling factor for the EM score

output_objects.append(emr)
```

4.3.3. Defining the sampling protocol

Sampling begins by randomizing the coordinates of the starting particles using `shuffle_configuration` [see Note 13]. Because this randomization generally places beads of neighboring residues far apart, we first optimize the positions of these flexible beads using steepest descent minimization for 500 steps based on only the connectivity restraint. We then add the balance of the scoring function terms to the model prior to the main sampling step.

```
IMP.pmi.tools.shuffle_configuration(root_hier,
                                    max_translation=50)
dof.optimize_flexible_beads(500)

evr.add_to_model()
emr.add_to_model()
xlr.add_to_model()
sr.add_to_model()
```

We implement a Monte Carlo sampling scheme with replica exchange using the PMI `ReplicaExchange0` macro. Within this macro, we set the directory where all output files will be placed, `global_output_directory`, and the `number_of_frames` to generate. The final line of the script executes the sampling macro.

```
rex=IMP.pmi.macros.ReplicaExchange0(mdl,
        root_hier=root_hier,           # the system root hierarchy
        crosslink_restraints= [xlr],   # This allows viewing of crosslinks in Chimera
        monte_carlo_sample_objects=dof.get_movers(), # all objects to be moved
        global_output_directory='run1/' # Set the output directory for this run.
        output_objects=output_objects,    # Write these items to the stat file
        monte_carlo_steps=10,             # Number of MC steps between writing frames
        number_of_best_scoring_models=0,  # set >0 to store best scoring PDB files
        number_of_frames=10000)           # Total number of frames to generate

rex.execute_macro()
```

4.4. Running the modeling script

Modeling analysis requires at least two independent sampling runs be performed. For each run, in `modeling.py` the `global_output_directory` keyword can be set to `run1`, `run2`, ..., `runX`.

The modeling script can be run on a single processor using the following command:
`python ../modeling.py`

or in parallel using N processors using:
`mpirun -np N python ../modeling.py`

A parallel invocation of IMP will run replica exchange with `N` replicas. A serial run will run a basic Monte Carlo protocol with one replica.

Raw output will be written to the `./runX/output` folder, as specified in the replica exchange macro. Within this folder, stat files contain tabulated statistics for each frame. In the `rmf` directory, model coordinates for the lowest temperature replica are stored. These can be opened directly in Chimera and the “trajectories” observed.

4.5. Analysis

Analysis is performed using scripts located in `./analysis/scripts/`. The already-generated sampling output will be analyzed here; it is contained in the folders `./modeling/run1` and `./modeling/run2`.

Analysis is performed in a new directory: `./analysis/tutorial_analysis/`.

4.5.1. Filtering good scoring models

The `select_good_scoring_models.py` script filters models based on score and parameter thresholds. In this script, required flags are: `-rd`, which specifies the directory containing sampling output folders; `-rp`, which defines the prefix for the sampling output folders; `-sl`, which defines the stat file keywords [see Note 13] that we wish to filter on; `-pl`, which specifies the keywords that will be written to the output file; `-alt` and `-aut`, which specify, respectively, the lower and upper threshold for each keyword in `-sl` that define acceptance. The `-mlt`

and `-mut` keywords, which are optional, define thresholds for restraints made of multiple components (such as crosslinks).

Here, we first use crosslink satisfaction as an initial filtering criterion because we usually have an *a priori* estimate of the false positive rate and/or cutoff distance [see Note 14]. For this simulated system, we only accept models with 100% satisfaction of crosslinks by setting both `-alt` and `-aut` to 1.0. A crosslink is satisfied if the distance is between 0.0 and 30.0 Å, as delineated by the `-mlt` and `-mut` keywords, respectively. We specify that connectivity, crosslink data score, excluded volume, EM, SAXS and total scores be printed as well.

```
python ../scripts/select_good_scoring_models.py -rd ../../modeling -rp run -sl
"CrossLinkingMassSpectrometryRestraint_Distance_" -pl
ConnectivityRestraint_None CrossLinkingMassSpectrometryRestraint_Data_Score
ExcludedVolumeSphere_None GaussianEMRestraint_None SAXSRestraint_Score
Total_Score -alt 1.0 -aut 1.0 -mlt 0.0  -mut 30.0
```

This script creates a directory `./filter/` and a file, `./filter/models_scores_ids.txt`, that contains the model index, its run, replica ID, frame ID, scores, and sample ID for each model. We can now use the script `plot_score.py` to plot the distribution of SAXS, EM, connectivity and excluded volume scores from this first set of filtered models to determine a reasonable threshold for accepting or rejecting a model.

```
python ../scripts/plot_score.py ./filter/model_ids_scores.txt
SAXSRestraint_Score

python ../scripts/plot_score.py ./filter/model_ids_scores.txt
GaussianEMRestraint_None
```

The resulting histograms (`SAXSRestraint_score.jpg and GaussianEMRestraint_None.jpg`) are roughly Gaussian, Based on these distributions we set our criteria for good scoring models as those whose EM and SAXS scores are >1 standard deviation below the mean, except for connectivity, which is well satisfied in almost all models and EM, which has a large tail. [see Note 15] Our high score thresholds are 2.0 for EM, and 4.554 for SAXS, 1.0 for connectivity and 4.916 for excluded volume.

We rerun `select_good_scoring_models.py` adding the extra keywords and score thresholds. We add the extra flag, `-e`, to extract Rich Molecular Format (RMF) files of all good scoring models. These thresholds return 1618 good scoring models. [see Note 16]

```
python ../scripts/select_good_scoring_models.py -rd ../../modeling -rp run -sl
"CrossLinkingMassSpectrometryRestraint_Distance_" GaussianEMRestraint_None
SAXSRestraint_Score ConnectivityRestraint_None ExcludedVolumeSphere_None -pl
ConnectivityRestraint_None CrossLinkingMassSpectrometryRestraint_Data_Score
ExcludedVolumeSphere_None Total_Score -alt 1.0 -50 -50.0 0.0 0.0 -aut 1.0 2.0
4.554 1.0 4.916 -mlt 0.0 0.0 0.0 0.0 0.0 -mut 30.0 0.0 0.0 0.0 0.0 -e
```

The output directory, `good_scoring_models`, contains folders `sample_A` and `sample_B`, which hold the RMF files of the good scoring models for each independent run (or set of runs). The file `model_ids_scores.txt` contains the model index, its run, replica ID, frame ID, scores, and sample ID for each model.

4.5.2. Determining sampling precision, clustering, and computing localization densities

The `Master_Sampling_Exhaustiveness_Analysis.py` script is used to calculate the sampling precision of the modeling. During this step, multiple tests for convergence are performed on the two samples determined in step 4.5.1, models are clustered, and localization densities are computed.

First, we create a file, `density_ranges.txt`, in the `tutorial_analysis/` directory with a single line that defines components using PMI selection tuples on which we calculate localization densities. [see Note 17] Here, we create three localization densities, one for the entire actin molecule and one each for the structured residues of each of the other two molecules.

```
density_custom_ranges={"Actin":['A'],"Gelsolin":[(1,126,'G')],"Tropomysin":[(145,324,'G')]}
```

We now run the script for testing sampling exhaustiveness.

```
python ../scripts/Master_Sampling_Exhaustiveness_Analysis.py -n actin -p
good_scoring_models/ -d density_ranges.txt -m cpu_omp -c 8 -a -g 0.1
```

e

system name, `actin`, defines the labels for the output files. The `-a` flag aligns all models [see Note 18] and `-g` determines the step size in Å for calculating sampling precision. [see Note 19] This routine can be run in parallel using the `-m cpu_omp` flag [see Note 20] and `-c N`, where `N` is the number of processors. The `-p` flag defines the path to the good scoring model directory.

The results of the convergence tests are summarized in the output figure [Fig. 6] `actin_convergence.png`, which identifies our sampling precision of 3.5 Å, with one dominant cluster, one minor cluster and one cluster of insignificant size. Text files containing this information are also produced. [see Note 21] Output also includes localization densities for each cluster, which are contained in separate directories (`cluster.0`, `cluster.1`, …). Within these directories are a representative RMF file `cluster_center_model.rmf3` and localization densities for each subunit defined in the `density_ranges.txt` file. [see Note 22]

4.5.3. Visualizing models

The cluster RMF files and localization densities can be visualized using UCSF Chimera version >= 1.13. Example scripts for visualizing all localization densities are provided in `./analysis/scripts/chimera_scripts`.

At this point, one must decide if the models are helpful in answering our biological questions. In the case of this tutorial, the PPI is localized to within a few Å and we can make predictions as to what residues may be important for this interaction. If our models are not well enough resolved, more information may have to be added through additional experiments, addition of constraints to the sampling, change in system representation, and/or additional sampling. We can iterate this process until we are satisfied with our output models.

4.5.4. Additional model validation

Additional validation of the final model ensemble can be performed by rerunning the above protocol while omitting one or more of the input data points. Ideally, models generated with only a subset of the data will not differ significantly from the original models. Further, any information not used in the modeling process can be used as a validation of the final model ensemble (section 2.4).

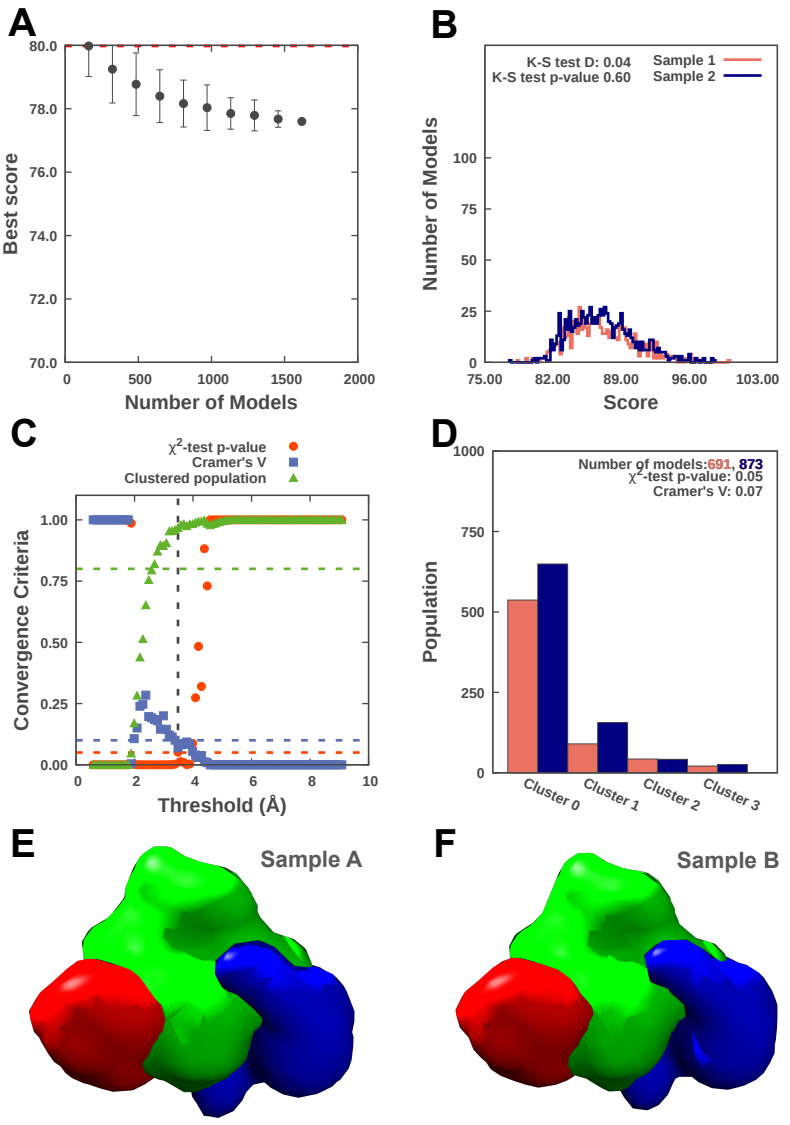

Figure 6. Results for sampling exhaustiveness protocol for modeling in complex of actin and tropomodulin-gelsolin chimera. **A.** Results of test 1, convergence of the model score, for the 1618 good-scoring models; the scores do not continue to improve as more models are computed essentially independently. The error bar represents the standard deviations of the best scores, estimated by repeating sampling of models 10 times. The red dotted line indicates a lower bound reference on the total score. **B.** Results of test 2, testing similarity of model score distributions between samples 1 (red) and 2 (blue); the difference in distribution of scores is significant (Kolmogorov-Smirnov two-sample test p-value less than 0.05) but the magnitude of the difference is small (the Kolmogorov-Smirnov two-sample test statistic *D* is 0.02); thus, the two score distributions are effectively equal. **C.** Results of test 3, three criteria for determining the sampling precision (Y-axis), evaluated as a function of the RMSD clustering threshold (X-axis). First, the p-value is computed using the χ^2-test for homogeneity of proportions (red dots). Second, an effect size for the χ^2-test is quantified by the Cramer's *V* value (blue squares). Third, the population of models in sufficiently large clusters (containing at least 10 models from each sample) is shown as green triangles. The vertical dotted grey line indicates the RMSD clustering threshold at which three conditions are satisfied (p-value > 0.05 [dotted red line], Cramer's V < 0.10 [dotted blue line], and the population of clustered models > 0.80 [dotted green line]), thus defining the sampling precision of 3.5 Å. **D.** Populations of sample 1 and 2 models in the clusters obtained by threshold-based clustering using the RMSD threshold of 3.5 Å. Cluster precision is shown for each cluster. **E.** and **F.** Results of test 4: comparison of localization probability densities of models from sample A and sample B for the major cluster (84% population). The cross-correlation of the density maps of the two samples is 0.99 for the gelsolin (red) and tropomysin (blue) maps and 0.97 for the actin map (green).

4.6. Storing and reporting results in the wwPDB

For our modeling to be reproducible - a key requirement for the 4-stage modeling procedure [Fig. 1] and for science in general - the modeling protocol, all of the input data we used, and the final output models, should be deposited in a public location, ideally the nascent PDB-Dev repository (https://pdb-dev.wwpdb.org/).

4.6.1. Modeling protocol

The modeling protocol includes the entire procedure of converting raw input data to output models, and so comprises both the set of IMP Python scripts described above and any procedures used to prepare IMP inputs, such as comparative modeling of subunits, segmentation of an EM density, and processing of XL-MS data to get a set of proximate residues. An excellent way to store and disseminate such a protocol is by using a source control system with a publicly accessible web frontend, such as GitHub (as is used for this tutorial). Integrative modeling is an inherently collaborative process. Source control makes it straightforward to track changes to all of the protocol scripts and data by local and remote collaborators. All protocol files should be deposited in a permanent location with a fixed Digital Object Identifier (DOI). A number of free services are available for deposition of such files, such as Zenodo (https://zenodo.org) and FigShare (https://figshare.com), where a snapshot of a GitHub repository for the published work can be deposited. For an example, see ref. [38]

4.6.2. Input data

Each piece of input data used should also be publicly available. Where possible, this data should be deposited in a repository specific to the given experimental technique and referenced from the model mmCIF file. For example, all of the crystal structures used in this example are simply referenced by their PDB IDs. Where such a repository does not exist, the data files should be made available at a DOI. The simplest way to archive these files is to store them in the same GitHub repository used for the modeling protocol. If derived data are used, the modeling protocol should indicate where the original raw data came from.

4.6.3. Output models

A decision needs to be made about which models to deposit. Generally, a representative sample of each cluster should be deposited, together with the localization densities of the entire cluster.

The mmCIF file format allows for multiple models, potentially at multiple scales, in multiple states, and/or different time points, to be stored in a single file together with pointers to the input data and modeling protocol. Implementation of this format in IMP is still under development. The functionality will extract information from the RMF files output by the IMP modeling and combine it with metadata extracted from each experimental input. This file can be visualized in UCSF ChimeraX, [39] and similar files from real modeling runs can be deposited in PDB-Dev and cited in publications.

6. Notes

1. Other sampling methods include Rapidly Exploring Random Trees (RRT) for searching dihedral space, [40] divide-and-conquer message passing methods [41] for large discrete spaces, conjugate gradients and molecular dynamics.

2. In this case, the user may reformulate the representation by adding a state to the system. (Note 8)

3. In general, an ensemble of models can be visualized as a localization probability density map (localization density). The map specifies the probability of any volume element being occupied by a given bead in superposed good scoring models.

4. Simulated EM maps can be created map can be created in IMP using the following command: `simulate_density_from_pdb <file.pdb> <output.mrc> <resolution> <a/pixel>`

5. Spherical beads are applied to every 10 residues with smaller beads applied to loops of smaller length.

6. These residues are also assigned to rigid_body 1 to improve sampling. All beads within rigid bodies are, by default, allowed to be flexible.

7. The file `./modeling/modeling_manual.py` contains this exact system built manually using PMI commands instead of a topology file. PMI commands allow significantly more flexibility in model design.

8. To add a second state with the same topology, this line can be repeated, or to use a different topology, `bs.add_state(t2)` can be invoked with a different topology file.

9. For coarse-grained models, a molecular mechanics force field is not applicable. The CHARMM force field can be applied to enforce stereochemistry on atomic models, however. See the examples in the `IMP.atom` module to learn how to implement this restraint.

10. Model SAXS profiles can be computed using residues, CA atoms, heavy atoms or all atoms, depending on the resolution of the model. The recommended `maxq` values are dependent on this choice. At residue resolution, the fit is only valid up until q ~ 0.15; for heavy atoms q = 0.4; and for all atoms, the fit is valid out to q = 1.0 (the maximum value).

11. See `derived_xls.dat` and the `modeling.py` script for a more in-depth explanation of crosslink keys.

12. The shuffle algorithm fails if it cannot find a configuration without any overlap between components. If this happens, try increasing the `max_translation` parameter. Don't set this too high as you'll spend way too much time getting your system to move back together.

13. A list of acceptable keywords can be determined by running `../scripts/plot_stat.py ./path/to/stat/file -pk`.

14. For scores whose thresholds are not known *a priori*, one can perform a multi-stage filtering process as outlined in the above protocol.

15. Currently, the choice of filtering criteria is very subjective. Ideally, a fully Bayesian framework will allow for objective weighting of different restraints and allow for filtering at single likelihood. Until then, the choice of a score or parameter that represents a "good scoring model" should be carefully thought out by the modeler and reported in the text.

16. In general, we require at least 1000 or more models for assessing sampling exhaustiveness. Our score thresholds were chosen in order to have a reasonable number (1000 - 20000) models for analysis. If we have too few models, the satisfaction criteria should be relaxed, or more sampling should be performed to find more satisfactory models. Too many models (>20,000) will make subsequent processing more computationally intensive; in this case satisfaction criteria can be made stricter, or one can pass a random subset of these models to the sampling convergence protocol.

17. An explanation of the PMI selection format can be found at https://github.com/salilab/pmi/wiki/PMI-Tuple-Selection-Format

18. One can choose whether to align models (`-a` option) or not. Alignment of models is sometimes not necessary, e.g. when one has a medium resolution or better EM map.

19. For calculating sampling precision, the grid size is the step size at which clustering is performed between the minimum and maximum RMSDs in the dataset. This tutorial uses 0.1 Å to get a very precise estimate of the sampling precision; however this results in a very long calculation. In practice, especially for larger systems whose sampling precision will be much lower, one would choose a larger value to make calculation more efficient.

20. If alignment is necessary, the GPU mode of pyRMSD generally increases performance significantly. It is invoked by using `-m cuda`.

21. The output of the protocol can be readily plotted using any plotting software. Example scripts in `./analysis/scripts/gnuplot_scripts` can be used to obtain the plots in Figure 6.

22. Sometimes, there are too many clusters to visualize at the determined sampling precision. In this case, we can rerun clustering using a threshold worse than the sampling precision to get fewer clusters to visualize. In that case, the skip option (`-s`) along with the value of clustering threshold (`-ct`) allows one to bypass RMSD and sampling precision calculation and get the clusters and their densities, as follows: `python ../scripts/Master_Sampling_Exhaustiveness_Analysis.py -n`

`actin -d density_custom.txt -ct 4.39 -a -s`. Note that this clustering threshold should always be worse than the sampling precision.

23. Built-in Chimera color names can be found at: https://www.cgl.ucsf.edu/chimera/docs/UsersGuide/colortables.html

24. These keywords are specifically for completely disordered domains or short helical components. For `IDEAL_HELIX`, a single helix will be created for that component.

References

1. Mitra K, Frank J (2006) RIBOSOME DYNAMICS: Insights from Atomic Structure Modeling into Cryo-Electron Microscopy Maps. Annu Rev Biophys Biomol Struct 35:299–317 . doi: 10.1146/annurev.biophys.35.040405.101950

2. Robinson CV, Sali A, Baumeister W (2007) The molecular sociology of the cell. Nature 450:973–982

3. Sali A, Glaeser R, Earnest T, Baumeister W (2003) From words to literature in structural proteomics. Nature 422:216–225

4. Schmeing TM, Ramakrishnan V (2009) What recent ribosome structures have revealed about the mechanism of translation. Nature 461:1234–1242 . doi: 10.1038/nature08403

5. Blundell TL, Johnson LN (1976) Protein Crystallography. Academic Press, New York

6. Chiu W, Baker ML, Jiang W, et al (2005) Electron Cryomicroscopy of Biological Machines at Subnanometer Resolution. Structure 13:363–372 . doi: 10.1016/j.str.2004.12.016

7. Lučić V, Leis A, Baumeister W (2008) Cryo-electron tomography of cells: connecting structure and function. Histochem Cell Biol 130:185–196 . doi: 10.1007/s00418-008-0459-y

8. Stahlberg H, Walz T (2008) Molecular Electron Microscopy: State of the Art and Current Challenges. ACS Chem Biol 3:268–281 . doi: 10.1021/cb800037d

9. Parrish JR, Gulyas KD, Finley RL (2006) Yeast two-hybrid contributions to interactome mapping. Curr Opin Biotechnol 17:387–393 . doi: 10.1016/j.copbio.2006.06.006

10. Fernandez-Martinez J, Phillips J, Sekedat MD, et al (2012) Structure–function mapping of a heptameric module in the nuclear pore complex. J Cell Biol 196:419–434 . doi: 10.1083/jcb.201109008

11. Gingras A-C, Gstaiger M, Raught B, Aebersold R (2007) Analysis of protein complexes using mass spectrometry. Nat Rev Mol Cell Biol 8:645–654 . doi: 10.1038/nrm2208

12. Ward AB, Sali A, Wilson IA (2013) Integrative Structural Biology. Science 339:913–915 . doi: 10.1126/science.1228565

13. Lasker K, Förster F, Bohn S, et al (2012) Molecular architecture of the 26S proteasome holocomplex determined by an integrative approach. Proc Natl Acad Sci USA 109:1380–1387

14. Simon B, Madl T, Mackereth CD, et al (2010) An efficient protocol for NMR-spectroscopy-based structure determination of protein complexes in solution. Angew Chem Int Ed Engl 49:1967–1970 . doi: 10.1002/anie.200906147

15. Baù D, Sanyal A, Lajoie BR, et al (2011) The three-dimensional folding of the α-globin gene domain reveals formation of chromatin globules. Nat Struct Mol Biol 18:107–114 . doi: 10.1038/nsmb.1936

16. Viswanath S, Bonomi M, Kim SJ, et al (2017) The molecular architecture of the yeast spindle pole body core determined by Bayesian integrative modeling. Mol Biol Cell 28:3298–3314 . doi: 10.1091/mbc.E17-06-0397

17. Kim SJ, Fernandez-Martinez J, Nudelman I, et al (2018) Integrative structure and functional anatomy of a nuclear pore complex. Nature 555:475–482 . doi: 10.1038/nature26003

18. Molnar K, Bonomi M, Pellarin R, et al (2014) Cys-Scanning Disulfide Crosslinking and Bayesian Modeling Probe the Transmembrane Signaling Mechanism of the Histidine Kinase, PhoQ. Structure 22:1239–1251

19. Webb B, Viswanath S, Bonomi M, et al (2018) Integrative structure modeling with the Integrative Modeling Platform: Integrative Structure Modeling with IMP. Protein Sci 27:245–258 . doi: 10.1002/pro.3311

20. Shen M, Sali A (2006) Statistical potential for assessment and prediction of protein structures. Protein Sci 15:2507–2524 . doi: 10.1110/ps.062416606

21. Sippl MJ (1990) Calculation of conformational ensembles from potentials of mena force. J Mol Biol 213:859–883 . doi: 10.1016/S0022-2836(05)80269-4

22. Brooks BR, Brooks CL, Mackerell AD, et al (2009) CHARMM: The biomolecular simulation program. J Comput Chem 30:1545–1614 . doi: 10.1002/jcc.21287

23. Weiner SJ, Kollman PA, Case DA, et al (1984) A new force field for molecular mechanical simulation of nucleic acids and proteins. J Am Chem Soc 106:765–784 . doi: 10.1021/ja00315a051

24. Šali A, Blundell TL (1993) Comparative Protein Modelling by Satisfaction of Spatial Restraints. J Mol Biol 234:779–815 . doi: 10.1006/jmbi.1993.1626

25. Kelley LA, Mezulis S, Yates CM, et al (2015) The Phyre2 web portal for protein modeling, prediction and analysis. Nat Protoc 10:845–858 . doi: 10.1038/nprot.2015.053

26. Hanot S, Bonomi M, Greenberg CH, et al (2018) Bayesian multi-scale modeling of macromolecular structures based on cryo-electron microscopy density maps. . doi: 10.1101/113951

27. Kawabata T (2008) Multiple Subunit Fitting into a Low-Resolution Density Map of a Macromolecular Complex Using a Gaussian Mixture Model. Biophys J 95:4643–4658 . doi: 10.1529/biophysj.108.137125

28. Metropolis N, Rosenbluth AW, Rosenbluth MN, et al (1953) Equation of State Calculations by Fast Computing Machines. J Chem Phys 21:1087–1092 . doi: 10.1063/1.1699114

29. Swendsen RH, Wang J-S (1986) Replica Monte Carlo Simulation of Spin-Glasses. Phys Rev Lett 57:2607–2609 . doi: 10.1103/PhysRevLett.57.2607

30. Viswanath S, Chemmama IE, Cimermancic P, Sali A (2017) Assessing Exhaustiveness of Stochastic Sampling for Integrative Modeling of Macromolecular Structures. Biophys J 113:2344–2353 . doi: 10.1016/j.bpj.2017.10.005

31. Alber F, Dokudovskaya S, Veenhoff LM, et al (2007) The molecular architecture of the nuclear pore complex. Nature 450:695–701

32. Sali A, Berman HM, Schwede T, et al (2015) Outcome of the First wwPDB Hybrid/Integrative Methods Task Force Workshop. Struct Lond Engl 1993 23:1156–1167 . doi: 10.1016/j.str.2015.05.013

33. Burley SK, Kurisu G, Markley JL, et al (2017) PDB-Dev: a Prototype System for Depositing Integrative/Hybrid Structural Models. Struct Lond Engl 1993 25:1317–1318 . doi: 10.1016/j.str.2017.08.001

34. Vallat B, Webb B, Westbrook JD, et al (2018) Development of a Prototype System for Archiving Integrative/Hybrid Structure Models of Biological Macromolecules. Struct Lond Engl 1993. doi: 10.1016/j.str.2018.03.011

35. Gil VA, Guallar V (2013) pyRMSD: a Python package for efficient pairwise RMSD matrix calculation and handling. Bioinformatics 29:2363–2364 . doi: 10.1093/bioinformatics/btt402

36. Rao JN, Dominguez R (2014) Complex of ATP-actin With the C-terminal Actin-Binding Domain of Tropomodulin. . doi: 10.2210/pdb4pki/pdb

37. Schneidman-Duhovny D, Hammel M, Sali A (2010) FoXS: A Web Server for Rapid Computation and Fitting of SAXS Profiles. Nucleic Acids Res 38:541–544

38. Robinson P, Trnka M, Pellarin R, et al (2015) Molecular architecture of the yeast Mediator complex. eLife 10.7554/eLife.08719

39. Goddard TD, Huang CC, Meng EC, et al (2018) UCSF ChimeraX: Meeting modern challenges in visualization and analysis: UCSF ChimeraX Visualization System. Protein Sci 27:14–25 . doi: 10.1002/pro.3235

40. Carter L, Kim SJ, Schneidman-Duhovny D, et al (2015) Prion protein-antibody complexes characterized by chromatography-coupled small-angle X-ray scattering. Biophys J 109:793–805

41. Lasker K, Topf M, Sali A, Wolfson HJ (2009) Inferential optimization for simultaneous fitting of multiple components into a cryoEM map of their assembly. J Mol Biol 388:180–194