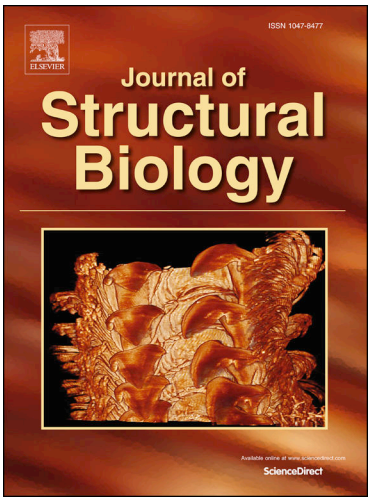
Journal Pre-proofs



Integration of software tools for integrative modeling of biomolecular systems

Matthew Hancock, Thomas-Otavio Peulen, Ben Webb, Billy Poon, James S Fraser, Paul Adams, Andrej Sali

PII: S1047-8477(22)00011-9
DOI: https://doi.org/10.1016/j.jsb.2022.107841
Reference: YJSBI 107841

To appear in: *Journal of Structural Biology*

Received Date: 13 October 2021
Revised Date: 28 January 2022
Accepted Date: 4 February 2022

Please cite this article as: Hancock, M., Peulen, T-O., Webb, B., Poon, B., Fraser, J.S., Adams, P., Sali, A., Integration of software tools for integrative modeling of biomolecular systems, *Journal of Structural Biology* (2022), doi: https://doi.org/10.1016/j.jsb.2022.107841

Integration of software tools for integrative modeling of biomolecular systems

Matthew Hancock[1,2], Thomas-Otavio Peulen[2], Ben Webb[2] , Billy Poon[3] ,

James S Fraser[2,6], Paul Adams[3,4] , Andrej Sali*[2,5,6]

* Please direct all correspondence to Andrej Sali.

Email: sali@salilab.org. Phone: 1-415-514-4227.

Mailing Address: 1700 4th St, Rm 503B, San Francisco, CA 94158, United States

1. Biophysics Graduate Program, University of California, San Francisco, MC 2240 1600 16th St, San Francisco, California 94143, United States
2. Department of Bioengineering and Therapeutic Sciences, UCSF Box 0775 1700 4th St, University of California, San Francisco, San Francisco, California 94158, United States
3. Molecular Biophysics and Integrated Bioimaging Division, Lawrence Berkeley National Laboratory, Building 33 1 Cyclotron Rd, Berkeley, California 94270, United States
4. Department of Bioengineering, University of California, Berkeley, MC 1762 306 Stanley Hall, Berkeley, California 94720, United States
5. Department of Pharmaceutical Chemistry, University of California, San Francisco, UCSF Box 2880 600 16th St, San Francisco, California 94143, United States
6. Quantitative Biosciences Institute (QBI), University of California, San Francisco, 1700 4th St, San Francisco, California, United States

Author emails: matthew.hancock@ucf.edu, thomas-otavio.peulen@ucsf.edu, ben@salilab.org, BKPoon@lbl.gov, jfraser@fraserlab.com, PDAdams@lbl.gov, sali@salilab.org

Keywords: integrative modeling; structural modeling; integrative structural biology; software integration

Abstract

Integrative modeling computes a model based on varied types of input information, be it from experiments or prior models. Often, a type of input information will be best handled by a specific modeling software package. In such a case, we desire to integrate our integrative modeling software package, *Integrative Modeling Platform* (IMP), with software specialized to the computational demands of the modeling problem at hand. After several attempts, however, we have concluded that even in collaboration with the software's developers, integration is either impractical or impossible. The reasons for the intractability of integration include software incompatibilities, differing modeling logic, the costs of collaboration, and academic incentives. In the integrative modeling software ecosystem, several large modeling packages exist with often redundant tools. We reason, therefore, that the other development groups have similarly concluded that the benefit of integration does not justify the cost. As a result, modelers are often restricted to the set of tools within a single software package. The inability to integrate tools from distinct software negatively impacts the quality of the models and the efficiency of the modeling. As the complexity of modeling problems grows, we seek to galvanize developers and modelers to consider the long-term benefit that software interoperability yields. In this article, we formulate a demonstrative set of software standards for implementing a model search using tools from independent software packages and discuss our efforts to integrate the IMP and the crystallography suite *Phenix* within the Bayesian modeling framework.

Introduction

Introduction to integrative modeling

Integrative modeling combines information of different types into a model (Alber et al., 2007; Rout and Sali, 2019). When all available information is used, the accuracy, precision, and completeness of the model are maximized. An example of an integrative model is the double-helical structure of DNA, which could only be resolved through a joint consideration of a fiber X-ray diffraction pattern of the DNA, data about composition and stoichiometry of the component nucleotides, as well as theoretical information about physiochemical nucleotide complementarity (Watson and Crick, 1953). Modern integrative modeling of biomolecular structures similarly considers experimental data (*e.g.*, an X-ray diffraction pattern, a cryo-electron microscopy (cryo-EM) density map, and nuclear magnetic resonance (NMR) spectra) and prior information (*e.g.*, a molecular mechanics force field, a statistical potential, and previously obtained structural models). As the complexity (*e.g.*, size, resolution, heterogeneity, and dynamics) of biomolecular structural models grows, integration of diverse and often sparse experimental data will be critical for maximally exploiting experimental techniques and their complementarities (Figure 1) (Sali, 2021).

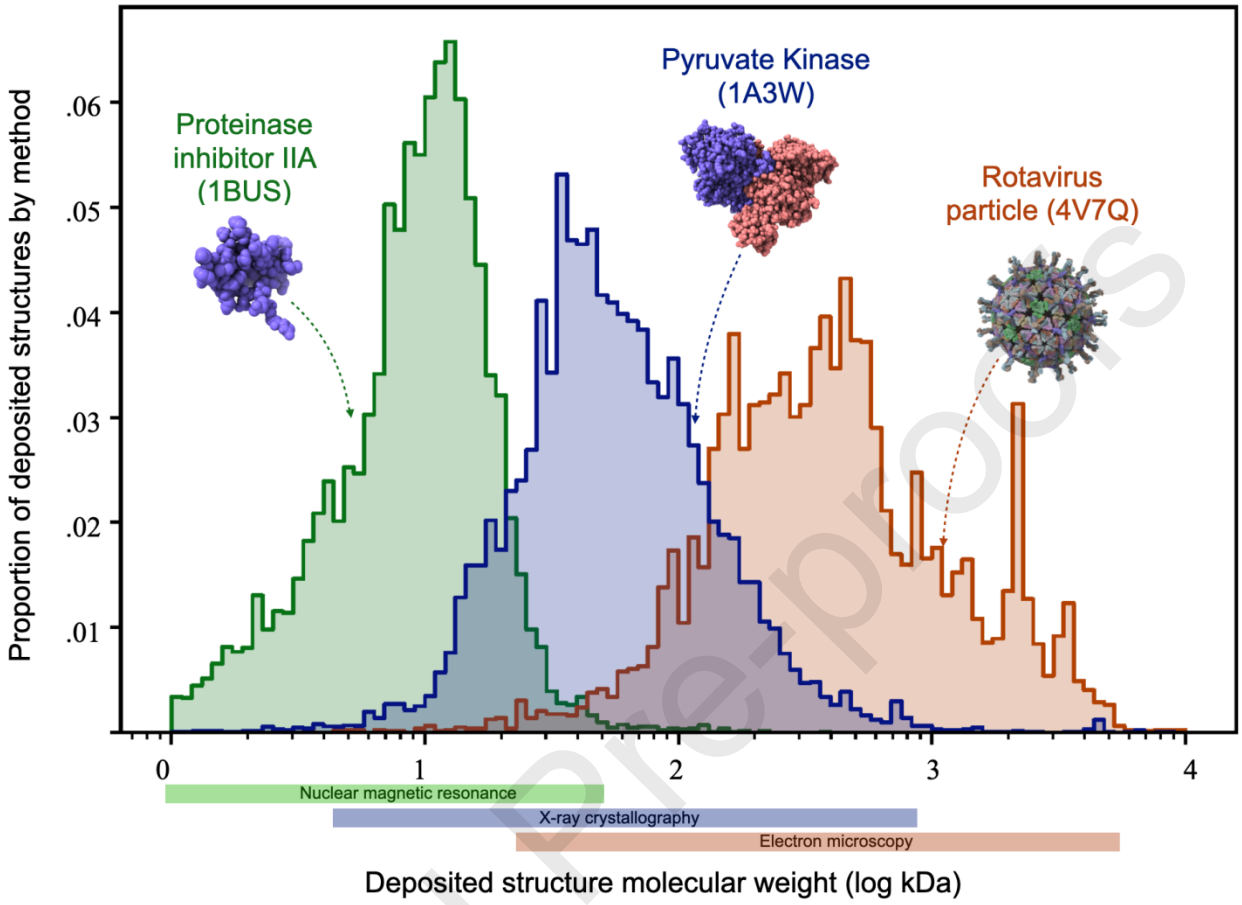


Figure 1: Coverage of molecular weight by structural technique.

Histogram of the molecular weight of structures resolved by solution-state NMR spectroscopy, macromolecular X-ray crystallography, and electron microscopy (EM) deposited in the Protein Data Bank (PDB) (Berman et al., 2003). Each technique has a different coverage with respect to the molecular weight of the studied system. This variation illustrates one reason why it is desirable to integrate varied types of information as well as the software tools used to compute models from them.

Developing software modeling tools is challenging

Modeling is often only possible through computation using software tools. The development of software tools for modeling is challenging because it demands both domain knowledge and technical

expertise. Domain knowledge encompasses both an understanding of the general concepts of the field and a thorough understanding of both the theory and practices of the leveraged experimental techniques. The technical prerequisites include the ability to formulate the theory into stable numerical algorithms and to develop software of sufficient quality. Moreover, software needs to incorporate new technical and scientific advancements. Fulfilling the above prerequisites is difficult when developing software for an individual experimental technique. The challenge is compounded in the development of integrative modeling software, where several sources of experimental information generally are combined. Nevertheless, there is a large number of independent and complex integrative modeling software packages (Adams et al., 2010; Das and Baker, 2008; Dimura et al., 2016; Dominguez et al., 2003; Hsieh et al., 2017; Hua et al., 2018; Hummer and Köfinger, 2015; Karakaş et al., 2012; Leaver-Fay et al., 2011; Russel et al., 2012; Schneidman-Duhovny et al., 2005; Schwieters et al., 2018; Serra et al., 2017; Trussart et al., 2015; van Zundert et al., 2016), as reviewed (Sali, 2021).

Collaborative development maximizes the efficiency and quality of modeling

Given the technical and scientific difficulty of developing software modeling tools, it is desirable to tackle integrative modeling as a collaborative development effort of multiple research groups and development teams. Such collaborative development has the benefits of better integration of domain expertise into modeling software. There is also a benefit in the distribution of development costs over multiple research groups. For these reasons, collaborative software development maximizes both the efficiency of the software development and the quality of the resulting models.

Integration of software tools is an efficient approach to collaborative development

One way to achieve collaborative development is by developing software that is designed to solve a modeling problem through integration with one or more additional modeling tools. Such integration can occur between tools within and across integrative modeling software packages. The ability to combine the tools from existing software packages can extend the set of methods that can be

implemented with these tools. It is often better to approach an integrative modeling problem by mixing and matching existing tools rather than by building new ones because the developer benefits from prior work. For example, Rosetta (Leaver-Fay et al., 2011) integrates high-level tools within Rosetta through RosettaScripts (Fleishman et al., 2011) and lower-level Rosetta functionality with tools outside of Rosetta through the C++/Python layers, such as *phenix.rosetta_refine (DiMaio et al., 2013)*.

Why is integration of software tools challenging?

There are numerous structural biology software packages, including tens of integrative modeling programs alone (Rout and Sali, 2019). While there is consensus that software interoperability is beneficial, little has been done to address the issue. It is often attractive to implement new tools within one's own software ecosystem, even if similar tools already exist elsewhere. It is not unusual to see, for example, unique implementations of vector classes within modeling packages. Possible reasons include the difficulty of coordinating a large number of contributors, lack of support and motivation for rigorous development and maintenance standards for academic software, and the incentive for individual programmers and research groups to publish new software.

Opportunity for standardization

As the complexity of integrative modeling problems grows, they will be increasingly difficult to solve by relying on tools from a single software package. Integration can be done on an *ad hoc* basis, where specific tools are combined when needed. Such an integration is not efficient, however, as the pool of tools grows because any integration would incur additional development costs. An alternative is the adoption of software standards for tools developed by the integrative modeling community. Software standards offer guarantees on some aspect of the software's implementation or function. Well-defined standards would benefit the integrative modeling field by ensuring that software tools are of sufficient quality and generality to be interoperable with other software tools that adhere to the standards. If a standard is adopted by the community, an integrative modeler may be able to readily mix-and-match software that has been produced

from multiple development groups. Similar to the PDB/mmCIF standards for archival, the nature and extent of the standards must be agreed upon by the integrative modeling development community.

Article overview

In this article, we develop a demonstrative software standard for integrating one or more independent tools in a model search. We begin by describing modeling as a 5-step search for a model that satisfies input information and distinguishing between informed and uninformed model searches. We then describe how the model search is achieved through basic function definitions. We illustrate the standards by integrating *Phenix* and *Integrative Modeling Platform* (IMP) for computing an atomic model from X-ray crystallography datasets and a molecular mechanics force field. Key to the integration is the factorization of the model posterior density into likelihoods and priors. We discuss our attempts to integrate IMP and *Phenix* tools as independent processes *via* file input/output and within a single process using both libraries' application programming interface (API). We conclude by discussing how software integration may increase the quality and potential complexity of the model as well as the efficiency of the modeling.

Approach

Modeling as a search

A model is a depiction of a system or process that we would like to inform from input information, consisting of experimental data and prior information (Rout and Sali, 2019). A model can then be used to rationalize input information and make testable predictions. Modeling is the search for a set of models consistent with the input information. Ideally, we aim to find all models that satisfy the input information, reflecting the uncertainty of the input information. It is convenient to divide the search into the following three steps: (*i*) defining the model representation that specifies all degrees of freedom whose values are determined by modeling, (*ii*) defining a scoring function for ranking alternative models for their agreement with the input information, and (*iii*) generating a sample of good-scoring models. As an aside, these models can be optionally filtered based on the input information and should also be validated before interpretation (Rout and Sali, 2019).

Multistate model of Nup133 computed by a model search

For example, a model search is used to compute a multi-state model of the Nup133 nucleoporin from small-angle x-ray scattering (SAXS), electron microscopy class averages, and cross-linking mass spectrometry (XL-MS) data (Kim et al., 2014). To reflect the structural heterogeneity of Nup133 in solution, the authors defined the model representation as an ensemble of fully atomic structures. The degrees of freedom, to be fit to the input information, include the number of models in the ensemble as well as the positions of atoms in each structure. Therefore, the objective of the model search is to find all Nup133 ensembles that satisfy the input information. A sample of Nup133 ensembles was generated *via* molecular dynamics simulations (MD) such that sufficient coverage of the energy landscape was achieved. The scoring function then evaluated the consistency of any model ensemble with the SAXS profile, the EM class averages, and chemical cross-links by simulating data via physical principles and comparing it to the

observed data. The model search framework is a general description of modeling that can describe most modeling protocols.

Informed vs uninformed search

Generally, integrative modeling software either explicitly or implicitly implements tools for each step of the model search. The model search can be categorized as either uninformed or informed, relative to some input information. In the uninformed search, candidate models are systematically generated to explore the search space without consideration for a specified subset of input information (Grosan and Abraham, 2011). For example, the minimal ensemble approach to computing protein structure ensembles based on SAXS data generates an ensemble of structures without consideration of the experimental data (Köfinger et al., 2019). In the informed search, a specified subset of input information is used to bias the generation of solutions (Grosan and Abraham, 2011). For example, partial derivatives based on the SAXS data could be used to guide sampling.

It is easier to isolate software tools in an uninformed search because outputs of relevant modeling steps can be combined as an additional post-processing step. However, due to a large model space generally required to be searched when solving integrative modeling problems, we are interested in the integration of software tools that enable informed search, in addition to the uninformed search. Informed search demands the passage of information between the tools that implement representation, scoring, and sampling during the modeling procedure.

Demonstrative software standards (Figure 2)

While tools within a given software package generally pass information to each other, they have not been designed to do so across different software packages. Interoperability may sometimes be achieved by engineering connections between a particular set of software tools in an *ad hoc* fashion. However, to maximize generalizability, it is better if information is communicated in well-defined channels defined by

a software standard. Here, we develop basic standards to illustrate how information passing could be accomplished between independent modeling tools.

To facilitate the integration of two modeling tools, they should share the minimal amount of information necessary. It is also desirable that the tools be limited in scope to maximize modularity. For example, if a tool implemented a specific modeling step (*i.e.,* representation, scoring, sampling, and optionally filtering plus validation). Tools with a well-defined purpose that hide their implementation details help manage the technical complexity of an integrative modeling problem. In the example of the Nup133 model search, a distinct software tool could be used to implement the multi-state model representation, the molecular dynamics sampler, the SAXS scoring function, the XL-MS scoring function, and the EM scoring function. The tools must be able to communicate with one another, but at the same time, they should be encapsulated from each other's technical complexity.

To design message passing, we first define the function of a model representation, scoring function, and sampling algorithm tool based on an informed search. The model representation manages the model state (the current value of the model parameters), which we partition between structural, $\mathbf{X}$, and nuisance parameters, $\boldsymbol{\sigma}$. The model representation manages the model state at step i of the model search, $\{\mathbf{X}, \boldsymbol{\sigma}\}_{\mathbf{i}}$, as well as previously visited states, $\{\mathbf{X}, \boldsymbol{\sigma}\}_0$,..., $\{\mathbf{X}, \boldsymbol{\sigma}\}_{\mathrm{i}-1}$. The scoring function computes the scores $\boldsymbol{s}$, an assessment of the compatibility of $\{\mathbf{X}, \boldsymbol{\sigma}\}_{\mathbf{i}}$ with input information, $\mathbf{D}$. The scoring function also returns heuristics, $\mathbf{h}$, as a function of $\mathbf{D}$, which help inform the search process (*eg,* gradients for finding local minima in the search space), $f(\{\mathbf{X}, \boldsymbol{\sigma}\}_{\mathrm{i}}, \mathbf{D}) = \mathbf{s}, \mathbf{h}$. A sampling tool updates the model state based on $\{\mathbf{X}, \boldsymbol{\sigma}\}_{\mathbf{i}}$, $\mathbf{s}$, and $\mathbf{h}$, $g(\{\mathbf{X}, \boldsymbol{\sigma}\}_{\mathrm{i}}, \boldsymbol{s}, \boldsymbol{h}) = \{\mathbf{X}, \boldsymbol{\sigma}\}_{\mathrm{i}+1}$.

Based on the above definitions, the minimal informed search is implemented by the following 4 functions that facilitate communication between the model representation, the scoring function, and the sampling algorithm tools (Figure 2). First, the scoring function must be able to access the model state from the model representation (*get_state*). Second, the sampling algorithm must be able to access the current model

parameters from the model representation (*get_state*). Third, the sampling algorithm must be able to also access the score and heuristic information from the scoring function (*get_score*). Finally, the sampling algorithm must be able to update the current model parameters (*update_state*).

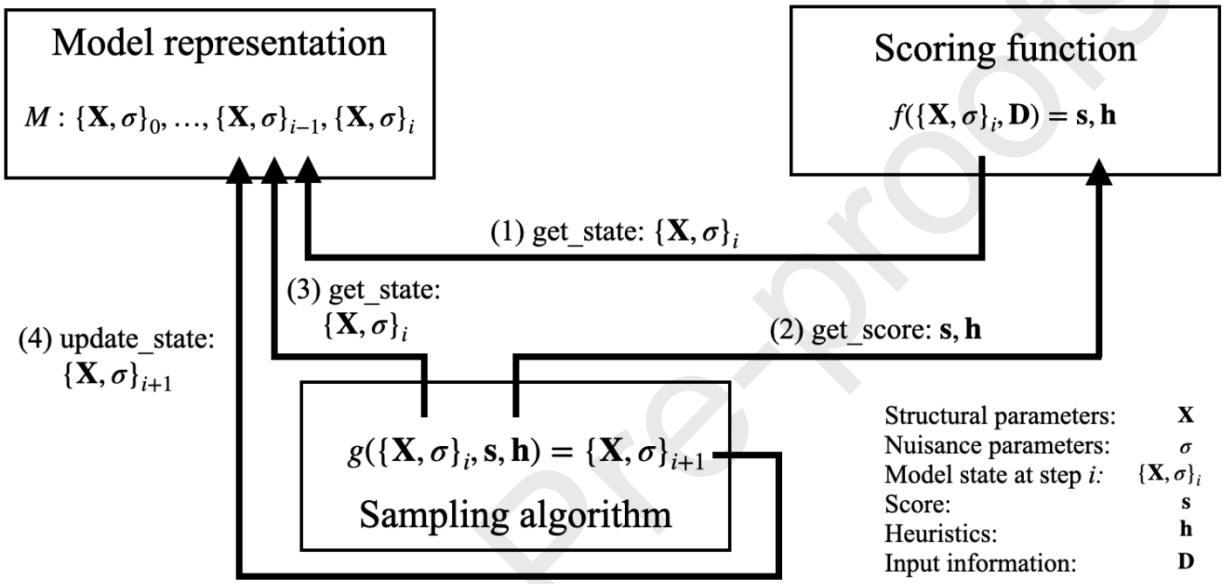


Figure 2. Class diagram for proposed standard for implementing a model search.

A model search may be implemented through the communication of software tools implementing a modeling step (representation, scoring, sampling). A box represents an independent tool and the arrows represent function calls.

Our demonstrative standard is sufficiently general to enable the mixing-and-matching of modeling tools. For example, when computing a model of Nup133, an integrative modeler wishes to score their model against multiple sources of input information (SAXS, XL-MS, and EM) that are not easily handled by a single modeling tool. 3 distinct scoring tools may be drawn from different modeling packages specialized to the computational demands unique to each experimental datatype, Optimally, these scoring function tools operate independently on each information source to manage the complexity of the tool's implementation.

If all tools provide a uniform interface for returning the score and computed heuristics (*get_score*), they may be used interchangeably while remaining isolated from one another.

Illustrative modeling problem

We demonstrate the software standard by integrating IMP and *Phenix* to solve a specific problem in X-ray crystallography. Namely, we are interested in computing a model of a set of atomic protein structures (multi-state model), based on multiple diffraction datasets collected at different temperatures and physical principles. Such a multi-state model can be useful for mapping the dynamics and allostery of proteins (Fraser et al., 2011; Keedy et al., 2015). Next, we describe the three necessary steps in modeling.

Input Information. We are interested in informing our model by both structure factors from the X-ray crystallography experiments in addition to an empirical molecular mechanics force field. Satisfaction of the X-ray crystallography data restrains the overall model geometry while an empirical potential energy function restrains the stereochemistry and nonbonded interactions of local sets of atoms. Utilization of experimental X-ray diffraction datasets in conjunction with a force-field has been applied previously in computing X-ray models (Brunger et al., 1989; Burnley et al., 2011).

Representation. The multi-state model *M* is defined by Cartesian atomic coordinates for each of a small number of discrete structural states of a protein; the model also includes the relative weight of each state.

Bayesian scoring function. In general, Bayes' theorem states that the posterior model density, $p(\mathrm{M} \mid \mathrm{D}, \mathrm{I})$, (the conditional probability density of model, *M*, given experimental measurements, *D*, and prior information, *I*) is proportional to the product of the data likelihood, $p(\mathrm{D} \mid \mathrm{M}, \mathrm{I})$, (the probability of *D* given *M* and *I*) and prior distribution, $p(\mathrm{M} \mid \mathrm{I})$, (the probability of *M* given *I*):

$$p(\mathrm{M} \mid \mathrm{D}, \mathrm{I}) \propto p(\mathrm{D} \mid \mathrm{M}, \mathrm{I}) \times p(\mathrm{M} \mid \mathrm{I})$$

For our multi-temperature model where we have multiple diffraction datasets, D_i, we assume that the likelihood is the product of independent likelihoods for each diffraction dataset. The posterior model density is:

$$p(\mathrm{M} \mid \mathrm{D}, \mathrm{I}) \propto \prod p(\mathrm{D}_i \mid \mathrm{M},\mathrm{I}) \times p(\mathrm{M}\mid \mathrm{I})$$

where $p(\mathrm{D}_i|\mathrm{M},\mathrm{I})$ is the likelihood for the diffraction data from the *i-th* experiment. In Bayesian modeling, the model is not a single model instance, rather the model is the posterior density over the entire model space spanned by the degrees of freedom defined by the model representation. The uncertainty of the model is the posterior model density spread. Bayesian modeling is conducive for integrative modeling because likelihoods and priors can be combined from diverse experimental datasets and prior information. As is often the case with probabilistic modeling, the score, s, is the negative logarithm of the model posterior density:

$$\mathrm{s} = -\log p(\mathrm{M} \mid \mathrm{D}, \mathrm{I})$$

$$s = -\log p(\mathrm{M} \mid \mathrm{I}) - \log \prod p(\mathrm{D}_i \mid \mathrm{M},\mathrm{I})$$

$$s = -\log p(\mathrm{M} \mid \mathrm{I}) - \sum \log p(\mathrm{D}_i \mid \mathrm{M},\mathrm{I})$$

Though the scores can be weighed as pure probabilities, it is often useful to weigh the likelihood and priors so that the gradients have comparable magnitudes (Brunger et al., 1989). The weights, w_1, w_2, may be optimized to a target function (*eg*, Phenix.refine) or empirically chosen:

$$s = -\mathrm{w}_1 \log p(\mathrm{M} \mid \mathrm{I}) - \mathrm{w}_2 \sum \log p(\mathrm{D}_i \mid \mathrm{M},\mathrm{I})$$

Sampling. A sample can be generated from the model posterior *via* Molecular Dynamic simulation where atomic positions are updated based on a force computed from the potential energy surface. The stochastic Monte Carlo method, where parameter moves are accepted and rejected based on relative energy levels, is also useful for generating molecular ensembles. Furthermore, enhanced sampling variations of the Monte Carlo method that leverage derivatives (*e.g Hamiltonian* Monte Carlo) are useful for improving sample convergence from complex posterior model densities.

To implement the representation, scoring, and sampling outlined above, we integrated software tools from the *Phenix* software suite and IMP modeling package using our demonstrative standard.

Phenix

Phenix (Python-based Hierarchical Environment for Integrated Xtallography) provides a suite of programs for manipulating experimental data, computing models, and validating structures from cryo-EM and X-ray/neutron/electron crystallography data. *Phenix* includes tools for the entire data processing and model generation workflow, including computing data quality indicators (phenix.xtriage), maximum likelihood estimation of phases from molecular replacement *via* a homologous structure (phenix.phaser (McCoy et al., 2007)), phase optimization (phenix.density_modification (Terwilliger et al., 2020)), model building (phenix.autobuild (Terwilliger et al., 2008)), refinement of the model to better fit both experimental and empirical restraints (phenix.refine (Afonine et al., 2012)), and finally model validation (access to MolProbity webserver (Williams et al., 2018)). Despite a large number of algorithms in *Phenix*, there would be great benefit from integration with IMP, for example by providing access to the flexible model representations, incorporation of non-crystallographic information, and enhanced sampling techniques in IMP.

IMP

Integrative Modeling Platform (IMP) is open-source software that contains a large number of libraries and programs for flexibly computing integrative models of biomolecular systems (integrativemodeling.org) (Russel et al., 2012). IMP supports a diverse set of model representations that can be flexibly coarse-grained to suit the problem. Restraints can be formulated to score models against various experimental data (*eg*, chemical cross-links identified by mass spectrometry, electron microscopy density maps, and small-angle X-ray scattering profiles) as well as prior models (*eg*, excluded volume, comparative models, molecular mechanics force fields, and statistical potentials). Models can be sampled or enumerated through numerical integration techniques, variations of the Monte Carlo method, as well as Molecular and Brownian Dynamics simulations. IMP's relative strengths include a large variety of model representations, scoring functions based on different data, and sampling schemes, all of which can be mixed

and matched relatively easily with each other to facilitate integrative structure modeling. Another distinction is an increasingly Bayesian perspective on uncertainties in input information, model representations, and scoring functions. In contrast, IMP does not include tools for X-ray crystallography. However, *Phenix* is a premier program for this task.

Integration of Phenix and IMP (Figure 3)

As IMP and *Phenix* use independent modeling frameworks, we were required to implement the standard functions by using and modifying software tools from both packages. Engineering the interfaces, therefore, required familiarity with the application and implementation of tools from both software ecosystems. For example, IMP and *Phenix* employ their own model hierarchy to represent atomic structures. Yet for the model search, there must be a shared definition of the model representation. Efficiently managing and interconverting between the unique model hierarchies proved challenging. Such challenges of handling specific modeling packages further motivates the need for general standards to facilitate software interoperability without additional development effort.

The multi-state model representation is defined in IMP (IMP.Model). The sampler is also defined in IMP (IMP.MolecularDynamics). We implemented 2 scoring function tools, to evaluate the prior and likelihood respectively. The distinction between data likelihood and prior provides an opportunity for distributing an evaluation of the scoring function across both *Phenix* and IMP, taking advantage of the comparative strengths of each software package. The prior scoring function uses the IMP.atom library to evaluate stereochemical and non-bonded scores based on the CHARMM22 empirical force field. The likelihood scoring function uses *Phenix*'s maximum likelihood target function for a given set of experimentally observed structure factors. Evaluation of the crystallography likelihood includes several computationally demanding tasks such as the inference of distribution parameters and determination of the solvent region in the unit cell. Both the prior and likelihood scoring function tools were implemented with

an identical interface to return the score and gradients, ensuring compatibility with IMP.Model and IMP.MolecularDynamics.

The model search could be implemented in two ways: in separate runtime environments *via* data integration or in the same runtime environment *via* library integration, as follows.

Data integration. Our first attempt to integrate *Phenix* and IMP was by exchanging data between independent executions of custom scoring evaluation programs written separately from the *Phenix* (Computational Crystallography Toolbox (CCTBX)) and IMP libraries, respectively. The stochastic sampling algorithm proposes a move and saves the coordinates to a disk file in the PDB format, which is then read separately by the two evaluation programs. The *Phenix* program computes the likelihood while the IMP program computes the prior independently. The posterior is the product of these two terms. The advantage of this strategy is that the runtime environments are completely separated. However, the design presents a runtime challenge because all model data must be saved from the IMP address space to disk and then be read by the IMP and *Phenix* programs. As the structure factor calculations using Fast Fourier Transform (FFT) are extremely fast, the addition of the computational overhead presented a significant challenge to generating and scoring a sufficiently large sample. *Phenix* can compute 6224 structure factors for a ubiquitin molecule in ~0.015 seconds on a single computational core. *Phenix* can read and write a PDB file in ~0.08 seconds, while IMP is even slower. As millions of samples may be necessary to sufficiently sample the Bayesian posterior model density, this additional overhead makes the data integration expensive. For the informed search to be computationally feasible, the likelihood and prior evaluations must occur in the same runtime environment, which is achieved *via* library integration.

Library integration. We also engineered evaluation of both the likelihood and prior with *Phenix* and IMP, respectively, within the same runtime environment. A single evaluation function accepts a model and computes both the likelihood and prior using *Phenix* and IMP library calls, respectively. To integrate the scoring functionalities of *Phenix* and IMP, it was essential to develop a code for translating between the IMP and CCTBX hierarchical representations. As a result, the IMP sampling algorithm can make proposals

based on IMP's implementation of the model representation that is automatically reflected in CCTBX's implementation representation and can be used natively with other CCTBX tools. Although the release of both IMP and *Phenix's* underlying libraries (CCTBX) in conda-forge enables a consistent Python environment, we opted to build our integration in C++ for its performance advantages, followed by wrapping in Python for usability. We handled the technical integration of both IMP and *Phenix* shared dynamic (.so) libraries along with their dependencies through a custom compilation of IMP facilitated by CMake.

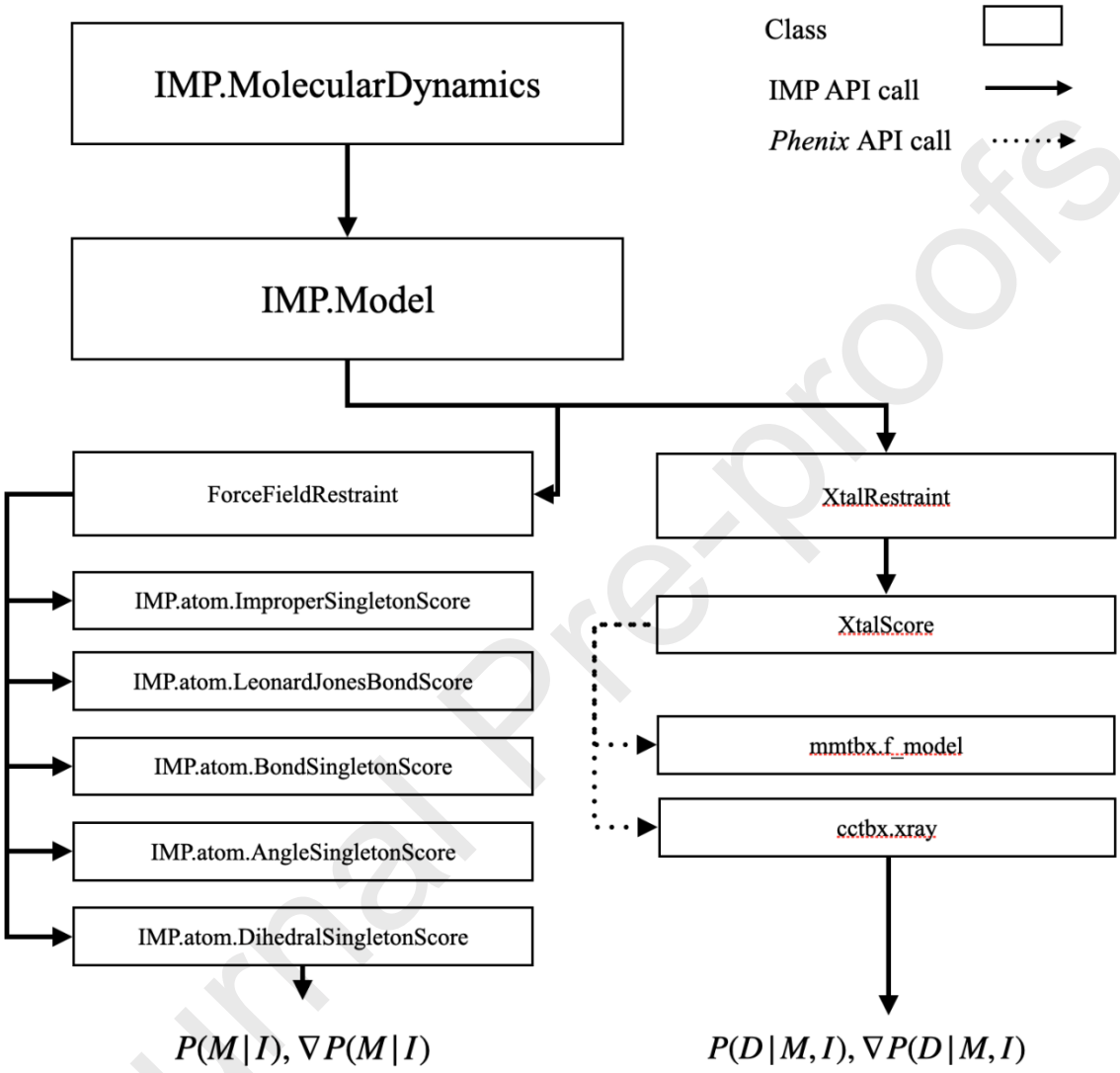


Figure 3. Workflow for integrating modeling software, exemplified by Phenix and IMP.

The sampler is implemented by IMP.MolecularDynamics. The model representation is implemented by IMP.Model. Two scoring tools are used: ForceFieldRestraint for computing the prior based on an empirical force field and XtalRestraint for computing the likelihood based on the observed diffraction data. IMP.MolecularDynamics calls to the model representation evaluate both restraints. The restraint then calls the functions to compute the score and gradients. For ForceFieldRestraint, calls were made to the IMP.atom

library to return individual stereochemical and nonbonded scores which are combined to compute in the total prior and gradients. For XtalRestraint, calls were made to *Phenix* functions within the mmtbx and cctbx libraries. IMP.MolecularDynamics updates the model parameters based on Newton's second law of motion where the force is derived from the sum of the returned gradients. The modularity of the design enables the substitution of alternative model representation and sampling tools (IMP.MonteCarlo). The isolation of *Phenix* and IMP scoring evaluations demonstrates how software integration is facilitated by the factorization of the Bayesian posterior model density into a data likelihood and a prior.

Conclusions

We were successful in incorporating IMP and *Phenix* functions, data structures, and numerical calculators in a model search. Rather than integrating IMP and *Phenix* tools in an *ad hoc* fashion, we organized them as proposed by our standard where the scoring function, sampling, and model representation tools communicate through defined channels. Importantly, the crystallographic likelihood and derivative calculator were completely independent of the molecular mechanics force field likelihood and derivative calculator. Using the model search implemented through the integration of IMP and *Phenix*, we generated a sample of the model posterior density from the structure of SARS-Cov main protease (PDB ID: 2H2Z) (Figure 4). Structures contained within the sample are drawn from the potential energy landscape formulated from the satisfaction of empirical stereochemical and non-bonded relationships as well as the observed X-ray data.

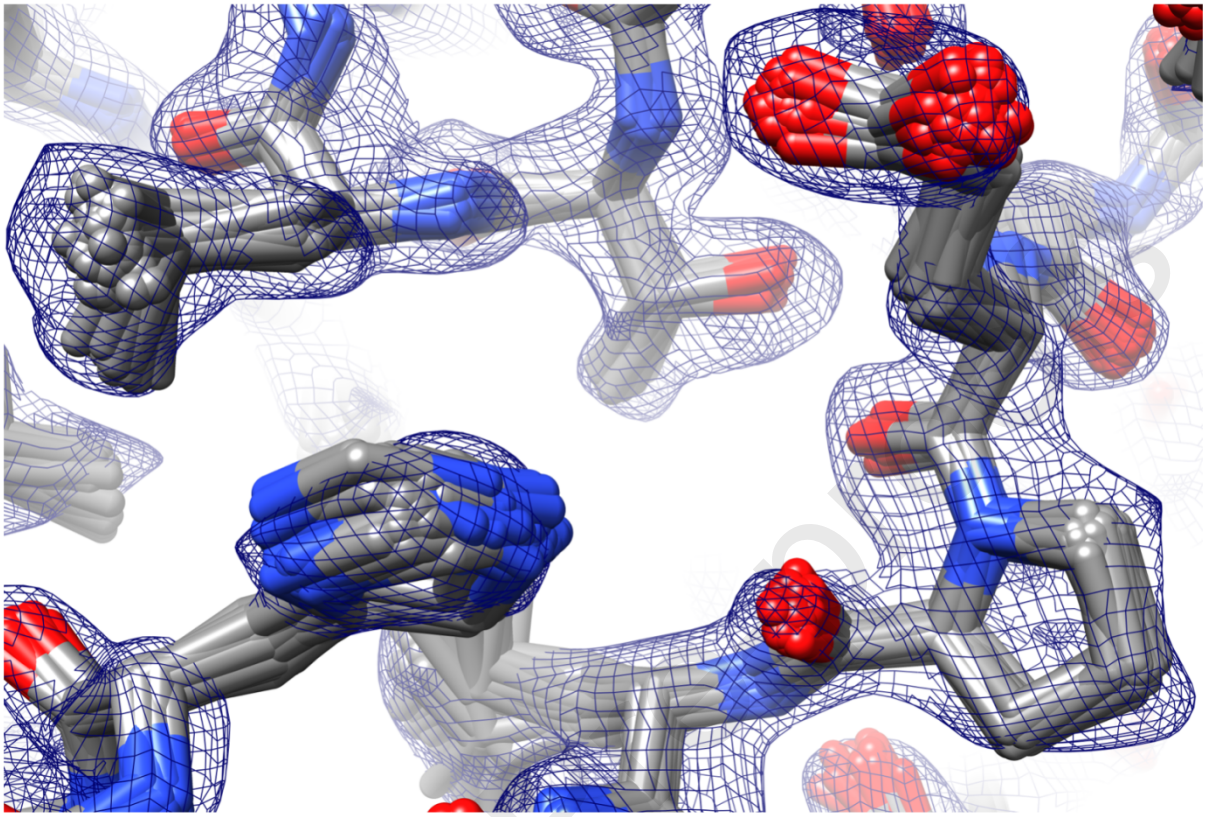


Figure 4. Sample of SARS-CoV main protease posterior model density evaluated through integration of IMP and *Phenix*.

Sample of Histidine 246, Proline 241, Glutamic Acid 240, and Leucine 202 conformations from the SARS-Cov main protease model posterior density. The sample of 100 structures was generated *via* Molecular Dynamics sampling of the model posterior density. The posterior was evaluated as the product of the molecular mechanics prior and the X-ray likelihood, computed by IMP and *Phenix* libraries respectively. The model posterior density sample is overlaid by the all features (2Fo-Fc) map.

As introduced above, the first major advantage of integration is that little crystallography-specific source code must be implemented in IMP. By leveraging the crystallographic functionality of *Phenix*, we do not have to implement a large number of crystallographic data structures and subroutines in IMP. CCTBX and IMP consist of 6210 (955,323 lines of code) and 3450 unique source files (327,966 lines of

code), respectively, supported by multiple groups around the world. Direct integration of source has historically been the *de facto* approach to accommodating new data types in IMP. This integration saves significant development time and also prevents inflation of either codebase. We also benefit from the significant amount of previous *Phenix* development and will continue to benefit from future *Phenix* development. IMP also benefits from the *Phenix* authors' expertise in computational crystallography, which includes significant runtime optimizations (*e.g.*, testing whether FFT or direct summation is faster for a given crystal system).

Secondly, the standard is sufficiently general to enable substitution of other model representation, sampling algorithm, or scoring function tools. For example, the modularity is well suited for introducing additional scoring function tools as they simply need to parse the model representation and return the computed score plus heuristics. Using the Bayesian framework, new tools could be easily introduced for computing likelihoods from new forms of experimental data, for example nuclear magnetic resonance (NMR) restraints, or prior information, for example statistical potentials from the PDB.

More complex sampling and scoring procedures may also be employed. The optimal strategy for sampling a model space is often unknown prior to modeling. Methods often rely on iterations of sampling procedures that may vary the sampling parameters (*eg,* temperature in simulated annealing) or the scoring functions (optimization of satisfaction of one source of input information over the other). Based on our framework, the model representation and scoring functions, are encapsulated from the details of the sampling procedure (in other words, *g*). New sampling procedures can easily be substituted so long as they can access the model representation to update the parameter state and accept scores and heuristics from the scoring function.

Discussion

Next, we discuss the implications of software integration for the field of integrative modeling. In integrative modeling, the goal is to build increasingly complex models based on increasingly varied sets of data (Sali, 2021). Key to any modeling is input information, which determines the type of model that can be computed; the choice of model representation is of course also informed by the questions asked of a model. In addition to model representation, a general description of modeling requires a Bayesian posterior model density that specifies the probability density of a model, given the input information, and a scheme for sampling this posterior density. Using Bayes' theorem, the posterior can be factorized into data likelihoods, which depend on data, and priors, which depend on prior information. The evaluation of a posterior based on independent likelihoods and priors is simple in concept, but may be technically challenging; in other words, writing the code that implements the component likelihoods and priors may require a significant amount of effort and expertise that is difficult to duplicate by non-experts and wasteful to duplicate by experts. Thus, software integration can be seen as the major method for maximizing the quality and efficiency of modeling based on varied data and prior models. If the experts encoded their expertise in the code that can be easily mixed and matched, integrative modelers would in turn be able to rigorously combine likelihoods and priors for all the available input information to solve their integrative modeling problems efficiently. To illustrate this point in more detail, we discuss three specific examples of the posterior model density factorization next.

The first example is that used above (Approach), corresponding to computing an atomic multi-state model based on crystallography data and physical principles. Posterior model density is factorized into a data likelihood based on X-ray diffraction patterns and a prior based on a molecular mechanics force field. Consequently, input information is conveniently isolated in the software (IMP for prior models and *Phenix* for diffraction patterns) that is best suited to evaluate a model based on it. The separation of software manages the technical complexity of the prior and likelihood implementation.

The second example is computing a coarse-grained structural model of the hetero-heptameric Nup84 complex based on a negative-stain electron microscopy map and residue-specific cross-links as well as prior models of the subunits(Shi et al., 2014). Posterior model density is factorized into a data likelihood based on the map, a data likelihood based on the cross-links, a prior based on prior subunit models, and a prior based on excluded volume. Although all the terms were evaluated in IMP in this case, it is conceivable that a more accurate encoding of a subset of input information is or will be available in another software package. In such a case, software integration would facilitate computing a higher quality model more efficiently.

The third example is computing a multi-scale model of glucose-stimulated secretion in human pancreatic beta-cells, based on 8 prior models of different aspects of different parts of the entire system(Raveh et al., 2021). These prior models are a coarse-grained spatiotemporal simulation of insulin vesicle trafficking, docking, and exocytosis; a molecular network model of glucose-stimulated insulin secretion signaling; a network model of insulin metabolism; a structural model of glucagon-like peptide-1 receptor activation; a linear model of a pancreatic cell population; and ordinary differential equations for systemic postprandial insulin response. When dealing with a complex multi-scale model, it is often not reasonable to assume independence of the input information. In this case, the prior models must be coupled by additional terms in the scoring function. In addition to the prior models, simple models of statistical coupling between the prior models are also defined. The prior models and the couplers are the priors in a posterior model density for a model of the entire system. Bayesian metamodeling estimates the posterior density via backpropagation. Thus, Bayesian metamodeling decomposes the problem of modeling a large, complex system into smaller, more tractable modeling problems. It is likely that more sophisticated and physically realistic coupling of prior models would be facilitated by software integration, where each type of prior model is evaluated in a separate specialized code developed by experts in the domain of that model.

Conclusions

In summary, software interoperability would greatly benefit the field of integrative modeling as the integration of software specialized for handling specific types of information supports more efficient building of higher quality and more complex models. We proposed a demonstrative standard that facilitates simple software integration by representing modeling as a model search with defined information passing. We then implemented the above standards to integrate *Phenix* and IMP. Finally, we discussed the ability of the Bayesian formulation to facilitate collaborative integrative modeling by mixing-and-matching priors and likelihoods. Ultimately, we suggest that the software development community in the field of integrative modeling consider the definition and adoption of *de facto* protocol standards for improving the interoperability of their software.

Acknowledgments

This work was funded by NIH grants R01 GM083960 (Sali), R01 GM123159 (Fraser), P41 GM109824 (Sali), P01 AG002132 (Sali), P50 AI150476 (Sali), and U19 AI135990 (Sali) as well as NSF grants DBI-1756250 (Sali) and DBI-1832184 (Sali). The development of Phenix is supported by NIH grants P01GM063210 and R24GM141254 to PDA, the Phenix Industrial Consortium, and in part by the US Department of Energy under Contract DE-AC02-05CH11231.

References

Adams, P.D., Afonine, P.V., Bunkóczi, G., Chen, V.B., Davis, I.W., Echols, N., Headd, J.J., Hung, L.-W., Kapral, G.J., Grosse-Kunstleve, R.W., McCoy, A.J., Moriarty, N.W., Oeffner, R., Read, R.J., Richardson, D.C., Richardson, J.S., Terwilliger, T.C., Zwart, P.H., 2010. PHENIX: a comprehensive Python-based system for macromolecular structure solution. Acta Crystallogr. D Biol. Crystallogr. 66, 213–221.

Afonine, P.V., Grosse-Kunstleve, R.W., Echols, N., Headd, J.J., Moriarty, N.W., Mustyakimov, M., Terwilliger, T.C., Urzhumtsev, A., Zwart, P.H., Adams, P.D., 2012. Towards automated crystallographic structure refinement with phenix.refine. Acta Crystallogr. D Biol. Crystallogr. 68, 352–367.

Alber, F., Dokudovskaya, S., Veenhoff, L.M., Zhang, W., Kipper, J., Devos, D., Suprapto, A., Karni-Schmidt, O., Williams, R., Chait, B.T., Rout, M.P., Sali, A., 2007. Determining the architectures of macromolecular assemblies. Nature 450, 683–694.

Berman, H., Henrick, K., Nakamura, H., 2003. Announcing the worldwide Protein Data Bank. Nat. Struct. Biol. 10, 980.

Brunger, A.T., Karplus, M., Petsko, G., 1988. Crystallographic Refinement by Simulated Annealing: Application to Crambin. Acta Crystallogr. A. 64, 61–69.

Burnley, B.T., Afonine, P.V., Adams, P.D., Gros P., 2011. Modeling dynamics in protein crystal structures by ensemble refinement. Elife 1. https://doi.org/10.7554/eLife.00311

Das, R., Baker, D., 2008. Macromolecular modeling with rosetta. Annu. Rev. Biochem. 77, 363–382.

DiMaio, F., Echols, N., Headd, J.J., Terwilliger, T.C., Adams, P.D., Baker, D., 2013. Improved low-resolution crystallographic refinement with Phenix and Rosetta. Nat. Methods 10, 1102–1104.

Dimura, M., Peulen, T.O., Hanke, C.A., Prakash, A., Gohlke, H., Seidel, C.A., 2016. Quantitative FRET studies and integrative modeling unravel the structure and dynamics of biomolecular systems. Curr.

Opin. Struct. Biol. 40, 163–185.

Dominguez, C., Boelens, R., Bonvin, A.M.J.J., 2003. HADDOCK: a protein-protein docking approach based on biochemical or biophysical information. J. Am. Chem. Soc. 125, 1731–1737.

Fleishman, S.J., Leaver-Fay, A., Corn, J.E., Strauch, E.-M., Khare, S.D., Koga, N., Ashworth, J., Murphy, P., Richter, F., Lemmon, G., Meiler, J., Baker, D., 2011. RosettaScripts: a scripting language interface to the Rosetta macromolecular modeling suite. PLoS One 6, e20161.

Fraser, J.S., van den Bedem, H., Samelson, A.J., Lang, P.T., Holton, J.M., Echols, N., Alber, T., 2011. Accessing protein conformational ensembles using room-temperature X-ray crystallography. Proc. Natl. Acad. Sci. U. S. A. 108, 16247–16252.

Grosan, C., Abraham, A., 2011. Intelligent Systems: A Modern Approach. Springer Berlin Heidelberg.

Hsieh, A., Lu, L., Chance, M.R., Yang, S., 2017. A Practical Guide to iSPOT Modeling: An Integrative Structural Biology Platform. Adv. Exp. Med. Biol. 1009, 229–238.

Hua, N., Tjong, H., Shin, H., Gong, K., Zhou, X.J., Alber, F., 2018. Producing genome structure populations with the dynamic and automated PGS software. Nat. Protoc. 13, 915–926.

Hummer, G., Köfinger, J., 2015. Bayesian ensemble refinement by replica simulations and reweighting. J. Chem. Phys. 143, 243150.

Karakaş, M., Woetzel, N., Staritzbichler, R., Alexander, N., Weiner, B.E., Meiler, J., 2012. BCL::Fold--de novo prediction of complex and large protein topologies by assembly of secondary structure elements. PLoS One 7, e49240.

Keedy, D.A., Kenner, L.R., Warkentin, M., Woldeyes, R.A., Hopkins, J.B., Thompson, M.C., Brewster, A.S., Van Benschoten, A.H., Baxter, E.L., Uervirojnangkoorn, M., McPhillips, S.E., Song, J., Alonso-Mori, R., Holton, J.M., Weis, W.I., Brunger, A.T., Soltis, S.M., Lemke, H., Gonzalez, A., Sauter, N.K., Cohen, A.E., van den Bedem, H., Thorne, R.E., Fraser, J.S., 2015. Mapping the conformational landscape of a dynamic enzyme by multitemperature and XFEL crystallography. Elife 4. https://doi.org/10.7554/eLife.07574

Kim, S.J., Fernandez-Martinez, J., Sampathkumar, P., Martel, A., Matsui, T., Tsuruta, H., Weiss, T.M., Shi, Y., Markina-Inarrairaegui, A., Bonanno, J.B., Sauder, J.M., Burley, S.K., Chait, B.T., Almo, S.C., Rout, M.P., Sali, A., 2014. Integrative structure-function mapping of the nucleoporin Nup133 suggests a conserved mechanism for membrane anchoring of the nuclear pore complex. Mol. Cell. Proteomics 13, 2911–2926.

Köfinger, J., Różycki, B., Hummer, G., 2019. Inferring Structural Ensembles of Flexible and Dynamic Macromolecules Using Bayesian, Maximum Entropy, and Minimal-Ensemble Refinement Methods, in: Bonomi, M., Camilloni, C. (Eds.), Biomolecular Simulations: Methods and Protocols. Springer New York, New York, NY, pp. 341–352.

Leaver-Fay, A., Tyka, M., Lewis, S.M., Lange, O.F., Thompson, J., Jacak, R., Kaufman, K., Renfrew, P.D., Smith, C.A., Sheffler, W., Davis, I.W., Cooper, S., Treuille, A., Mandell, D.J., Richter, F., Ban, Y.-E.A., Fleishman, S.J., Corn, J.E., Kim, D.E., Lyskov, S., Berrondo, M., Mentzer, S., Popović, Z., Havranek, J.J., Karanicolas, J., Das, R., Meiler, J., Kortemme, T., Gray, J.J., Kuhlman, B., Baker, D., Bradley, P., 2011. ROSETTA3: an object-oriented software suite for the simulation and design of macromolecules. Methods Enzymol. 487, 545–574.

McCoy, A.J., Grosse-Kunstleve, R.W., Adams, P.D., Winn, M.D., Storoni, L.C., Read, R.J., 2007. Phaser crystallographic software. J. Appl. Crystallogr. 40, 658–674.

Raveh, B., Sun, L., White, K.L., Sanyal, T., Tempkin, J., Zheng, D., Bharat, K., Singla, J., Wang, C., Zhao, J., Li, A., Graham, N.A., Kesselman, C., Stevens, R.C., Sali, A., 2021. Bayesian metamodeling of complex biological systems across varying representations. bioRxiv. https://doi.org/10.1101/2021.03.29.437574

Rout, M.P., Sali, A., 2019. Principles for Integrative Structural Biology Studies. Cell 177, 1384–1403.

Russel, D., Lasker, K., Webb, B., Velázquez-Muriel, J., Tjioe, E., Schneidman-Duhovny, D., Peterson, B., Sali, A., 2012. Putting the pieces together: integrative modeling platform software for structure determination of macromolecular assemblies. PLoS Biol. 10, e1001244.

Sali, A., 2021. From integrative structural biology to cell biology. J. Biol. Chem. 296, 100743.

Schneidman-Duhovny, D., Inbar, Y., Nussinov, R., Wolfson, H.J., 2005. PatchDock and SymmDock: servers for rigid and symmetric docking. Nucleic Acids Res. 33, W363–7.

Schwieters, C.D., Bermejo, G.A., Clore, G.M., 2018. Xplor-NIH for molecular structure determination from NMR and other data sources. Protein Sci. 27, 26–40.

Serra, F., Baù, D., Goodstadt, M., Castillo, D., Filion, G.J., Marti-Renom, M.A., 2017. Automatic analysis and 3D-modelling of Hi-C data using TADbit reveals structural features of the fly chromatin colors. PLoS Comput. Biol. 13, e1005665.

Shi, Y., Fernandez-Martinez, J., Tjioe, E., Pellarin, R., Kim, S.J., Williams, R., Schneidman-Duhovny, D., Sali, A., Rout, M.P., Chait, B.T., 2014. Structural characterization by cross-linking reveals the detailed architecture of a coatomer-related heptameric module from the nuclear pore complex. Mol. Cell. Proteomics 13, 2927–2943.

Terwilliger, T.C., Grosse-Kunstleve, R.W., Afonine, P.V., Moriarty, N.W., Zwart, P.H., Hung, L.W., Read, R.J., Adams, P.D., 2008. Iterative model building, structure refinement and density modification with the PHENIX AutoBuild wizard. Acta Crystallogr. D Biol. Crystallogr. 64, 61–69.

Terwilliger, T.C., Ludtke, S.J., Read, R.J., Adams, P.D., Afonine, P.V., 2020. Improvement of cryo-EM maps by density modification. bioRxiv. https://doi.org/10.1101/845032

Trussart, M., Serra, F., Baù, D., Junier, I., Serrano, L., Marti-Renom, M.A., 2015. Assessing the limits of restraint-based 3D modeling of genomes and genomic domains. Nucleic Acids Res. 43, 3465–3477.

van Zundert, G.C.P., Rodrigues, J.P.G.L.M., Trellet, M., Schmitz, C., Kastritis, P.L., Karaca, E., Melquiond, A.S.J., van Dijk, M., de Vries, S.J., Bonvin, A.M.J.J., 2016. The HADDOCK2.2 Web Server: User-Friendly Integrative Modeling of Biomolecular Complexes. J. Mol. Biol. 428, 720–725.

Watson, J.D., Crick, F.H., 1953. Molecular structure of nucleic acids; a structure for deoxyribose nucleic acid. Nature 171, 737–738.

Williams, C.J., Headd, J.J., Moriarty, N.W., Prisant, M.G., Videau, L.L., Deis, L.N., Verma, V., Keedy,

D.A., Hintze, B.J., Chen, V.B., Jain, S., Lewis, S.M., Arendall, W.B., 3rd, Snoeyink, J., Adams, P.D., Lovell, S.C., Richardson, J.S., Richardson, D.C., 2018. MolProbity: More and better reference data for improved all-atom structure validation. Protein Sci. 27, 293–315.

Credit author statement

Matthew Hancock: Conceptualization, Methodology, Software, Investigation, Formal analysis, Visualization, Writing – Original Draft, Writing – Review & Editing.

Thomas-Otavio Peulen: Conceptualization, Writing- Original draft preparation.

Ben Webb: Software.

Billy Poon*:* Software**.**

James S Fraser: Supervision, Conceptualization, Resources

Paul Adams: Supervision, Conceptualization, Resources

Andrej Sali: Supervision, Conceptualization, Writing- Reviewing and Editing, Resources, Project Administration, Funding Acquisition

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: