Ensembler: Enabling high-throughput molecular simulations at the superfamily scale

Daniel L. Parton,[1] Patrick B. Grinaway,[1] and John D. Chodera[1,*]

[1]*Computational Biology Program, Sloan Kettering Institute,*
Memorial Sloan Kettering Cancer Center, New York, NY 10065

(Dated: March 6, 2015)

The rapidly expanding body of available genomic and protein structural data provides a rich resource for understanding protein dynamics with biomolecular simulation. While computational infrastructure has grown rapidly, simulations on an *omics* scale are not yet widespread, primarily because software infrastructure to enable this has not kept pace. It should now be possible to study protein dynamics across entire (super)families, exploiting the variety of available structural biology data and conformational similarities across homologous proteins. Here, we present a new tool for enabling high-throughput simulation in the genomics era. **Ensembler** takes any set of sequences—from a single sequence to an entire superfamily—and shepherds them through various stages of modeling and refinement to produce simulation-ready structures. This includes comparative modeling to all relevant PDB structures (which may span multiple conformational states of interest), reconstruction of missing loops, addition of missing atoms, culling of nearly identical structures, assignment of appropriate protonation states, solvation in explicit solvent, and refinement with molecular simulation to ensure stable simulation. The output of this pipeline is an ensemble of structures ready for subsequent molecular simulations using computer clusters, supercomputers, or distributed computing projects like Folding@home. **Ensembler** automates much of the time-consuming process of preparing protein models suitable for simulation, while allowing scalability up to entire superfamilies. A particular advantage of this approach can be found in the construction of kinetic models of conformational dynamics—such as Markov state models—which benefit from a diverse array of initial configurations that span the accessible conformational states to aid sampling. We demonstrate the power of this approach by constructing models for all catalytic domains in the human tyrosine kinase family, using all available kinase catalytic domain structures from any organism as structural templates.

Ensembler is free and open source software licensed under the GNU General Public License (GPL) v2. It should run on all major operating systems, and has been tested on Linux and OS X. The latest release can be installed via the `conda` package manager, and the latest source can be downloaded from `https://github.com/choderalab/ensembler`.

Keywords: molecular dynamics simulation; comparative modeling

I. INTRODUCTION

Proteins play a diverse variety of roles in living organisms, and the understanding of their function—and how mutations can cause dysfunction and disease—is the preoccupation of much of modern biology. The diminishing cost of nucleic acid sequencing technologies has produced an enormous wealth of genomic data, yielding a large collection of protein-coding open reading frames that provide basic information about these proteins (at the level of primary amino acid sequences) for numerous organisms [CITE]. Complementing this, large-scale structural biology efforts such as the Protein Structure Initiative (PSI) and Structural Genomics Consortium (SGC) have yielded a great number of protein structures, allowing comparative modeling to provide insight into the static structures many of these proteins adopt [CITE review of structural biology efforts or current comparative modeling?].

Static structures, however, provide only a snapshot of the rich dynamical behavior of proteins. Many functional properties—such as the ability to bind small molecules or interact with signaling partners—often require conformational changes at many levels, from reorganization of sidechains at binding interfaces to loop motions to large scale folding-unfolding events.

Molecular dynamics simulations have proven to be a useful tool for revealing the dynamics of individual proteins, with a number of mature software packages and forcefields available for biomolecular simulation. Advances in computing architectures—especially the recent emergence of GPUs as a technology for providing a hundredfold increase in computational power per unit cost for a variety of applications—and the proliferation of scalable computing technologies (such as distributed computing platforms like Folding@home [CITE], GPUGrid [CITE], and Compernicus [CITE]) now provide unprecedented hardware platforms on which to study the dynamics of these proteins. In parallel, techniques for aggregating molecular dynamics simulation data to survey the conformational and kinetic landscapes of biomolecules, such as Markov state modeling (MSM) approaches [CITE MSM reviews], are now reaching maturity.

Despite this, a critical gap remains in our ability to bridge genome-scale sequence information and molecular simulations to enable the study of entire families or superfamilies of proteins in a single organism or across organisms. Molecular simulations must largely be set up by hand, with little in the way of automation available to provide practitioners a way of studying many members of a family in a manner that exploits their similarity, and especially in cases where only a subset of members may have structural data.

* Corresponding author; john.chodera@choderalab.org

Complicating matters further, in protein families known to be able to adopt multiple conformations—such as kinases—structural data may only exist for one or two conformations for any individual member of the family for which there is structural data. This poses a challenge for biomolecular simulation and analysis methods such as MSMs, which can provide detailed insight but require global coverage of the conformational landscape to realize their full potential.

Here, we present the first steps toward a resolution of this problem: a fully automated open source framework for building simulation-ready protein models scalable from single sequences to entire superfamilies. We demonstrate the utility of this tool by constructing models for the entire set of human tyrosine kinase catalytic domains, and show that the resulting models provide good coverage of known functionally relevant regions of structure space. While this tool was originally constructed to form the foundation for a new era of superfamily-scale molecular simulations for the Folding@home project, we anticipate its utility is far broader.

II. DESIGN AND IMPLEMENTATION

Ensembler is written in Python, and can be used via a command-line tool (`ensembler`) or via a flexible Python API.

The **Ensembler** modeling pipeline entails a series of stages which are performed in a defined order. A visual overview of the pipeline is shown in Fig. 1. The various stages of this pipeline are described in detail below.

[JDC: We could really help the reader if we preface each section here with a bit of an introduction of what we're trying to accomplish in each stage. Otherwise, I worry that each section is a long list of things we do without reference to an overall concept of what the stage is trying to accomplish or why certain decisions were made.]

1. *Target selection*

The `ensembler` command-line tool provides methods for selecting targets from either UniProt (a freely accessible resource for protein sequence and functional data—uniprot.org) or TargetExplorer (a database framework for aggregating various types of biological data; work to be published). [JDC: Does it make sense to omit discussion of TargetExplorer since it's not clear how useful this will be to users?] This tool allows the user to easily select a single protein, many proteins, or an entire superfamily. [JDC: Can you give a clearer picture of how this is done? Does the user just specify different flags for these categories, or would they have to construct different Uniprot searches?] The output from this stage is a FASTA-formatted text file containing the selected target sequences and identifiers [JDC: What kind of identifiers? Arbitrary, Uniprot, etc?], such that the output of other software that produces FASTA-formatted sequence files can be used as input to the next stage.

To select targets from UniProt, a query string must be constructed by the user that conforms to the same syntax as the search function on the UniProt website. [JDC: What command-line flag or API call is used to select targets from UniProt?] For example, `'domain:"Protein kinase" AND taxonomy:9606 AND reviewed:yes'` would select all human protein kinases which have been reviewed by a human curator. **Ensembler** is designed to work with protein *domains*, rather than full-length proteins, and the desired protein domain(s) can be selected using a regular expression. For example, the string `'^Protein kinase(?!; truncated)(?!; inactive)'` would match domains annotated as "Protein kinase", "Protein kinase; 1" or "Protein kinase; 2", but would exclude the domains "Protein kinase; truncated" and "Protein kinase; inactive". The program then performs this search on UniProt, retrieves the data, extracts sequences and identifiers [JDC: UniProt identifiers?], and saves these to a FASTA-format text file.

2. *Template selection*

As for target selection, the `ensembler` tool provides methods for selecting template structures (from which models will be built) from various sources—UniProt [CITE], the Protein Data Bank (PDB; pdb.org) [JDC: Do you use the RCSB or some other PDB site?], or a TargetExplorer database [JDC: Again, not sure if it is helpful to leave this in unless it will be immediately useful to the reader]. From the user perspective, selection of templates from UniProt proceeds in a similar fashion as described above; a used-specified UniProt query string is used to retrieve the list of PDB entries and residue spans associated with each UniProt entry mathing the query. [JDC: Can you be more precise what is meant by "residue span" here?] Structures that include the desired domain [JDC: Can you elaborate on how this is determined?] are retrieved from the RCSB in PDB format. Data from the SIFTS service (www.ebi.ac.uk/pdbe/docs/sifts) (CITE: Velankar Nucleic Acids Res 2013), which provides residue-level mappings between PDB and UniProt entries, is then used to filter out PDB chains with $< 70\%$ resolved residues within the domain span. Template sequences and structures are then extracted and written as FASTA-format sequence and PDB-format coordinate files, respectively.

Selection of template structures from the RCSB simply requires specifying a list of PDB IDs. [JDC: Who specifies the list of PDB IDs? How are they specified? Can this be built automatically, or does the user need to specify these?] These are matched to UniProt entries via the SIFTS service, and the same procedure is then followed to extract template sequences and structures.

Unresolved template loops can optionally be remodeled with a kinematic closure algorithm [CITE], which is provided via the loopmodel tool of the Rosetta software suite (CITE: Rosetta and/or loopmodel). Because shorter loops need to be built by the subsequent model-building stage, prebuilding template loops tends to provide higher-quality models following the subsequent modeling process.

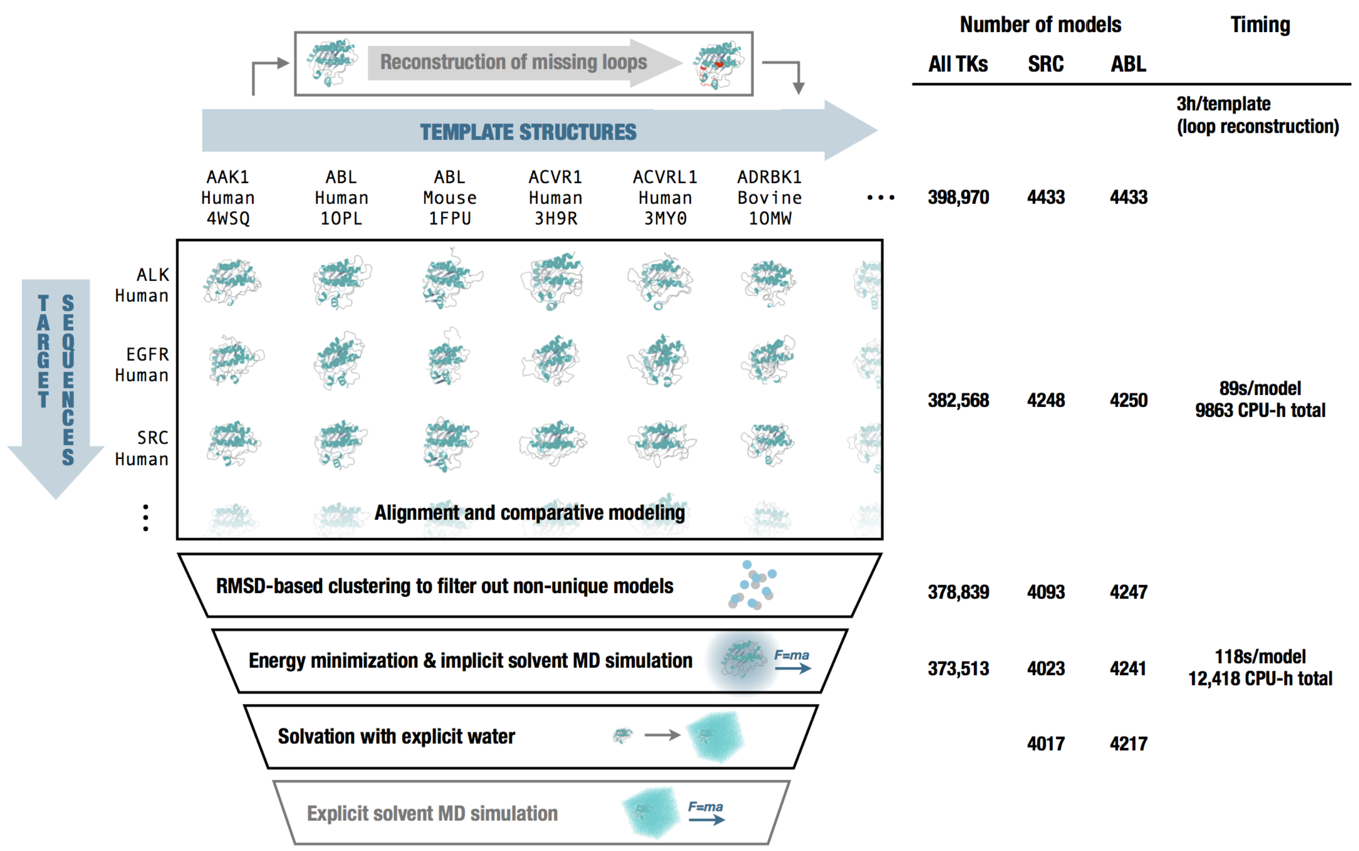
Reconstruction of missing loops
TEMPLATE STRUCTURES
AAK1 Human 4WSQ
ABL Human 1OPL
ABL Mouse 1FPU
ACVR1 Human 3H9R
ACVRL1 Human 3MY0
ADRBK1 Bovine 1OMW
TARGET SEQUENCES
ALK Human
EGFR Human
SRC Human
Alignment and comparative modeling
RMSD-based clustering to filter out non-unique models
Energy minimization & implicit solvent MD simulation
F=ma
Solvation with explicit water
Explicit solvent MD simulation
F=ma
Number of models
All TKs
SRC
ABL
Timing
3h/template (loop reconstruction)
398,970 4433 4433
382,568 4248 4250
89s/model 9863 CPU-h total
378,839 4093 4247
373,513 4023 4241
118s/model 12,418 CPU-h total
4017 4217

FIG. 1. **Diagrammatic representation of the various stages of the Ensembler pipeline.** The number of viable models surviving each stage of the pipeline for are shown, either for all tyrosine kinases (*All TKs*) or representative individual kinases (*SRC* and *ABL*). In addition, the typical timing on a cluster (containing Intel Xeon E5-2665 2.4GHz hyperthreaded processors and NVIDIA GTX-680 or GTX-Titan GPUs) is reported to convey resources required per model and for modeling the entire set of tyrosine kinases. Note that *CPU-h* denotes the number of hours consumed by the equivalent of a single hyperthread—parallel execution can reduce wall clock time nearly linearly.

3. *Modeling*

In this stage, models are generated for each target-template pair, using the Modeller automodel function [CITE: Modeller], which implements comparative structure modeling by satisfaction of spatial restraints [CITE: Sali Blundell J Mol Biol 1993; Fiser Sali Prot Sci 9 2000]. While Modeller can generate alignments automatically, we utilize the BioPython `pairwise2` module (CITE: BioPython)—which uses a dynamic programming algorithm—with the PAM 250 scoring matrix of Gonnet *et al.* [CITE: Gaston Gonnet Science 1992], which we have empirically found to produce better quality alignments for purposes of high-throughput model building.

All chains of template structures that contain the template sequence are utilized in the modeling phase, which can sometimes cause models to be nearly identical. Since the goal is to provide good coverage of conformation space, **Ensembler** filters out nearly identical models using structural similarity-based clustering. The mdtraj [CITE: mdtraj] Python library is used to calculate RMSD with a fast quaternion characteristic polynomial (QCP) [JDC: Cite Theobald QCP papers] implementation, and the leader algorithm is then used to populate clusters. A minimum distance cutoff (which defaults to 0.6 Å) is used to retain only a single model per cluster. [JDC: Which atoms are used in the RMSD comparison? All atoms, heavy atoms, or CA only?]

4. *Refinement*

Models are then refined with a steepest descent energy minimization [JDC: I think OpenMM uses L-BFGS] and a short molecular dynamics (MD) simulation in implicit solvent. This is implemented using the OpenMM molecular simulation toolkit (link and CITE: OpenMM), chosen for its flexible Python API, and high performance GPU-acclerated simulation code. The Amber99SB-ILDN force field is used [CITE: amber99sbildn refs] with a modified generalized Born solvent model (GBSA-OBC) (CITE: GBSA-OBC). The simulation is run for a default of 100 ps to filter out poor quality models (where atomic overlaps that cannot be resolved by energy minimization would cause the simulation to explode) and help relax models for subsequent production simulation. [JDC: What criteria were applied to filter out poor models? Do we only look for thrown exceptions or NaNs? Or do we use an energy filtering criteria too?]

While protein-only models may be sufficient for structural analysis or implicit solvent simulations, **Ensembler** also provides a stage for solvating models with explicit water and performing a round of explicit-solvent MD refinement/equilibration under isothermal-isobaric (NPT) conditions. The solvation step solvates each model for a given target with the same number of waters to facilitate the integration of data from multiple simulations, such as the construction of MSMs. The target number of waters is selected by first solvating each model with a specified padding distance (default: 10 Å), then taking a percentile value from the distribution (default: 68th percentile). [JDC: Would be useful to explain why we are doing this.] Models are resolvated with the target number of waters by first solvating with zero padding, then incrementally increasing the box size and resolvating until the target is exceeded, then finally deleting sufficient waters to match the target value. The explicit solvent MD simulation is also implemented using OpenMM, with the Amber99SB-ILDN force field and TIP3P water [JDC: CITE]. [JDC: We should allow other water models in OpenMM too, such as TIP4P-Ew?]

[JDC: In the Discussion, let's be sure to talk about the limitations and what could be improved or added in the future. For example, we don't yet handle counterions (e.g. structural Zn^{2+}), prosthetic groups (e.g. heme), or cofactors (e.g. ATP) yet. We don't handle post-translational modifications either (such as phosphorylation, methylation, glycosylation, etc.). It's a good idea to suggest that this is an important first step toward enabling superfamily- and genomics-scale modeling, but there's a lot of work yet to be done.]

5. *Packaging*

Ensembler provides a packaging module which can be used to compress models in preparation for data transfer, or to prepare models with the appropriate directory and file structure for subsequent production simulations on the distributed computing platform Folding@home (CITE: F@H).

6. *Provenance*

To aid the user in tracking the provenance of each model, each pipeline function also outputs a metadata file, which helps to link data to the software version used to generate it (both **Ensembler** and its dependencies), and also provides timing and performance information, and other data such as hostname.

7. *Rapidly modeling a single template*

For users interested in simply using **Ensembler** to rapidly generate a set of models for a single template sequence, **Ensembler** provides a command-line tool `quickmodel`, which performs the entire pipeline for a single target with a small number of templates. For larger numbers of models (such as entire protein families), modeling time is greatly reduced by using the main modeling pipeline parallelized via MPI, which distributes computation across each model (or across each template, in the case of the loop reconstruction code), scaling (in a "pleasantly parallel" manner) up to the number of models generated.

III. RESULTS

IV. AVAILABILITY AND FUTURE DIRECTIONS

V. ACKNOWLEDGMENTS

The authors are grateful to Kyle A. Beauchamp (MSKCC), Robert McGibbon (Stanford), Arien S. Rustenburg (MSKCC) for many excellent software engineering suggestions. The authors thank Sonya M. Hanson (MSKCC), Nicholas M. Levinson (?), Markus A. Seeliger (Stony Brook), Diwakar Shukla (Stanford), and Avner Schlessinger (Mount Sinai) for helpful scientific feedback on modeling kinases. The authors are grateful to Benjamin Webb and Andrej Sali (UCSF) for help with the MODELLER package, Peter Eastman and Vijay Pande (Stanford) for assistance with OpenMM, and Marilyn Gunner (CCNY) for assistance with MCCE2. DLP and this work was supported in part by the generous support of a Louis V. Gerstner Young Investigator Award.