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 Center, Memorial Sloan Kettering Cancer Center, New York, NY 10065*

(Dated: February 6, 2015)

The rapidly expanding body of available genomic and protein structural data provides a rich resource for the field of biomolecular simulation. However, simulations on an omics scale are not yet widely performed, partly because software has had trouble keeping pace. For example, it should be possible to study proteins across entire (super)families, and to do so in a way which exploits the entire variety of available structural biology data. 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 of interest - and shepherds them through various stages of: comparative modeling to all relevant PDB structures, reconstruction of missing loops, addition of missing atoms, culling by close structural similarity, assignment of protonation states, solvation, and refinement with molecular simulation. The output is an ensemble of structures ready for subsequent parallel or distributed molecular simulations. This automates much of the time-consuming process of preparing protein models suitable for simulation, while also allowing this process to be scaled to the superfamily scale. A particular advantage of this approach can be found in the construction of kinetic models of conformational dynamics - such as Markov state models - for which a diverse array of starting configurations is expected to aid sampling. We demonstrate the power of this approach by constructing initial models for all catalytic domains in the human tyrosine kinase family, using all kinase catalytic domain structures from any organism as structural templates. Ensembler should run on all major operating systems, and has been tested on Linux and OS X. The program is free of charge, and is made available under the terms of the GNU General Public License (GPL) v2. The latest release can be installed via the conda package manager, and the latest source can be downloaded from `https://github.com/choderalab/ensembler`.

I. INTRODUCTION

II. DESIGN AND IMPLEMENTATION

Ensembler is written in Python, and can be used via a command-line tool (`ensembler`), or via the 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, and a detailed description follows.

1. *Target selection*

The output from this stage is simply a FASTA-formatted sequence file containing the selected target sequences and identifiers. 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). This allows the user to easily select a single protein, many proteins, or an entire superfamily. Alternatively, the targets file can be generated using any other software, and stored at the appropriate filepath.

The method for selecting targets from UniProt is described here. A query string is required as input, using the same syntax as the search function on the UniProt website. 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 extracts sequences and identifiers from the UniProt data, and saves these to a FASTA-format text file.

2. *Template selection*

As for target selection, the `ensembler` tool provides methods for selecting templates from various resources—UniProt, the Protein Data Bank (PDB; pdb.org) or a TargetExplorer database. From the user perspective, selection from UniProt proceeds in a similar fashion as described above. The returned data for each UniProt entry includes an up-to-date list of PDB structures and their residue spans. PDB files are downloaded from the PDB for structures which in-

* Corresponding author; john.chodera@choderalab.org

clude the desired domain. 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 from the PDB simply requires specifying a list of PDB IDs. 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, which is provided via the loopmodel tool of the Rosetta software suite (CITE: Rosetta and/or loopmodel). This tends to provide higher-quality models following the subsequent modeling process.

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). Modeller requires the user to first provide a target-template sequence alignment. This is implemented in Ensembler using 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).

Non-unique models are then filtered out using structural similarity-based clustering. The mdtraj (CITE: mdtraj) Python library is used to calculate RMSD with a fast quaternion characteristic polynomial (QCP) implementation, and the leader algorithm is then used to populate clusters. A minimum distance cutoff (default: 0.6 Å) is used to retain only a single model per cluster.

4. *Refinement*

Models are then refined with a steepest descent energy minimization and a short molecular dynamics (MD) simulation with 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. This refinement process helps to prepare models for subsequent production simulation, and also helps to filter out poor quality models.

Ensembler also provides optional routines for solvating models with explicit solvent and performing a second MD refinement. The solvation step solvates each model for a given target with the same number of waters, as this is (currently) a requirement for building MSMs from multiple independent MD trajectories. 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). 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.

5. *Packaging*

Finally, 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. *Other features*

The command-line tool also provides a `quickmodel` function, which performs the entire Ensembler pipeline for a single target with a small number of templates. For larger numbers of models (such as entire protein families), the main pipeline functions should be used. The modeling and refinement functions use MPI to trivially parallelize computation across each model (or across each template, in the case of the loop reconstruction code).

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.

III. RESULTS

IV. AVAILABILITY AND FUTURE DIRECTIONS

V. ACKNOWLEDGMENTS

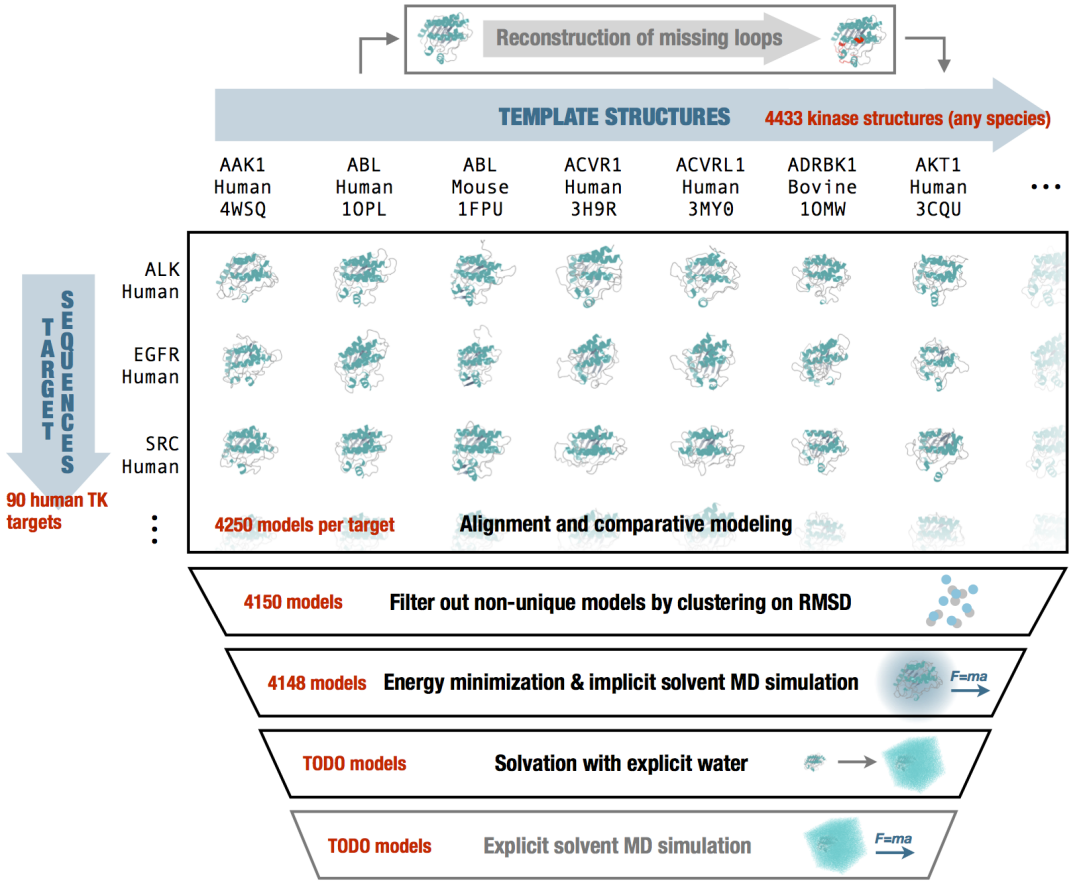

FIG. 1. Ensembler pipeline