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 14, 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 simulations at this scale have not kept pace. It should now be possible to study protein dynamics across entire (super)families, exploiting both 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 and filtering 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. [JDC: Prior sentence is redundant?] 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; distributed simulation

I. INTRODUCTION

Recent advances in genomics and structural biology have helped generate an enormous wealth of protein data at the level of amino-acid sequence and three-dimensional structure. However, proteins typically exist as an ensemble of thermally accessible conformational states, and static structures provide only a snapshot of their rich dynamical behavior. Many functional properties—such as the ability to bind small molecules or interact with signaling partners—require transitions between states, encompassing anything from reorganization of sidechains at binding interfaces to domain motions to large scale folding-unfolding events. Drug discovery could also benefit from a more extensive consideration of protein dynamics, whereby small molecules might be selected based on their predicted ability to bind and trap a protein target in an inactive state [1].

Molecular dynamics (MD) simulations have the capability, in principle, to describe the time evolution of a protein in atomistic detail, and have proven themselves to be a useful tool in the study of protein dynamics. A number of mature software packages and forcefields are available, and much recent progress has been driven by advances in computing architecture. For example, many MD packages are now able to exploit GPUs, which provide greatly improved simulation efficiency per unit cost relative to CPUs, while distributed computing platforms such as Folding@home [CITE], GPUGrid [CITE], and Copernicus [CITE] allow scalability on an unprecedented level. In parallel, methods for building human-understandable models of protein dynamics from noisy simulation data, such as Markov state modeling (MSM) approaches, are now reaching maturity [CITE MSM reviews]. MSM methods in particular have the advantage of being able to aggregate data from multiple independent MD trajectories, facilitating parallelization of production simulations and thus greatly alleviating overall computational cost. There also exist a number of mature software packages for comparative modeling of protein structures, in which a target protein sequence is modeled using one or more structures as templates [CITE Modeller and Rosetta and a recent homology modeling review].

However, it remains difficult for researchers to exploit the full variety of available protein sequence and structural data in simulation studies, largely due to limitations in software architecture. For example, the set up of a biomolecular simulation is typically performed manually, encompassing a series of fairly standard (yet time-consuming) steps such as the choice of protein sequence construct and starting struc-

* Corresponding author; john.chodera@choderalab.org

ture, addition of missing residues and atoms, solvation with explicit water and salt buffer, choice of simulation parameters, and system relaxation with energy minimization and one or more short MD simulations. For this reason, simulation studies typically consider only one or a few proteins and starting configurations.

The ability to fully exploit the large base of available protein sequence and structural data in biomolecular simulation studies could open up many interesting avenues for research, enabling the study of entire protein families or superfamilies across multiple organisms. The similarity between members of a given protein family could be exploited to generate arrays of conformational models, which could be used as starting configurations to aid sampling in MD simulations. This approach would be highly beneficial for many MD methods, such as MSM construction, which require global coverage of the conformational landscape to realize their full potential, and would also be particularly useful in cases where structural data is present for only a subset of the members of a protein family. It would also aid in studying protein families known to have multiple metastable conformations—such as kinases—for which the combined body of structural data for the family may cover a large range of these conformations, while the available structures for any individual member might encompass only one or two distinct conformations.

Here, we present the first steps toward bridging the gap between biomolecular simulation software and *omics*-scale sequence and structural data: a fully automated open source framework for building simulation-ready protein models in multiple conformational substates scalable from single sequences to entire superfamilies. **Ensembler** provides functions for selecting target sequences and homologous template structures, and (by interfacing with a number of external packages) performs pairwise alignments, comparative modeling of target-template pairs, and several stages of model refinement. As an example application, we have constructed models for the entire set of human tyrosine kinase catalytic domains, using all available structures of protein kinase domains (from any species) as templates. This results in a total of almost 400,000 models, and we demonstrate that these provide wide-ranging coverage of known functionally relevant conformations. By using these models as starting configurations for highly parallel MD simulations, we expect their structural diversity to greatly aid in sampling of conformational space. We anticipate that the tool will prove to be useful in a number of other ways. For example, the generated models could represent valuable data sets even without subsequent production simulation, allowing exploration of the conformational diversity present within the available structural data for a given protein family. Furthermore, the automation of simulation set up provides an excellent opportunity to make concrete certain "best practices", such as the choice of simulation parameters.

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 comprises 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.] [DLP: Good point. I've added in brief introductions for each section.] [JDC: Can you do a bit more here? I feel that the reader may need more orientation. You've essentially just added a sentence or two at the beginning of each stage that doesn't really enlighten the user as to your terminology (*tempalte* and *target*), your motivation for *why* things are done each stage, or *how* the user is to select among the various options available.]

1. *Target selection*

The first stage entails the selection of a set of target protein sequences. [JDC: Maybe explain what is meant by "target protein sequences"? These are the sequences the user is interested in modeling.]

These targets can be defined manually, simply by providing a FASTA-formatted text file containing the desired target sequences with arbitrary identifiers. The `ensembler` command-line tool also allows targets to be selected from UniProt—a freely accessible resource for protein sequence and functional data (uniprot.org) [JDC: Isn't there a real citation for UniProt?], using the subcommand `gather_targets`. The user specifies a query string with the `--query` flag, which conforms to the same syntax as the search function available on the UniProt website. For example, `--query 'mnemonic:SRC_HUMAN'` would select the full-length human Src sequence, while `--query 'domain:"Protein kinase" AND taxonomy:9606 AND reviewed:yes'` would select all human protein kinases which have been reviewed by a human curator. In this way, the user may select a single protein, many proteins, or an entire superfamily. The program outputs a FASTA file, setting the UniProt mnemonic (e.g. `SRC_HUMAN`) as the identifier for each target protein.

In many cases, it will be desirable to build models of an isolated protein domain, rather than the full-length protein. The `gather_targets` subcommand allows protein domains to be selected from UniProt data by passing a regular expression string to the `--domains` flag. For example, the above `--query` flag for selecting all human protein kinases returns UniProt entries with domain annotations including "Protein kinase", "Protein kinase 1", "Protein kinase 2", "Protein kinase; truncated",

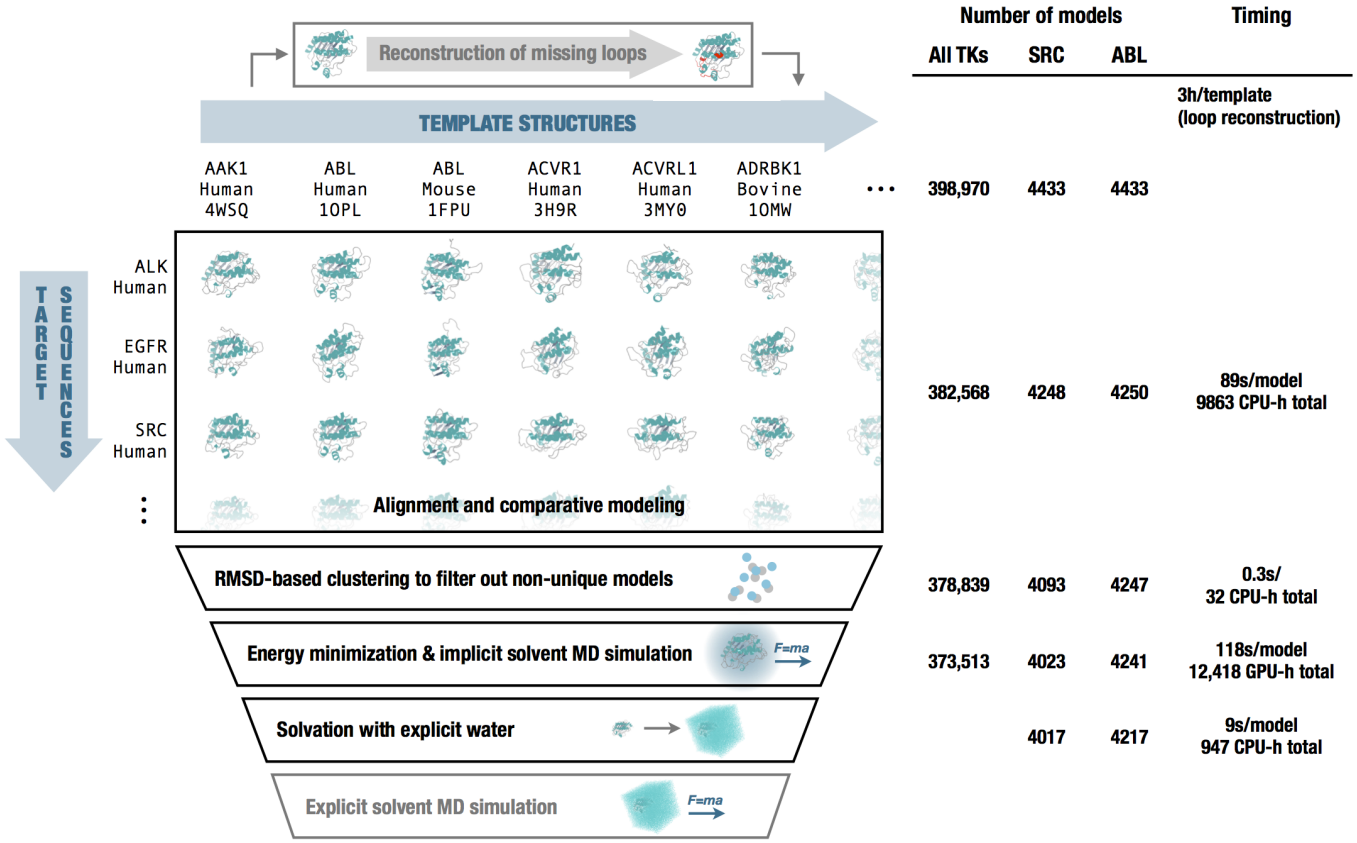


FIG. 1. Diagrammatic representation of the various stages of the Ensembler pipeline. The number of viable models surviving each stage of the pipeline are shown, either for all tyrosine kinases (*All TKs*) or representative individual kinases (*SRC* and *ABL*). In addition, the typical timing on a cluster (containing Intel Xeon E5-2665 2.4GHz hyperthreaded processors and NVIDIA GTX-680 or GTX-Titan GPUs) is reported to exemplify the 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.

"Protein kinase; inactive", "SH2", "SH3", etc. To select only domains of the first three types, the following regular expression could be used: `'^Protein kinase(?!; truncated)(?!; inactive)'`. In this case, target identifiers are set with the form `[UniProt mnemonic]_D[domain index]`, where the latter part represents a 0-based index for the domain—necessary because a single target protein may contain multiple domains of interest. Example identifiers: `JAK1_HUMAN_D0`, `JAK1_HUMAN_D1`. [JDC: Does it make sense to set some of these coded examples off on their own lines?]

2. *Template selection*

The second stage entails the selection of templates and storage of associated structures, sequences and identifiers. [JDC: Again, can you provide more information about *why* this is being done? What the motivation is, and how the user might expect to select these?]

This data can be provided manually, by storing the sequences and identifiers in a FASTA file, and the structures as PDB-format coordinate files with filenames matching the identifiers in the sequence file. The structure residues must also match those in the sequence file.

The `ensembler gather_templates` subcommand also provides methods for selecting template structures from either UniProt or the Protein Data Bank (PDB;), specified by the `--gather_from` flag. Both methods select templates at the level of PDB chains—a PDB structure containing multiple chains with identical sequence spans (e.g. for crystal unit cells with multiple asymmetric units) would thus give rise to multiple template structures.

Selection of templates from the PDB simply requires passing a list of PDB IDs as a comma-separated string, e.g. `--query 2H8H,1Y57`. Specific PDB chain IDs can optionally also be selected via the `--chainids` flag. The program retrieves structures from the PDB server, as well as associated 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. The SIFTS data is used to extract template sequences, retaining only residues which are resolved and match the equivalent residue in the UniProt sequence—non-wildtype residues are thus removed from the template structures. Furthermore, PDB chains with less than a given percentage of resolved residues (default: 70%)

are filtered out. Sequences are stored in a FASTA file, with identifiers of the form `[UniProt mnemonic]_D[UniProt domain index]_[PDB ID]_[PDB chain ID]`, e.g. `SRC_HUMAN_D0_2H8H_A`. Matching residues then extracted from the original coordinate files and stored as PDB-format coordinate files.

Selection of templates from UniProt proceeds in a similar fashion as for target selection; the `--query` flag is used to select full-length proteins from UniProt, while the optional `--domains` flag allows selection of individual domains with a regular expression string. The returned UniProt data for each protein includes a list of associated PDB chains and their residue spans, and this information is used to select template structures, using the same method as for template selection from the PDB. If the `--domains` flag is used, then templates are truncated at the start and end of the domain sequence.

Unresolved template residues can optionally be remodeled with the `loopmodel` subcommand, which employs a kinematic closure algorithm [CITE] provided via the `loopmodel` tool of the Rosetta software suite (CITE: Rosetta and/or loopmodel). Because fewer loops need to be built during the subsequent model-building stage, prebuilding template loops tends to provide higher-quality models after completion of the **Ensembler** pipeline. Loop remodeling may fail for a small proportion of templates due to spatial constraints imposed by the original structure; the subsequent modeling step thus automatically uses the remodeled version of a template if available, but otherwise falls back to using the non-remodeled version. Furthermore, the Rosetta `loopmodel` program will not model missing residues at the termini of a structure—such residues spans are modeled in the subsequent stage.

3. Modeling

This stage entails the generation of models via comparative modeling of each target sequence onto each template structure. Non-unique models are filtered out using a RMSD-based clustering scheme.

Modeling is performed with 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 (for Cα atoms only) with a fast quaternion characteristic polynomial (QCP) [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.

4. Refinement

This stage entails the use of molecular dynamics simulations to refine the models built in the previous step. This helps to improve model quality and also prepares models for subsequent production simulation, including solvation with explicit water molecules, if desired.

Models are first subjected to energy minimization (using the L-BFGS algorithm [CITE]), followed by a short molecular dynamics (MD) simulation with an implicit solvent representation. 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. By default, the Amber99SB-ILDN force field is used [CITE: amber99sbildn refs] with a modified generalized Born solvent model (GBSA-OBC) (CITE: GBSA-OBC). The **Ensembler** API allows the use of any of the other force fields implemented in OpenMM. 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?] [DLP: We currently just filter out models which throw exceptions or NaNs.]

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.] [DLP: Addressed.] This helps to prevent models with particularly long, extended loops—such as those arising from template structures with unresolved termini—from imposing very large box sizes on the entire set of models. 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, using the Amber99SB-ILDN force field and TIP3P water [JDC: CITE] by default. Other force fields or water models such as TIP4P-

Ew [CITE]) can be specified via the **Ensembler** API. [JDC: We should allow other water models in OpenMM too, such as TIP4P-Ew?] [DLP: I forgot to mention this in the text previously - any of the OpenMM force fields can be chosen via the API. I've updated the text accordingly. Is this functionality sufficient? I guess it's ok to leave ff choice as an "advanced" feature which requires use of the API? Otherwise I could add a --water_model flag to the CLI, for example.]

[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, which is parallelized via MPI, distributing computation across each model (or across each template, in the case of the loop reconstruction code), and scaling (in a "pleasantly parallel" manner) up to the number of models generated.

III. RESULTS

[JDC: It would be useful to have some subheadings in this section to give it some internal organization.]

As a first application of **Ensembler**, we have built models for all 90 human tyrosine kinase (TK) domains listed in UniProt. [JDC: Is there a complete list of these somewhere? Maybe reference supplementary data?] TKs (and protein kinases in general) play important roles in many cellular processes and are involved in a number of types of cancer. [JDC: CITE] For example, mutations of Src are associated with colon, breast and prostate cancer [CITE: Src cancer involvement], while a translocation between the TK Abl and the pseudokinase Bcr is closely associated with chronic myelogenous leukemia [CITE: Abl cancer involvement]. Protein kinase domains are thought to have multiple accessible metastable conformation states, with a single active conformation, and much effort is directed at developing kinase inhibitor drugs which bind to and stabilize inactive conformations [CITE: Lee and Craik Science 2009]. [JDC: Lee and Craik do not discuss kinases, I don't believe; you'll have to find an accurate reference on kinase conformations.] Kinases are thus a particularly interesting subject for study with MSM methods [CITE: recent kinase MSM papers], and this approach stands to benefit greatly from the ability to exploit the full body of available genomic and structural data within the kinase family, e.g. by generating large numbers of starting configurations to be used in highly parallel MD simulation.

We selected all available structures of protein kinase domains (of any species) as templates, for a total of 4433 (398,970 target-template pairs). The templates were derived from 3028 individual PDB entries and encompassed 23 different species, with 3634 template structures from human kinase constructs.

The templates were first subjected to loop remodeling, using the `loopmodel` subcommand. Out of 3666 templates with one or more missing residues, 3134 were successfully remodeled. [JDC: Can you give some statistics on the distribution of loop lengths modeled? Why did loop modeling fail in the cases it did? Anything else you can say here beyond this one sentence?]

Following loop remodeling, the **Ensembler** pipeline was performed up to and including the implicit solvent MD refinement stage, which completed with 373,513 surviving models. In addition, the solvation stage was performed for two representative individual kinases (*Src* and *Abl*). The number of models which survived each stage are shown in Fig. 1, indicating that the greatest attrition occurred during the modeling stage. Fig. 1 also indicates the typical timing achieved on a cluster for each stage.

The distribution of RMSDs of the final models (relative to the highest sequence identity model for a given target) is shown in Fig. 3. The distributions are stratified based on the sequence identity between target and template, indicating that higher sequence identity templates result in lower RMSD models. The sequence identity stratifications were selected based on the sequence identity distribution (Fig. 2), which suggests an intuitive division into three categories, with 307,753 models in the 0-35% sequence identity range, 69,922 models in the 35-55% range, and 4893 models in the 55-100% range.

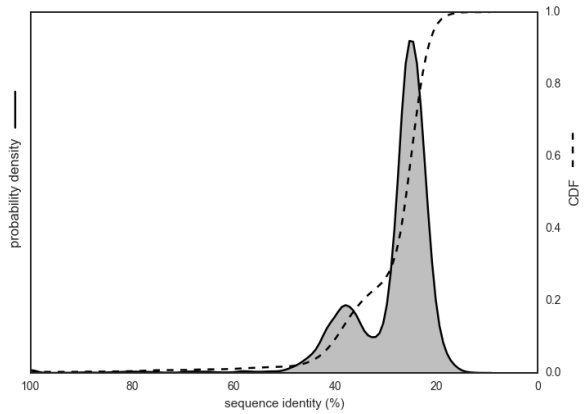


FIG. 2. Sequence identity distribution for human TK models. Distribution of sequence identities for all 373,513 models generated for the human tyrosine kinases. [DLP: should I mention the use of KDE smoothing?] [JDC: Yes.] Sequence identities are calculated from pairwise target-template alignments. The cumulative distribution function is shown by the dashed line. [JDC: Font size too small.]

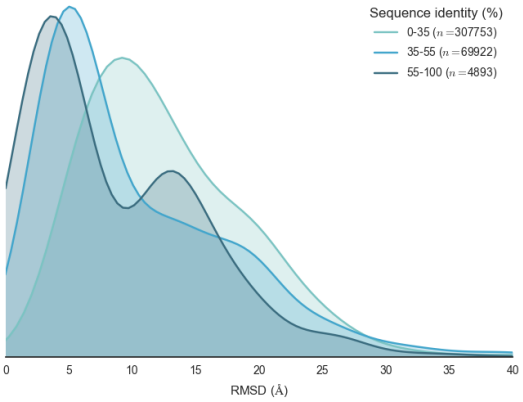


FIG. 3. RMSD distribution by sequence identity. RMSD distributions for all 373,513 human TK models, divided into three sequence identity ranges. For a given target, model RMSDs are calculated relative to the highest sequence identity model for that target. [JDC: Font size too small.]

IV. AVAILABILITY AND FUTURE DIRECTIONS

The latest release of **Ensembler** can be installed via the `conda` package manager for Python [**?**].

```
# conda install -c https://conda.binstar.org/omnia ensembler
```

Up to date instructions can be found at https://github.com/choderalab/ensembler. This will install all dependencies except for Modeller and (optionally) Rosetta, which are not available through the `conda` package manager, and thus must be installed separately by the user. The latest source can be downloaded from the above GitHub repository, which also contains instructions for building and installing the code.

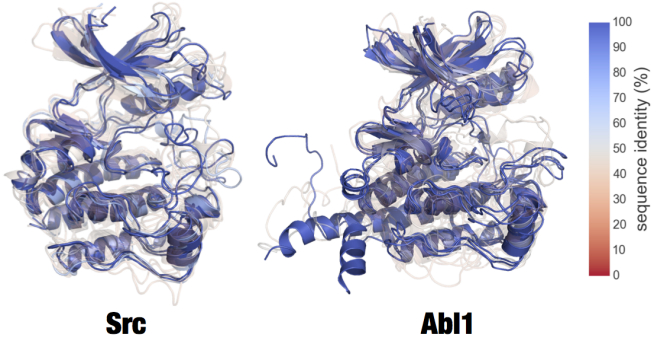
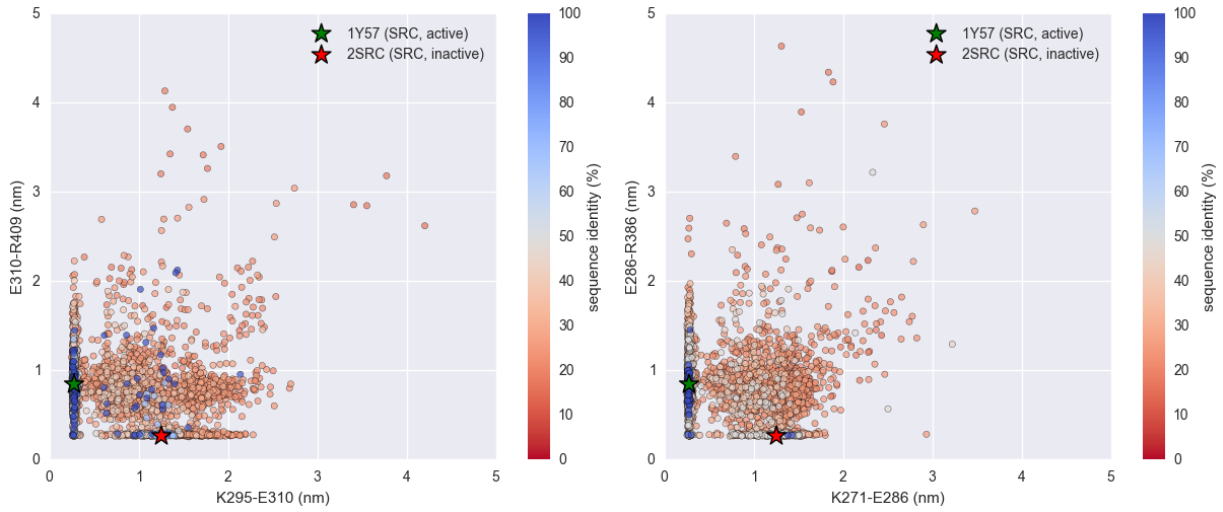


FIG. 4. Superposition of clustered models of Src and Abl. Superposed renderings of nine models each for Src and Abl, [JDC: Src and Abl, or Src and Abl1? The description should match the captions above.] giving some indication the diversity of conformations generated by Ensembler. The models for each target were divided into three sequence identity ranges (as in Fig. 3), and RMSD-based k-medoids clustering was performed to select three clusters from each. The models shown are the centroids of each cluster. Models are colored and given transparency based on their sequence identity, so that high sequence identity models are blue and opaque, while lower sequence identity models are transparent and red. [JDC: Font size too small.]

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.

[1] G. M. Lee and C. S. Craik, Science **324**, 213 (2009).



(a) Src (b) Abl

FIG. 5. E310-R409 and K295-E310 distances for models of Src and Abl, colored by sequence identity. [JDC: Font size too small if single-column, so I made this double-column. Not sure if that's what you want.] [JDC: Fill in rest of caption.]

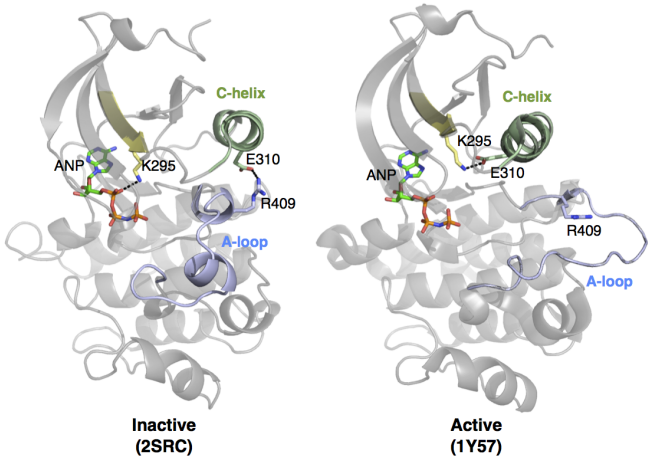


FIG. 6. Two structures of Src, indicating certain residues involved in activation. In the inactive state, E310 forms a salt bridge with R409. During activation, the C-helix (green) moves and rotates, orienting E310 towards the ATP-binding site and allowing it to instead form a salt bridge with K295. This positions K295 in the appropriate position for catalysis.