VOLUME FOUR

COMPUTATIONAL CRYSTALLOGRAPHY NEWSLETTER

JULY MMXIII

ENSEMBLE REFINEMENT, CABLAM

Table of Contents

Editor

Nigel W. Moriarty, NWMoriarty@LBL.Gov

PHENIX News

New programs

FEM: Feature Enhanced Maps (Pavel V. Afonine)

Interpretation of a crystallographic map is a means of obtaining an atomic representation of a crystal structure or the map itself may serve as the crystal model. There are number of factors that affect quality of crystallographic maps that in turn affect difficulty (or even feasibility) of their interpretation and quality of resulting model of crystal structure, and include:

- finite resolution of measured reflections;
- incompleteness of data (missing reflections within the resolution range of the measured data);
- experimental errors in measured reflections;
- errors in atomic model parameters.

These factors a) result in artificial peaks in the map that may be confused with the signal and therefore erroneously interpreted in terms of atomic model, b) introduce noise that may obscure the signal and c) may distort the signal in various ways.

Another fundamentally different contributor to the difficulty of map interpretation is that not all the signal has the same strength. For example, a strong signal arising from a heavy atom derivative may easily obscure a very weak signal (that may be at or below the noise level) arising from a partially occupied very mobile ligand or residue side chain alternative conformation or even hydrogen atoms.

The combination of the map artifacts arising from data and model errors with the signal variation problems makes map interpretation for macromolecules never unambiguous, trivial or unique.

A procedure is being developed that produces a map with less noise and stronger weak signal. In a nutshell it consists of two steps of map and Fourier map coefficients modification:

- *Eliminate noise as much as possible*. This is done by perturbing map just enough to shuffle the noise but leave the signal unchanged or changed insignificantly. Repeating such perturbation many times and combing resulting maps (by averaging or intersection, or both) is expected to reduce the artifacts but retain the signal. This is based on the Dusan Turk's idea of kick maps calculation, but takes it further by not only "kicking" atomic coordinates but also perturbing all possible variables that pertain to map calculation, such as data completeness, ways of filling missing F_{obs}, calculation of m and D for $2mF_{obs}$-DF_{model} map, varying components that go into Fmodel calculation, many trials of random elimination of largest F_{obs}-F_{model} outliers and more.
- *Equalize weak and strong signal*: equalize signal in the map such that strong and weak signal become of a similar strength. This is a known procedure of local map scaling used in some crystallographic applications (density modification and model building), and routinely used in digital image processing for local contrast improvement. There are multiple ways of doing this: from scaling map by standard deviation locally to local map histogram equalization. We are still investigating which of possible options performs best in this context.

More details, including usage instructions, can be found at www.phenix-online.org -> Presentations -> Feature Enhanced Map

The final algorithm and implementation details will be published elsewhere.

New tool: *phenix.real_space_refine*

An announcement of a new tool is included in the short communications section.

New features

New REEL features

Editing molecules and generating restraints has been simplified in REEL. The ability to add and delete atoms is now available on the toolbar. Naturally it is best to load a restraints CIF file (not a Chemical Components data CIF file) as the starting point and run *eLBOW* on the resulting molecule to generate the best possible restraints.

The ability to view the restraints actually used in a refinement has been added. By default, the bonding of a molecule loaded from a PDB file is performed using a distance-based approach. Calling REEL with both a PDB file and a geometry (`.geo`) file output from `phenix.refine` (or other restraints based program) will use the actual restraints defined and allow viewing of actual and ideal values.

Crystallographic meetings and workshops

Gordon Research Conference on Diffraction Methods in Structural Biology: Towards Integrative Structural Biology, Gordon Research Seminar, Bates College, Lewiston, ME, July 26-27, 2014

A seminar series preceding the GRC meeting.

Biophysical Society 58th Annual Meeting, February 15-19, 2014

An IYCr2014 symposium entitled "Celebrating 100 Years of Crystallography: X-Rays Are Photons Too" will be of interest to all crystallographers. This year the annual Biophysics101 session will be on tips for biophysicists who want to use the crystallographic technique. Check the website for details as the date approaches.

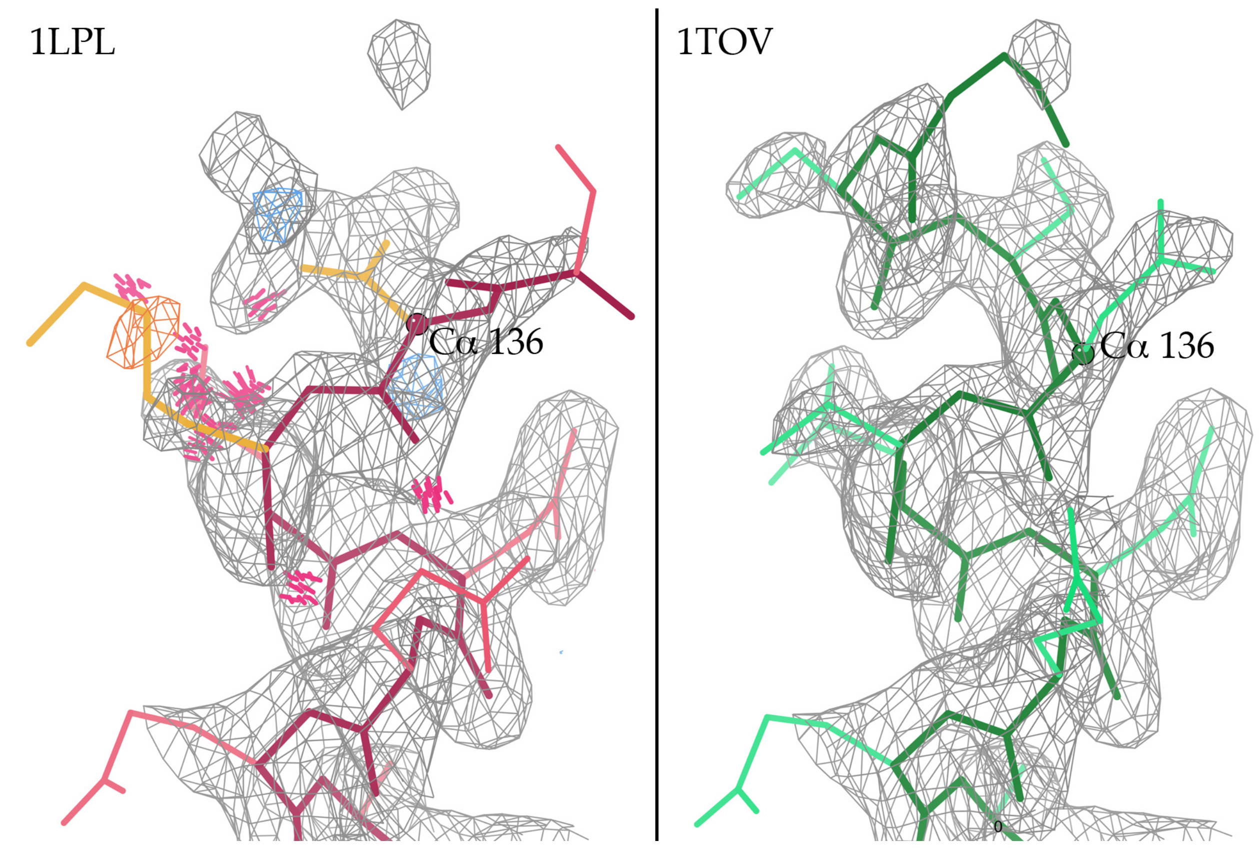


Figure 1. Example of CAP-Gly domain at 1.77Å resolution from 1LPL.

Gordon Research Conference on Diffraction Methods in Structural Biology: Towards Integrative Structural Biology, Gordon Research Conference, Bates College, Lewiston, ME, July 27-August 1, 2014

A very interesting and important meeting for protein crystallographers.

Expert advice

Fitting Tip #6 – Potential misfitting by a switch of sidechain vs mainchain

Jane Richardson and Bryan Arendall, Duke University

One thing that can sometimes go wrong in fitting a protein model is to put a sidechain into what should be backbone density and vice-versa. This usually happens either near a chain terminus or when coming in or out of a disordered loop. This is incorrect, of course, and will usually produce poor geometry even if that residue is really the chain terminus. But it is even more serious if there should have been more residues, since they will either be omitted or wrong because there is not connected density to fit more backbone out from the end of what should really have been the sidechain.

At a chain terminus: Figure 1 shows an example from the 1LPL CAP-Gly domain at 1.77Å resolution (Li 2002), where residues are missing at the N-terminus because of such a sidechain-mainchain switch at the start of an α-helix. The sidechain of Asp 136 was fit into electron density that is really helical backbone, and the mainchain can therefore only continue as far as the end of the Asp sidechain. The Cα of 136 is labeled and marked with a ball in both panels. The left panel shows model and map from the original 1LPL structure, where the fact that something is wrong can be diagnosed by all-atom steric clashes, difference-density peaks, and poor rotamers. The sidechain and mainchain were switched, producing a map that showed density for another residue; after 3 cycles of rebuilding all validation outliers were gone and nearly a full extra turn of helix was fit at the N-terminus (right panel). This change was the largest of several that gave a 4% drop in Rfree, redeposited as 1TOV (Arendall 2005).

Next to a disordered loop: Any transition between well-ordered structure and a disordered region is equivalent to a chain end, so sidechain-mainchain switches also occur at the ends of loops that are partially or fully disordered. An example is shown in Figure 2, from the 1INP inositol polyphosphate 1-phosphatase at 2.3Å resolution (York 1994). The top panel shows the 97-102 loop in 1INP as deposited, in its discontinuous and ambiguous electron density. That loop also has 4 steric clashes, 4 bond-angle outliers, a Ramachandran outlier, and a poor rotamer (not flagged in the figure). The center panel shows the switch of sidechain vs mainchain around the Cα of Glu 102, and the bottom panel shows the rebuilt loop in its improved density, with validation outliers corrected.

Such problems can occur even at fairly high resolution, because both people and software tend to treat backbone and sidechains as separate fitting tasks. If you want to try rebuilding a sidechain-mainchain switch, practice on 1LPL Asp136, 1INP Glu102, or 1BKR Met109 (harder - probably has two conformations).

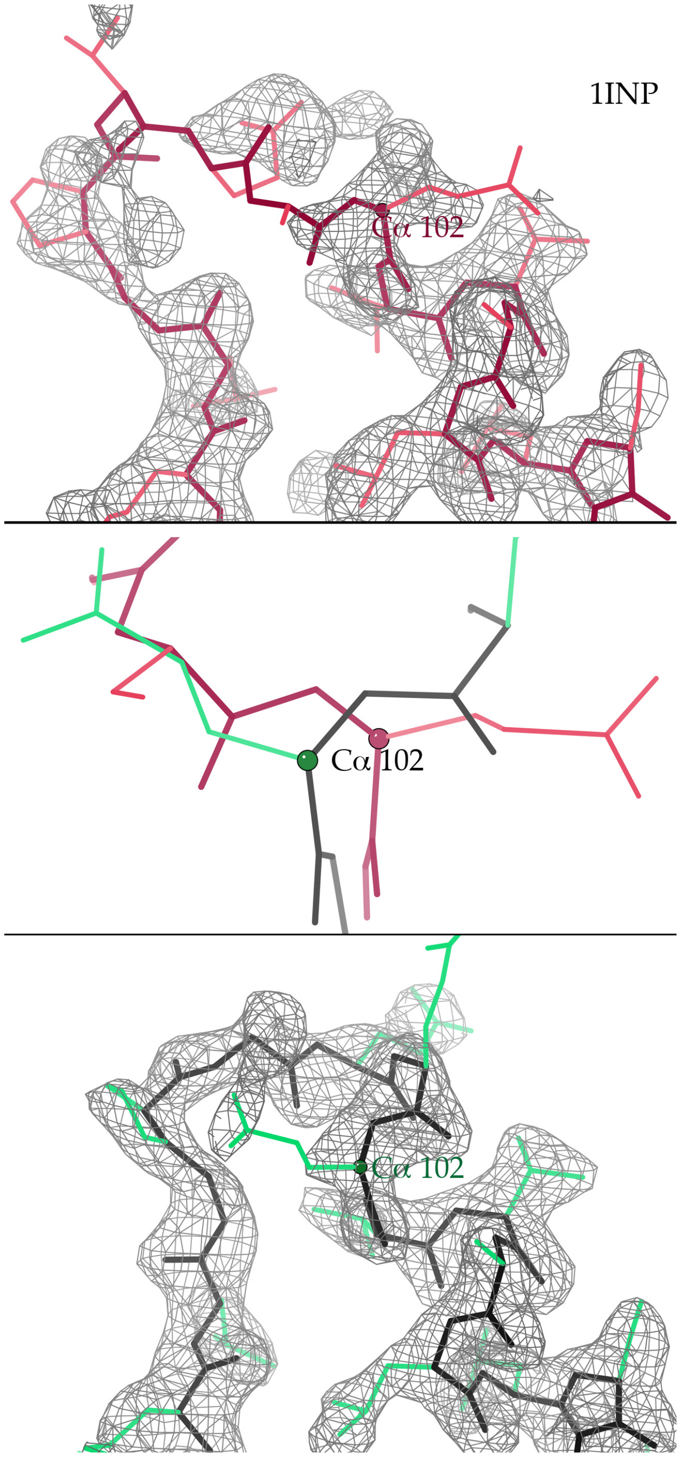


Figure 2. Example from 1INP.

References

Arendall WB III, Tempel W, Richardson JS, Zhou W, Wang S, Davis IW, Liu Z-J, Rose

JP, Carson WM, Luo M, Richardson DC, Wang B-C (2005). "A test of enhancing model accuracy in high-throughput crystallography." Journal of Structural and Functional Genomics 6:1-11.

Li S, Finley J, Liu Z-J, Qiu SH, Luan CH, Carson M, Tsao J, Johnson D, Lin G, Zhao J, Thomas W, Nagy A, Sha B, DeLucas LJ, Wang B-C, Luo M (2002). "Crystal structure of the cytoskeleton-associated protein glycine-rich (CAP-Gly) domain". Journal of Biological Chemistry 277:48596.

York JD, Ponder JW, Chen ZW, Mathews FS, Majerus PW (1994). "Crystal structure of inositol polyphosphate 1-phosphatase at 2.3Å resolution". Biochemistry 33: 13164.

FAQ

After I refine my model, why isn't atom X bound to atom Y when I look at it in a molecular viewer?

Most molecular viewers use the distance between atoms and knowledge of standard bond lengths to draw a line in the viewer pane that symbolises the bond. One reason for this is that the current PDB format does not enforce the specifying of bonds. This means that the lack of bond in a viewer does not mean that the refinement did not have a bond restraint between the specific atoms.

Running *phenix.refine* causes a file with an extension .geo to be written to disk (at the beginning of the refinement by default and optionally using the final model at the end of the refinement). This is the master file for restraints. All restraints used by the refinement engine are written to this file. The restraints are grouped by type and ordered with the restraints with largest residues first. It is human readable text with PDB strings for easy searching.

There are some simple tools for interrogating the information in this file but the most useful could be a new feature of REEL. The bonds can be read and displayed from the .geo using:

```
phenix.reel model.pdb model.geo
```

Other tools can be used at the command line and will be made available upon request.

Contributors

P. D. Adams, P. V. Afonine, B. Arendall, G. Bunkóczi, B. T. Burnley, N. Chakicherla, P. Gros, B. J. Hintze, N. W. Moriarty, D. C. Richardson, J. S. Richardson, C. J. Williams

CaBLAM: Identification and scoring of disguised secondary structure at low resolution

Christopher J. Williams, Bradley J. Hintze, David C. Richardson, and Jane S. Richardson

Department of Biochemistry, Duke University Medical Center, Durham, NC 27710

Correspondence email: christopher.j.williams@duke.edu

Low-resolution electron density presents challenges to fitting the details of the protein backbone. Density between 3 and 4Å generally lacks the details that would allow reliable placement of the backbone carbonyl oxygens. Low-resolution structures are therefore vulnerable to errors in peptide plane orientation. However, both humans and initial fitting programs tend to perform well in placing the general Cα trace, even into otherwise ambiguous density. CaBLAM identifies errors in peptide plane orientation and uses the relatively reliable Cα trace information to identify probable secondary structure disguised by these errors.

CaBLAM uses three parameters to identify modeling errors. μ_{in} and μ_{out} are two pseudodihedrals defined by Cα atom positions (Figure 1a). These parameters capture the Cα trace conformation around each residue. ν is a pseudodihedral that captures the orientation of a residue's peptide plane relative to its preceding neighbor (Figure 1b). ν is defined in a Cα-centric fashion to eliminate dependence on the positions of atoms that may not be reliably modeled. This parameter captures the majority of the outlier behaviors that CaBLAM targets.

These parameters were calculated for all residues in our Top8000 quality-filtered database that passed a B-factor filter of B<30 for all main chain atoms. Three-dimensional contours similar to those used in *phenix.rotalyze* and *phenix.ramalyze* (Chen *et al.*, 2010) were constructed from these high-quality reference data (Figure 2a). Glycine and proline residues showed significantly different behavior from the general case, and so received separate contours (not shown).

In analysis of a low-resolution structure, residues are compared against these contours. Those that fall outside the 95th percentile are considered outliers, and those that fall outside the 99th percentile are considered severe outliers. We

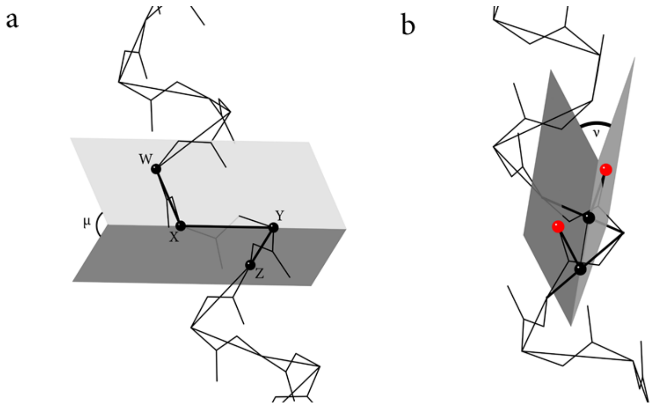


Figure 1: (a) Definition of the μ dihedral from four successive Cα positions. The μ_{in} and μ_{out} of adjacent residues overlap. The dihedral shown is both the μ_{out} of residue X and the μ_{in} of residue Y. (b) Defininition of the ν dihedral. To calculate ν, pseudoatom positions are constructed at the point on each Cα-Cα line closest to the carbonyl oxygen.

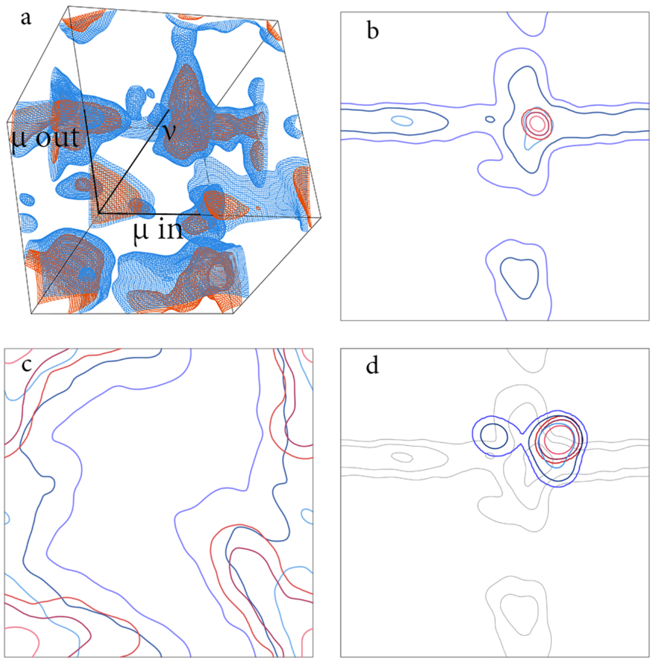


Figure 2: (a) Validation contours for general case residues in the CaBLAM parameter space. Contour levels shown are 99th and 99.5th percentiles. (b) Identification contours for α helix, (c) for β sheet, and (d) for 3_{10} helix, with α in the background for comparison. In b-d, the red contours represent a strict secondary structure definition, the blue a more permissive definition. The contour levels are 99th, 99.9th, and 99.99th percentiles.

found these cutoffs to provide a good balance between coverage of probable errors and acceptance of good structure.

A second set of contours is used to identify probable secondary structure. These contours were derived from the μ_{in} and μ_{out} parameters of residues in the Top8000 that matched hydrogen bonding patterns typical of each secondary structure type. Separate contours for α helix (Figure 2b), β sheet (2c), and 3_{10} helix (2d) were constructed.

Residues found to be CaBLAM parameter space outliers are compared against the 2-D secondary structure contours. Residues that fall within the 99.9th percentile are considered candidates for secondary structure. (In the region of overlap between α and 3_{10}, whichever receives the higher score is considered the more likely structure type.) Individual residues are assembled into complete secondary structure elements if at least three residues in a row match the secondary structure contours and continue for as many residues as the match persists.

Because the accuracy of the Cα trace through a residue is crucial to determining that residue's secondary structure type in CaBLAM, we have constructed an additional set of contours to check the reliability of local Cα geometry. These contours (not shown) are three-dimensional, using μ_{in}, μ_{out}, and the Cα virtual angle. Residues falling outside the 99.5th percentile are considered Cα geometry outliers and are excluded from further secondary structure assessments, as they cannot be guaranteed to have interpretable geometry. This cutoff was chosen as a breakpoint in contour behavior and for good coverage of the conformational space of the secondary structure contours.

CaBLAM is implemented inside PHENIX (Adams *et al*. 2010), making use of the `cctbx` utilities. The most comprehensive access to CaBLAM's validation is through the command line call:

```
phenix.cablam_validate \
  give_full_kin=True filename.pdb
```

This will return a multi-criterion-style kinemage with comprehensive CaBLAM validation and annotation. CaBLAM outliers are marked in purple, and severe CaBLAM outliers are marked in

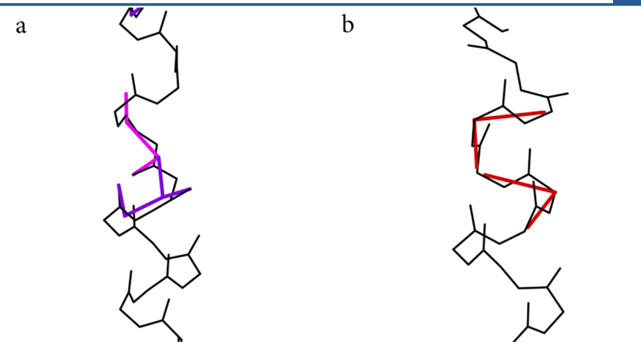


Figure 3: (a) Validation markup for CaBLAM outliers on a helix segment from 2o01. Purple denotes mild outliers, outside the 95th percentile contour. Pink denotes severe outliers, outside the 99th percentile contour. CaBLAM outlier markup follows the trace of the ν dihedral. (b) Validation markup for Cα geometry outliers, outside the 99.9th percentile contour.

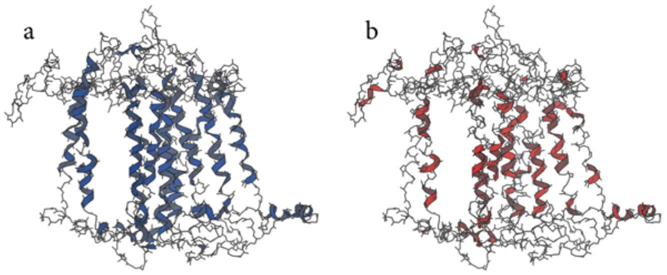


Figure 4: (a) Ribbons as drawn by `phenix.king` for secondary structure elements identified by CaBLAM in 2o01, a structure with many peptide plane orientation outliers. CaBLAM provides good coverage of the transmembrane helices. (b) Ribbons for the helices identified by `phenix.ksdssp`.

pink (Figure 3a). The markup follows the trace of the ν dihedral, since this is the most likely source of the error. Cα geometry outliers are marked in red (Figure 3b). That markup follows the Cα virtual angle of the outlier residue. Selecting a specific markup annotation in the KiNG window will display statistics on that residue's outlier level and its likelihood for secondary structure; for instance, a β-sheet outlier might show "general=0.7% (outlier), beta=85%, alpha=0%".

Rather than forcing users to interpret those statistics for themselves, CaBLAM also writes out pdb-style HELIX and SHEET records for the secondary structure elements it identifies. KiNG (Chen *et al*., 2009) interprets these records as ribbons for easy interpretation (Figure 4a). When `phenix.ksdssp` is used to return HELIX and SHEET records for the challenging Photosystem I structure `2o01`, the hydrogen-bonding-dependent analysis of `ksdssp` results in much less thorough identification of the structure's transmembrane helices (Figure 4b). Like many of the usual tools of structural biology, `ksdssp` was not designed to address the unusual challenges of low-resolution structures. By focusing on the relatively reliable Cα trace, CaBLAM goes some distance toward filling this gap in our validation capabilities.

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. and Zwart P.H.. (2010) PHENIX: a comprehensive Python-based system for macromolecular structure solution. Acta Cryst. D66:213-221.

Chen V.B., Davis I.W. and Richardson D.C. (2009) KiNG (Kinemage, Next Generation): A versatile interactive molecular and scientific visualization program. Protein Science 18:2403-2409.

Chen V.B., Arendall W.B., Headd J.J., Keedy D.A., Immormino R.M., Kapral G.J., Murray L.W., Richardson J.S. & Richardson D.C. (2010) MolProbity: all-atom structure validation for macromolecular crystallography Acta Cryst D66:12-21

Structural Classification of Allergen IgE Epitopes by Hierarchical Clustering

Naveen Chakicherla

Dougherty Valley High School, San Ramon, CA 94582; Intern, Lawrence Berkeley National Laboratory, Berkeley, CA 94720.

Correspondence email: naveenchaki@gmail.com

Abstract

Allergen epitopes derived from the Structural Database of Allergenic Proteins (SDAP) were modeled and classified structurally using single linkage hierarchical clustering. The largest single cluster in this study contained 231 epitopes representing 12 of the 16 species studied. Cluster analysis of 3D models generated for these epitopes revealed that, while sequence identity between the allergen epitopes was generally very low, epitopes in this cluster shared a common tertiary structure in the form of the W-shaped motif with a consensus sequence `CR+AKA--SK+SG`. The putative shared tertiary structural motif in the epitopes in the largest cluster could also contribute to the significant amount of cross-reactivity clinically observed in several allergens.

Introduction

Over the past decade, there has been a dramatic increase in both incidence and knowledge gain in the area of allergy medicine [1, 2]. The sequences of over 100 food allergens have been published and have been compiled into various public databases [2, 3]. Yet effective treatment of allergic response is still not established, prevention (avoidance) being the most touted cure. Surprisingly, analyses of plant food allergens show that most belong to only a small number of protein superfamilies. Most plant allergens belong to one of only four protein superfamilies, namely prolamins, cupins, profilins, and the Bet V1 family [4, 5]. The World Health Organization/Food and Agriculture Organization (WHO/FAO) guidelines define a protein as potentially allergenic if it either has an identity of at least six contiguous amino acids or a minimum of 35% sequence similarity over a window of 80 amino acids when compared with known allergens [6]. The epitopes, or antigenic determinant of protein antigens, are categorized as linear or conformational epitopes [7], which interact with the paratope based on linear sequence or 3-dimensional structure respectively. Conformational epitopes are stretches of amino acids that interact with an antigen and they are recognized on the basis of their 3D structure. Linear epitopes are recognized on the basis of their amino acid sequence. Although most epitopes are of the conformational kind, the allergen classification has been largely based on sequence similarity, and cross-reactivity predictions based on the WHO/FAO rules result in large numbers of false positives. Furthermore, sequence-based detection could miss several true positives. This study proposes the application of a clustering method to the 3-dimensional structures of allergenic proteins in order to better define the characteristics of the IgE epitopes of allergens in general.

Methods

Sequences of allergenic proteins with known and defined epitopes were obtained from the Structural Database of Allergenic Proteins (SDAP) [3]. In total, 41 full-length sequences of 29 different proteins were submitted to the homology modeling server AS2TS [8], resulting in 35 successful model building runs yielding five putative models for each sequence. Structures of epitopes, as listed in the SDAP, were extracted from these models using JMol [9].

In all, 1655 epitope models were generated, of which 170 models were of t-cell epitopes, 80 models of IgG epitopes and the remaining 1405 were of IgE epitopes. Of the total 1655 models, 446 had a file size of zero, since the segment of the allergen containing that epitope had not been modeled in AS2TS. A structural comparison between all remaining 1209 3D models was performed with SUBCOMB [10] with normalized spatial discrepancy (NSD). The 728424 structure superpositioning jobs were carried out using 30 2.4 GHz processors simultaneously, resulting in 1.3 CPU weeks in total, with a wall clock time of 7.5 hours. The resulting distance matrix was used to perform a

Table 1: Allergen types with known epitopes that were downloaded from SDAP and their representation in the largest cluster in the study.

Allergen	Species - Scientific Name	Species - Common Name	In Cluster 282
Ara h 1; Ara h 2.0101; Ara h 3	*Arachis hypogaea*	Peanut	Y
Asp f 1; Asp f 13; Asp f 2; Asp f 3	*Aspergillus fumigatus*	Fungi	Y
Bet v 1	*Betula verrucosa (Betula pendula)*	White birch	Y
Cha o 1	*Chamaecyparis obtusa*	Japanese Cypress	N
Cry j 1; Cry j 2;	*Cryptomeria japonica*	Japanese Cedar, sugi	Y
Jun a 1.010101; Jun a 3	*Juniperus ashei*	Mountain Cedar	N
Fag e 1;	*Fagopyrum esculentum*	Common Buckwheat	Y
Gly m glycinin G1; Gly m glycinin G2	*Glycine max*	Soybean	Y
Hev b 1; Hev b 3; Hev b 5	*Hevea brasiliensis*	Rubber (latex)	Y
Jug r 1	*Juglans regia*	English walnut	Y
Len c 1.0101	*Lens culinaris*	Lentil	N
Par j 1; Par j 2	*Parietaria judaica*	Pellitory-of-the-Wall	Y
Pen a 1	Penaeus aztecus	Shrimp	Y
Ves v 5	*Vespula vulgaris*	Yellow Jacket	N
Gal d 1	*Gallus domesticus*	Chicken	Y
Pen ch 18	*Penicillium chrysogenum (formerly P. notatum)*	Fungi	Y

cluster analysis with single linkage hierarchical clustering as available in the CCTBX [11].

The 19 clusters with ten or more members were further analyzed for composition, sequence identity, and other characteristics. The epitopes included in the largest cluster, cluster 282 were further analyzed. In order to compare their primary structures the PDB structures of the epitopes needed to be converted back to a primary sequence format. A custom application written by Sebastian Raschka, Michigan State University was used in this study to convert pdb files to fasta sequence format [12]. This custom tool takes a folder of multiple PDB files and writes the fasta sequences for all PDBs into one file that is placed in the same folder. Multiple alignment of the representative members of the largest cluster 282 was conducted using Clustal Omega [13]. In order to compare them structurally, the tool MAPSCI [17] was used.

Results

In this study, we analyzed the 29 allergens with epitopes that are to be found in the SDAP allergen database. These 29 allergens with epitopes originate from 16 distinct plant and animal species (table 1). Three-dimensional structural models were generated for the epitopes of these 29 allergens, resulting in 1209 3D models. Single linkage hierarchical clustering of the epitope structures resulted in a distribution of clusters as depicted in table 2. Clusters varied widely in size,

Table 2: Cluster distribution of the 270 clusters resulting from single linkage hierarchical clustering of structural models of all epitopes of the modeled allergens from SDAP.

Cluster Size	Occurrence
231	1
38	1
20	1
15	3
12	1
11	1
10	11
9	1
8	4
7	1
6	2
5	45
4	37
3	25
2	79
1	77

with the largest cluster having 231 members, while the smallest had one. There were 77 clusters with only one member, and these were eliminated. Clusters with ten or more epitope members were further analyzed. Of the 11 clusters with ten members each, nine clusters contained epitopes of the allergen Aspf1 from the fungi *Aspergillus fumigatus*, while the other two contain epitopes of the allergen, lipid transfer protein; P2 from the weed *Parietaria judaica*. The single cluster with 11 members had epitopes from the two allergens Pench18 and Hev b5 from the fungus *Penicillium chrysogenum* and the kiwi and potato homolog from rubber, *Hevea brasiliensis* respectively. The single cluster with 12 members included epitopes Ara h3 and Gly m glycinin G2 from peanut *Arachis hypogea*, and soybean *Glycine max* respectively. The three clusters with 15 members each all contained epitopes from the fungus *Aspergillus fumigatus.* The single cluster with 20 members had epitopes from the allergen Ara h 2 from peanut *Arachis hypogea* and from the allergen Fag e 1, from the common buckwheat *Fagopyrum esculentum*. The largest cluster containing 231 members is described further below.

Discussion

The majority of the clusters in this structural classification study were composed of epitopes from within the same species, some from different parts of the same protein model. This result suggests that epitopes within allergens are repeated at separate locations in the tertiary structure. However, there were some interesting clusters that showed membership from more than one species. The largest cluster, 282, with 231 members, was analyzed further. Of the 16 allergen species in the entire data set studied, epitopes of allergens from 12 species were included in this largest cluster. They included epitopes from *Arachis hypogaea (*peanut*), Aspergillus fumigatus (*fungus*), Betula verrucosa or Betula pendula (*white birch*), Cryptomeria japonica* (Japanese cedar, sugi), *Fagopyrum esculentum* (common buckwheat), *Glycine max* (soybean), *Hevea brasiliensis* (rubber (latex)), *Juglans regia* (English walnut),*Parietaria judaica* (Pellitory-of-the-Wall), *Penaeus aztecus* (shrimp), *Gallus domesticus* (chicken) and *Penicillium chrysogenum* formerly *P. notatum* (fungus). The four species not represented in this cluster included *Chamaecyparis obtuse* (Japanese Cypress), *Juniperus ashei* (mountain cedar), *Lens culinaris* (lentil) and *Vespula vulgaris* (yellow jacket).

The absence of the mountain cedar epitopes and the yellow jacket epitopes in cluster 282 is easily explained, because AS2TS failed to identify structural models in RCSB for the allergens from these two species. Their epitopes were therefore not represented in our final cluster analysis. The Japanese cypress allergen epitopes had 50 models generated in our analysis. However, it should be noted that the epitope in all cases of this allergen were classified by SDAP as t-cell epitopes that presumably has a different structure than the IgE type epitope that is prevalent in all other epitopes in the cluster 282 (the largest cluster). Hong *et. al.* have shown that pepsin digestion eliminates IgE reactivity but maintains T-cell reactivity suggesting that the two epitopes have different structure and or location in the allergen [16]. The lentil allergen did in fact have 12 models generated by AS2TS, one set modeled using the A chain of the adzuki bean 7S globulin 1 protein structure 2ea7-a and the second set modeled using the PDB structure of a hypothetical Coenzyme A-binding protein from *Thermus thermophiles* bacterium, 1iuk-a.

Sequence analysis of all epitopes in cluster 282 using Clustal Omega gave an alignment length of 30 and an average identity of 7.27% (for default settings) and an alignment length of 25 and an alignment length of 24 and average identity of 8.53% for two HMM iteration and one guide-tree iteration. When all FASTA sequences with exact sequence matches to other FASTA from the same allergen and epitope were removed, as were 1-mer epitopes, this resulted in a smaller cluster of 53 unique sequences. This set gave similar results as the entire set with average length of 6.9 residues, an alignment length of 23 and an average identity of 6.17%.

However, multiple structural analysis of essentially the same set of epitopes from cluster 282 using MAPSCI [17] (epitope PDBs with fewer than four C-alphas were removed, resulting in 45 PDBs) showed a consensus motif in several members of this cluster (See figure 1 and table 3). This motif consisted of two parts with a consensus sequence of `CR+AKA- -SK+SG`. The 3D structural visualization of these epitopes reveals a

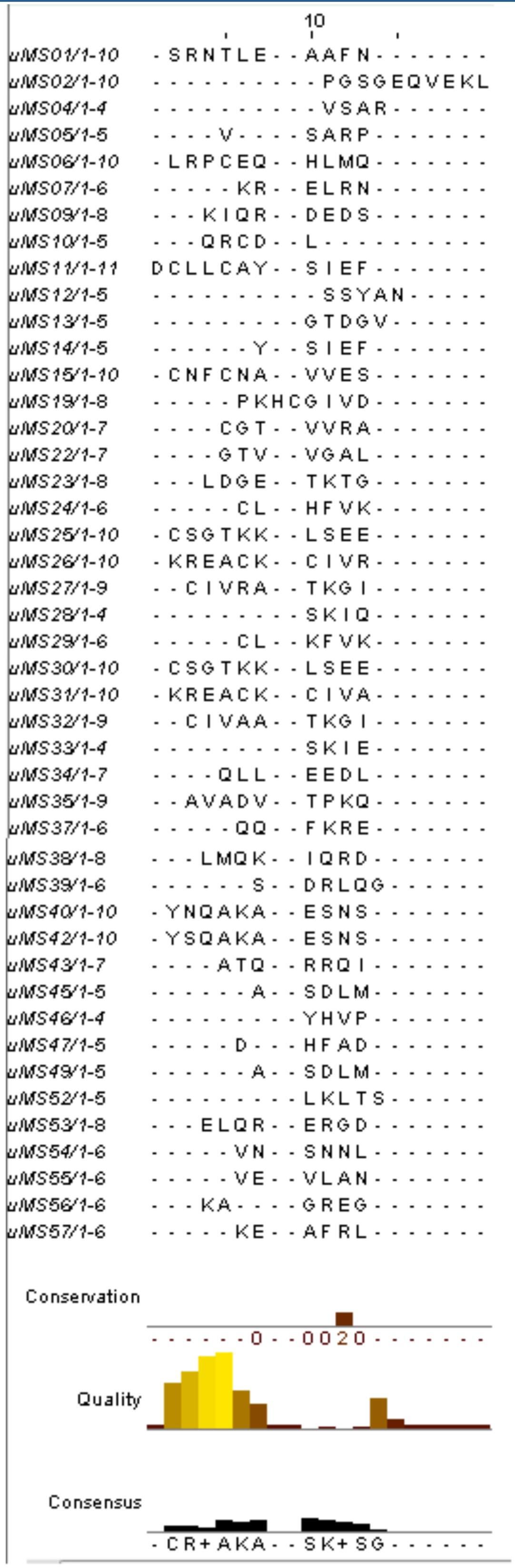
Table 3: Summary of cluster 282 epitopes analyzed using MAPSCI.

MAPSCI code	Allergen	SDAP Source Number	AS2TS model number	PDB template	PDB template chain	Epitope type	Epitope start residue	Epitope stop residue
uMS01	arah1	p43237	1	1iuk	a	IgE	311	320
uMS02	arah1	p43237	1	1iuk	a	IgE	559	568
uMS04	arah1	p43237	3	3s7i	a	IgE	578	587
uMS05	arah1	p43237	5	1dgw	y	IgE	578	587
uMS06	arah2.0101	9186485	1	1w2q	a	IgE	39	48
uMS07	arah2.0101	9186485	1	1w2q	a	IgE	127	132
uMS09	arah2.0101	9186485	2	3ob4	a	IgE	49	56
uMS10	arah2.0101	9186485	5	1pnb	b	IgE	143	150
uMS11	Gald1	P01005	1	2p6a	d	IgE	64	74
uMS12	Gald1	P01005	1	2p6a	d	IgE	95	99
uMS13	Gald1	P01005	1	2p6a	d	IgG	55	59
uMS14	Gald1	P01005	1	2p6a	d	IgG	70	74
uMS15	Gald1	P01005	1	2p6a	d	IgG	189	198
uMS19	Parj1	P43217	3	1t12	a	IgE	72	79
uMS20	Parj1	O04404	1	1mid	a	IgE	41	47
uMS22	Parj1	Q40905	1	1t12	a	IgE	41	47
uMS23	Parj1	Q40905	1	1t12	a	IgE	72	79
uMS24	Parj2	P55958	1	2alq	a	IgE	45	50
uMS25	Parj2	P55958	1	2alq	a	IgE	61	70
uMS26	Parj2	P55958	1	2alq	a	IgE	77	86
uMS27	Parj2	P55958	1	2alq	a	IgE	83	91
uMS28	Parj2	P55958	1	2alq	a	IgE	122	129
uMS29	Parj2	O04403	1	fk5	a	IgE	45	50
uMS30	Parj2	O04403	1	fk5	a	IgE	61	70
uMS31	Parj2	O04403	1	fk5	a	IgE	77	86
uMS32	Parj2	O04403	1	fk5	a	IgE	83	91
uMS33	Parj2	O04403	1	fk5	a	IgE	122	129
uMS34	Pena1	11893851	3	2b9c	b	IgE	85	99
uMS35	Pench18	7963902	5	1sh7	a	IgE	401	410
uMS37	arah2.0101	15418705	1	1w2q	a	IgE	127	132
uMS38	arah2.0101	15418705	2	3obq4	a	IgE	49	56
uMS39	arah2.0101	15418705	2	3obq4	a	IgE	117	122
uMS40	Aspf1	166486	1	1jbs	a	IgE	51	60
uMS42	aspf1	p04389	1	1jbs	a	IgE	51	60
uMS43	Aspf2	P79017	1	1eb6	a	IgE	58	64
uMS45	Aspf2	P79017	1	1eb6	a	IgE	181	185
uMS46	Aspf2	P79017	1	1eb6	a	IgE	189	192
uMS47	Aspf2	P79017	1	1eb6	a	IgE	200	204
uMS49	Aspf3	664852	1	1eb6	a	IgE	115	125
uMS52	Cryj2	P43212	5	2x3h	a	T-cell	400	414
uMS53	Fage1	2317670	1	3fz3	b	IgE	360	367
uMS54	Fage1	2317670	1	3fz3	b	IgE	411	416
uMS55	Fage1	2317674	1	2e9q	a	IgE	460	465
uMS56	Fage1	2983941	1	3qac	a	IgE	460	465
uMS57	Fage1	2983941	1	3qac	a	IgE	506	511

W-shaped "basket" motif (figure 2). The first part of the motif showed better quality in the multiple alignment, but with lower consensus than the second part. The second half of the motif showed greater conservation and was more consistently present in all the epitopes in the cluster 282, although the quality was overall lower.

Conclusion

The decision tree approach to evaluating the potential allergenicity of genetically engineered food products [14, 15] proposed in 1996 has since



```
                          10
                   '      '      '
uMS01/1-10    -SRNTLE--AAFN-------
uMS02/1-10    ----------PGSGEQVEKL
uMS04/1-4     ----------VSAR------
uMS05/1-5     ----V----SARP-------
uMS06/1-10    -LRPCEQ--HLMQ-------
uMS07/1-6     -----KR--ELRN-------
uMS09/1-8     ---KIQR--DEDS-------
uMS10/1-5     ---QRCD--L----------
uMS11/1-11    DCLLCAY--SIEF-------
uMS12/1-5     ----------SSYAN-----
uMS13/1-5     ---------GTDGV------
uMS14/1-5     ------Y--SIEF-------
uMS15/1-10    -CNFCNA--VVES-------
uMS19/1-8     -----PKHCGIVD-------
uMS20/1-7     ----CGT--VVRA-------
uMS22/1-7     ----GTV--VGAL-------
uMS23/1-8     ---LDGE--TKTG-------
uMS24/1-6     -----CL--HFVK-------
uMS25/1-10    -CSGTKK--LSEE-------
uMS26/1-10    -KREACK--CIVR-------
uMS27/1-9     --CIVRA--TKGI-------
uMS28/1-4     ---------SKIQ-------
uMS29/1-6     -----CL--KFVK-------
uMS30/1-10    -CSGTKK--LSEE-------
uMS31/1-10    -KREACK--CIVA-------
uMS32/1-9     --CIVAA--TKGI-------
uMS33/1-4     ---------SKIE-------
uMS34/1-7     ----QLL--EEDL-------
uMS35/1-9     --AVADV--TPKQ-------
uMS37/1-6     -----QQ--FKRE-------
uMS38/1-8     ---LMQK--IQRD-------
uMS39/1-6     ------S--DRLQG------
uMS40/1-10    -YNQAKA--ESNS-------
uMS42/1-10    -YSQAKA--ESNS-------
uMS43/1-7     ----ATQ--RRQI-------
uMS45/1-5     ------A--SDLM-------
uMS46/1-4     ---------YHVP-------
uMS47/1-5     -----D---HFAD-------
uMS49/1-5     ------A--SDLM-------
uMS52/1-5     ---------LKLTS------
uMS53/1-8     ---ELQR--ERGD-------
uMS54/1-6     -----VN--SNNL-------
uMS55/1-6     -----VE--VLAN-------
uMS56/1-6     ---KA----GREG-------
uMS57/1-6     -----KE--AFRL-------

Conservation
              ------0--0020-------
Quality

Consensus
              -CR+AKA--SK+SG------
```

Figure 1. Structure based sequence alignment of members of Cluster 282 using MAPSCI shows a consensus sequence of CR+AKA..SK+SG.

been practiced widely to determine allergenic nature of novel protein products. This strategy uses primary sequence analysis and immunochemistry along with gene source to predict the allergenicity of the novel protein. We have presented here a structural classification of the epitopes from all allergens having epitopes in the Structural Database of Allergenic Proteins, SDAP. The largest single cluster in this study contained 231 epitopes representing 12 of the 16 species studied. Cluster analysis of 3D models generated for these allergens revealed that, while sequence identity between the allergen epitopes was generally very low, epitopes in this cluster shared common tertiary structure in the form of the W-shaped motif with consensus sequence `CR+AKA--SK+SG`. This study lends support to the recommendation that the decision tree approach to evaluating the potential allergenicity of genetically engineered food products, from the FAO/WHO, should include a stronger component of structural identity, in addition to the current focus on sequence identity. The putative shared tertiary structure motif in the epitopes in the largest cluster in the analysis could also contribute to the significant amount of cross-reactivity clinically observed in several allergens.

Acknowledgements

This study was funded in part by a Student Assistantship in the Physical Biosciences Division at Lawrence Berkeley National Laboratory (Berkeley Lab) in the lab of Dr. Peter H. Zwart. I would like to especially thank my advisor and mentor, Peter H. Zwart, for his guidance and encouragement. I would also like to thank the Berkeley Center of Structural Biology for allowing me the use of their computational facilities for this research. I would like to acknowledge the support of Sebastian Raschka, Ph.D. student at Michigan State University for a tool that converts pdb files to fasta files.

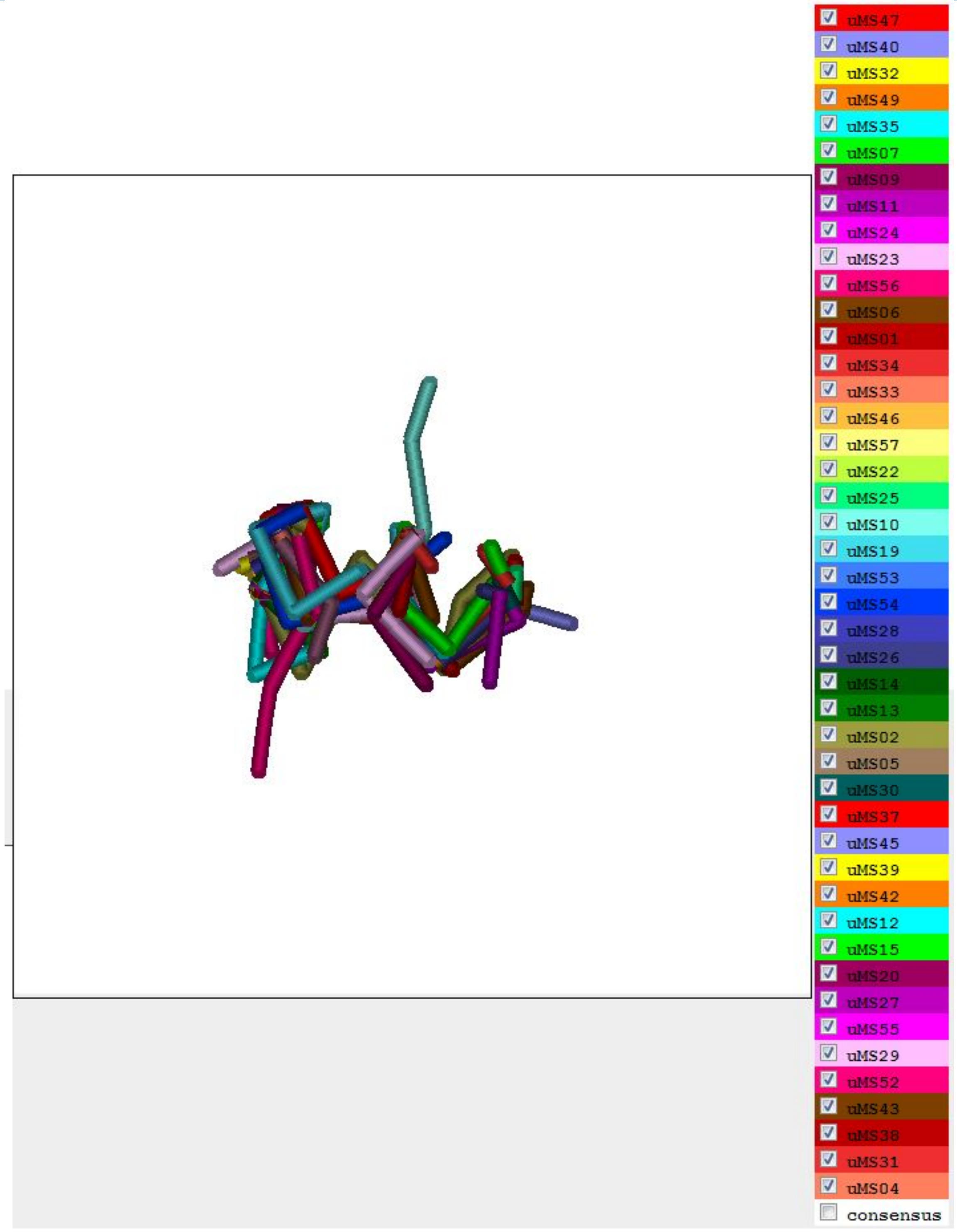


Figure 2. 3D visualization of the members of cluster 282 showing the W-shaped "basket" motif of the cluster. The members of this alignment are available in table 3.

References

1. Ovidiu Ivanciuc, Tzintzuni Garcia, Miguel Torres, Catherine H. Schein, Werner Braun (2009). *Molecular Immunology* 46, 559–568.
2. Christian Radauer, Merima Bublin, Stefan Wagner, Adriano Mari, Heimo Breiteneder (2008) *J Allergy Clin Immunol* 2008;121:847-52.
3. Ivanciuc, O., Schein. C. H., and Braun, W. (2003). *Nucleic Acids Res.* 31(1). 359-362.
4. Heimo Breiteneder, E.N. Clare Mills (2005). *Biotechnology Advances* 23, 395–399.
5. Christian Radauer, and Heimo Breiteneder (2007). *J Allergy Clin Immunol*. 120:518-25.
6. FAO/WHO, 2001, http://www.fao.org/es/ESN/food/pdf/allergygm.pdf; 2003, http://www.codexalimentarius.net/download/report/46/Al0334ae.pdf
7. Ovidiu Ivanciuc, Catherine H. Schein, Tzintzuni Garcia, Numan Oezguen, Surendra S. Negi, Werner Braun. (2009). *Regulatory Toxicology and Pharmacology* 54, S11–S19.
8. Zemla A. (2003). *Nucleic Acids Research*, Vol. 31, No. 13, pp. 3370-3374.
9. Jmol: an open-source Java viewer for chemical structures in 3D. http://www.jmol.org/
10. Kozin, M. B., and Svergun, D. I. (2001) *J. Appl. Crystallogr.* 34, 33.
11. http://cctbx.sourceforge.net/
12. http://sebastianraschka.com/webapps.html
13. Fabian Sievers, Andreas Wilm, David Dineen, Toby J Gibson, Kevin Karplus, Weizhong Li, Rodrigo Lopez, Hamish McWilliam, Michael Remmert, Johannes Söding, Julie D Thompson, and Desmond G Higgins (2011). *Mol Syst Biol.* 7: 539.
14. Dean D. Metcalfe, James D. Astwood, Rod Townsend, Hugh A. Sampson, Steve L. Taylor & Roy L. Fuchs (1996). *Critical Reviews in Food Science and Nutrition* 36, Suppl. 001.
15. http://www.fao.org/docrep/007/y0820e/y0820e05.htm
16. Hong SJ, Michael JG, Fehringer A, Leung DY. (1999). *J Allergy Clin Immunol*. Aug;104(2 Pt 1):473-8.
17. Ivaylo Ilinkin, Jieping Ye, Ravi Janardan (2010). *BMC Bioinformatics*, 11:71.

New tool: phenix.real_space_refine

Pavel V. Afonine[a], Jeffrey J. Headd[b], Thomas C. Terwilliger[c] and Paul D. Adams[a,d]

[a]*Physical Biosciences Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720-8235*
[b]*Department of Biochemistry, Duke University Medical Center, Durham, NC 27710*
[c]*Los Alamos National Laboratory, Los Alamos, NM 87545-0001*
[d]*Department of Bioengineering, University of California at Berkeley, Berkeley, CA 94720-1762*

Correspondence email:e-mail: PAfonine@lbl.gov

The aim of this article is to announce and briefly document *phenix.real_space_refine* – a new command line tool for refinement of a model against a map.
The program is being developed with the following goals:

- fit a model as well as possible into a map that can vary in quality: from low to high resolution;
- produce models without poor validation metrics, such as clashes or rotamer outliers;
- correct obvious model completeness problems, such as incomplete residues or simple alternative conformations (that are unambiguously defined by the map);
- fast execution;
- have a large convergence radius.

Not all the above goals are implemented at present and are therefore the subject of future work. Also, the program is still under active development, which implies that some protocols or/and functionality may change or improve in future versions.

The current workflow of *phenix.real_space_refine* includes:

- reading a PDB file with an initial model and a map that can be given as an actual map in CCP4 or X-plor format or as Fourier coefficients (MTZ file). There are no assumptions made about the technique used to obtain the map (X-ray, neutron, electron microscopy, etc);
- refinement macro-cycle (repeated execution of various refinement techniques listed below);
- the results of running *phenix.real_space_refine* are two PDB files: one contains the final refined model and the other one is a multiple model file containing all intermediate models that were generated during refinement.

The refinement macro-cycle consists of five refinement steps:

1. *Model morphing* (Terwilliger *et al*., 2012) is performed first and repeated until it shifts the model by no more than 0.3Å or the number of iterations exceeds a specified threshold.
2. *Local real-space fitting* of individual residues that the program identifies as outliers. Residues that need local real-space refinement (outliers) are chosen based on map correlation, rotameric state or clashes with neighboring atoms. First, the procedure fits main-chain atoms and then performs local side-chain rotamer searches to find a valid rotameric state that best fits the map. For each newly fit residue torsion angle restraints are added to maintain its current conformation in subsequent global refinement (so the residue doesn't become an outlier again). These restraints are discarded or updated at the next macro-cycle.
3. *Global gradient-driven* (lbfgs) refinement using the target function, $T = T_{MAP} + w * T_{RESTRAINTS}$, where the restraints target, $T_{RESTRAINTS}$, consists of a set of standard restraints (Grosse-Kunstleve *et al*., 2004; Afonine *et al*., 2012) as well as optional extra restraints such as secondary structure, C_β deviation and Ramachandran plot restraints (Headd *et al*., 2012). The weight 'w' is determined automatically best testing of several values: this array of values is tried until a value is found that yields a refined model with bond and angle deviations from ideal values lower than predefined limits (which are user-controlled parameters). T_{MAP} is the negated sum of map values calculated at atomic centers.
4. *Simulated Annealing* (SA) is optional using the same implementation from *phenix.refine* (Afonine *et al*., 2012).
5. *Rigid-body* refinement is also optional and currently can only be performed on the entire

model (as opposed to smaller rigid groups) limiting its usefulness somewhat.

Current known limitations, inefficiencies and planned future functionalities include:

- model may not contain hydrogen atoms;
- morphing is done only on the whole model and not on individual parts;
- the procedure could be stabilized by allowing single atoms to be harmonically restrained to their current positions;
- there is not yet an GUI (command line only);
- there are some missing restraint features: (no reference model or NCS);
- there is no model completion;
- non amino-acid residues are not handled in morphing and local real-space fitting.
- There is no ADP (individual, TLS) and occupancy refinement.

A detailed description of the program and underlying tools will be published elsewhere.

Usage examples:

```
phenix.real_space_refine model.pdb ccp4_formatted_map.map
```

```
phenix.real_space_refine model.pdb map_coefficients.mtz
```

```
phenix.real_space_refine model.pdb map_coefficients.mtz label='FOFCWT,PHFOFCWT'
```

To see all parameters: `phenix.real_space_refine -h`

References

1. 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 Cryst. D68, 352-367.
2. Grosse-Kunstleve, R.W., Afonine, P.V. and Adams, P.D. (2004). "cctbx news: Geometry restraints and other new features". Newsletter of the IUCr Commission on Crystallographic Computing, 4, 19-36.
3. Headd, J.J., Echols, N., Afonine, P.V., Grosse-Kunstleve, R.W., Chen, V.B., Moriarty, N.W., Richardson, D.C., Richardson, J.S. & Adams, P.D. (2012). "*Use of knowledge-based restraints in phenix.refine to improve macromolecular refinement at low resolution*". Acta Cryst. D68, 381-390.
4. Terwilliger, T.C., Read, R.J., Adams, P.D., Brunger, A.T., Afonine, P.V., Grosse-Kunstleve, R.W. & Hung, L.-W. (2012). "*Improved crystallographic models through iterated local density-guided model deformation and reciprocal-space refinement*". Acta Cryst. D68, 861-870.

cctbx tools for transparent parallel job execution in Python. III. Remote access

Gábor Bunkóczi

Department of Haematology, University of Cambridge, Cambridge, UK

Correspondence email: gb360@cam.ac.uk

Introduction

CPU development in the past decade, especially the introduction of multi-core CPUs, has made an increasing number of structural biology problems computationally feasible, given the availability of a suitably large computing cluster. Although such a cluster is not always available, the necessary computing power is often present, but distributed over a number of workstations. Harnessing the collective power of this workstation pool can be a sufficient substitute to a computing cluster for certain calculations. However, a number of these processing nodes must be accessed remotely.

On the other hand, even when a cluster is available, access is often complicated by network security rules and usage restrictions that can explicitly forbid users running specific processes (e.g. graphical user interfaces) on the cluster's head node (although it is more frequent that system administrators simulate direct access to the cluster with remote command execution). While such a solution is transparent for most uses, it can interfere with the `processing` (Bunkóczi & Echols, 2012) module (for brevity, the `libtbx.queuing_system_utils` package will be implicit in all unqualified module names, except stated otherwise), because of potentially different base directories used when translating relative file paths. In addition, the overhead of job polling can increase significantly, since each request must be authenticated by the selected network protocol (normally `ssh`). Finally, this mode of access implicitly assumes a shared file system, and network latency can become a problem.

The `libtbx.queuing_system_utils.remote` module offers a convenient and transparent solution to these problems.

Remote execution

The core architecture resembles a common client-server setup, although in this case the server process is started up by the client after establishing a network connection with the remote machine, and can only use a single communication channel, namely the one that is open towards the client that initiated the connection. The server process has two responsibilities: (1) job execution and (2) data exchange with the client. Management of job execution is done using the `scheduling` module (Bunkóczi & Echols, 2012). Scheduling is, however, delegated to the client, which is responsible for tracking how many jobs are being executed. The server behaves as an "infinite resource", and every time a new job execution request is made, it is fulfilled. Data exchange with the client is managed by the main thread. This task is very much I/O bound, and it spends most of its time waiting for requests. To achieve efficient and regular job lifecycle management, job execution is handled by a separate thread.

On the client side, a proxy object is created that holds communication streams to the server, and acts as the only point of contact. Similarly to the `processing.QueueHandler` object, it exports a `Job` factory function that can be used to create a remote job on the server machine. This creates an object that shares the same interface as `multiprocessing.Process` or `threading.Thread` from the Python Standard Library. Job execution data is sent over serialized (using the Python Standard Library `pickle` module) to the server, where it gets reconstructed and executed. Results are sent back using a similar mechanism. Therefore, input data (including the function to execute) and results have to be pickleable. It is important to note that the client object does not have its own thread: submission, job status checks and result downloads are performed on the main thread, whenever requested. This may add a certain overhead to the whole program, but this is normally not serious.

For the network communication to work, the `remote` module on both ends of the connection should be the same version. In addition, for serialization to work, input and output objects (including the called function) should also be the same version.

The `remote` module tries to resemble direct execution as closely as possible. For example, anything that is printed by the executing process at the remote machine is transferred and printed onto the local computer's terminal.

Management of the network connection has been kept entirely independent in the design. The client only requires the input and output file handles of the server to maintain communication, and would therefore work with any network protocols (although the server code may need adjustment).

Fetching job results

It is assumed that each remote job returns a value that needs to be sent back to the client (note that the design of the `scheduling` module allows the execution of jobs without capturing their return value). This is normally done by passing one end of a data queue to the underlying execution process. However, for a remote job this is done using a two-stage approach.

1. The server process configures each `ExecutionUnit` with a `RetrieveProcessor`. The appropriate data queue is selected at time of creation. This is used to return results from jobs to the server's `Manager` object.
2. The server sends this result back to the client. The client attaches it to the corresponding `Job` object, and the parent `ExecutionUnit` can extract it using a `FetchProcessor`, which is a new processor class developed for this task.

An alternative option would be to pass the details of a data queue to the remote machine, and make the executing processes return their result via a network socket. This would transform the whole return procedure into a single-step process, and would also make the `FetchProcessor` superfluous. Unfortunately, this approach makes the assumption that the local machine can be contacted from the remote (which is often invalid), and would also require development of network server process that would handle incoming connections and channel results to the main program.

Configuration

Remote execution is not tied to any particular execution mode: in fact, it would be possible (even if less useful) for a remote machine to delegate job execution even further to another remote machine. Execution details are controlled by the programmer, and are communicated to the server process via command-line arguments. The current server implementation provides two customizations: a factory function for the executing process and a factory function for the `RetrieveProcessor`'s data queue. Naturally, these factory functions have to be serialized to be able to act as command line arguments, but the `remote` module provides convenience functions to do this (Figure 1).

Changes in the `scheduling` module

For the `remote` module, it is important that job lifecycle management and resource allocation are separated. This saves a tiny amount of unnecessary scheduling overhead, but more importantly, eliminates a possible mismatch between the allocated number of CPUs at the local and the remote machine. To this end, the resource allocation responsibility of the `scheduling.Manager` object is separated out into a series of new components. These encapsulate various strategies in resource management:

- `Allocated`: maps jobs onto a number of predefined `ExecutionUnit` instances. This is the behaviour of the `scheduling.Manager` object.
- `Unlimited`: each new request results in the creation of a new `ExecutionUnit` instance (the exact type is described with a factory function). There is no limit to the number of concurrently active `ExecutionUnit`s. This can work well with common queuing systems, which also possess built-in scheduling functionality. After the job has finished, the `ExecutionUnit` is destroyed, and recreated when necessary.
- `UnlimitedCached`: similar to `Unlimited`, but after jobs finish, the idle `ExecutionUnit` is stored, and preferentially re-used. This is designed to save time on the data queue

```
from libtbx.queuing_system_utils import remote

import subprocess

server = subprocess.Popen(
  (
    "ssh",
    "mymachine",
    remote.server_process_command_line(
      job_factory = multiprocessing.Process,
      queue_factory = multiprocessing.Queue,
      ),
    ),
  stdin = subprocess.PIPE,
  stdout = subprocess.PIPE,
  )
client = remote.SchedulerClient(
  instream = server.stdout,
  outstream = server.stdin,
  )

...use client.Job as multiprocessing.Process...

client.close()
```

Figure 1
Creating a remote job class. Execution on the remote system is made using `multiprocessing.Process` (`job_factory` argument), and these processes use a `multiprocessing.Queue` data queue to return a result to the server process (`queue_factory` argument). In this case, `ssh` is used to establish a network connection between the two hosts, but alternatives are available (e.g. `paramiko`, https://github.com/paramiko/paramiko). Input and output file handles are opened and passed onto the client).

creation, since apart from an initial growth period, no new `ExecutionUnit` instances are created in a stationary execution flow.

The previous `scheduling.Manager` class is replaced with a function that creates the new `scheduling.Scheduler` class with the `Allocated` strategy for backward compatibility.

In addition, a new processor class (`FetchProcessor`) is provided to be used with primarily remote jobs. However, it can also be employed with other job types, although the use a data queue may not be possible to avoid (Figure 2).

Network optimizations

Network data exchange can be slow, since there is a delay for a request to reach the remote host and a response to arrive. For this reason, the module packages submission and status requests, and only executes them when a certain number of requests has been exceeded, or a certain time limit has been reached. Optimal parameters seem to depend on the speed of the network (fast/slow) and the number of CPUs allocated.

File operations

The module can send input data along through the network connection, and therefore does not depend on a shared file system. However, when multiple calculations require the same input data, it could be more efficient to transfer this as a file. While there is no generic support built into the module (since file transfer is entirely a network operation), the basic case when the file system is shared between the two systems is supported. This is achieved by changing the working

```
...create client...

from libtbx.queuing_system_utils import scheduling

manager = scheduling.Scheduler(
  handler = scheduling.Allocated(
    units = [
      scheduling.ExecutionUnit(
        factory = client.Job,
        processor = scheduling.FetchProcessor.Unpack(),
        )
      for i in range( 8 )
      ]
      + [
      scheduling.ExecutionUnit(
        factory = multiprocessing.Process,
        processor = scheduling.RetrieveProcessor(
          queue = multiprocessing.Queue(),
          ),
        priority = 1,
        )
      for i in range( 4 )
      ]
    )
  )

...use manager as before...
```

Figure 2

Using remote jobs with the `scheduling` module. The example allocates 12 CPUs, 4 on the local and 8 on a remote machine. It is also possible to use multiple remote machines. The `priority` argument controls the preference of selecting nodes. In the example, the CPUs in the local machine are used first.

directory of the server to correspond to that of the client, so that relative paths are equivalent.

Job and queue factories

The Python Standard Library `pickle` module does not support serialization for object methods. However, factory functions are sometimes objects methods, and it is important for efficiency reasons that the object instance is not recreated every time a factory is called. For example, when using the `Queue` method of a `multiprocessing.Manager` object to create a data queue, it is important that the `Manager` object is created only once, because it spawns a new process to manage the connections, and it would be very inefficient for each data queue to have its own managing process. The remote `module` provides the `RemoteFactory` class to help with this issue. It requires a deferred call to the object constructor, and the name of the object method that will act as the factory function. The object is created on first access, and is reused for subsequent calls. A full example is shown on Figure 3.

```
from libtbx.queuing_system_utils import scheduling
from libtbx.queuing_system_utils import remote
import subprocess
import multiprocessing

arg = remote.server_process_command_line(
  job_factory = multiprocessing.Process,
  queue_factory = remote.RemoteFactory(
    calculation = multiprocessing.Manager,
    method = "Queue",
    ),
  )

server = subprocess.Popen(
  ( "ssh", "mymachine", arg ),
  stdin = subprocess.PIPE,
  stdout = subprocess.PIPE,
  )
client = remote.SchedulerClient( server.stdout, server.stdin )

manager = scheduling.Scheduler(
  handler = scheduling.Allocated(
    units = [
      scheduling.ExecutionUnit(
        factory = client.Job,
        processor = scheduling.FetchProcessor.Unpack(),
        )
      for i in range( 4 )
      ]
    )
  )
import math
pfor = scheduling.ParallelForIterator(
  calculations = ( math.sin, ( i, ), {} ) for i in range( 100 ) ),
  manager = manager,
  )
for ( parms, res ) in scheduling.SubmissionOrder( parallel_for = pfor ):
  try:
    result = res()

  except Exception, e:
    print "Job: %s : exception '%s'" % ( parms, e )

  else:
    print result

client.close()
```

Figure 3
Parallel execution of an arbitrary calculation (in this case, `math.sin`) on a remote machine. Data queues are created using an object method, which is serialized using the `RemoteFactory` helper class.

Table 1
Benchmark results of executing 100 fast (100 ms) jobs on 4 CPUs using `multiprocessing.Process` as execution technology via a `scheduling.Manager` in various setups. The remote machine is about 5000 miles away from the local.

Description	**Timing**
Direct (local execution)	5.43 s
Same machine, remote execution via `subprocess`	10.15 s
Same machine, remote execution via SSH	10.58 s
Remote machine via remote execution (SSH)	15.36 s

Table 2
Benchmark of executing 10 jobs that take about 1 s each, using 10 CPUs on a Sun Grid Engine cluster. Sun Grid Engine is accessed using the `processing` module. The remote machine is the same as in the previous benchmark.

Description	**Timing**
Direct (local execution)	14.98 s
Remote machine via remote execution (SSH)	15.83 s

Overheads

Analysis of benchmarks (Table 1) shows that there seems to be a per-job overhead that consists of two parts:

1. Serialization overhead, which includes pickling and unpickling the requests and the results. This is about 50 ms/job, and calculated by comparing the timings of direct execution and the no-network remote execution.
2. Network overhead, which scales with the speed of the connection. For a very fast network (and physically close machines), this is negligible, but starts to increase with the distance. For the test case, it contributes an additional 50 ms overhead for machines that are about 5000 miles apart and connected by a fast network.

Although this looks quite significant (effectively tripling the execution time), it is only because the jobs are very fast, and also the job startup overhead is very small. When switching to longer calculations (Table 2), and execution technology that involves a larger overhead, the difference between the two are negligible.

References

Bunkóczi, G. & Echols, N. (2012). *Computational Crystallography Newsletter* **3**, 37-42.

phenix.ensemble_refinement: a test study of apo and holo BACE1

B. Tom Burnley and Piet Gros
Crystal and Structural Chemistry, Bijvoet Center for Biomolecular Research, Utrecht University

Correspondence email: B.T.Burnley@uu.nl & P.Gros@uu.nl

Introduction

`phenix.ensemble_refinement` (Burnley et al. 2012) combines molecular dynamic (MD) simulations with X-ray structure refinement to generate ensemble models fitted to diffraction data. It is an evolution of the 'time-averaging' method first proposed by Gros et al. in 1990 (Gros, van Gunsteren, and Hol 1990) and now utilizes a maximum-likelihood target function, a dual explicit-bulk solvent model and introduces a TLS fitting procedure that absorbs rigid-body motions such that short MD simulations can be used to sample local disorder. The resulting ensemble model represents, as a Boltzmann-weighted population of structures, the simulation trajectory and provides implicit modeling of anisotropic and anharmonic motions. This family of structures provides two main advantages: 1) a reduction in R_{free} compared with traditional single structure models and 2) quantification and visualization of protein dynamics that have been demonstrated to correlate with biological function. Methodological implementation and testing is described in detail in Burnley et al. 2012; here we focus on a practical usage of the method.

For this test case we examine β-Secretase (β -site amyloid precursor protein-cleaving enzyme1; BACE1), in the apo and holo state. BACE1 is a transmembrane aspartic protease that cleaves β-amyloid precursor protein and is the putative rate-limiting enzyme in the amyloid β-peptide (Aβ) pathway (Sinha 1999). Aβ is likely responsible for the Alzheimer's disease cascade and therefore BACE1 is a primary target for the development of inhibitors to treat/prevent Alzheimer's disease (Xu et al. 2011). The importance of this protein has resulted in multiple entries in the PDB (Berman et al. 2000). We selected two 1.6 Å resolution datasets deposited by Xu and co-workers (Xu et al. 2011) of the protease ectodomain in the apo state and holo state complexed with a nonpeptide inhibitor (*N0*-[(2*S*,3*R*)-4-(cyclopropylamino)-3-hydroxy-1-phenylbutan-2-yl]-5-(methylsulfonylamino)-*N*-[(1*R*)-1-phenylethyl]benzene-1,3-dicarboxamide; BSIIV) for analysis by ensemble refinement (PDB codes 3TPJ and 3TPP respectively). Here we detail the phenix.ensemble_refinement process, from preparing input data through parameter optimization to structure analysis.

Preparing input data

`phenix.ensemble_refinement` uses a set of input files similar to `phenix.refine (Adams et al. 2010),` i.e. structure (e.g. .pdb format) and reflection files (e.g. .mtz) are required and specified parameter definitions (e.g. .eff) and ligand restraints (e.g. .cif) should be supplied as needed. Ensemble refinement is dependent on the quality of the input structure therefore the input structure should be post-traditional refinement and thus ready for deposition, or as deposited, in the PDB. It should be noted that if TLS was used in the refinement protocol the TLS contributions must be present in the atom records (see `phenix.tls`).

For this test case the PDB structures and structure factors were downloaded from the PDB_REDO server (Joosten et al. 2010). One of the advantages of the PDB_REDO server is that it automatically builds and refines missing side-chains, and the usage of complete side-chains is recommended for ensemble refinement. However, we do not recommend building long sequences (>~4 residues) of highly disordered regions such as loops or termini if there is not sufficient electron density to support placement when using standard model building procedures. In the BACE1 structures loop 153-173 and loop 308-319 have missing residues, as do both the N and C termini and after examining in these regions. It was not possible to build any additional residues. `phenix.ready_set` was then used to generate ligand restraints and to add explicit hydrogen atoms to the PDB file. The structures were further refined using `phenix.refine` and TLS groups were determined using `phenix.find_tls`. Table 1 shows the refinement statistics. Prior to use with `phenix.ensemble_refinement` any alternative conformations should be removed and

Table 1. BACE1 apo and holo dataset and refinement statistics as deposited in the PDB and following re-refinement using `phenix.refine` and `phenix.ensemble_refinement`.

	Apo	Holo
PDB code	3tpj	3tpp
Space group	*C*222₁	*C*222₁
Unit-cell parameters		
a (Å)	104.4	104.5
b (Å)	128.7	128.2
c (Å)	76.7	76.5
Resolution range (Å)	35.8 - 1.6	49.1 - 1.6
PDB structure		
R_{work} (%)	17.8	15.5
R_{free} (%)	19.6	18.0
phenix.refine		
R_{work} (%)	14.4	14.0
R_{free} (%)	15.8	16.1
Geometric rmsd		
Bond lengths (Å)	0.0	0.0
Bond angles (°)	1.3	1.4
Dihedral angles (°)	12.5	13.1
Ramachandran		
Disallowed residues (%)	0.0	0.0
Allowed residues (%)	2.3	1.3
Favored residues (%)	97.7	98.7
phenix.ensemble_refinement		
R_{work} (%)	12.1	12.1
R_{free} (%)	14.7	14.6
Geometric rmsd (centroid)		
Bond lengths (Å)	0.008	0.007
Bond angles (°)	1.113	1.151
Dihedral angles (°)	8.38	8.51
Geometric rmsd (per structure)		
Bond lengths (Å)	0.014	0.013
Bond angles (°)	1.713	1.728
Dihedral angles (°)	17.12	17.22
Ramachandran (centroid)		
Disallowed residues (%)	1.6	0.5
Allowed residues (%)	1.3	1.9
Favored residues (%)	97.0	97.6
Ramachandran (per structure)		
Disallowed residues (%)	2.5	2.6
Allowed residues (%)	4.4	5.1
Favored residues (%)	93.1	92.3

occupancies re-adjusted. Alternative conformations will be sampled automatically during the simulation.

Both 3TPJ and 3TPP datasets were collected at 100 K and exhibit the $C222_1$ space group, also both have similar unit cell parameters and resolution ranges (Table 1). When possible it is advantageous to use analogous datasets such as these as it allows for more direct structural comparison. Care should be taken if, for example, datasets have different space groups or cell parameters as differences in crystal packing may influence atomic fluctuations and must be considered when comparing the resultant ensembles.

Running ensemble refinement

`phenix.ensemble_refinement` can be run from the command line or from the *PHENIX* GUI. There are several parameters that should be defined and/or optimized by the user. Parameter optimization is covered in the next section; here we examine the parameters defined for the BACE1 simulations that remain consist across all runs.

TLS groups used in the TLS fitting procedure should be defined by the user and do not correlate with those used in standard refinement (i.e. it is not recommended to use `phenix.find_tls` for ensemble refinement). Each defined TLS group will fit the rigid-body motion of that group such that they are not required to be sampled during the simulation. This reduces the conformational space that needs to be sampled in the simulation and simplifies the output ensemble. In practice it is best to use complete chains or domains for the defined groups (e.g. tls_group_selections = 'chain a') unless there is good reason to do otherwise. In the case of BACE1 it has a defined 2-lobe architecture so each lobe is assigned to a separate TLS group. The N terminal lobe occurs between residues -4 to 180 and the C-terminal lobe from residue 181 to 386. Both the apo and holo structures contain a number of non-solvent ligands and these must be assigned to the correct TLS group. Visual inspection in a molecular graphics program, e.g. PyMOL (Schrödinger 2010) is used to determine which lobe each ligand is closest to. For the apo dataset the following TLS selections were used to define two groups: tls_group_selections = 'resseq -4:180 or resseq 394 or resseq 395 or resseq 398:400 or resseq 402:406', tls_group_selections = 'resseq 181:386 or resseq 396:397 or resseq 401 or resseq 999'. The selection strings can be defined using the standard nomenclature. It is not necessary to include water atoms as these are automatically assigned to the nearest TLS groups during the simulation.

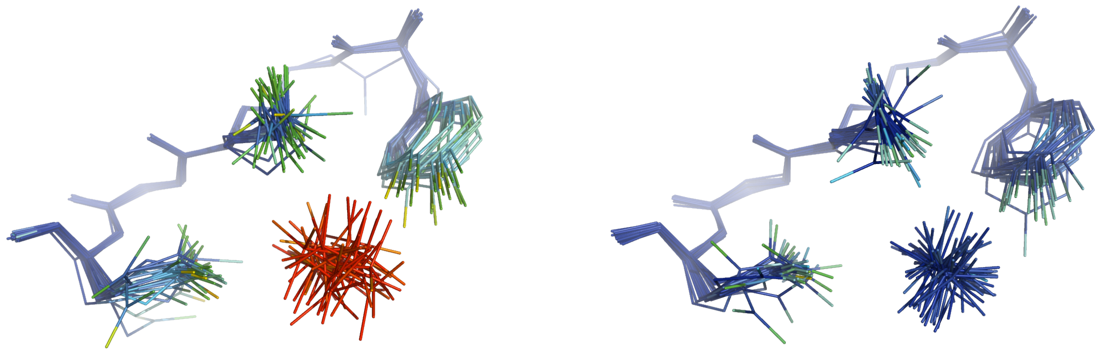


Figure 1. Effect of atom occupancy on kinetic energy. Two simulations of holo BACE1 were performed with difference occupancies for the SO_4^{2-} hetgroup, 1.0 on the left and 0.4 on the right. Atoms are colored by kinetic energy ranging from 0.0 to 2.0 kcal/mol (blue to red).

For weakly bound non-specific ligands it may be appropriate to add harmonic restraints to prevent stochastic displacement during the simulation. For apo BACE1 harmonic restraints we defined as: harmonic_restraints.selections = 'resname so4 or resname cl or resname ure'. This restrains all selected atoms to the x,y,z co-ordinate positions given in the input structure. To allow for any uncertainty in position the harmonic restraints have a 'flat bottom' so deviations less than a given slack value (distance in Å) will generate no force (e.g. harmonic_restraints.slack = 2.0). For the holo structure it is not necessary to add harmonic restraints for the high-affinity, well-ordered, inhibitor.

At present non-solvent ligands with non-unity occupancies cannot be replaced/reinserted in the simulation *a la* the explicit solvent atoms (see Burnley et al. 2012 for details). Atom occupancy is defined from the input structure and if applicable this should be refined, when possible, using single-structure methods prior to ensemble refinement. Errors in occupancy can be detected during ensemble refinement: an atom with too high occupancy will sample a greater amplitude and frequency of disorder than neighboring atoms and exhibit excessive kinetic energy. Atomic kinetic energies (kcal/mol x 10^{-1}) can be outputted in the final PDB ensemble in the occupancy column by setting the parameter: `output_running_kinetic_energy_in_occupancy_column=True`. Figure 1 shows a sulphate group in holo BACE1, where two simulations were performed with occupancies of 1.0 and 0.4 (0.4 was calculated using combined occupancy and ADP refinement in `phenix.refine`). An occupancy of 1.0 results in the sulphate group sampling excessively high kinetic energies whereas at the refined occupancy of 0.4 these atoms exhibit kinetic energies similar to that of the neighboring environment. For datasets where stable refinement of occupancies is not possible, e.g. at low resolution, we recommend running multiple simulations with different occupancy values (e.g. 0.3,0.4,0.5…) .

Finally the water picking parameters can be adjusted depending on the dataset resolution. As standard the water positions are assessed using cutoffs of 3.0 and 1.0 σ for mFo-DFm and 2mFo-DFm maps respectively. For structures with resolution better than ~1.7 Å, better results maybe obtained by lowering the mFo-DFm threshold (e.g. `ensemble_ordered_solvent.primary_map_cutoff = 2.5`). For both the apo and holo BACE1 structures better quality mFo-DFm difference maps and lower R_{free} values where found when using a cutoff of 2.5 σ was used (values of 2.5 and 3.0 σ were tested).

The simulation can be started from the GUI or from the command line using `phenix.ensemble_refinment`. The simulation length depends on several factors including: the number of atoms in the ASU, dataset resolution and hardware configuration. In our experience runtime is typically between several hours and a few days. Using a 1.9 GHz processor for BACE1 the CPU time was 3.6 days. While the simulations are running no user input is required and, due to the length of CPU time, it is highly recommended to test alternative parameters in parallel.

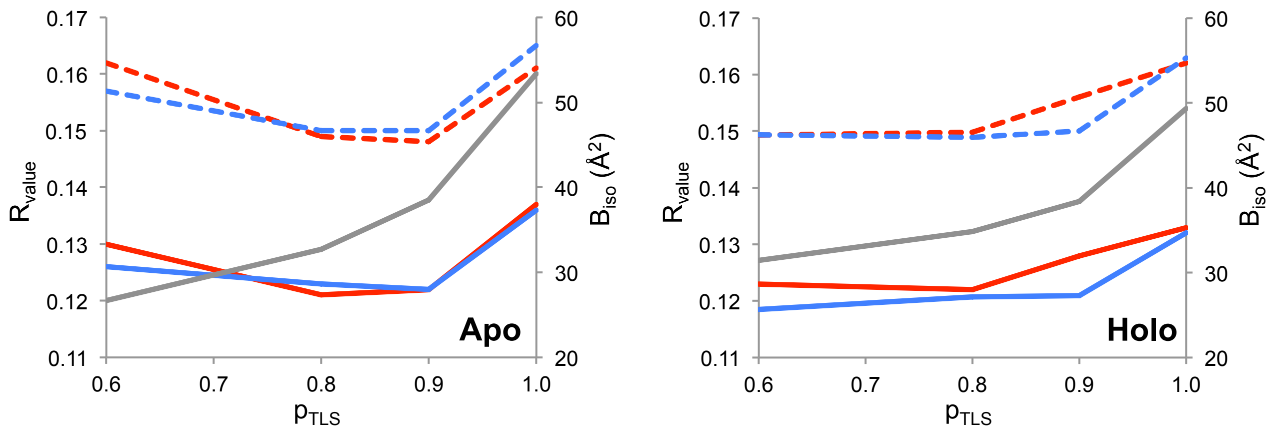


Figure 2. Ensemble refinement parameter optimization for apo and holo BACE1 datasets. An array of p_{TLS} values (0.6 – 1.0) and T_{bath} (2 and 10 K shown in red and blue respectively) was performed and the resulting R_{work} and R_{free} values are shown in solid and dashed lines respectively. The isotropic equivalent of the fitted TLS B-factors are shown in grey, averaged over all non-solvent, non-hydrogen atoms.

Optimizing simulation parameters

There are two main empirical parameters (p_{TLS}, T_{bath}) that affect the running and the outcome of ensemble refinement, and it is critical to optimize both on a dataset by dataset basis.

p_{TLS} (e.g. `ptls=0.9`) defines the fraction of atoms included in the TLS fitting procedure (for details see Burnley et al. 2012). The optimum value for this parameter cannot be determined *a priori*. Setting up a parallel array of simulations with different values (e.g. 1.0, 0.9, 0.8, 0.6) is recommended. The TLS fitting process requires that the number of non-hydrogen, non-solvent atoms per TLS group multiplied by the p_{TLS} fraction to be greater than 63 (enforcing a data:parameter ratio greater than 3 for TLS fitting). If this is not the case, p_{TLS} will be automatically increased. In the case that a TLS group has less than 63 non-solvent, non-hydrogen atoms then an isotropic model, based on the Wilson B-factor, is automatically applied.

T_{bath} (e.g. `wxray_coupled_tbath_offset=5`) set the temperature bath offset (K) and, counter intuitively, T_{bath} controls the X-ray weight. The X-ray weight is modulated in situ such that the simulation runs at the target temperature (e.g. `cartesian_dynamics.temperature=300`). The simulation uses velocity scaling to maintain the target temperature. This is coupled to a temperature thermostat which is set as: thermostat = cartesian_dynamics.temperature - wxray_coupled_tbath_offset. The non-conservative time-averaged X-ray force generates heat (Gros, van Gunsteren, and Hol 1990), thus larger offsets increase the X-ray weight. The default value is 5 K and, for example, values of 2 and 10 K may also be tested.

For both the apo and holo BACE1 datasets a parallel array of eight separate simulations was performed with a grid of p_{TLS} (1.0, 0.9, 0.8 and 0.6) and T_{bath} (2 and 10 K), as shown in Figure 2. A shallow optimum in R_{values} is observed and with optimum values of 0.9 and 2 K for p_{TLS} and T_{bath} respectively for the apo dataset and 0.6 and 2 K for the holo dataset.

Ensemble refinement is based on MD simulations that are stochastic by nature. Therefore to aid analysis it is useful to run a number of random seed repeats (e.g. `random_seed=26556257`). This seed is used in the initial velocity assignment and the alternative trajectories that arise can be used as a guide to the reproducibility of the simulation as a whole and also the individual atom fluctuations (see Burnley et al., 2012, Figures 4, 5 & 10). In total five random number seed repeats where run for both BACE1 datasets producing consistent global R_{values}, see Table 2, the best of which (as judged by R_{work} *and* R_{free} *values*) were then selected for the subsequent structural analysis and the refinement statistics for these are reported in Table 1. In this case we found that the R_{values} improve for both the apo and holo datasets after ensemble refinement.

Table 2. R_{values} and number of structures in output PDB ensemble model for five random number seed repeats of `phenix.ensemble_refinement` for apo and holo BACE1.

	Apo			Holo		
Repeat	R_{work} (%)	R_{free} (%)	*Structures*	R_{work} (%)	R_{free} (%)	*Structures*
1	12.1	14.7	134	12.1	14.6	107
2	12.1	14.7	178	11.9	14.7	134
3	12.2	14.8	107	12.1	14.8	134
4	12.1	14.8	134	12.3	14.9	89
5	12.2	14.8	134	12.4	15.0	134
Mean	*12.1*	*14.8*	*137*	*12.2*	*14.8*	*120*

Two sets of geometry statistics are reported in Table 1 for the ensemble structures: 'centroid' and 'per structure' (see Burnley et al. 2012 for complete definitions). This is because the ensemble as a whole is fitted to the data. Therefore the structures represent a distribution that reflects fluctuations around ideal geometric values, i.e. each individual restraint in each individual structure does not necessarily have to fit the ideal geometric value and a degree of variation is expected. However, considering the model as a whole ensemble, the restraints deviations can be expected to oscillate around an ideal value therefore 'centroid' statistics are provided. The geometric deviations are also calculated for each structure separately and then averaged across the ensemble giving the 'per structure' values. While large deviations can be observed when evaluated per structure, the centroids of the distribution show fluctuations around the correct value for the BACE1 datasets, see Table 1.

By default phenix.ensemble_refinement outputs the following files: log (.log), map coefficients (.mtz), geometry of input structure (.geo) and the ensemble structure (.pdb.gz). As the ensemble PDB structure can contain hundreds of models it is zipped to reduce size and should be unzipped (using gunzip for instance) before use. During the simulation thousands of sets of coordinates are sampled, so to simplify the final model the minimum number of models is found that reproduces the R_{free} value of the whole trajectory within a 0.25 % tolerance. To generate these reduced ensembles every 1st,2nd,3rd…nth model of the total number of coordinates sets stored during the acquisition phase of the simulation are selected and the R_{values} of these concerted ensembles are recalculated. The relationship between the number of structures and the R_{values} is shown for apo and holo BACE1 in Figure 3. There is some fluctuation in the recalculations and as such the number of models in the final PDB can vary as is shown in Table 2 for the five number seed repeats.

Structural analysis

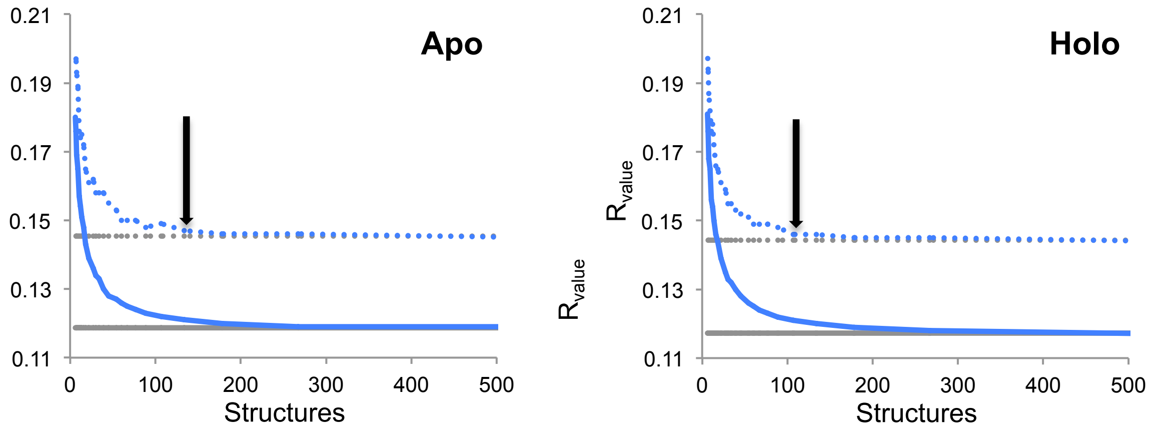


Figure 3. Determination of the number of models in the output ensemble PDB. For the best apo and holo simulation the charts shows the recalculation of R_{work} and R_{free} (solid and dashed line respectively) with different numbers of structures in the ensemble models (blue), the minimum number of models to reproduce complete trajectory R_{values} (shown in grey) within a 0.25% tolerance is selected for output (highlighted with arrow). In this case, apo and holo BACE1 contained 134 and 107 models.

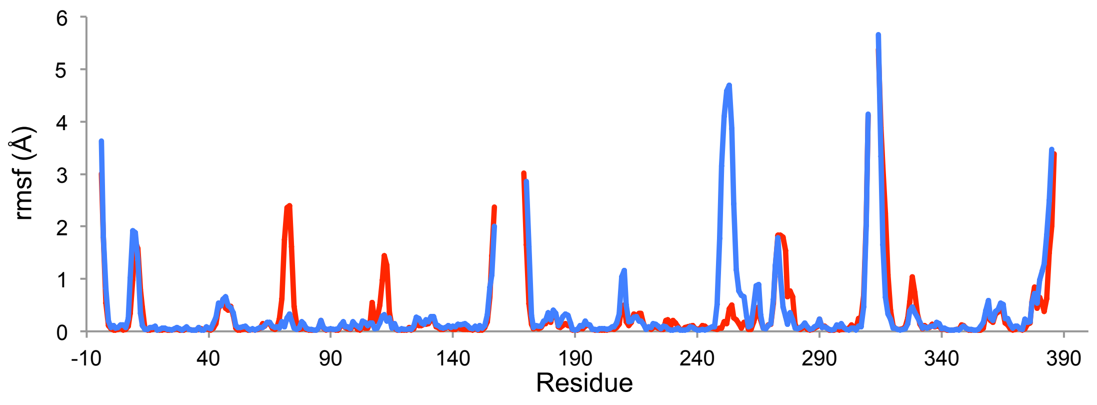


Figure 4. Backbone rmsf of apo (red) and holo (blue) BACE1 ensembles. Human aspartic protease residue numbering is used. Values were calculated used ens_tools.py script for PyMOL.

Visual analysis of the structures and maps can be done using standard programs like PyMOL and COOT (Emsley et al. 2010). When using PyMOL the models can be viewed sequentially as a movie using the play button or superimposed together via the show all states option (via Movie header or the command 'set all_states, 1'). If the user wants to view atomistic atom B-factors or kinetic energies (see above) for each individual model in the ensemble the additional option 'discrete=1' should be specified with the load command. Viewing in COOT may be improved by reducing the number of models in the ensemble to between 5 to 50 and this can be done using command line program `phenix.ensemble_reducer`, which allows the further truncation of the ensemble. Once the ensemble models have been produced no manual model building is possible. If, on inspection, an error has occurred, for instance a ligand has become displaced from its binding site than the simulation parameters need to altered, e.g. harmonic restraints maybe added or adjusted, and ensemble refinement should be re-run.

The structural fluctuation in the ensembles can be quantified by calculating the rmsf and this is shown per residue for apo and holo BACE1 in Figure 4. Many programs can process ensemble structures to calculate rmsf and other metrics, and we have provided a script for PyMOL that can also do this. 'ens_tool.py' is available from $PHENIX/cctbx_project/mmtbx/refinement/ensemble_refinement/ and the command 'ens_rmsf, selection' will return the rmsf values for selected atoms. Figure 4 shows the highest deviations for both apo and holo BACE1 datasets occur in the N and C terminal regions as well loop 153-173 and loop 308-319. These regions contain missing residues so it is unsurprising to observed high proximal mobility located here. A high rmsf value is returned for tyr71 in the apo structure. This residue is located in the 'flap' region, a hairpin loop found in the N-terminal lobe and spanning residues 67-75. Other structures deposited in the PDB show that this loop can adopt multiple conformations. Figure 5 shows the flap region in the apo and holo forms of BACE1. When the active site is empty the flap region is mobile, however, upon addition of the inhibitor BSIIV the flap folds over the binding pocket such that tyr71 can make weak hydrophobic contacts with the inhibitor and thus trapping the flap in a single conformation. Figure 6 shows the remainder of the binding pocket and here a general tightening of the residues surrounding the ligand is observed upon binding. In particular, the conserved catalytic dyad formed by asp32 and asp228 becomes more ordered in presence of the inhibitor, see Figure 7. Interesting, the holo structure is highly mobile between residues 248-256, a surface turn is located in the C-terminal lobe, whereas the apo structure is more rigid. This pattern is repeated for all five random number seed repeats for both apo and holo datasets, and could have functional role for instance in offsetting the loss of entropy upon binding and/or as a method of signaling.

In conclusion, `phenix.ensemble_refinement` is suitable for standard protein datasets and does not require specialist experimental techniques or hardware. For both apo and holo BACE1 it provides multi-structure models that fit well to the diffraction data. The TLS fitting procedure absorbs global motions and this gives the

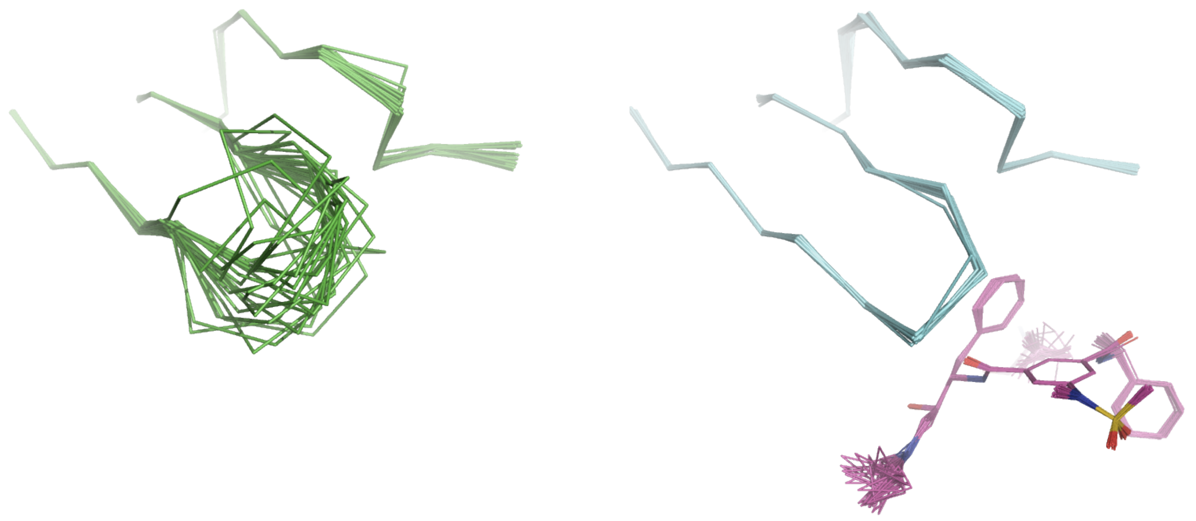


Figure 5. Comparison of the 'flap' region in the ensemble structures of apo (left, green ribbon) and holo (right, blue ribbon) BACE1. The hairpin loop (residues 67-75) becomes significantly more ordered in the presence of the inhibitor. 25 structures are shown for each ensemble.

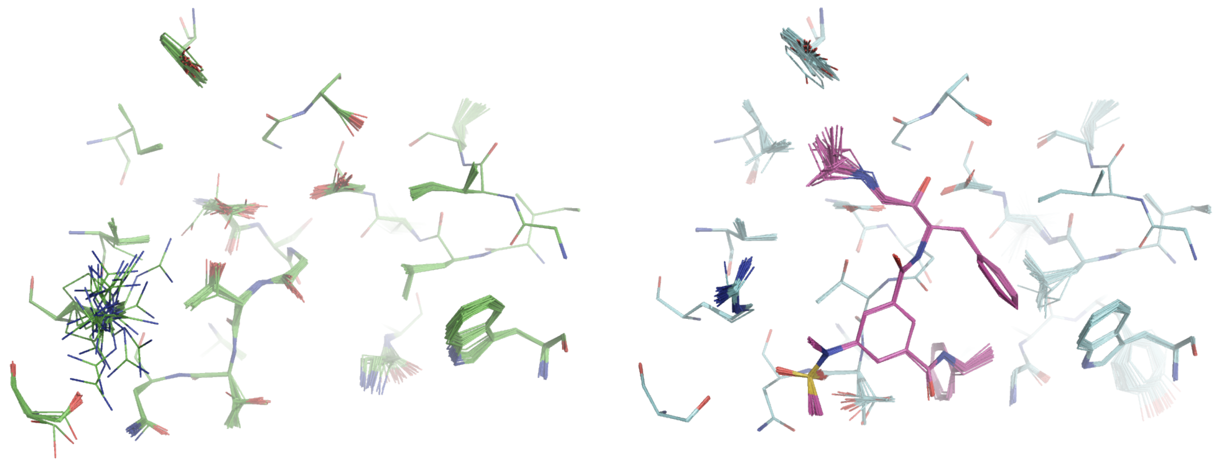


Figure 6. The binding pocket of apo (left, green carbon atoms) and holo (right, blue carbon atoms) BACE1 rigidifies in the presence of the inhibitor (right, pink carbon atoms). 25 structures are shown for each ensemble.

opportunity to visualize and quantify the intramolecular dynamics of the molecule. As such, it provides a tool to probe structure, function and dynamics from diffraction data.

Caveat Emptor

There are some important caveats that should be noted to prevent accidental misinterpretation of the models produced using ensemble refinement. 1) Ensemble refinement is recommended for use with datasets with a resolution of 3 Å or better. 2) The number of structures in the output ensemble is variable and should not be over interpreted. 3) The ensembles are a Boltzmann-weighted population and therefore contain high-energy conformations. 4) Dynamic rates cannot be calculated from the simulations. The X-ray force is non-conservative and accelerates atomic sampling. This allows movements that occur at timescales several orders of magnitude greater than a 100ps simulation, e.g. side chain sampling or loop motions, to be modeled in the (relatively) short simulation time but therefore the rate of conformation exchange cannot be derived. 5) No information regarding correlated motion is provided by the time- and spatially-averaged X-ray restraints although local correlations may be observed via short-range steric interactions.

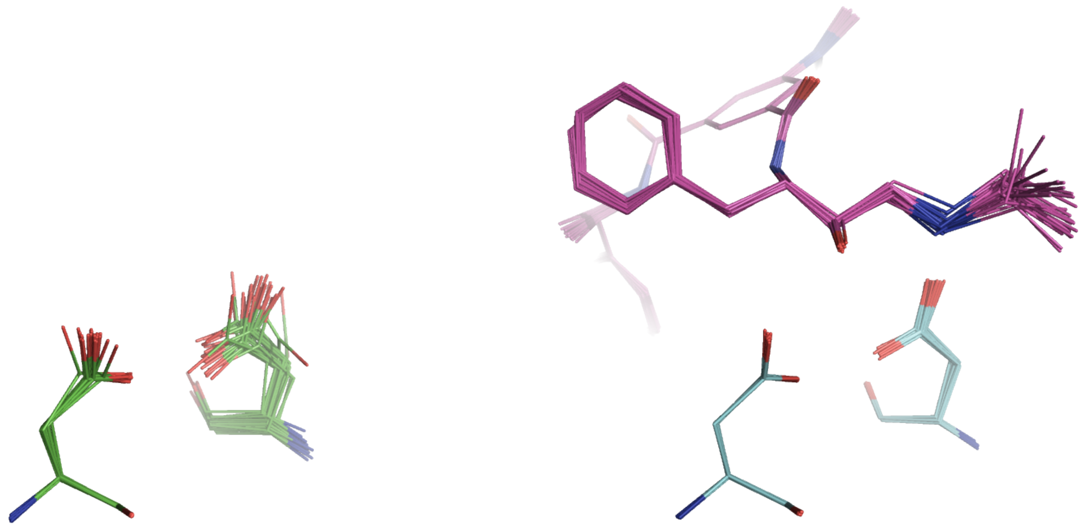


Figure 7. Both asp32 and asp228 residues that form the conserved catalytic dyad exhibit higher mobility in the absence of the inhibitor. Apo BACE1 ensemble is shown left (green carbon atoms) and holo ensemble is shown right (blue and pink for protein and inhibitor carbon atoms respectively). 25 structures are shown for each ensemble.

References

Adams, P D, Pavel V Afonine, Gábor Bunkóczi, Vincent B Chen, Ian W Davis, Nathaniel Echols, Jeffrey J Headd, et al. 2010. "PHENIX: a Comprehensive Python-based System for Macromolecular Structure Solution." *Acta Crystallographica Section D* 66 (2) (February): 213–221. doi:10.1107/S0907444909052925.

Berman, Helen M., John Westbrook, Zukang Feng, Gary Gilliland, T. N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. 2000. "The Protein Data Bank." *Nucleic Acids Research* 28 (1): 235–242. doi:10.1093/nar/28.1.235.

Burnley, B. T., P. V. Afonine, P. D. Adams, and P. Gros. 2012. "Modelling Dynamics in Protein Crystal Structures by Ensemble Refinement." *eLife* 1 (0) (January 1): e00311–e00311. doi:10.7554/eLife.00311.

Emsley, P., B. Lohkamp, W. G. Scott, and K. Cowtan. 2010. "Features and Development of *Coot*." *Acta Crystallographica Section D Biological Crystallography* 66 (4) (March 24): 486–501. doi:10.1107/S0907444910007493.

Gros, P, W F van Gunsteren, and W G Hol. 1990. "Inclusion of Thermal Motion in Crystallographic Structures by Restrained Molecular Dynamics." *Science (New York, N.Y.)* 249 (4973) (September 7): 1149–1152.

Joosten, R. P., T. A. H. te Beek, E. Krieger, M. L. Hekkelman, R. W. W. Hooft, R. Schneider, C. Sander, and G. Vriend. 2010. "A Series of PDB Related Databases for Everyday Needs." *Nucleic Acids Research* 39 (Database) (November): D411–D419. doi:10.1093/nar/gkq1105.

Schrödinger, LLC. 2010. "The PyMOL Molecular Graphics System, Version 1.3r1."

Sinha, S. 1999. "Cellular Mechanisms of Beta -amyloid Production and Secretion." *Proceedings of the National Academy of Sciences* 96 (20) (September 28): 11049–11053. doi:10.1073/pnas.96.20.11049.

Xu, Yechun, Min-jun Li, Harry Greenblatt, Wuyan Chen, Aviv Paz, Orly Dym, Yoav Peleg, et al. 2011. "Flexibility of the Flap in the Active Site of BACE1 as Revealed by Crystal Structures and Molecular Dynamics Simulations." *Acta Crystallographica Section D Biological Crystallography* 68 (1) (December 9): 13–25.