Tips and tricks for manual model building of atomic models into cryoEM maps.

Oliver Clarke

An atomic model is a compact interpretation of the density map in light of prior knowledge (both specific and general).

- Aim is to build a model that is consistent with **both** the density map and everything we independently know about the structure/composition of the macromolecule of interest, both specifically and in terms of our general knowledge of protein structure and chemistry.

- At medium resolution (3-5 Å), this still requires manual building. Even the best autobuilt model still requires a lot of manual inspection and correction in most cases. (generates many fragments which need inspection, correction, merging)

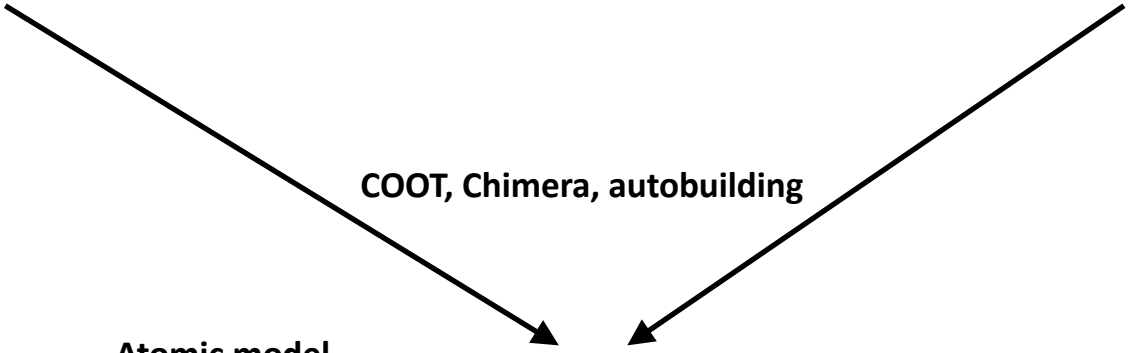
- Tradeoff between available prior knowledge and required resolution for atomic modelling – at the extremes, if a complete crystal structure is already available, 10Å data may be sufficient, while if no sequence/composition data is available even 3Å may not suffice.

Prior knowledge

- Protein sequence and derived info (secondary structure predictions, covariation/conservation, patterns of large/aromatic residues), disorder & contact prediction
- Crystal structures (+ homology models)
- Knowledge of protein structure, folding, chemistry, geometry.

Density map

- Resolution (+ local resolution, + map modification/sharpening)
- Patterns of large/small/absent sidechains
- Sharpening and density modification
- Conformational/compositional heterogeneity



COOT, Chimera, autobuilding

Atomic model

- If possible, unique model that agrees with both density map and priors
- Otherwise (and per region), specify ambiguity (w/UNK residues and numbering or Ca only model)
- Validation not just (or even mostly) about overfitting.
- Identify, analyse, fix errors.
- Direction and register of sequence fit.
- Ligand identification/assignment.
- No model is or ever will be perfect. That’s okay.

Before you start – make sure your maps are appropriately sharpened and low pass filtered! (and consider whether building is justified or whether further improvement of the reconstruction is required first)

- **Often it is helpful to build using multiple maps**. Assuming 3-3.5Å global res, I would suggest using a map filtered to the global resolution, one filtered to the best local resolution, and one filtered to ~4-4.5 Å (to better visualize connectivity).

- Try both simple B-factor sharpening and the approach used by *`phenix.auto_sharpen`*, which incorporates anisotropy removal. CisTEM *`sharpen_map`* also seems to give very good results in some cases.

- Also, if your map doesn't "look like" 4 Å, trust your eyes! If it is nominally 4Å and there are no sidechains visible, or your helices look "stretched", assess orientation bias (3D-FSC server: *https://3dfsc.salk.edu*), local resolution variation, and double check sharpening and masking parameters (are you *sure* you're looking at the sharpened map? Is the mask used for FSC calculation sensible?)

Prep for model building - what can we learn from the sequence alone?

Your protein sequence contains a lot of useful information which you can use to aid model building:

- Start by identifying boundaries of conserved domains (NCBI CDD: https://www.ncbi.nlm.nih.gov/Structure/cdd/; DELTA-BLAST also performs CD-search by default)
- Then identify suitable structural templates for building known domains: FUGUE, SPARKS-X, PHYRE2, MUSTER.
- Secondary structure, TM & disorder prediction (XtalPRED for overall summary; specific tools such as SPOT-DISORDER, SPIDER3 for best accuracy).
- Contact prediction from evolutionary couplings: EVFOLD & GREMLIN.
- Conservation analysis: Use favorite MSA algorithm (MUSCLE & CLUSTAL-OMEGA work well; TM-COFFEE, PRALINE-TM useful for membrane proteins) to create a sequence alignment of your protein with a few orthologs; gaps & insertions most commonly occur in loops/disordered regions. Useful as a guide during building.

Prep for model building - what can we learn from the sequence alone?

Your protein sequence contains a lot of useful information which you can use to aid model building:

- **Start by identifying boundaries of conserved domains (NCBI CDD: https://www.ncbi.nlm.nih.gov/Structure/cdd/; DELTA-BLAST also performs CD-search by default)**

- Then identify suitable structural templates for building known domains: FUGUE, SPARKS-X, PHYRE2, etc.

- Secondary structure, TM & disorder prediction (XtalPRED for overall summary; specific tools such as SPOT-DISORDER, SPIDER3 for best accuracy).

- Contact prediction from evolutionary couplings: EVFOLD & GREMLIN.

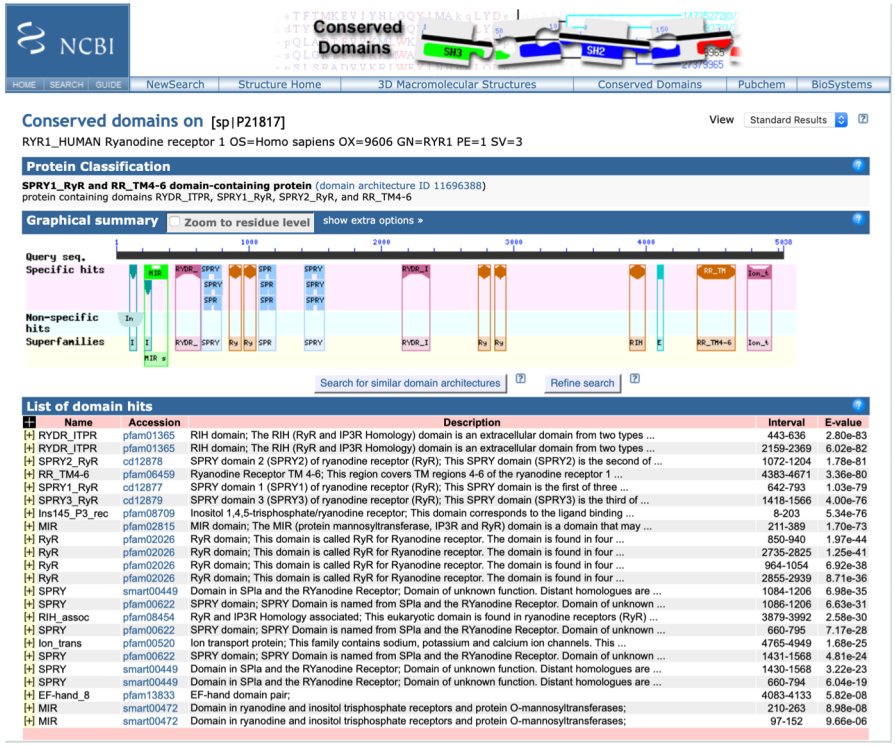
- Conservation analysis: Use favorite MSA algorithm (MUSCLE & CLUSTAL-OMEGA work well) to create a sequence alignment of your protein with a few orthologs; gaps & insertions most commonly occur in loops/disordered regions. Useful as a guide during building.

Prep for model building - what can we learn from the sequence alone?

Your protein sequence contains a lot of useful information which you can use to aid model building:

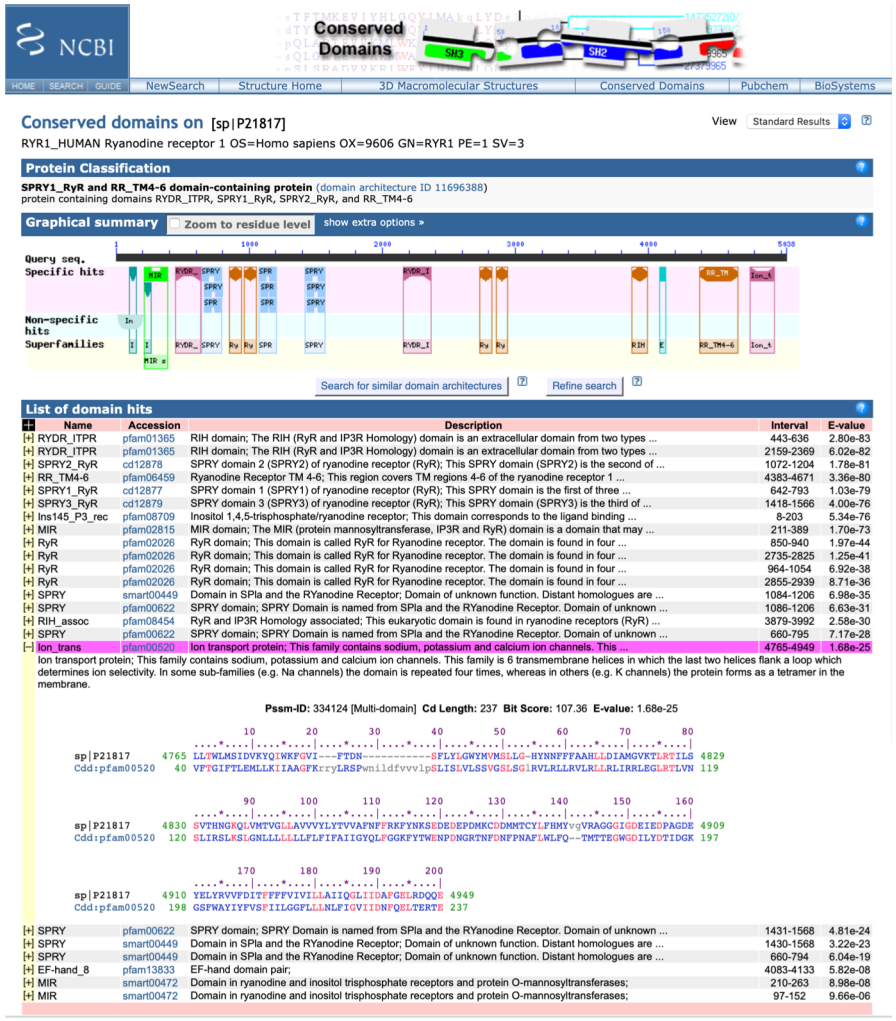
- **Start by identifying boundaries of conserved domains (NCBI CDD: https://www.ncbi.nlm.nih.gov/Structure/cdd/; DELTA-BLAST also performs CD-search by default)**
- Then identify suitable structural templates for building known domains: FUGUE, SPARKS-X, PHYRE2, etc.
- Secondary structure, TM & disorder prediction (XtalPRED for overall summary; specific tools such as SPOT-DISORDER, SPIDER3 for best accuracy).
- Contact prediction from evolutionary couplings: EVFOLD & GREMLIN.
- Conservation analysis: Use favorite MSA algorithm (MUSCLE & CLUSTAL-OMEGA work well; TM-COFFEE, PRALINE-TM useful for membrane proteins) to create a sequence alignment of your protein with a few orthologs; gaps & insertions most commonly occur in loops/disordered regions. Useful as a guide during building.

CDD provides a guide to domain level architecture, including sequence alignments & representative structures.

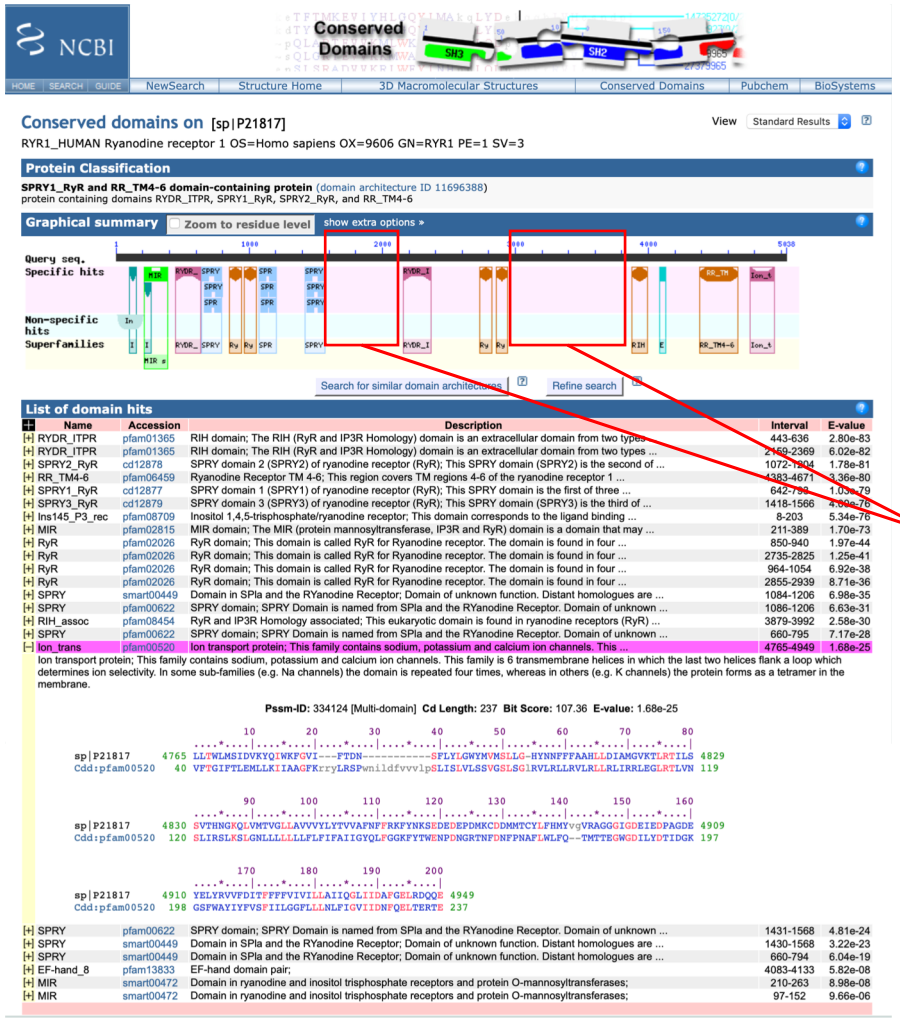


Name	Accession	Description	Interval	E-value
[+] RYDR_ITPR	pfam01365	RIH domain; The RIH (RyR and IP3R Homology) domain is an extracellular domain from two types ...	443-636	2.80e-83
[+] RYDR_ITPR	pfam01365	RIH domain; The RIH (RyR and IP3R Homology) domain is an extracellular domain from two types ...	2159-2369	6.02e-82
[+] SPRY2_RyR	cd12878	SPRY domain 2 (SPRY2) of ryanodine receptor (RyR); This SPRY domain (SPRY2) is the second of ...	1072-1204	1.78e-81
[+] RR_TM4-6	pfam06459	Ryanodine Receptor TM 4-6; This region covers TM regions 4-6 of the ryanodine receptor 1 ...	4383-4671	3.36e-80
[+] SPRY1_RyR	cd12877	SPRY domain 1 (SPRY1) of ryanodine receptor (RyR); This SPRY domain is the first of three ...	642-793	1.03e-79
[+] SPRY3_RyR	cd12879	SPRY domain 3 (SPRY3) of ryanodine receptor (RyR); This SPRY domain (SPRY3) is the third of ...	1418-1566	4.00e-76
[+] Ins145_P3_rec	pfam08709	Inositol 1,4,5-trisphosphate/ryanodine receptor; This domain corresponds to the ligand binding ...	8-203	5.34e-76
[+] MIR	pfam02815	MIR domain; The MIR (protein mannosyltransferase, IP3R and RyR) domain is a domain that may ...	211-389	1.70e-73
[+] RyR	pfam02026	RyR domain; This domain is called RyR for Ryanodine receptor. The domain is found in four ...	850-940	1.97e-44
[+] RyR	pfam02026	RyR domain; This domain is called RyR for Ryanodine receptor. The domain is found in four ...	2735-2825	1.25e-41
[+] RyR	pfam02026	RyR domain; This domain is called RyR for Ryanodine receptor. The domain is found in four ...	964-1054	6.92e-38
[+] RyR	pfam02026	RyR domain; This domain is called RyR for Ryanodine receptor. The domain is found in four ...	2855-2939	8.71e-36
[+] SPRY	smart00449	Domain in SPla and the RYanodine Receptor; Domain of unknown function. Distant homologues are ...	1084-1206	6.98e-35
[+] SPRY	pfam00622	SPRY domain; SPRY Domain is named from SPla and the RYanodine Receptor. Domain of unknown ...	1086-1206	6.63e-31
[+] RIH_assoc	pfam08454	RyR and IP3R Homology associated; This eukaryotic domain is found in ryanodine receptors (RyR) ...	3879-3992	2.58e-30
[+] SPRY	pfam00622	SPRY domain; SPRY Domain is named from SPla and the RYanodine Receptor. Domain of unknown ...	660-795	7.17e-28
[+] Ion_trans	pfam00520	Ion transport protein; This family contains sodium, potassium and calcium ion channels. This ...	4765-4949	1.68e-25
[+] SPRY	pfam00622	SPRY domain; SPRY Domain is named from SPla and the RYanodine Receptor. Domain of unknown ...	1431-1568	4.81e-24
[+] SPRY	smart00449	Domain in SPla and the RYanodine Receptor; Domain of unknown function. Distant homologues are ...	1430-1568	3.22e-23
[+] SPRY	smart00449	Domain in SPla and the RYanodine Receptor; Domain of unknown function. Distant homologues are ...	660-794	6.04e-19
[+] EF-hand_8	pfam13833	EF-hand domain pair;	4083-4133	5.82e-08
[+] MIR	smart00472	Domain in ryanodine and inositol trisphosphate receptors and protein O-mannosyltransferases;	210-263	8.98e-08
[+] MIR	smart00472	Domain in ryanodine and inositol trisphosphate receptors and protein O-mannosyltransferases;	97-152	9.66e-06

CDD provides a guide to domain level architecture, including sequence alignments & representative structures.



CDD provides a guide to domain level architecture, including sequence alignments & representative structures.



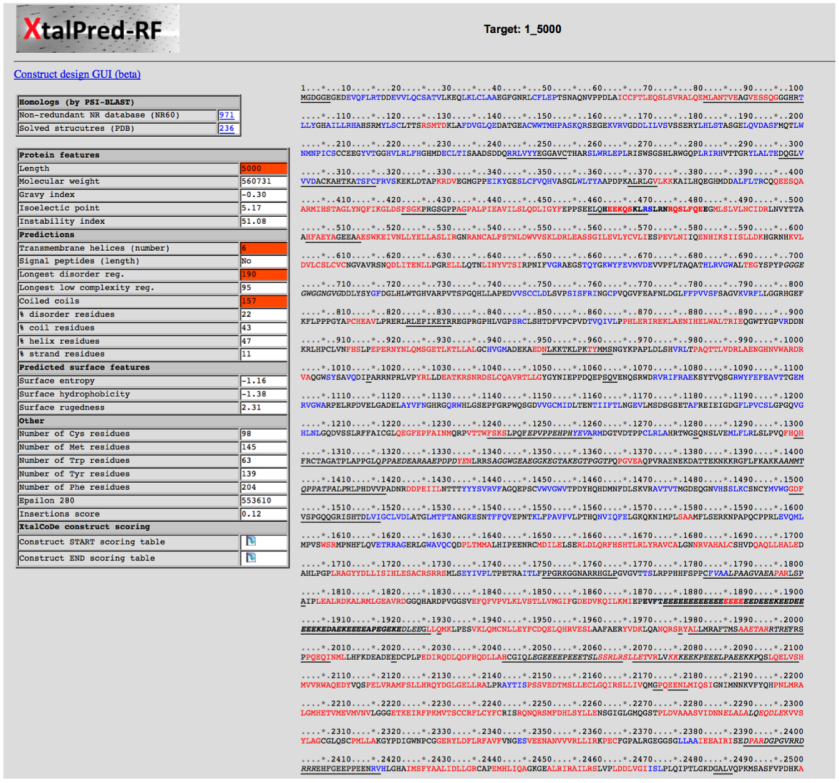
Once an initial trace is obtained for these regions, use DALI or PDBeFold to identify structural homologs that could not be identified by sequence alone.

Prep for model building - what can we learn from the sequence alone?

Your protein sequence contains a lot of useful information which you can use to aid model building:

- Start by identifying boundaries of conserved domains (NCBI CDD: https://www.ncbi.nlm.nih.gov/Structure/cdd/; DELTA-BLAST also performs CD-search by default)
- Then identify suitable structural templates for building known domains: FUGUE, SPARKS-X, PHYRE2, etc.
- **Secondary structure, TM & disorder prediction (XtalPRED for overall summary; specific tools such as SPOT-DISORDER, SPIDER3 for best accuracy).**
- Contact prediction from evolutionary couplings: EVFOLD & GREMLIN.
- Conservation analysis: Use favorite MSA algorithm (MUSCLE & CLUSTAL-OMEGA work well; TM-COFFEE, PRALINE-TM useful for membrane proteins) to create a sequence alignment of your protein with a few orthologs; gaps & insertions most commonly occur in loops/disordered regions. Useful as a guide during building.

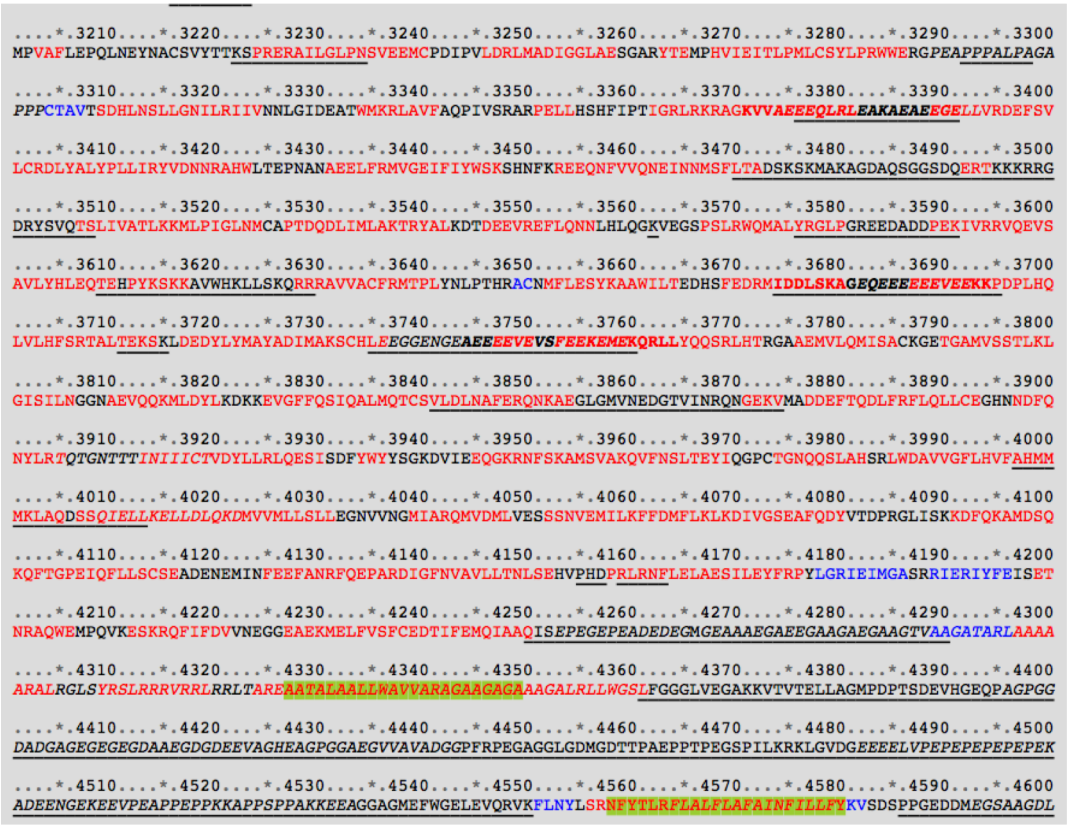
XtalPRED is a great tool for summarizing predicted sequence properties.

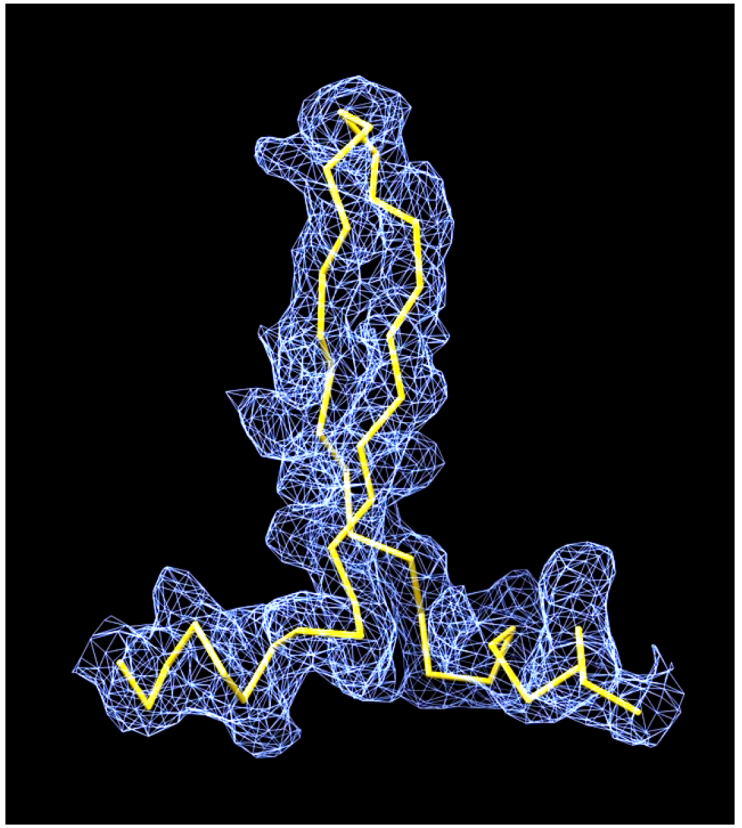


Highlights predicted secondary structure, disorder, low complexity regions on sequence in an easily digestible format. Useful to print and consult while building. Also provides list of structural homologs. (http://ffas.burnham.org/XtalPred-cgi/xtal.pl)

(Also consider using some of the newer single purpose neural-network based classifiers; e.g. SPIDER-3 & SPOT-DISORDER-SINGLE from Yaoqi Zhou lab: http://sparks-lab.org/index.php/Main/Services)

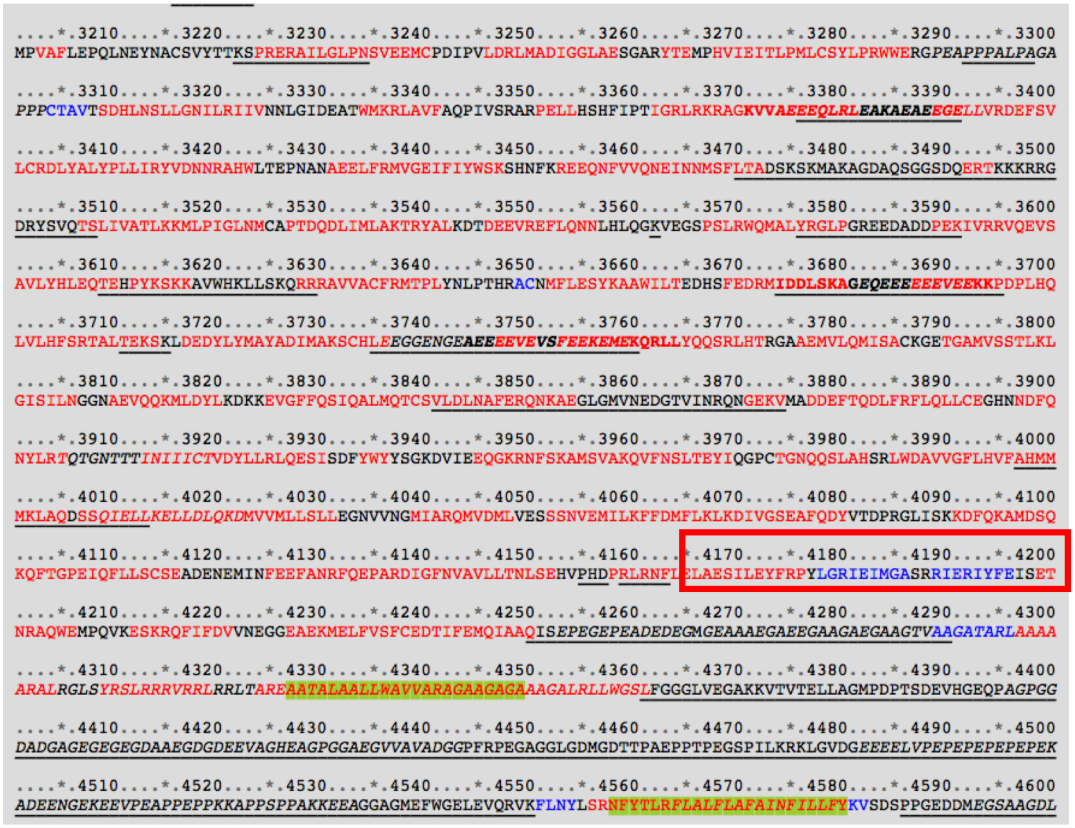
Secondary structure prediction is a very useful guide when building.

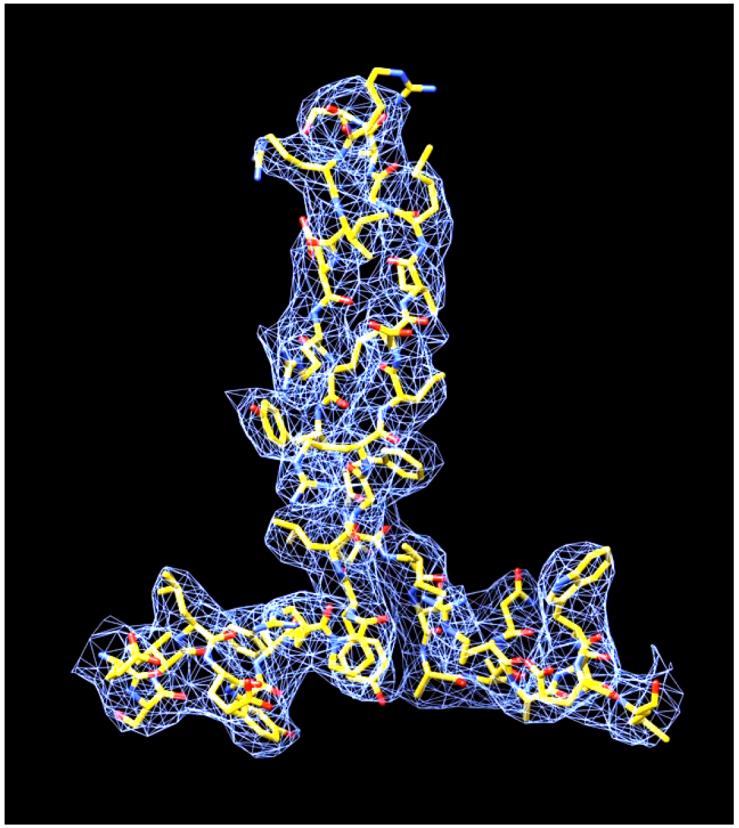




Where is this motif in the sequence?

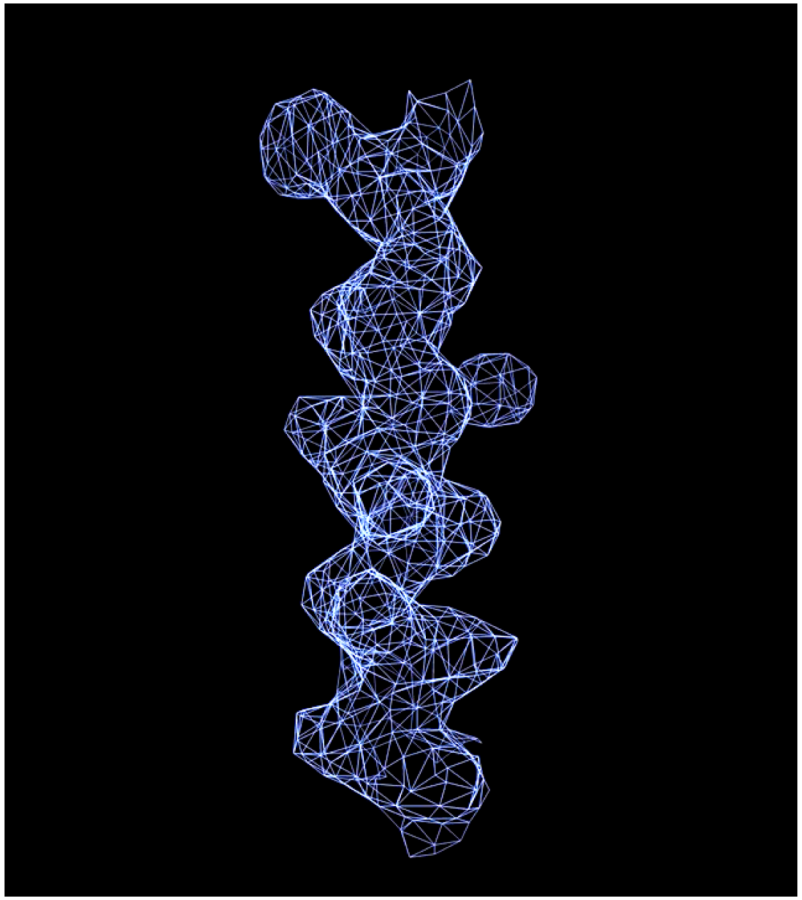
Secondary structure prediction is a very useful guide when building.



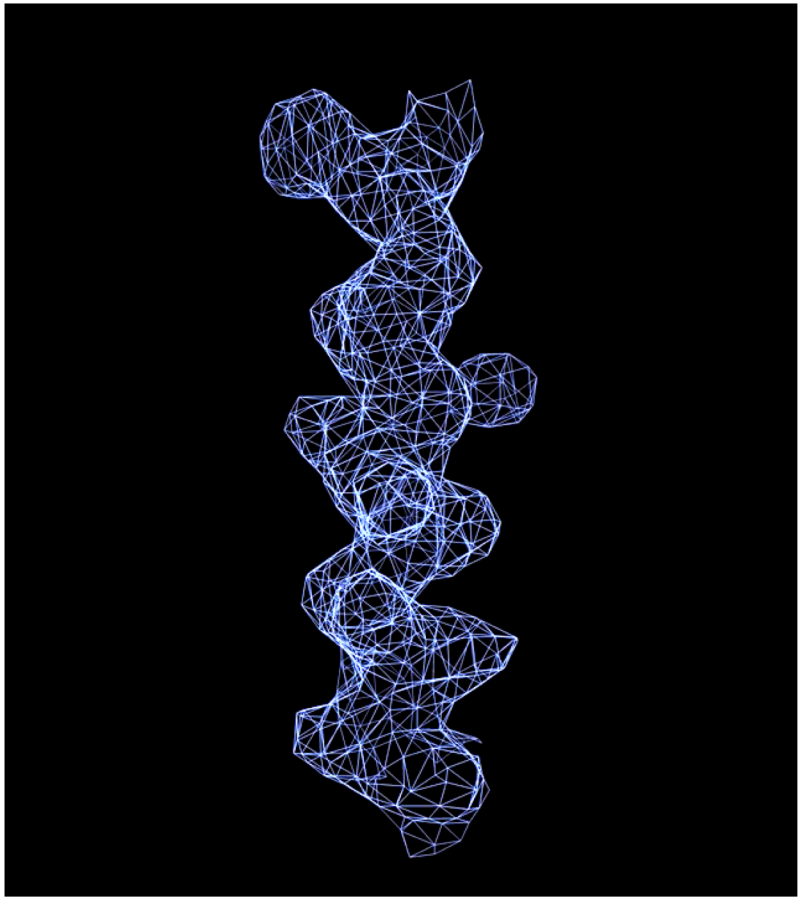


Secondary structure prediction is ~80% accurate. So if your model consistently disagrees with predicted secondary structure, look at it very closely!

What can we learn from the map alone?

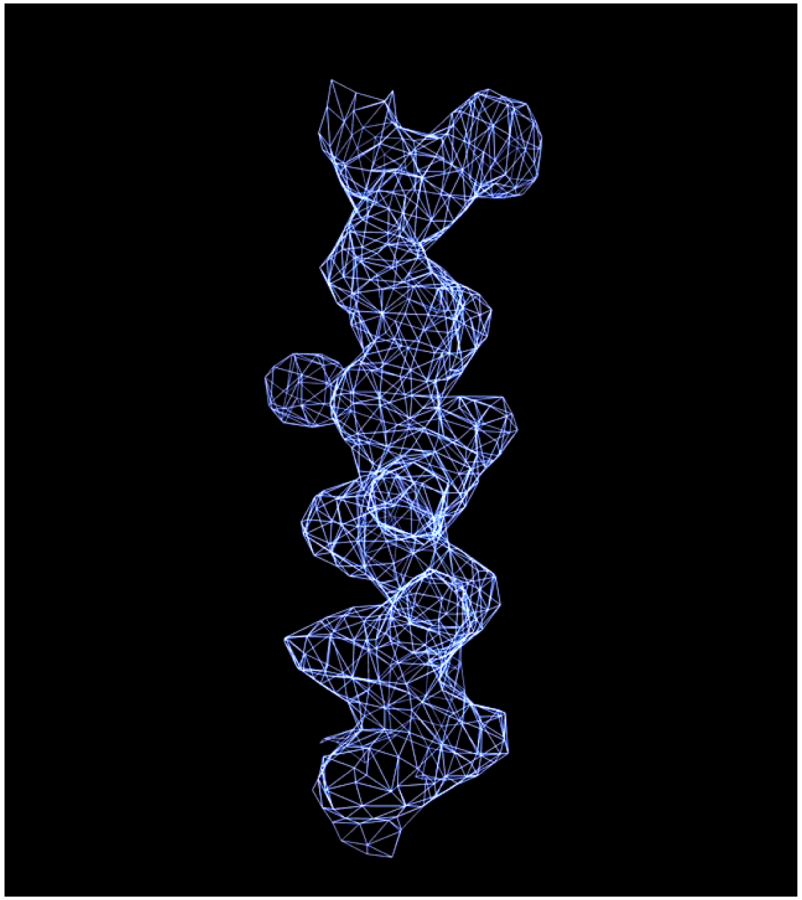


What can we learn from the map alone?

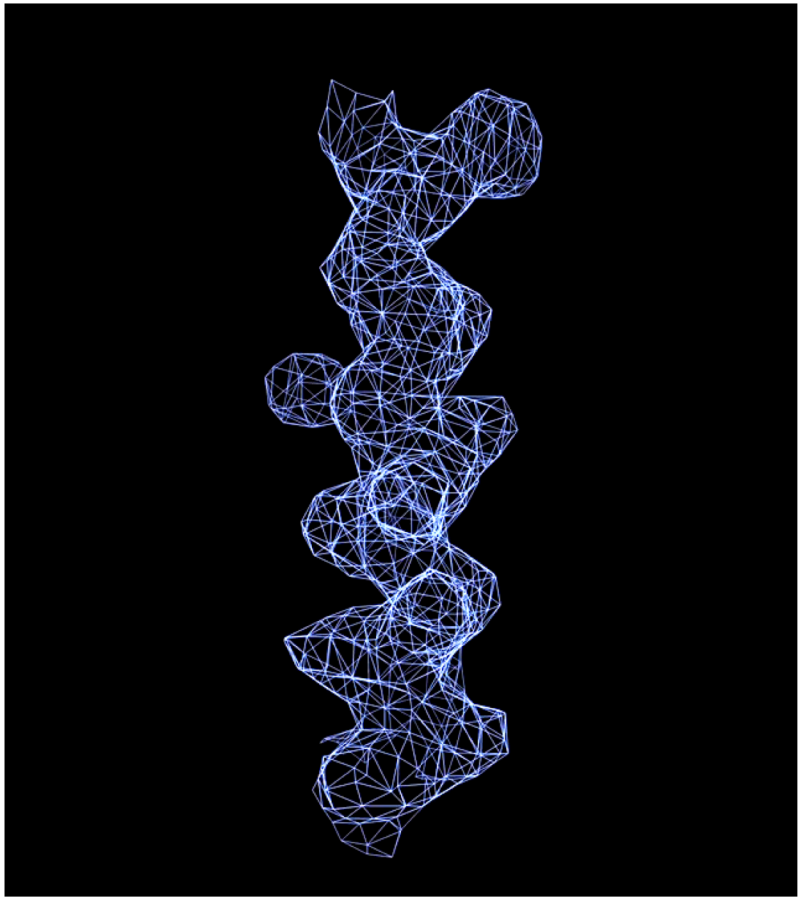


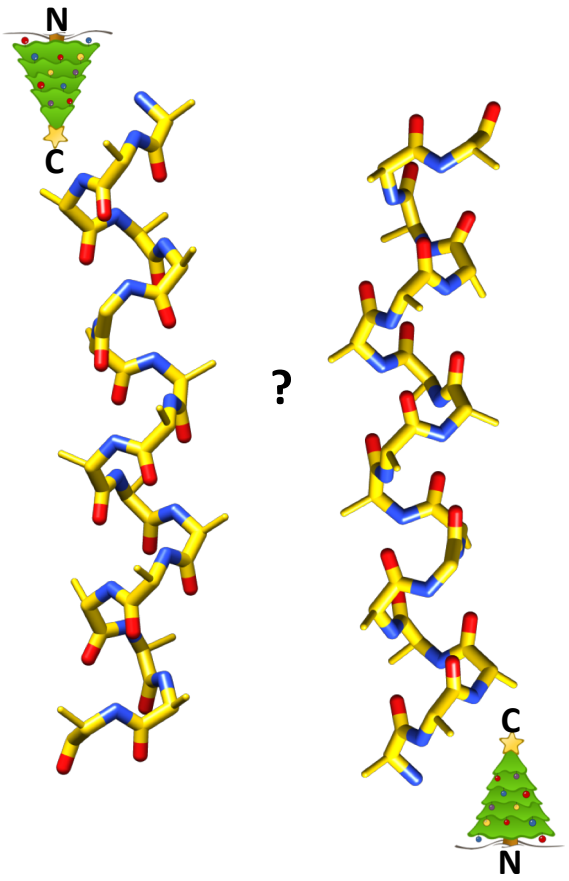
Left handed! Obvious here – can be less clear at lower res, so be careful.

OK, that's better! What can we learn from the map alone?

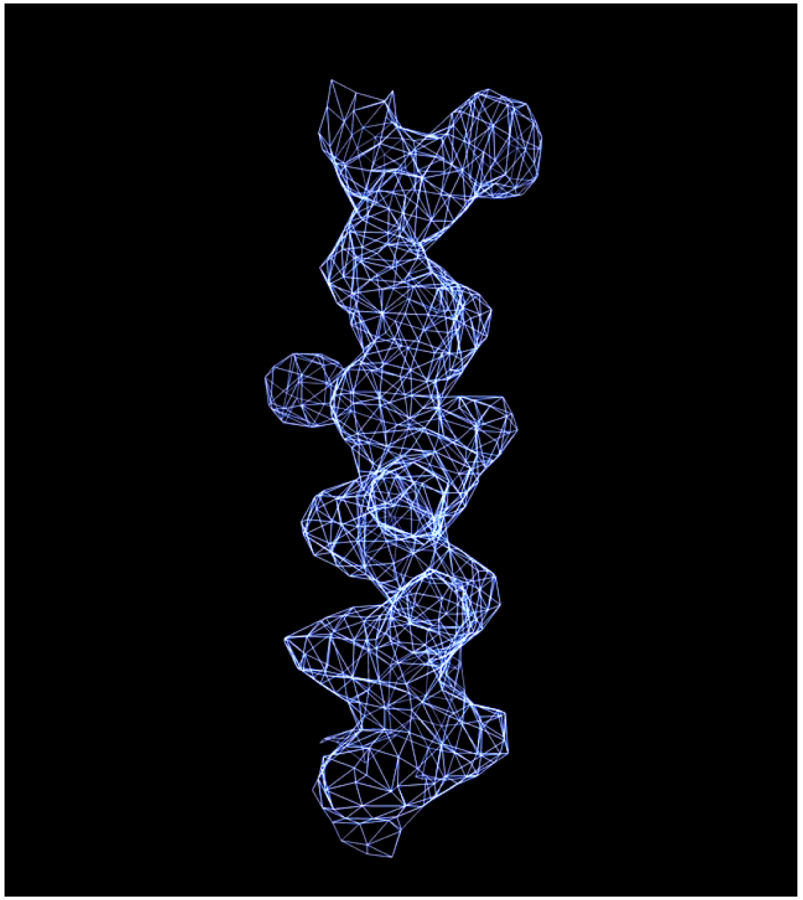


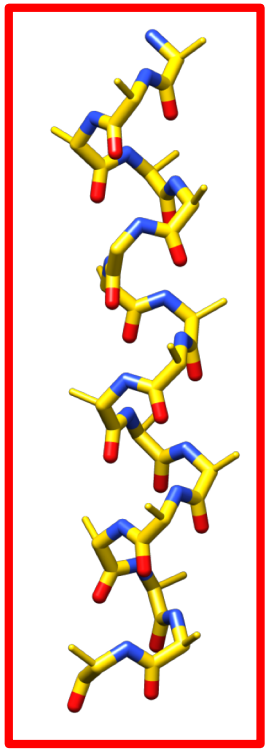
Which direction does the helix point?



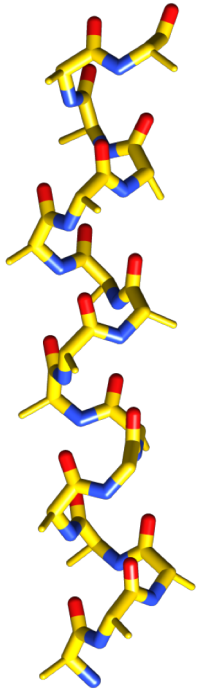


Which direction does the helix point?

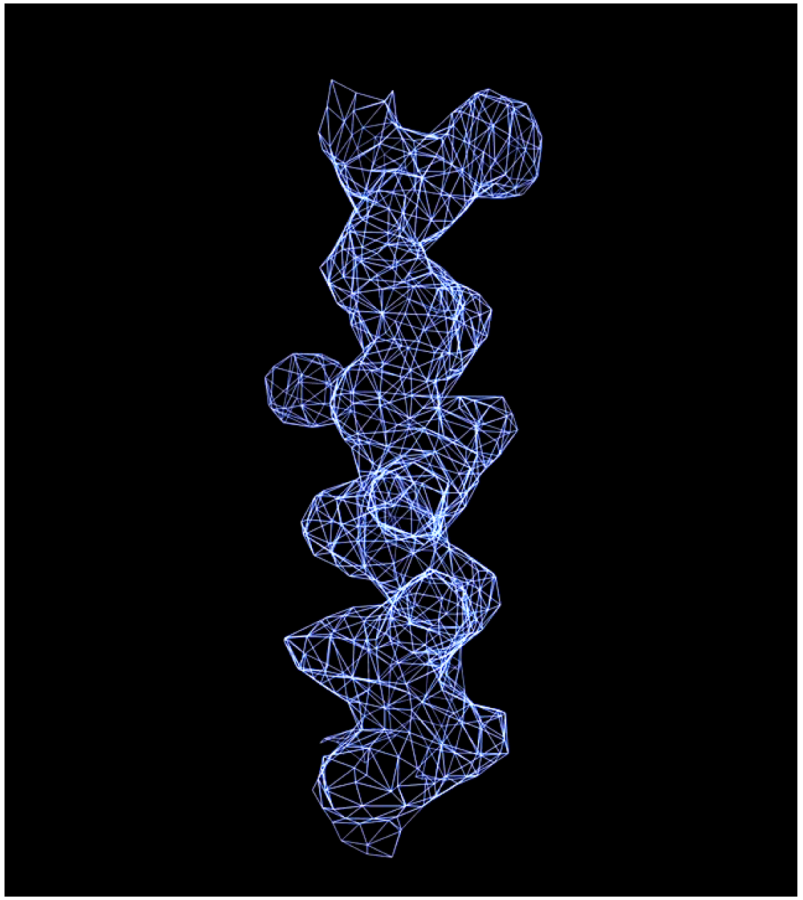


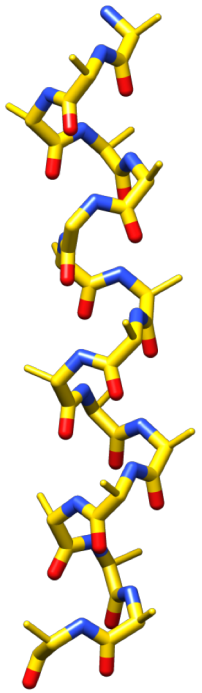


?

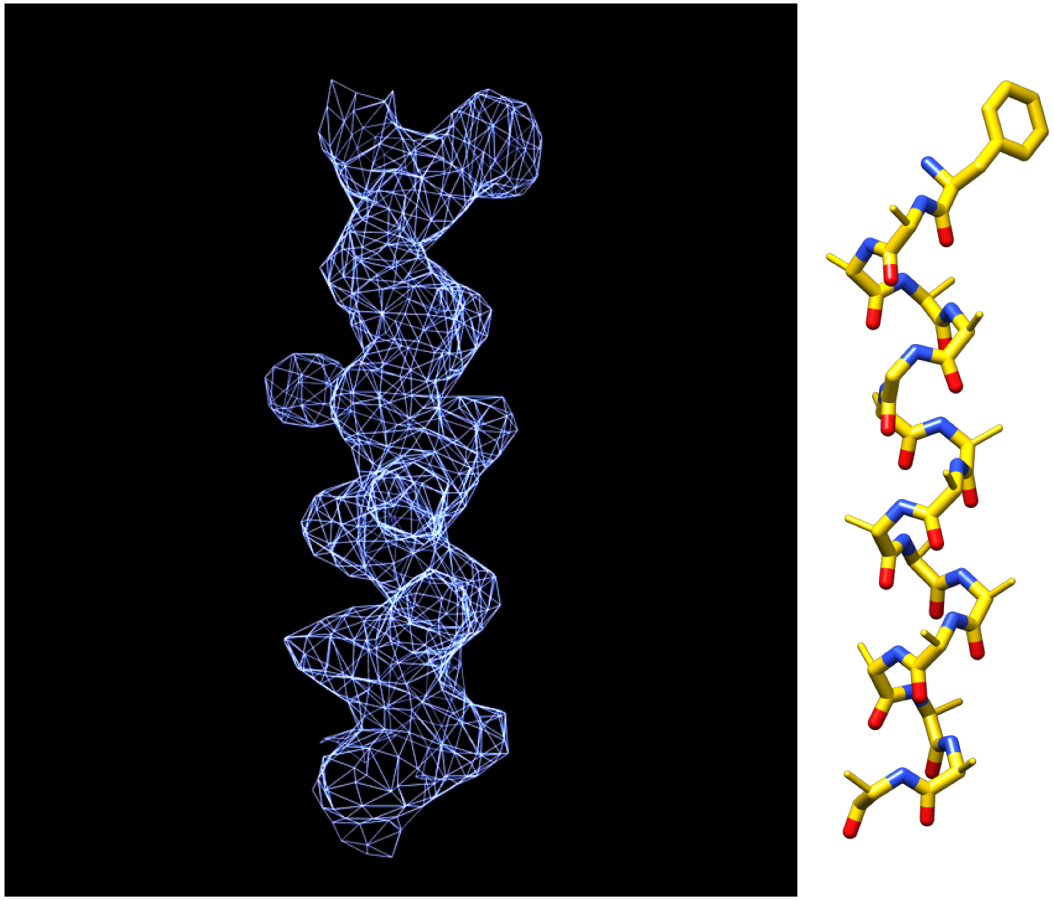


Can we identify any probable sidechains from the density?

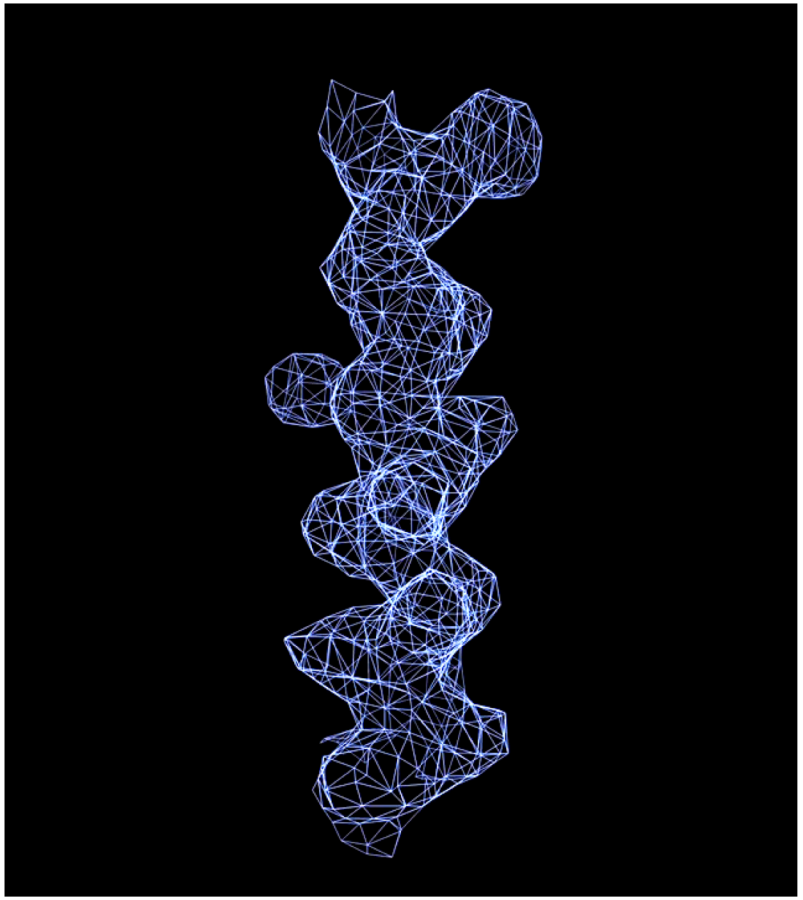


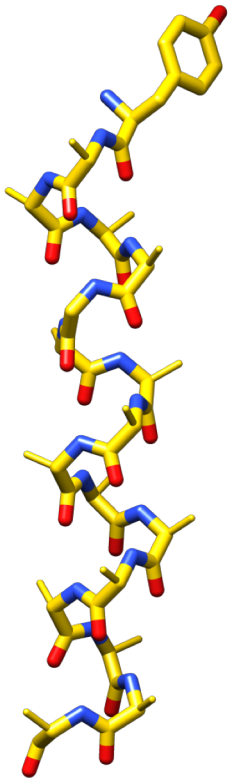


Can we identify any probable sidechains from the density?

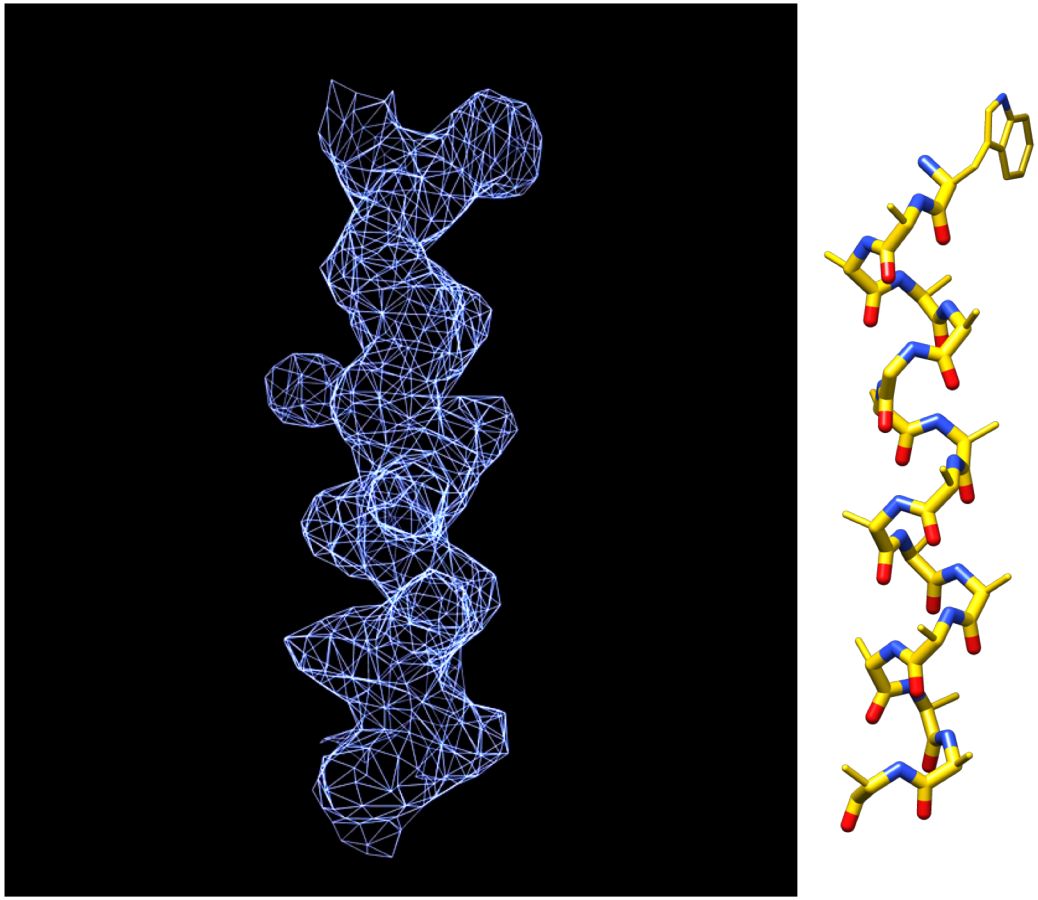


Can we identify any probable sidechains from the density?

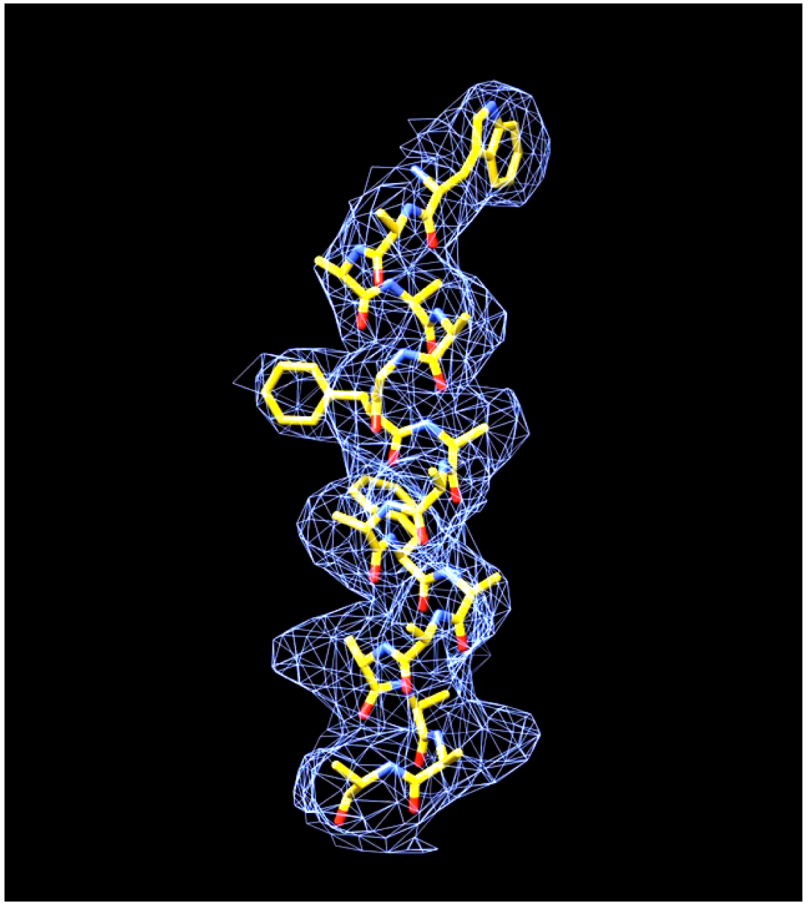


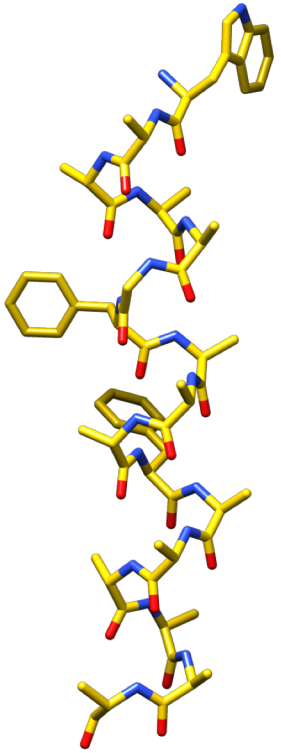


Can we identify any probable sidechains from the density?



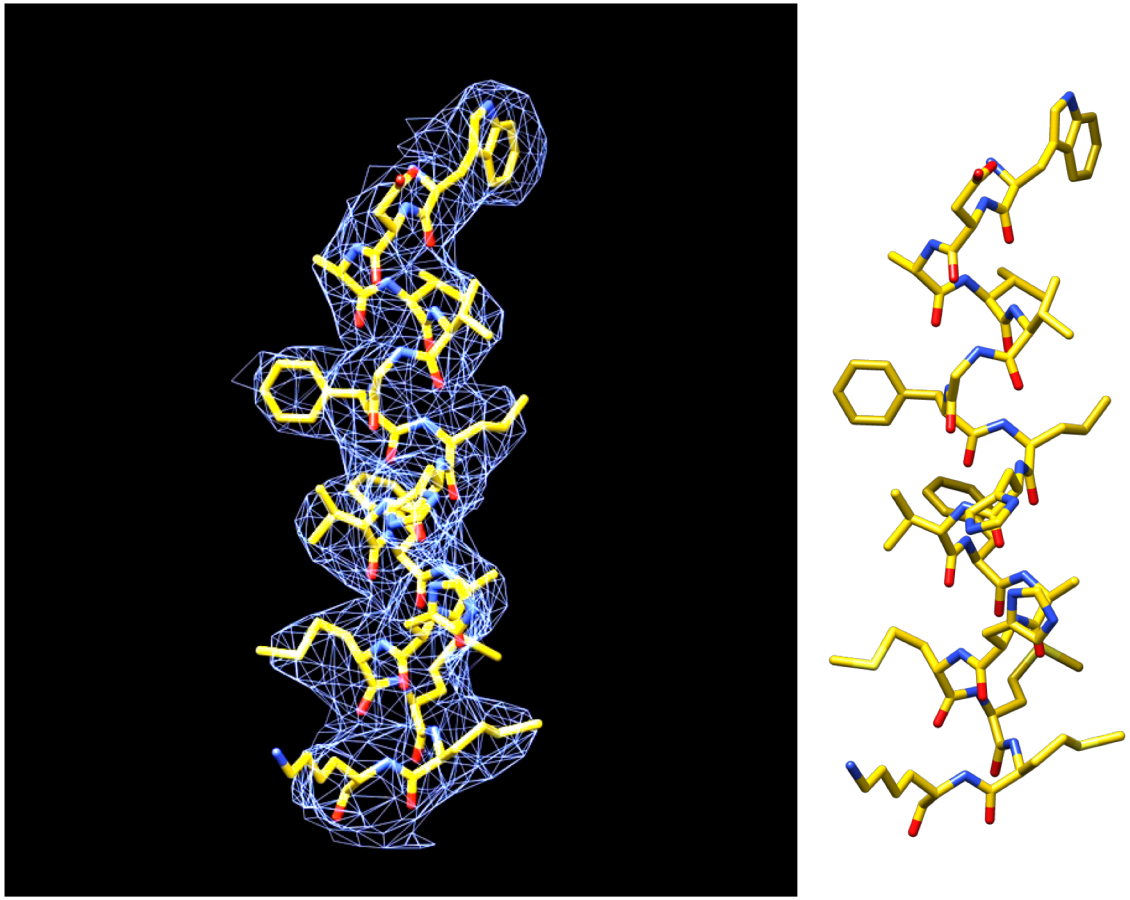
Test the initial hypothesis by extending sequence assignment along the chain.





…VFNSLTEYIQGPCTGNQQSLAHSRL**W**DAVVG**F**LHV**F**AHMMMMKLAQDSSQIELLKELLDLQ…

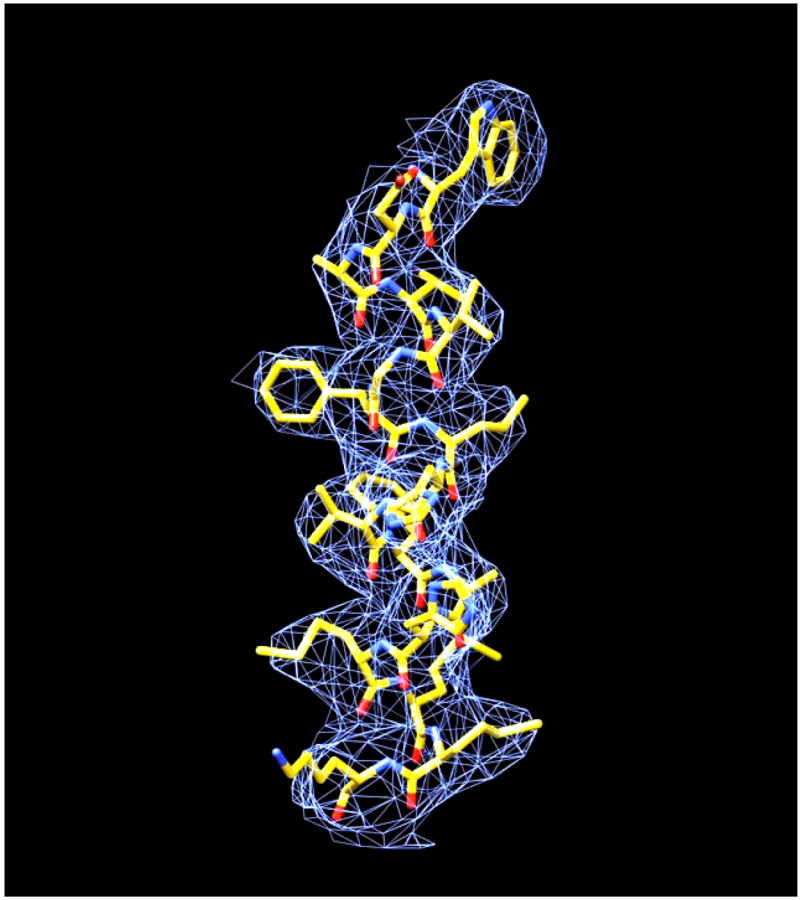
Test the initial hypothesis by extending sequence assignment along the chain.

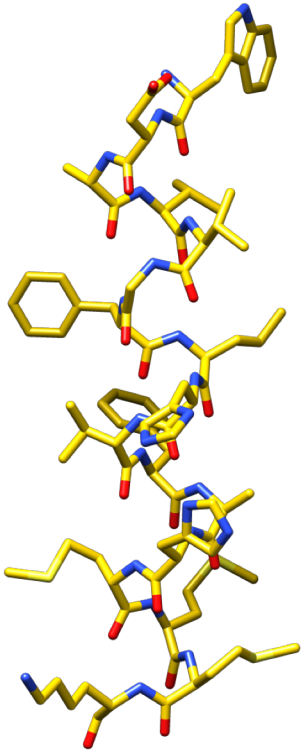


…VFNSLTEYIQGPCTGNQQSLAHSRL**WDAVVGFLHVFAHMMMK**LAQDSSQIELLKELLDLQ…

Test the initial hypothesis by extending sequence assignment along the chain.

Notice that the **absence** of large sidechain densities at small residue positions is just as valuable in validating the fit as the fit of large sidechains to the density.

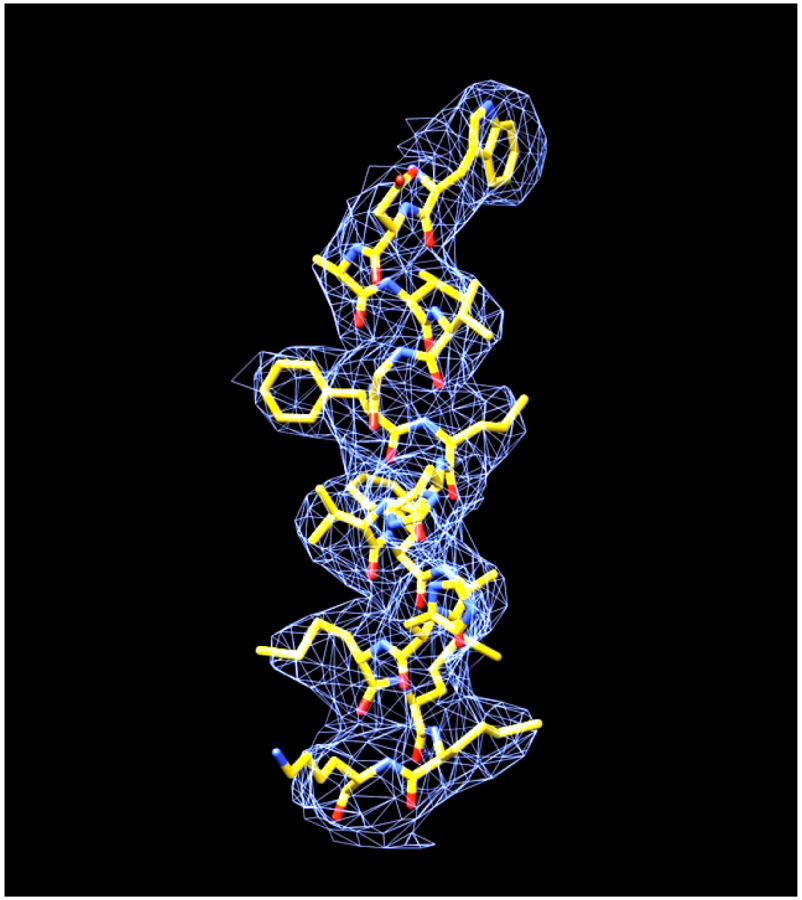


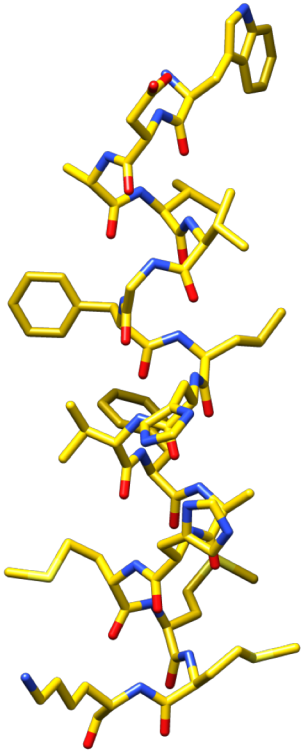


…VFNSLTEYIQGPCTGNQQSLAHSRL**WDAVVGFLHVFAHMMMK**LAQDSSQIELLKELLDLQ…

Test the initial hypothesis by extending sequence assignment along the chain.

Also, note that the information content of local regions varies. Consider "VTVVAASSTVV" vs "FGAAYWVTRA" – which is more likely to be uniquely identifiable from the map?





...VFNSLTEYIQGPCTGNQQSLAHSRL**WDAVVGFLHVFAHMMMK**LAQDSSQIELLKELLDLQ...

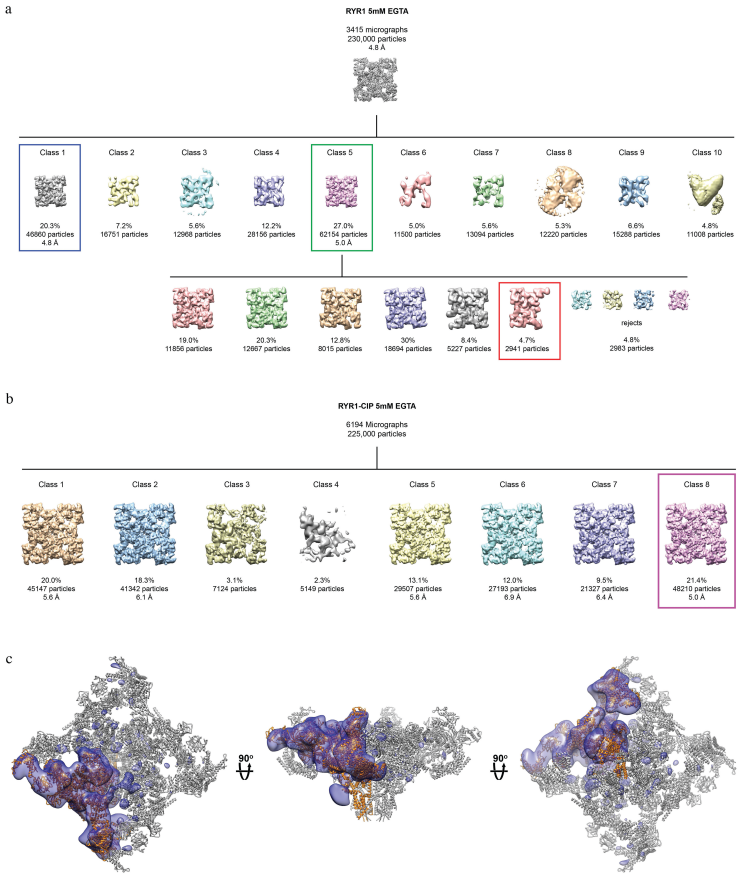
How to deal with uncertainty in sequence assignment and sidechain placement

- You will likely encounter situations where you cannot be certain of the local sequence register – what to do?

- No clear consensus, but I suggest assigning residue code as “UNK” and numbering to “best guess” value. A more granular way to quantify/convey uncertainty would be helpful!

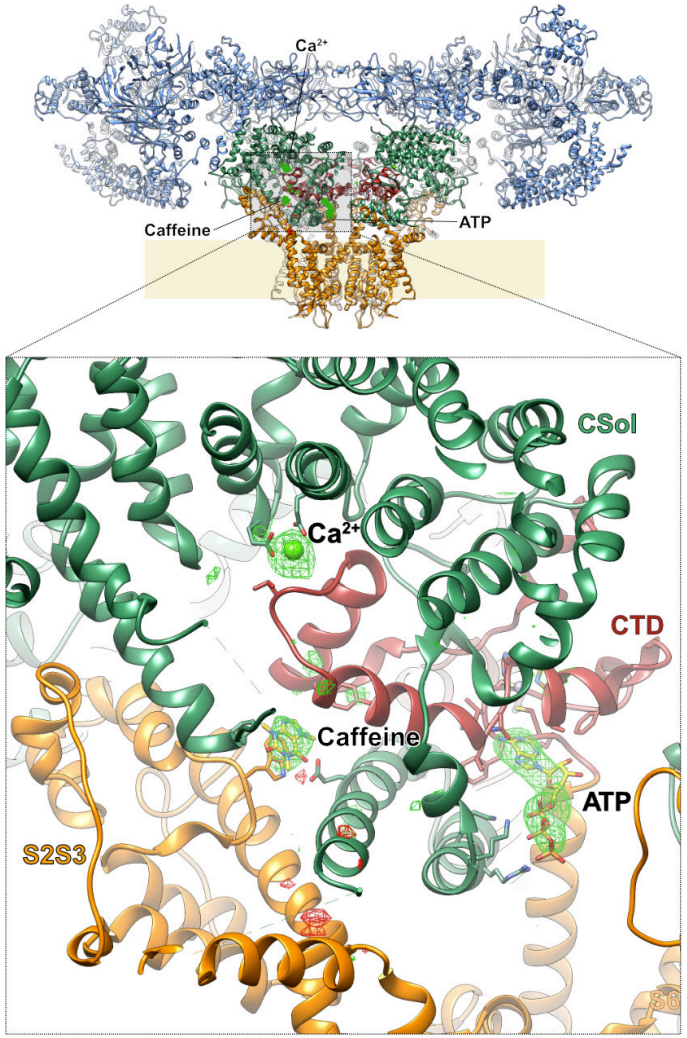
- Sidechain placement – two main camps – trim sidechains to density vs place them all (+/- zero occ.). The former may sound more conservative, but it can hide errors during validation (during analysis of clashes). Either is acceptable, just be consistent, and preferably outline the approach taken when writing up the structure.

Prior knowledge can come in many forms – use any and all available info to guide model building.

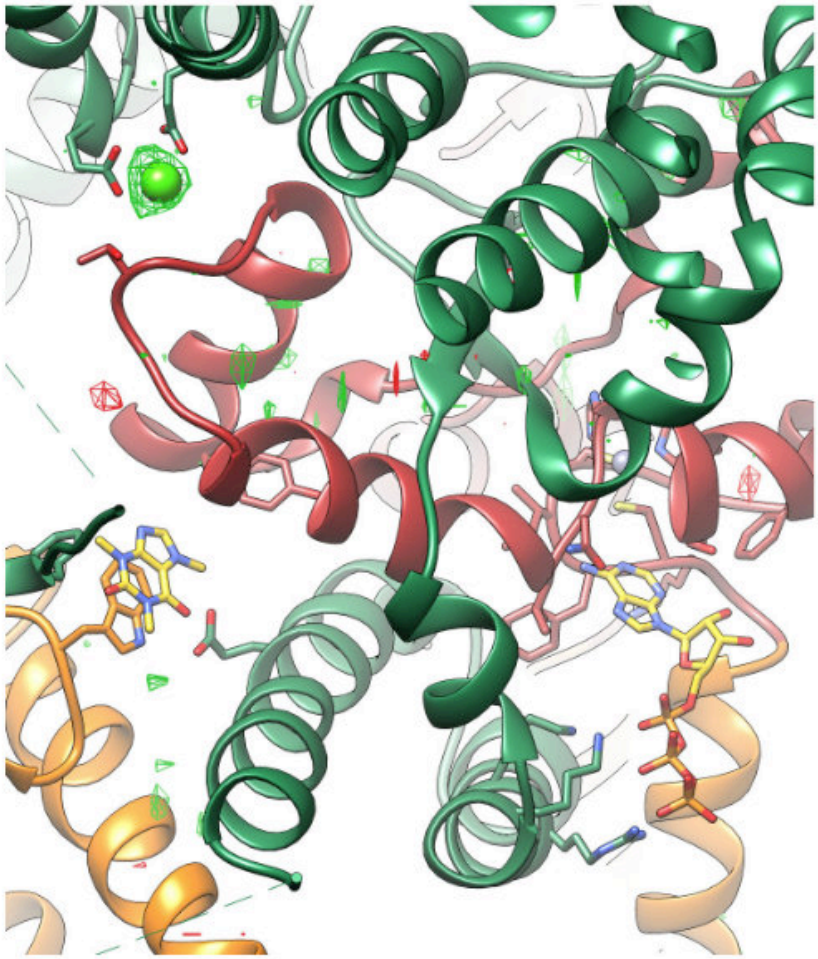


Here, serendipitous identification of a conformational class of RyR1 lacking density for one subunit aided identification of protomer boundaries. In other cases, cross-linking data or NS data on subcomplexes or Fab-complexes may be helpful.

In a similar manner, we can use locally aligned difference maps between holo and apo structures to locate ligands.

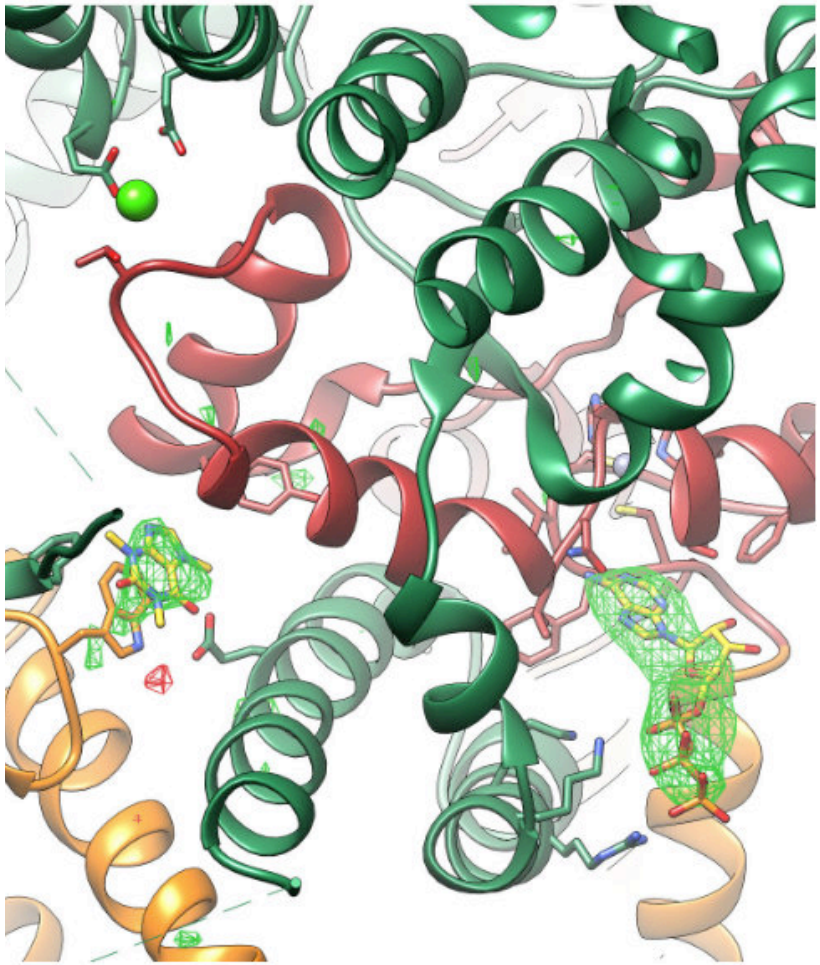


The three ligands are clustered around the C-terminal domain.



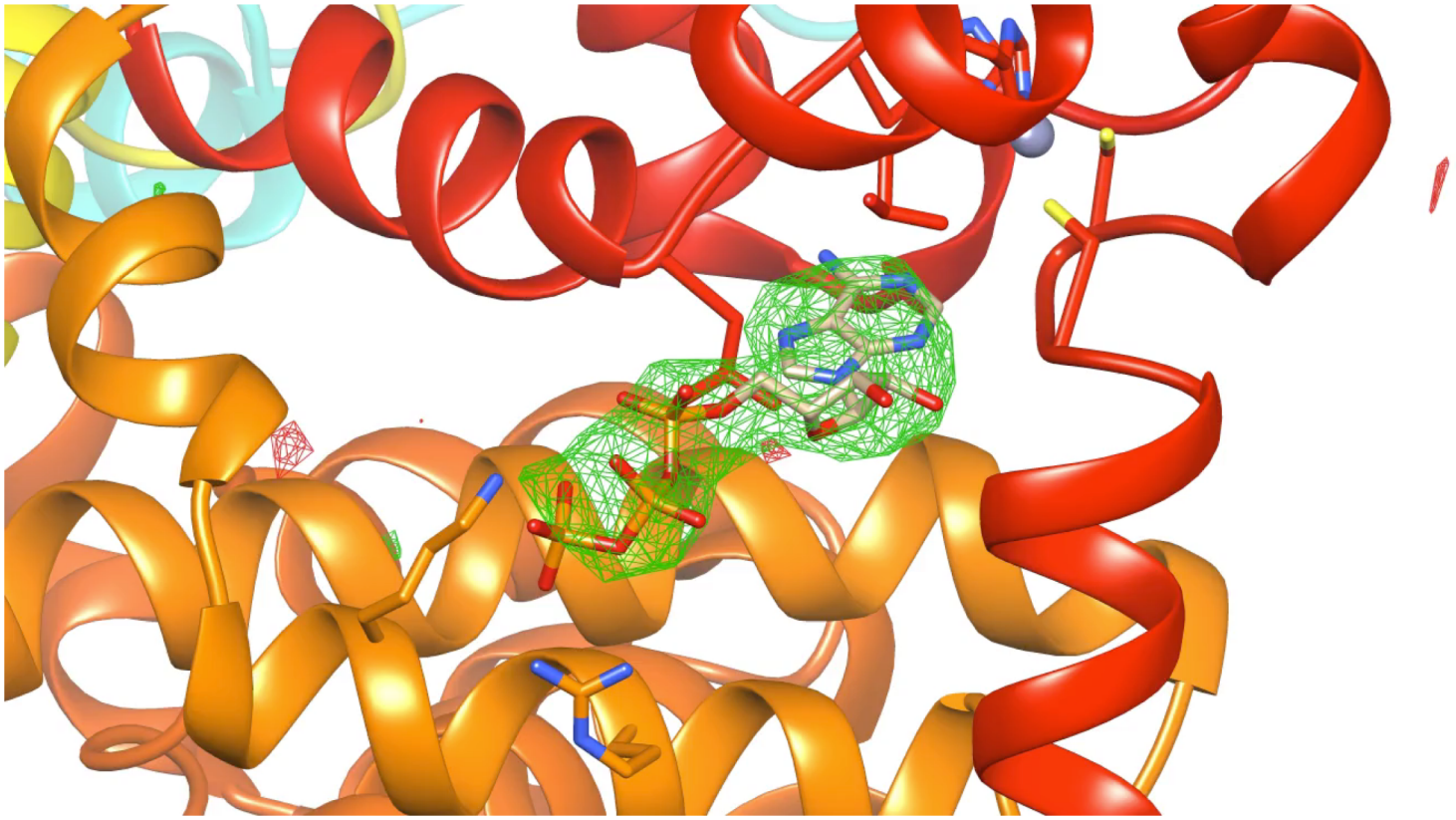
(Ca^{2+} only) minus (EGTA only)

The three ligands are clustered around the C-terminal domain.

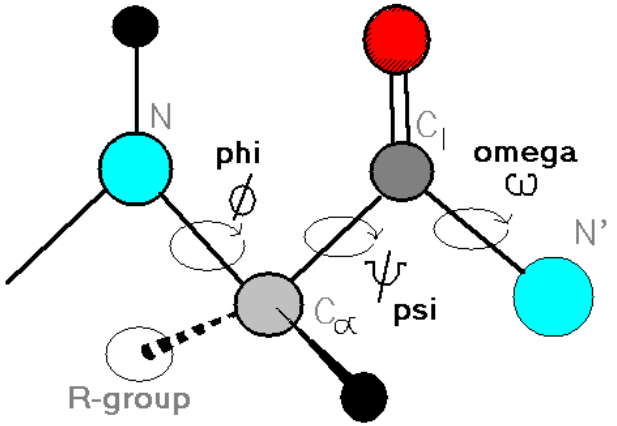


(ATP/Caffeine) minus (EGTA only)

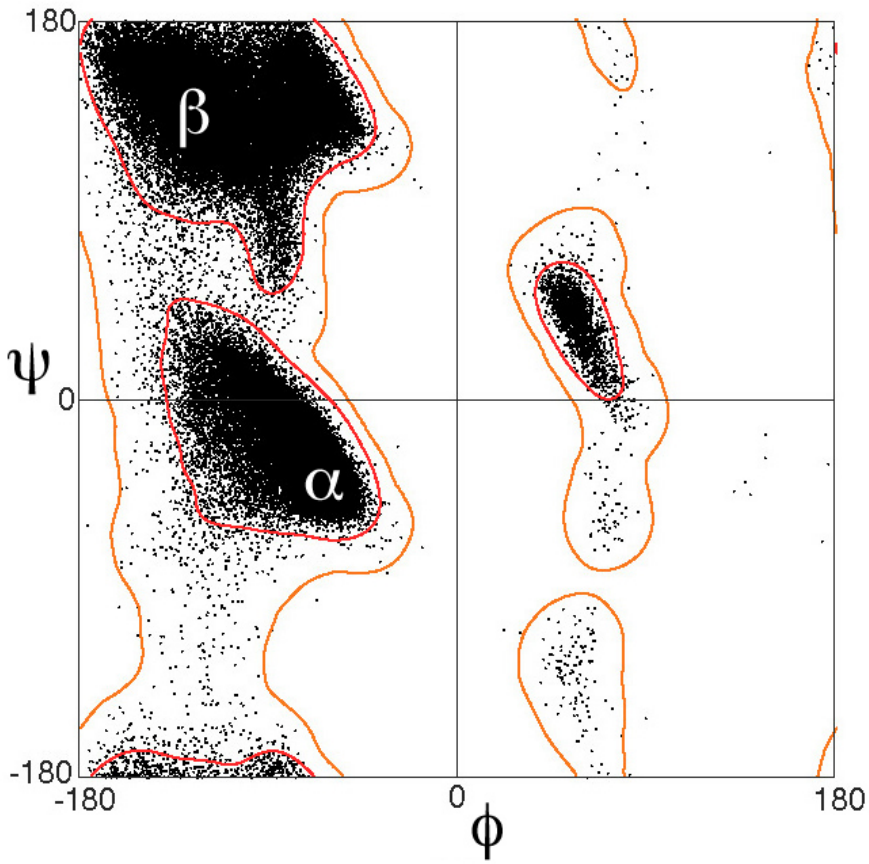
Very good difference density even at moderate (3.8Å) resolution. Highlights importance of phases!



Secondary structure and the Ramachandran plot

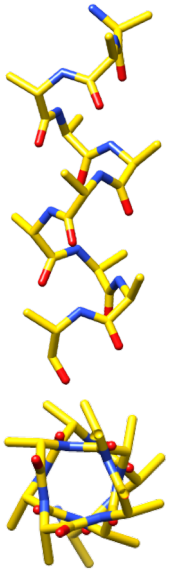


- Describes geometrically favored backbone torsions (omega generally 180, except for prolines)
- At high resolution, outliers may be justified by density
- At low resolution, we can't see carbonyls, so much harder to justify Ramachandran outliers.
- This is the general-case Rama plot - distribution is different for "special" residues (pro, gly, pre-pro)



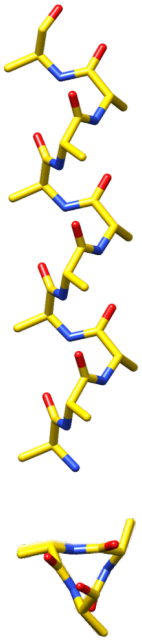
(http://www.biochem.ucl.ac.uk/~martin/c40/peptide.html)

Helices – alpha and 3_{10}



Alpha

- ~90%
- 3.6 residues per turn
- Fat

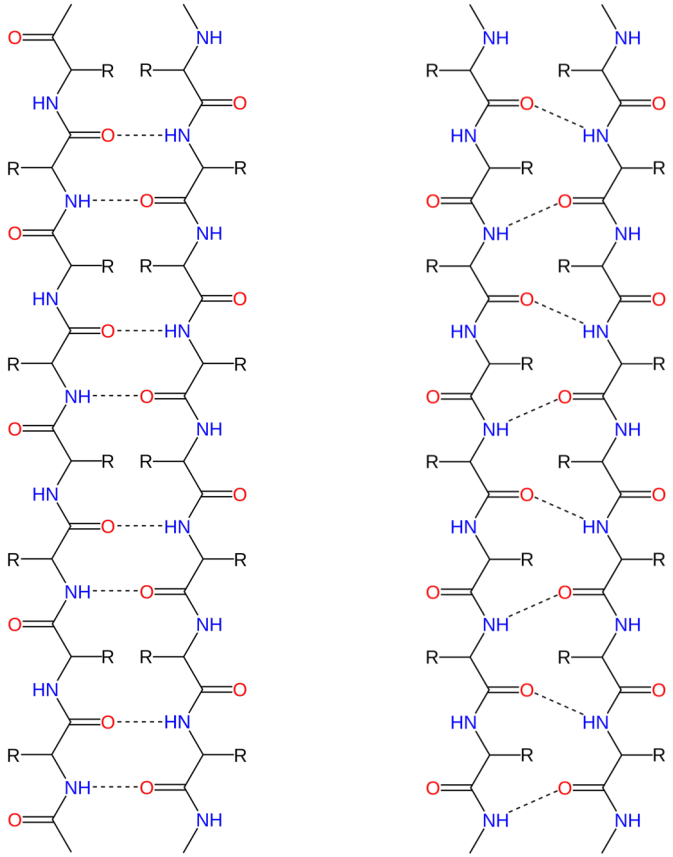


3_{10}

- ~10%. More common in TM? (e.g. S4 of VSD)
- 3 residues per turn. Triangular cross section.
- Skinny
- Can be tricky to identify at low resolution, can lead to register errors.

Beta sheets

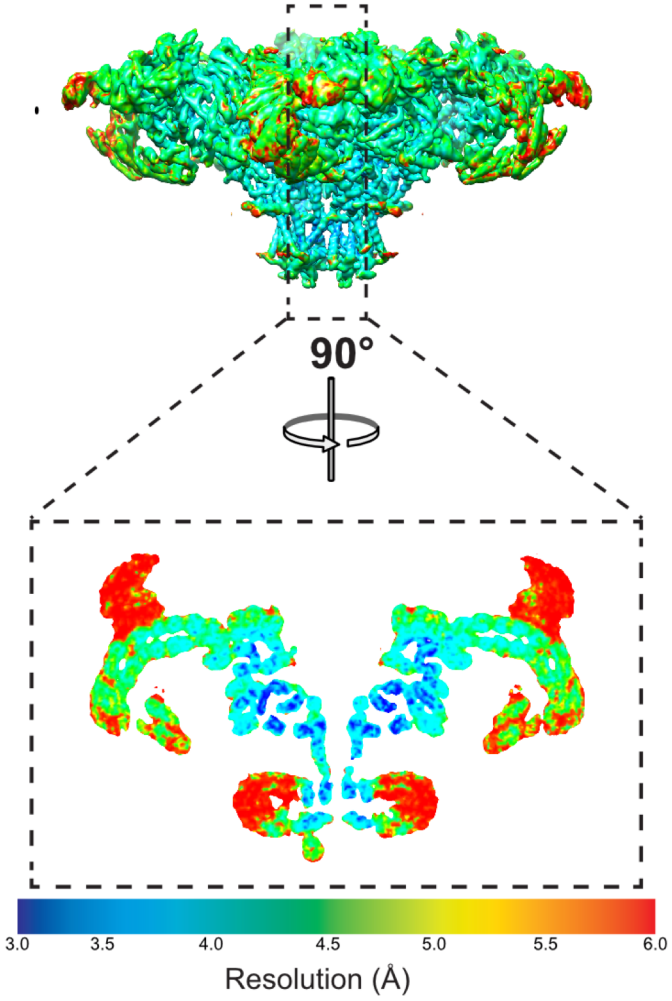
- Can be parallel or antiparallel in orientation (antiparallel more common and stable)]
- Twist of beta sheet varies, leading to more diversity than for alpha-helical structures.
- Harder to build at low resolution - whereas a helix can be placed at ~7 Å, adjacent strands can only be clearly separated at ~4.5 Å.



(https://en.wikipedia.org/wiki/Beta_sheet)

EM-specific considerations

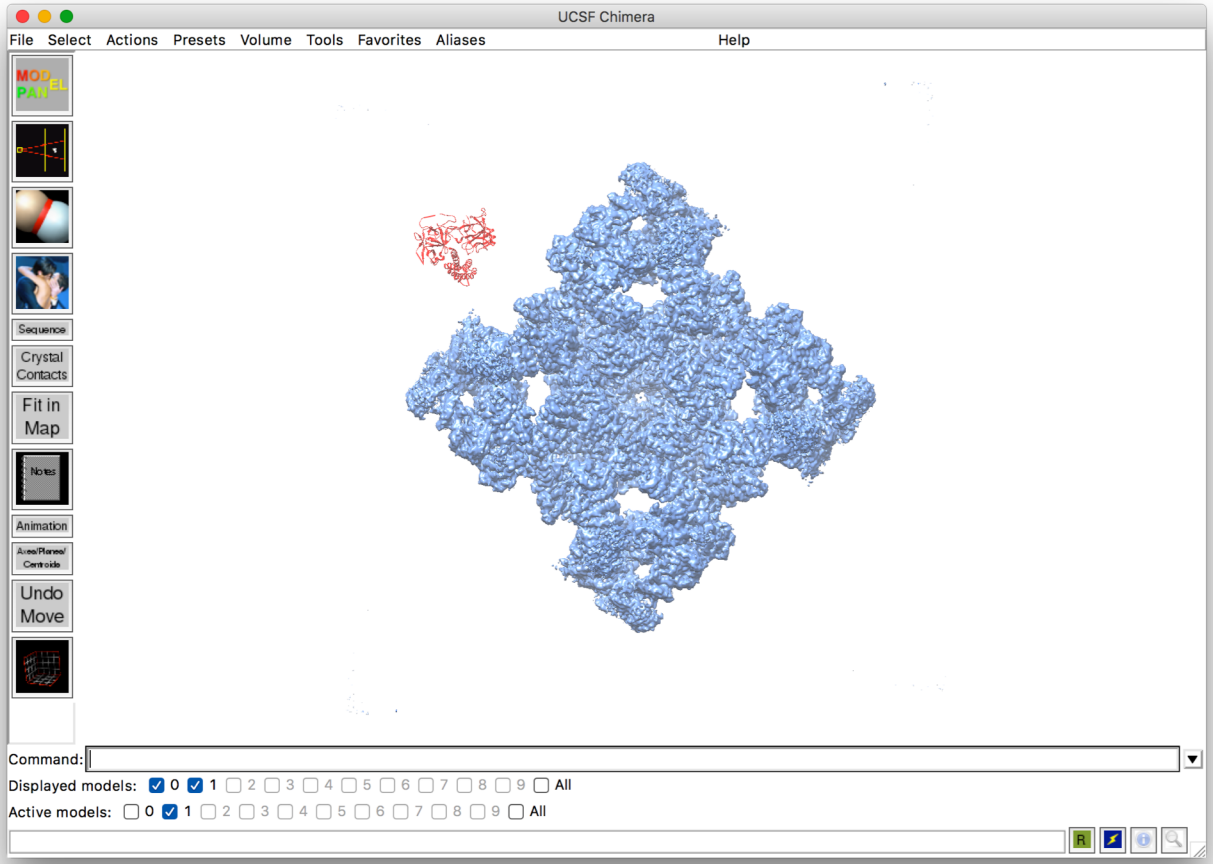
- No unambiguous sequence markers at low resolution (no equivalent of SeMet).
- No feedback from phase improvement, but also no model bias – WYSIWIG.
- Often substantial variation in local resolution – different strategies and levels of detail required for different regions. Map sharpening essential.
- ”Medium” resolution (4-6Å) much more common than for crystallography.
- Often have more than one map, with different composition or conformation (combine focused refinements in Chimera by taking max value at each voxel after alignment, e.g.: `vop maximum #1,2 ongrid #1`)



Building an initial model - where to start?

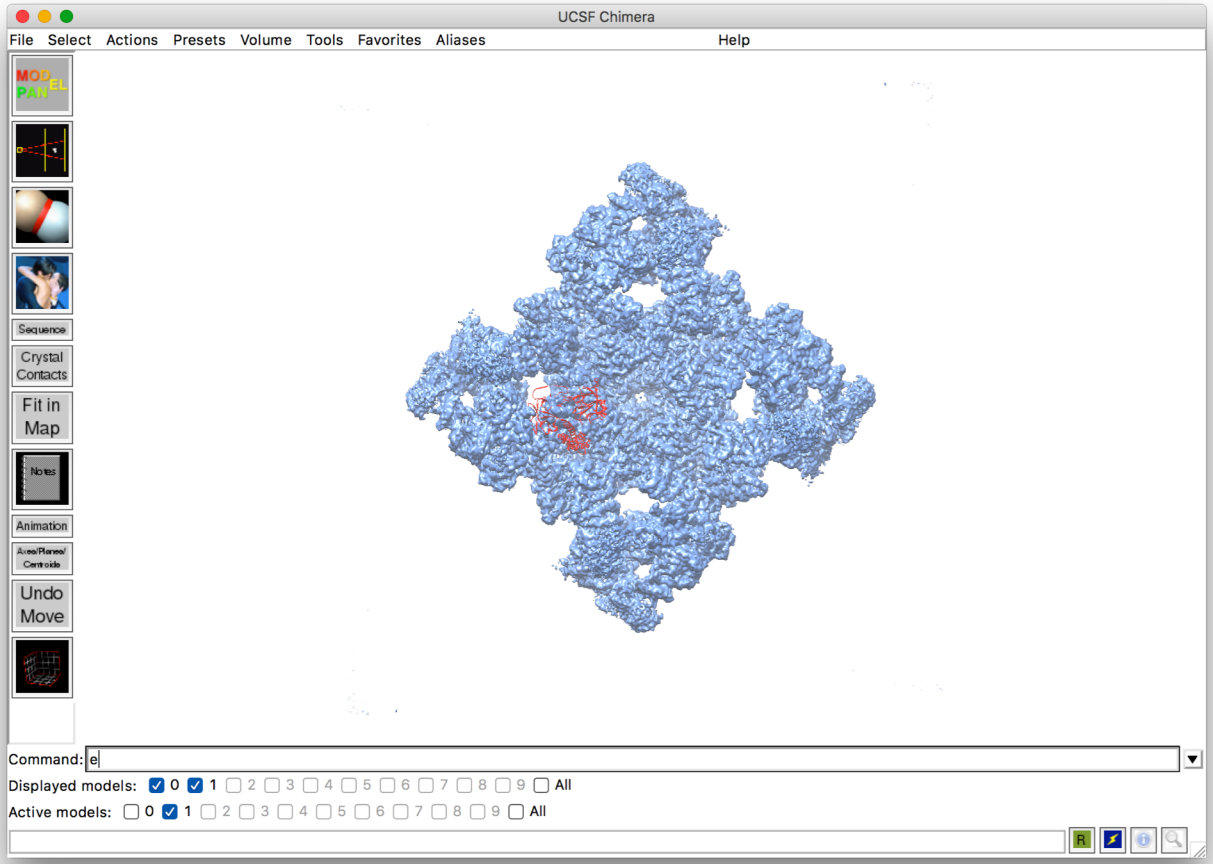
- If you have a crystal structure, of a fragment or a homology model of a domain, place it, and extend into density.
- If you have sufficient resolution, try autobuilding with phenix
- Otherwise, identify structurally distinctive motifs in the sequence – for example, a strongly predicted helix with three aromatic residues near the N-term end – and identify candidate locations in the density map. Extend and see if hypothesis still holds.

Using UCSF Chimera to fit solved domains



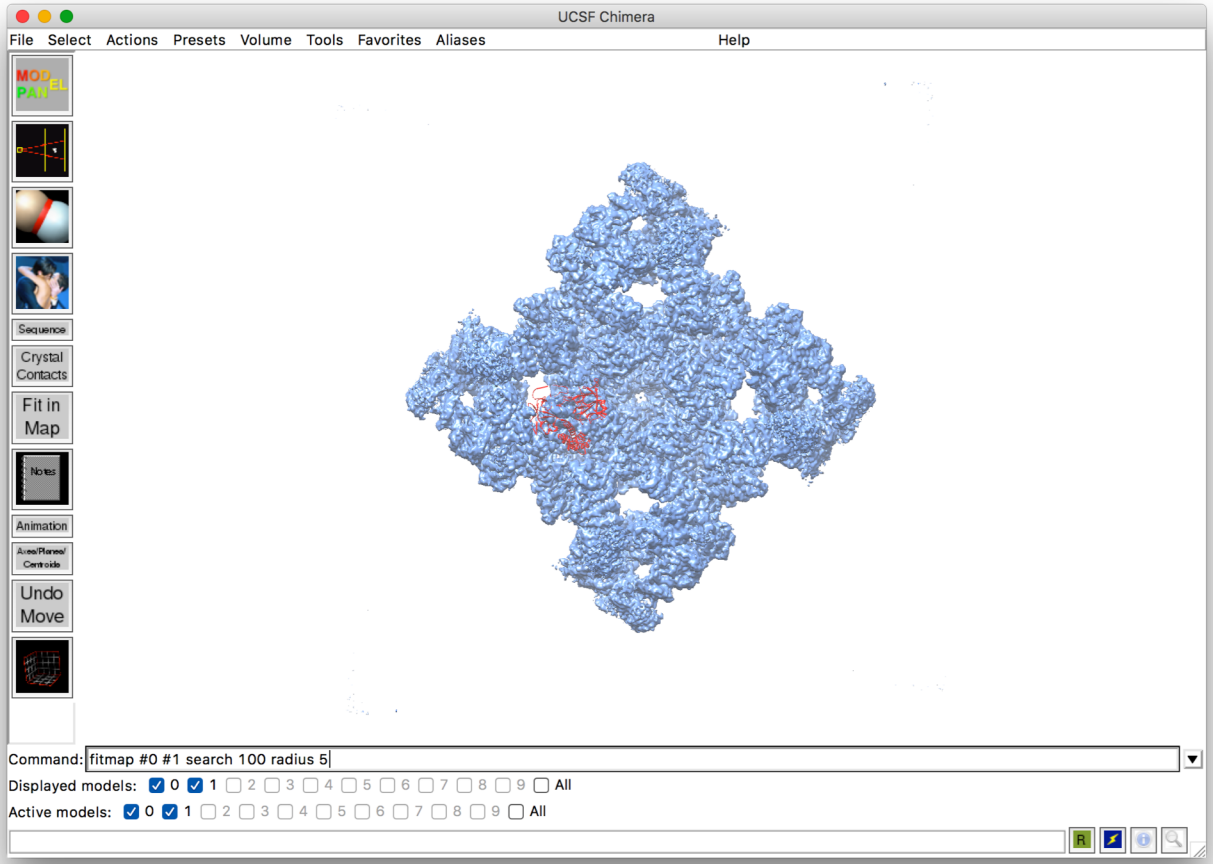
Start with map and model.

Using UCSF Chimera to fit solved domains



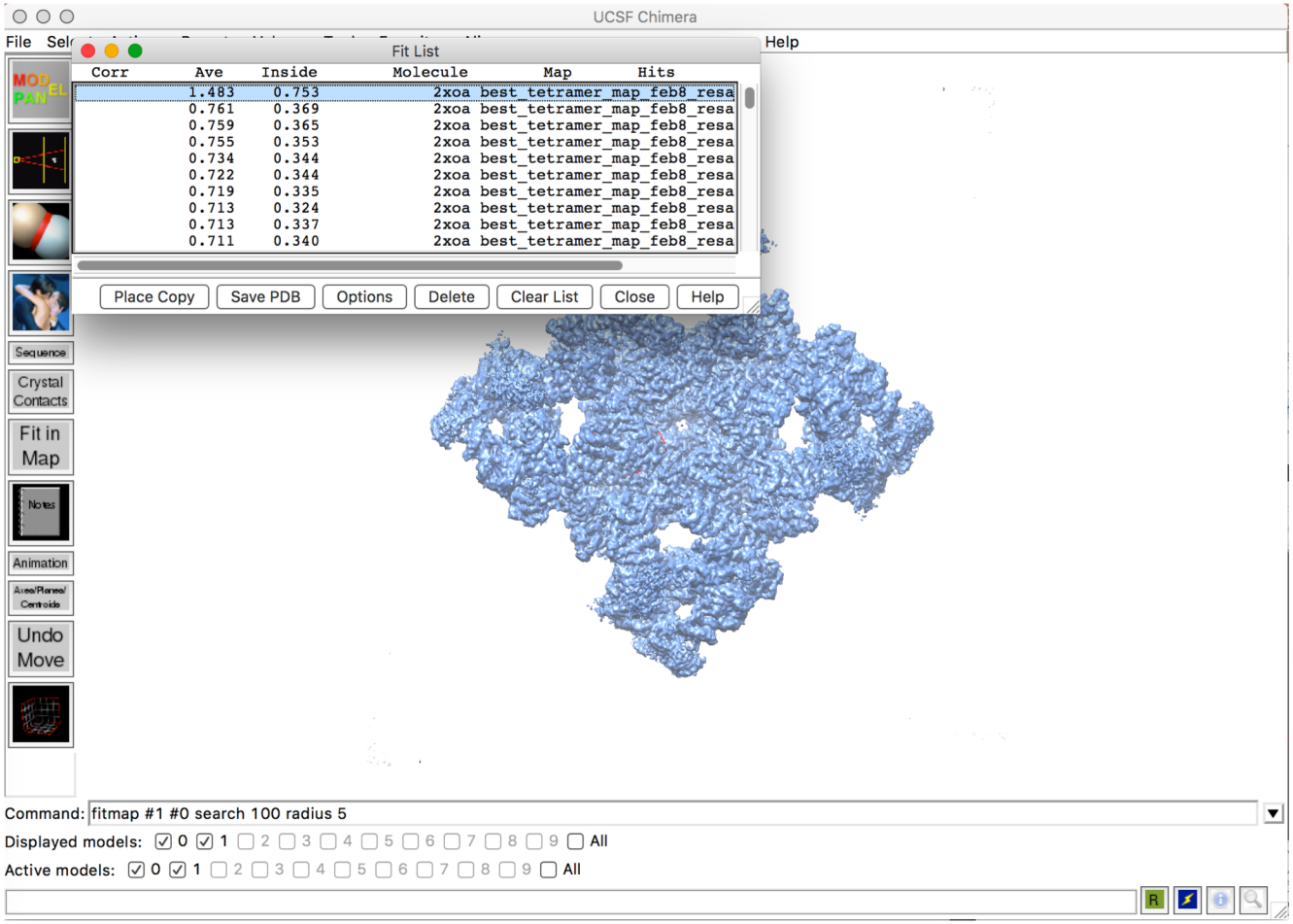
Move model to approximate position (if known, to save computation)

Using UCSF Chimera to fit solved domains



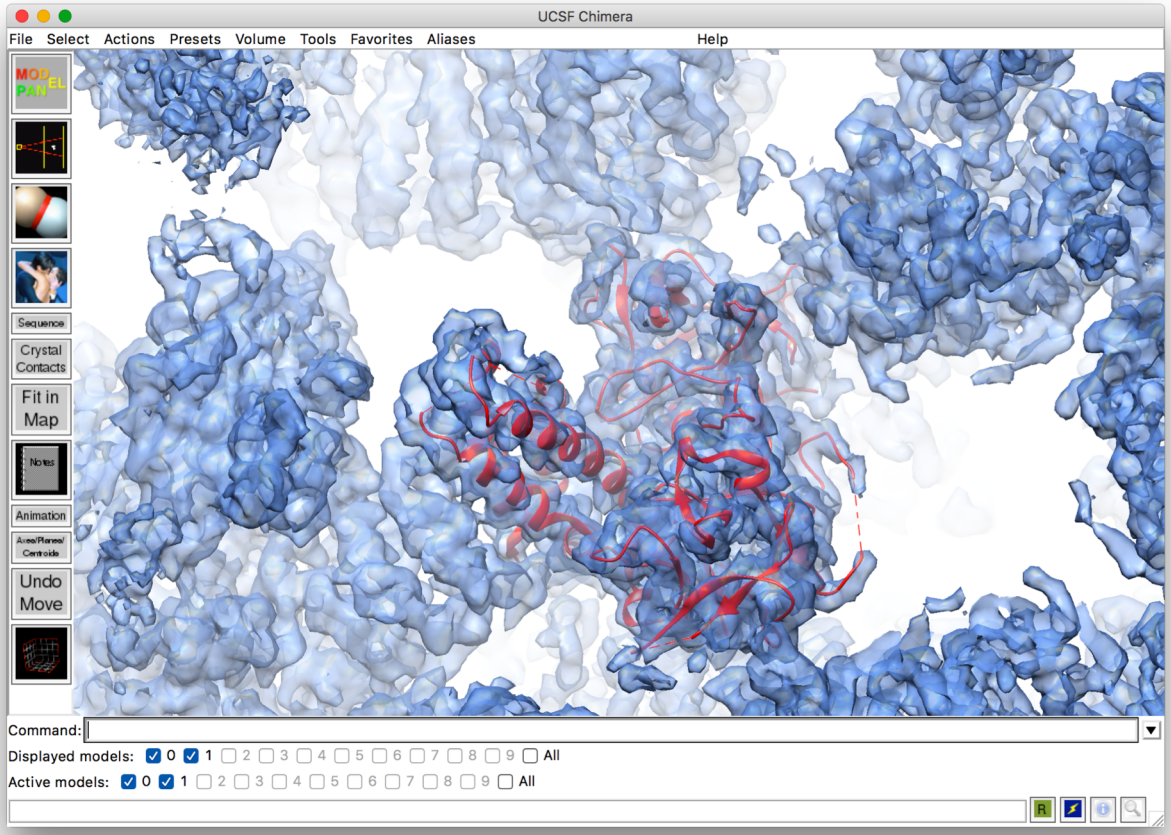
Run fitmap with ‘search’ (here 100 orientations) and ’radius’ (here 5 Å)

Using UCSF Chimera to fit solved domains



Chimera will return a list of candidate orientations, ranked by agreement with the map. Hopefully there will be a clear separation between the correct and incorrect solutions.

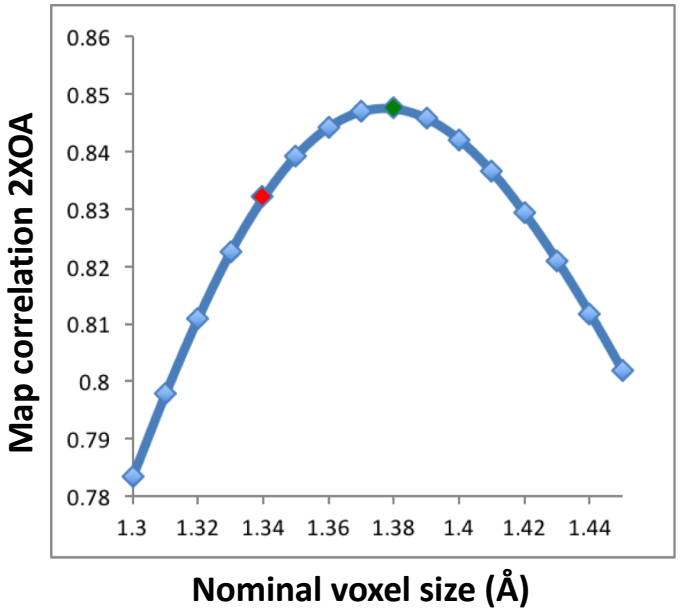
Using UCSF Chimera to fit solved domains

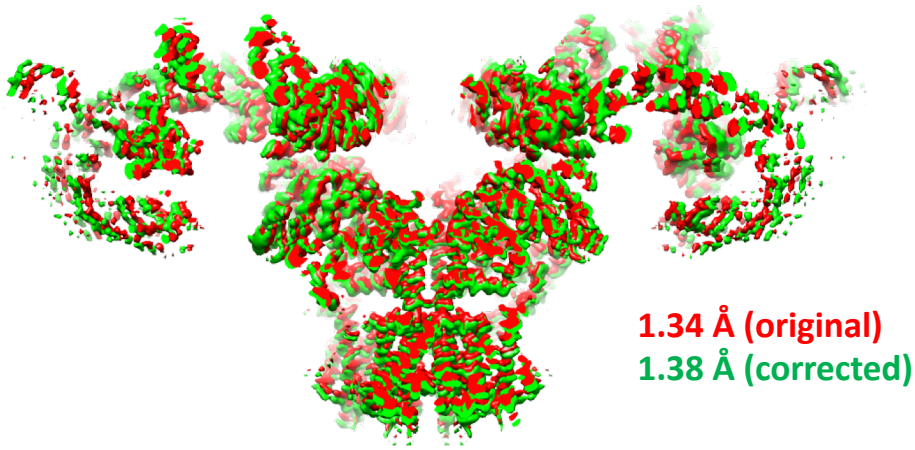


Chimera will return a list of candidate orientations, ranked by agreement with the map. Hopefully there will be a clear separation between the correct and incorrect solutions.

Using UCSF Chimera for voxel size calibration (of your map and others)

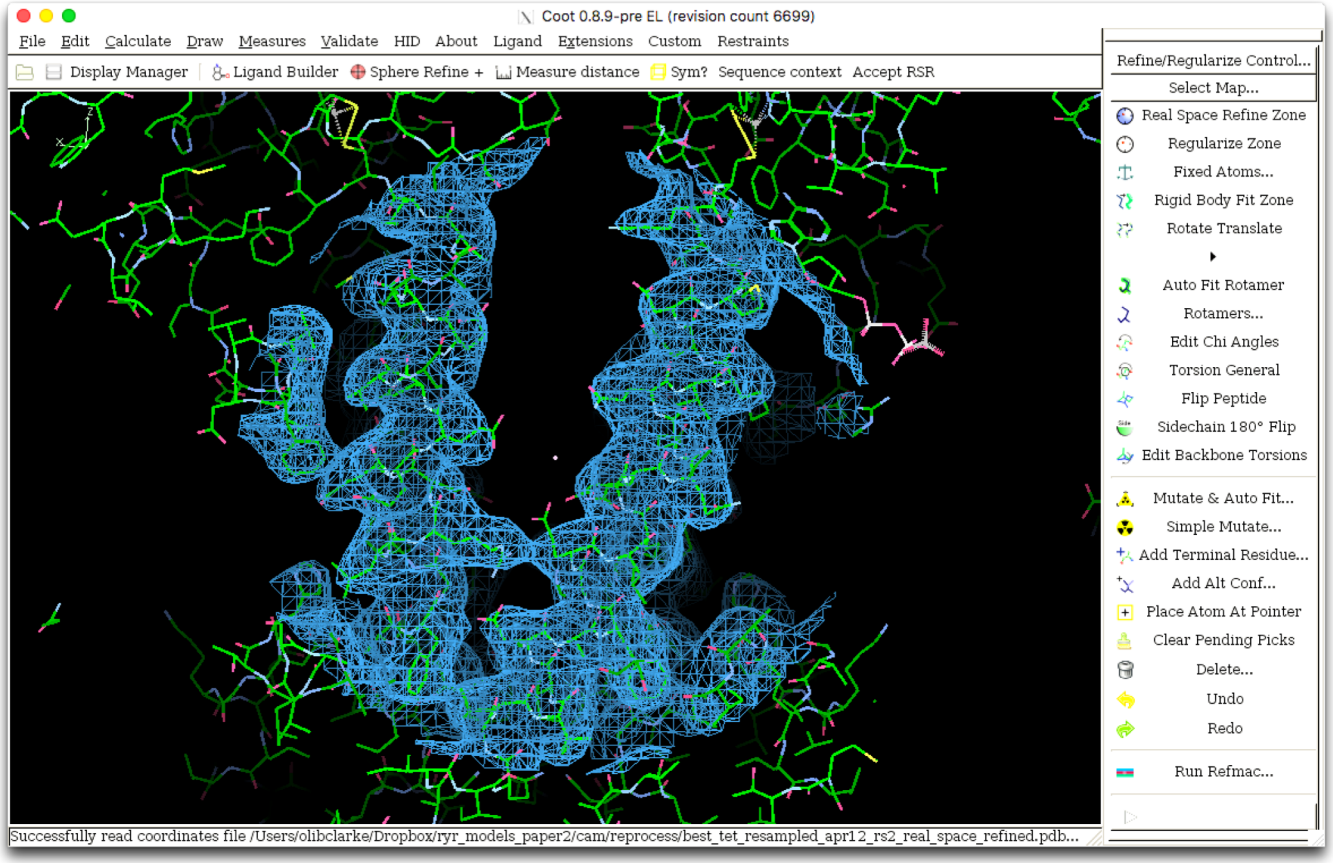
- Voxel size generally requires calibration against a crystal structure.
- Once calibrated, generally stable between samples/datasets at same magnification.
- Can calibrate by fitting in Chimera at range of nominal voxel sizes and measuring correlation.
- Incorrect voxel sizes are common in deposited maps - **be aware of this when comparing structures**. E.g. here there is a 3% difference – affects structural alignment, reported resolution (3.8 vs 3.9Å).





COOT – Crystallographic Object Oriented Toolkit

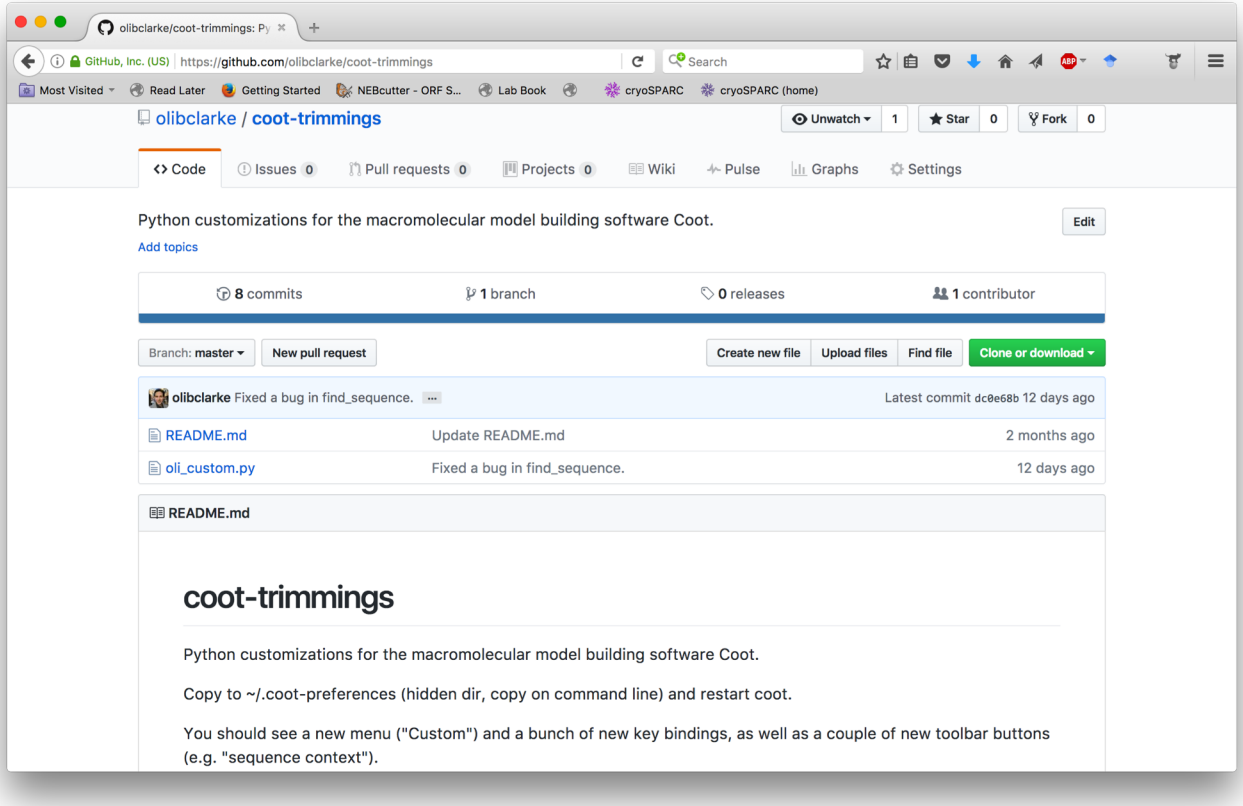
- Simple, intuitive interface for building and manipulating atomic models in density maps.
- Low computational requirements
- Extensive API – easy to script or modify (using simple Python code)
- On-the-fly sharpening and low pass filtering (for MTZ).



(Try the latest nightly with new features for EM, improved RSR: http://www.ccpem.ac.uk/download.php)

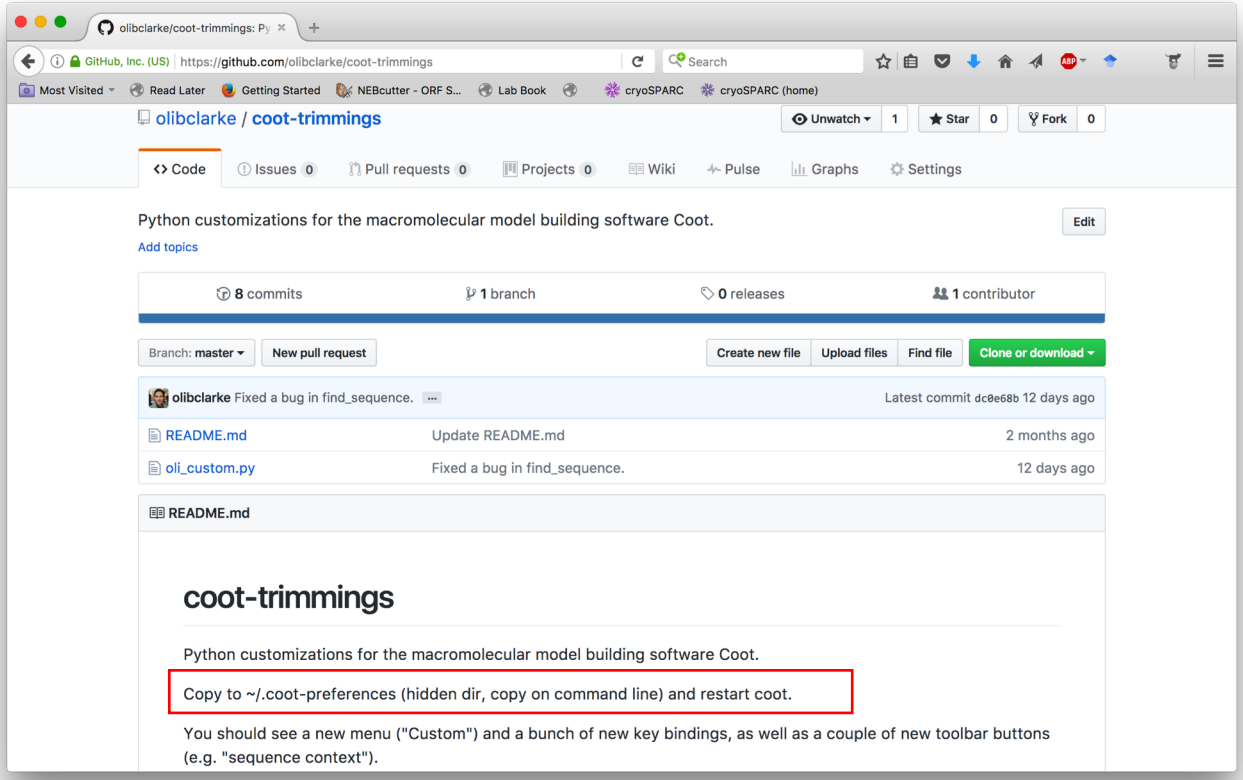
COOT – Crystallographic Object Oriented Toolkit

- Simple, intuitive interface for building and manipulating atomic models in density maps.
- Low computational requirements
- **Extensive API – easy to script or modify (using simple Python code)**
- On-the-fly sharpening and low pass filtering (for MTZ).



COOT – Crystallographic Object Oriented Toolkit

- Simple, intuitive interface for building and manipulating atomic models in density maps.
- Low computational requirements
- **Extensive API – easy to script or modify (using simple Python code)**
- On-the-fly sharpening and low pass filtering (for MTZ).



Any Python (or Scheme) file you put in ~/.coot-preferences will be executed when starting Coot. Can use this for extra key bindings, scripts, custom functions.

COOT – Crystallographic Object Oriented Toolkit

- Simple, intuitive interface for building and manipulating atomic models in density maps.
- Low computational requirements
- **Extensive API – easy to script or modify (using simple Python code)**
- On-the-fly sharpening and low pass filtering (for MTZ).

```
def mutate_by_entered_code():
  def mutate_single_letter(X):
    entry=str(X).upper()
    mol_id=active_residue()[0]
    ch_id=active_residue()[1]
    resno=active_residue()[2]
    ins_code=active_residue()[3]
    resname=residue_name(mol_id,ch_id,resno,ins_code)
    map_id=imol_refinement_map()
    aa_dic={'A':'ALA','R':'ARG','N':'ASN','D':'ASP','C':'CYS','E':'GLU','Q':'GLN','G':'GLY','H':'HIS','I':'ILE','L':'LEU','K':'LY
    nt_list=['A','C','T','G','U']
    if (resname in aa_dic.values()) and (aa_dic.get(entry,0)!=0):
      mutate(mol_id,ch_id,resno,ins_code,aa_dic.get(entry,0))
    elif (resname in nt_list) and (entry in nt_list):
      mutate_base(mol_id,ch_id,resno,ins_code,entry)
    else:
      info_dialog("Invalid target residue! Must be protein or nucleic acid, and entered code must be single letter.")
  generic_single_entry("New residue? (single letter code)","A","Mutate by single-letter code",mutate_single_letter)
```

```
#mutate active residue to entered residue code (upper or lower case single-letter)
add_key_binding("Mutate by single letter code","M",
lambda: mutate_by_entered_code())
```

COOT – Crystallographic Object Oriented Toolkit

- Simple, intuitive interface for building and manipulating atomic models in density maps.
- Low computational requirements
- **Extensive API – easy to script or modify (using simple Python code)**
- On-the-fly sharpening and low pass filtering (for MTZ).

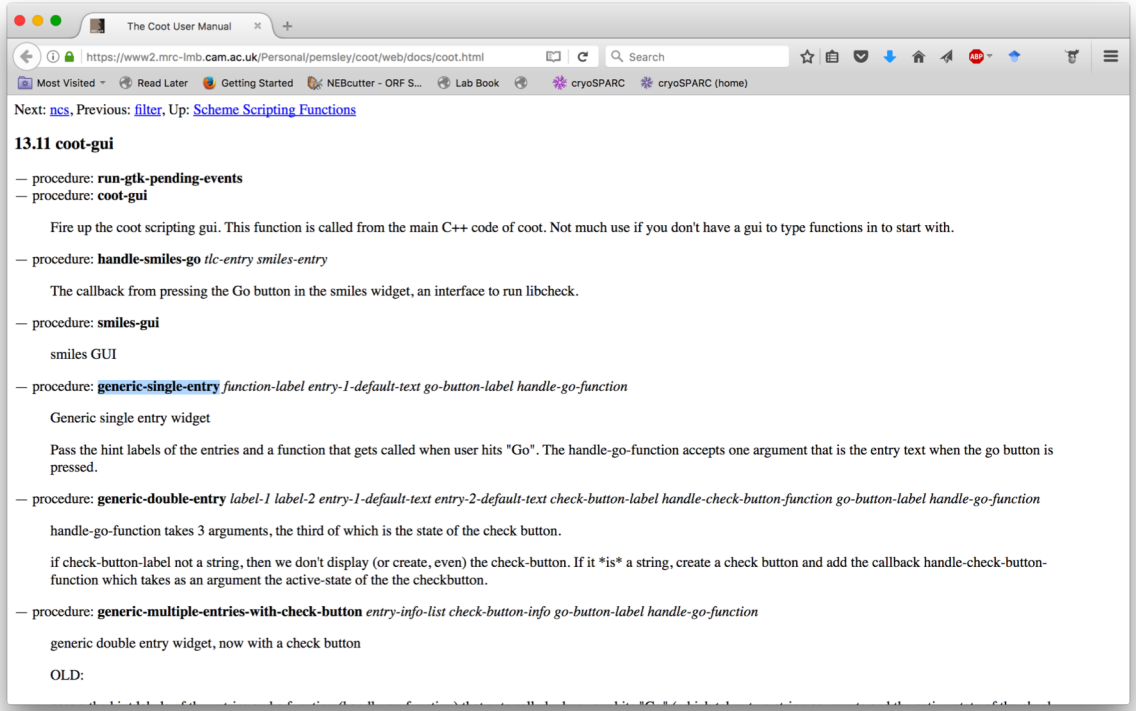
```
def mutate_by_entered_code():
  def mutate_single_letter(X):
    entry=str(X).upper()
    mol_id=active_residue()[0]
    ch_id=active_residue()[1]
    resno=active_residue()[2]
    ins_code=active_residue()[3]
    resname=residue_name(mol_id,ch_id,resno,ins_code)
    map_id=imol_refinement_map()
    aa_dic={'A':'ALA','R':'ARG','N':'ASN','D':'ASP','C':'CYS','E':'GLU','Q':'GLN','G':'GLY','H':'HIS','I':'ILE','L':'LEU','K':'LY
    nt_list=['A','C','T','G','U']
    if (resname in aa_dic.values()) and (aa_dic.get(entry,0)!=0):
      mutate(mol_id,ch_id,resno,ins_code,aa_dic.get(entry,0))
    elif (resname in nt_list) and (entry in nt_list):
      mutate_base(mol_id,ch_id,resno,ins_code,entry)
    else:
      info_dialog("Invalid target residue! Must be protein or nucleic acid, and entered code must be single letter.")
  generic_single_entry("New residue? (single letter code)","A","Mutate by single-letter code",mutate_single_letter)
```

```
#mutate active residue to entered residue code (upper or lower case single-letter)
add_key_binding("Mutate by single letter code","M",
lambda: mutate_by_entered_code())
```

Many pre-packaged functions available in COOT API. Mostly documented in online manual. Very easy to write your own! Useful e.g. for scripting domain-wise rigid body refinement.

COOT – Crystallographic Object Oriented Toolkit

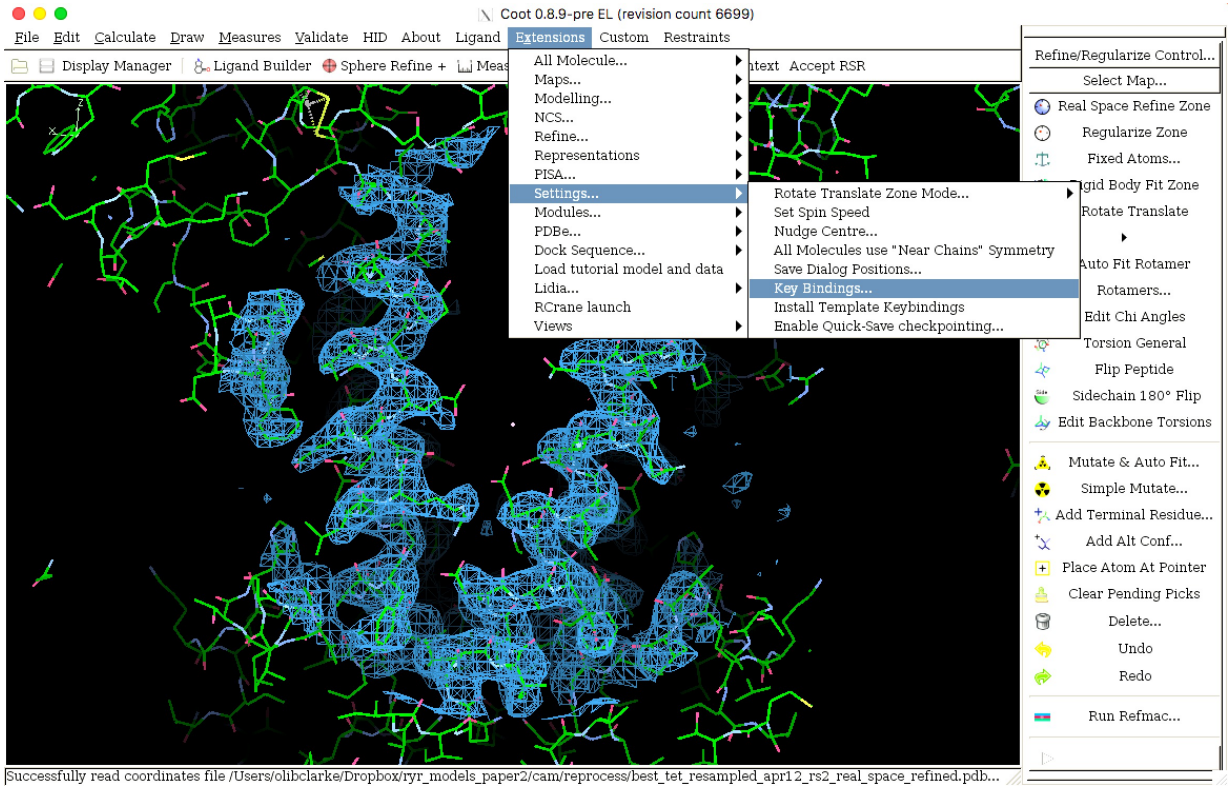
- Simple, intuitive interface for building and manipulating atomic models in density maps.
- Low computational requirements
- **Extensive API – easy to script or modify (using simple Python code)**
- On-the-fly sharpening and low pass filtering (for MTZ).



Many pre-packaged functions available in COOT API. Mostly documented in online manual.

COOT – Crystallographic Object Oriented Toolkit

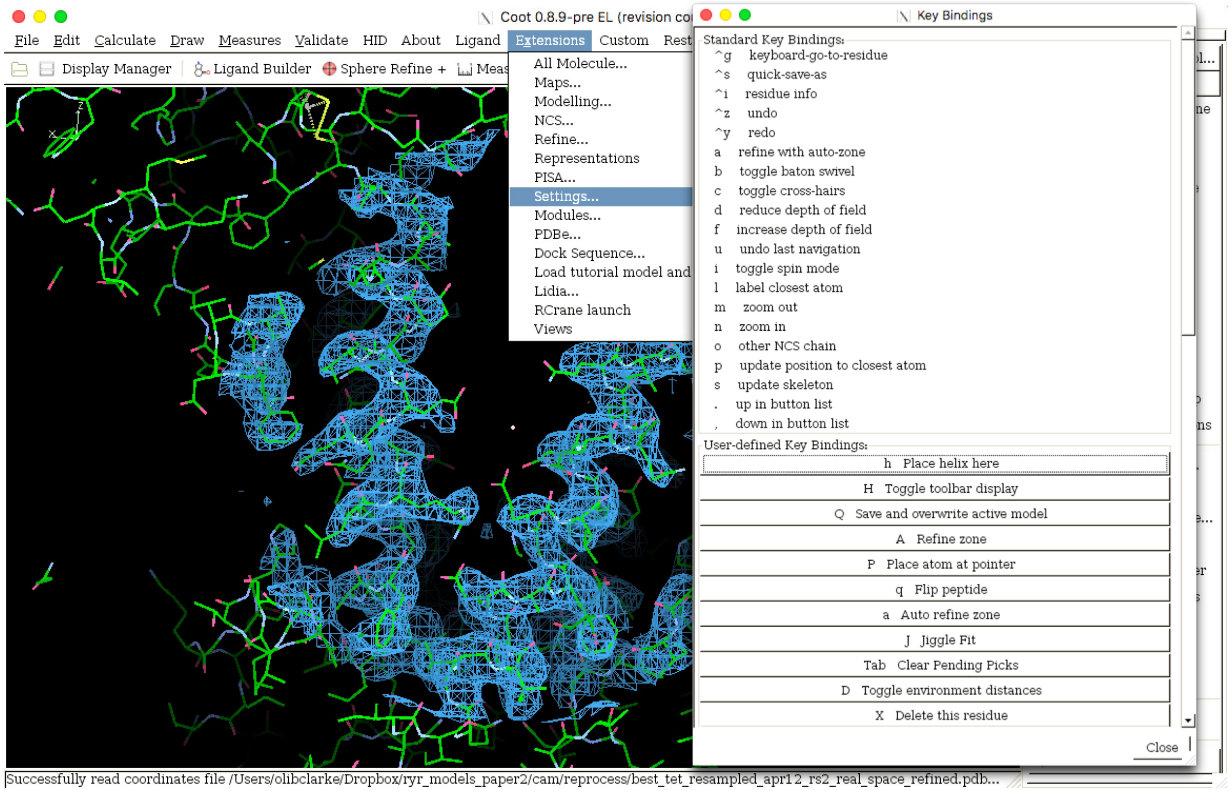
- Simple, intuitive interface for building and manipulating atomic models in density maps.
- Low computational requirements
- **Extensive API – easy to script or modify (using simple Python code)**
- On-the-fly sharpening and low pass filtering (for MTZ).



Lots of key bindings, and easy to define custom keys. Learn them. They make everything much faster.

COOT – Crystallographic Object Oriented Toolkit

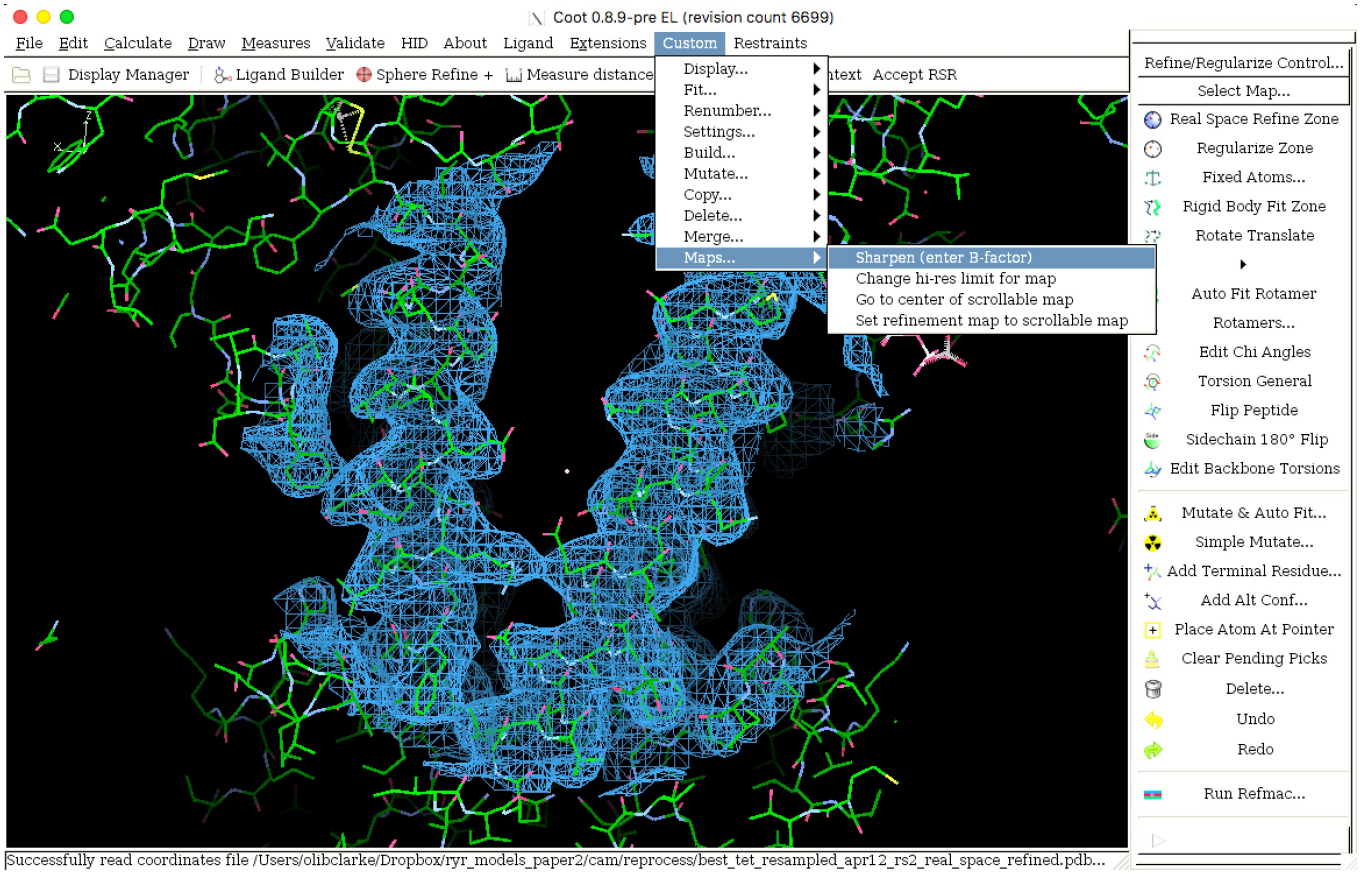
- Simple, intuitive interface for building and manipulating atomic models in density maps.
- Low computational requirements
- **Extensive API – easy to script or modify (using simple Python code)**
- On-the-fly sharpening and low pass filtering (for MTZ).



Lots of key bindings, and easy to define custom keys. Learn them. They make everything much faster.

COOT – Crystallographic Object Oriented Toolkit

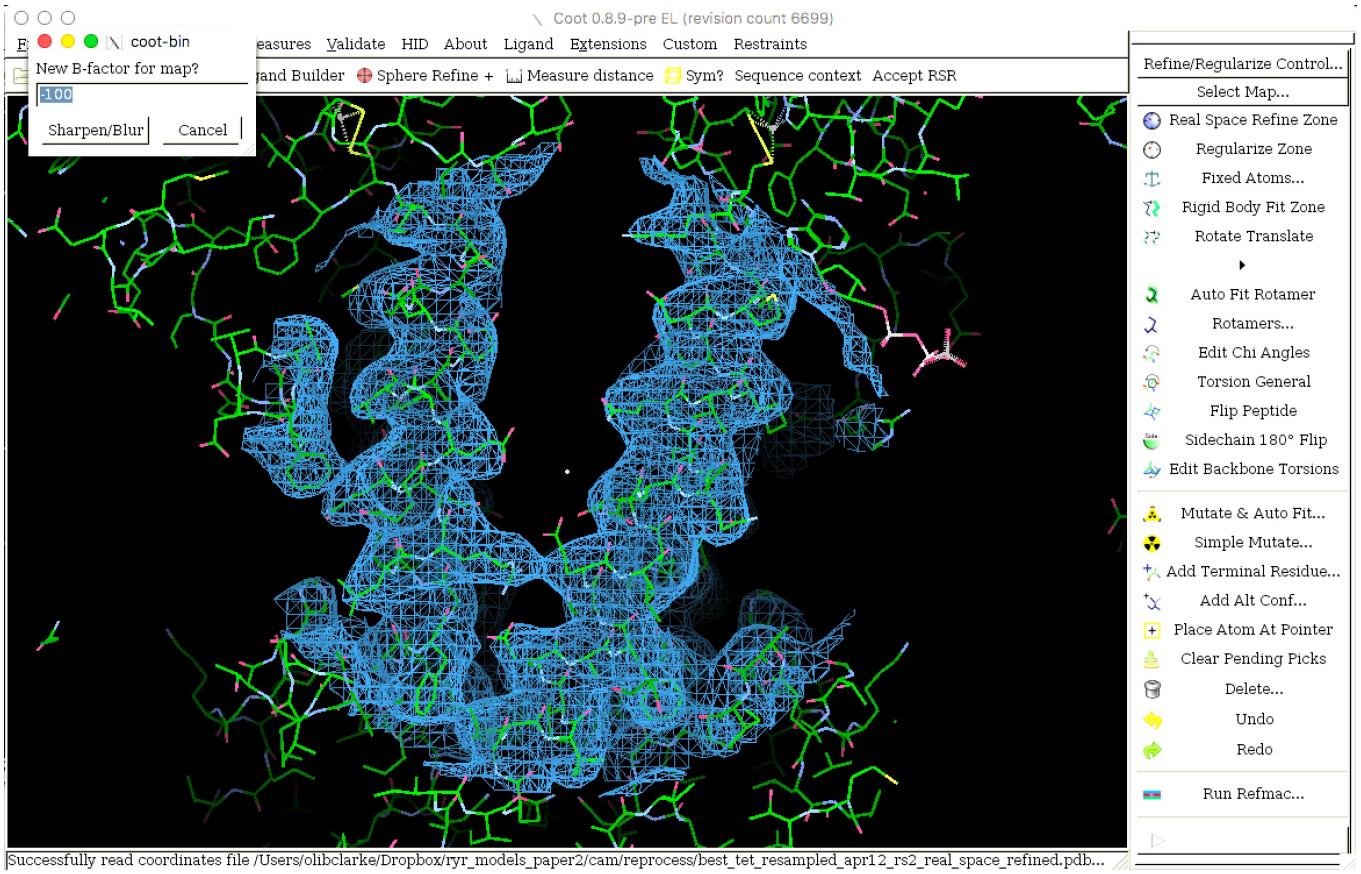
- Simple, intuitive interface for building and manipulating atomic models in density maps.
- Low computational requirements
- Extensive API – easy to script or modify (using simple Python code)
- **On-the-fly sharpening and low pass filtering (for MTZ).**



You can convert an mrc to mtz using *`phenix.map_to_structure_factors`*.

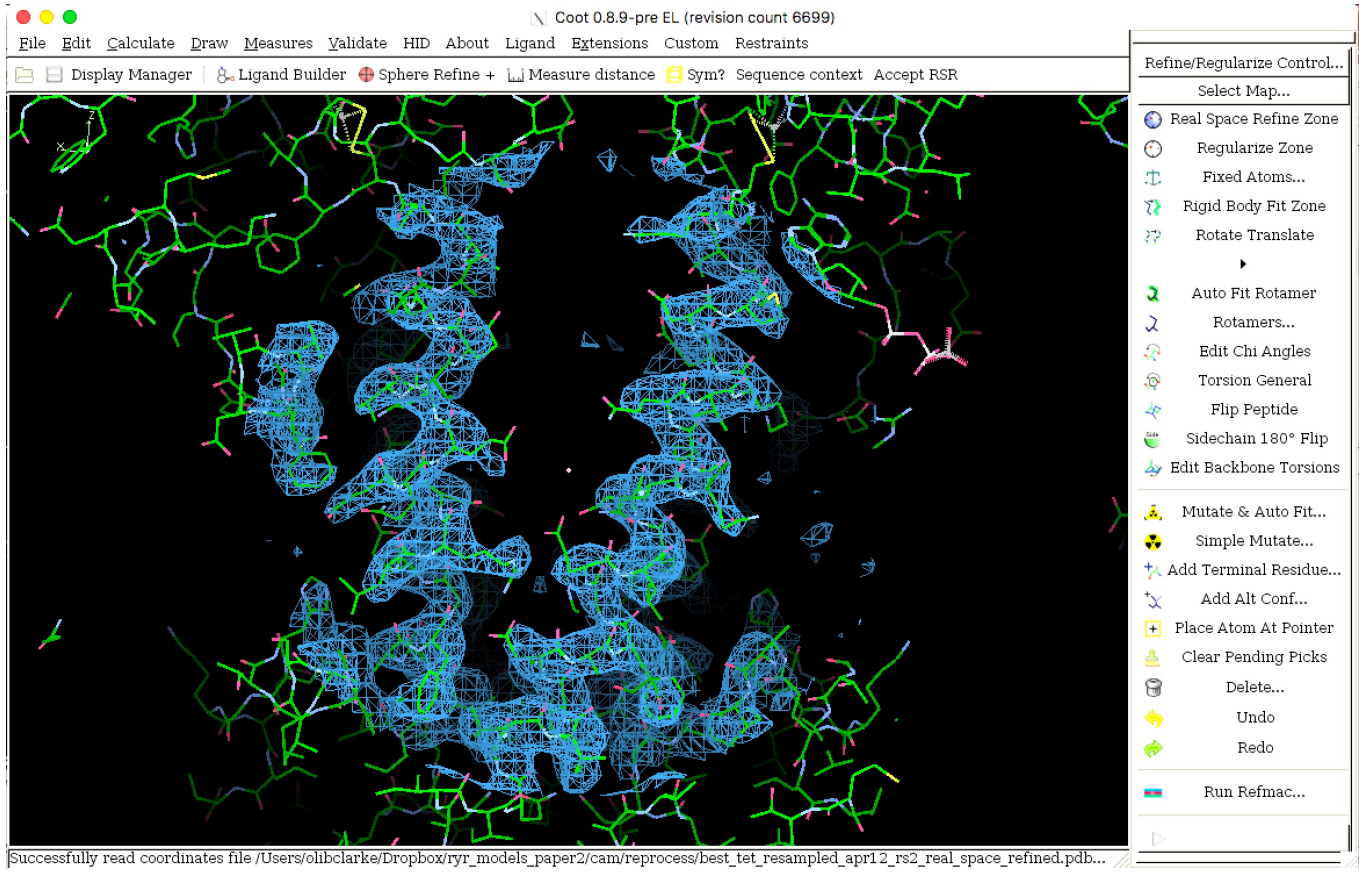
COOT – Crystallographic Object Oriented Toolkit

- Simple, intuitive interface for building and manipulating atomic models in density maps.
- Low computational requirements
- Extensive API – easy to script or modify (using simple Python code)
- **On-the-fly sharpening and low pass filtering (for MTZ).**



COOT – Crystallographic Object Oriented Toolkit

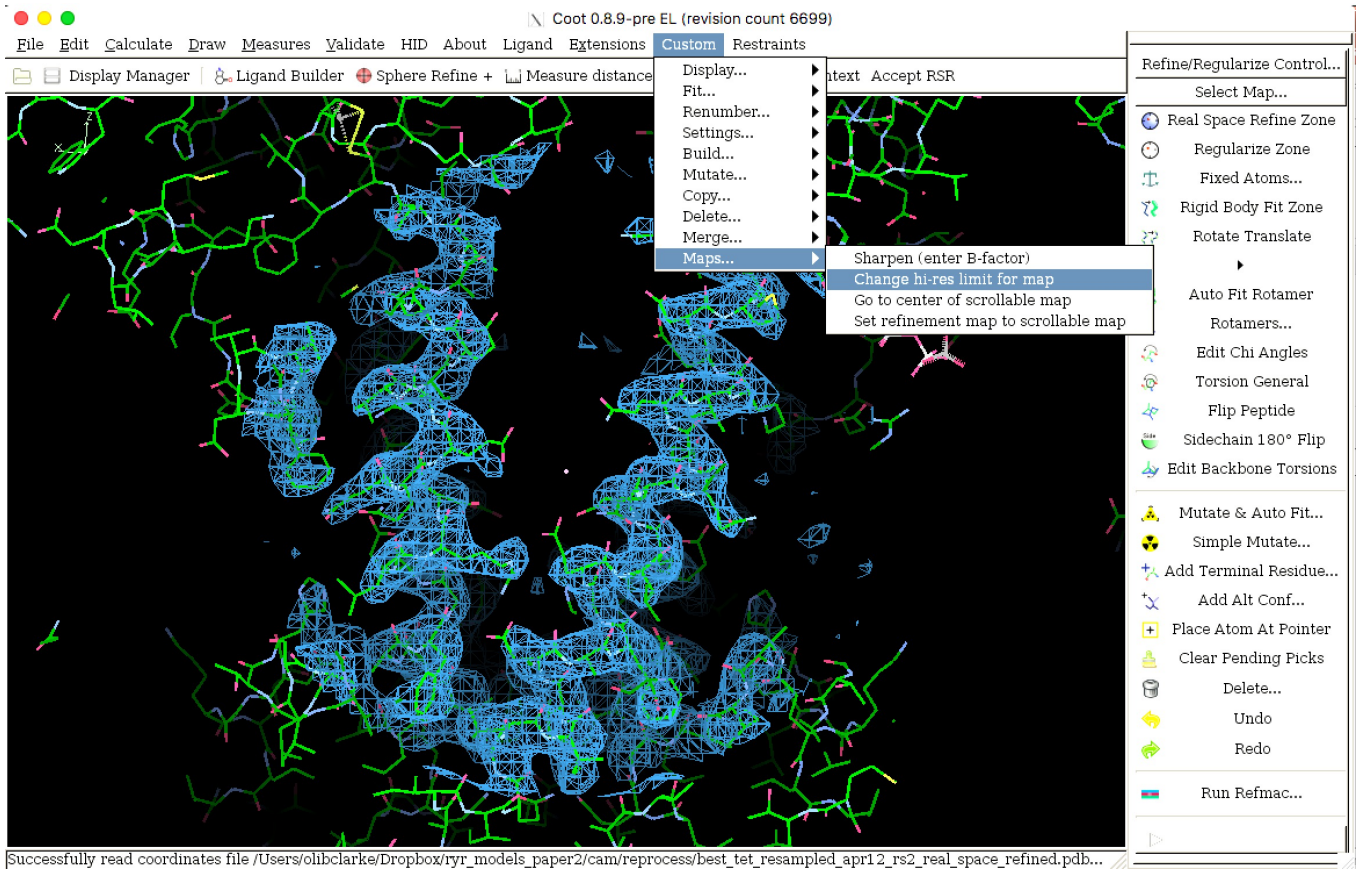
- Simple, intuitive interface for building and manipulating atomic models in density maps.
- Low computational requirements
- Extensive API – easy to script or modify (using simple Python code)
- **On-the-fly sharpening and low pass filtering (for MTZ).**



The optimal sharpening B-factor will vary across the map (assuming some variation in local resolution). On-the-fly adjustment is therefore very useful. (*`phenix.auto_sharpen`* is good for determining

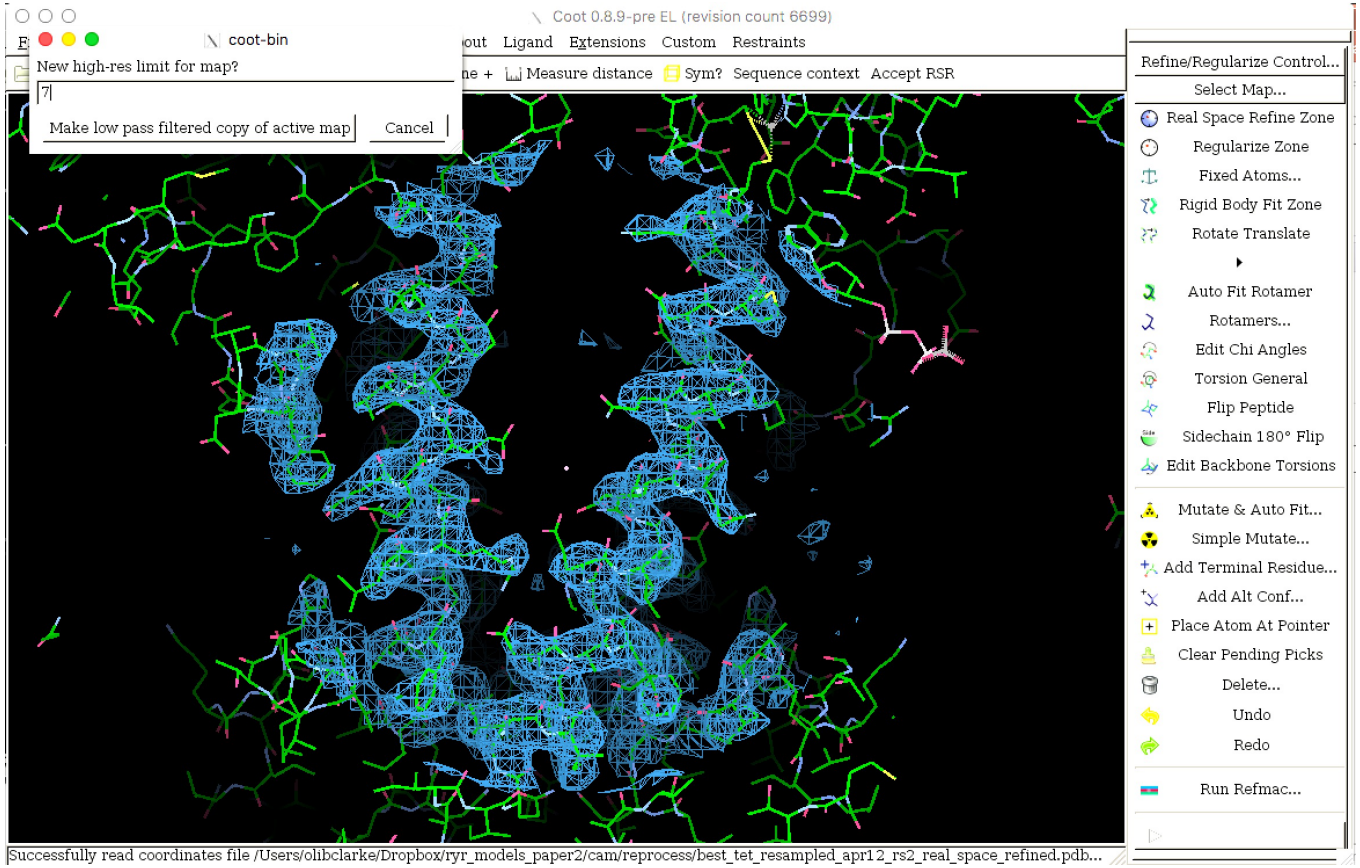
COOT – Crystallographic Object Oriented Toolkit

- Simple, intuitive interface for building and manipulating atomic models in density maps.
- Low computational requirements
- Extensive API – easy to script or modify (using simple Python code)
- **On-the-fly sharpening and low pass filtering (for MTZ).**



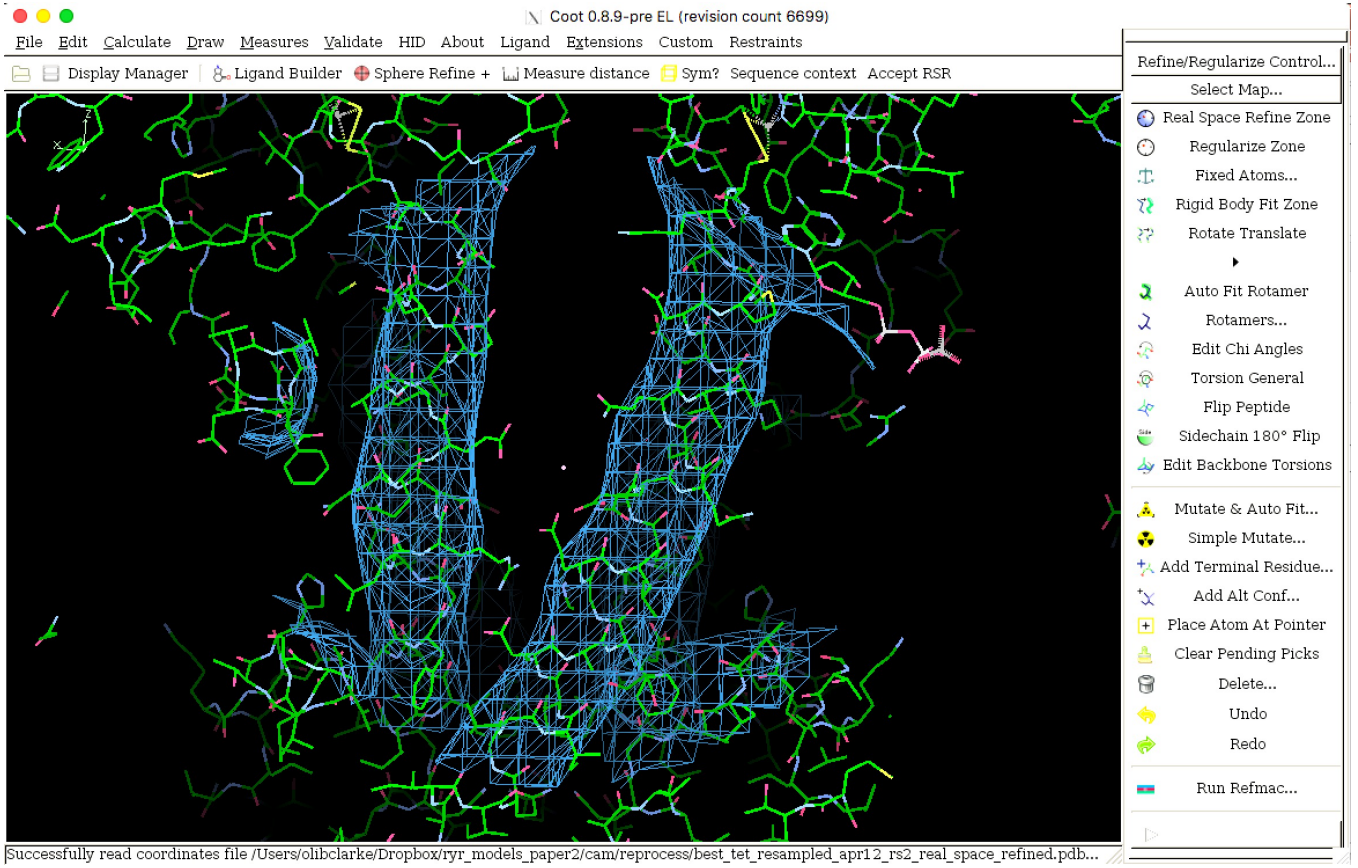
COOT – Crystallographic Object Oriented Toolkit

- Simple, intuitive interface for building and manipulating atomic models in density maps.
- Low computational requirements
- Extensive API – easy to script or modify (using simple Python code)
- **On-the-fly sharpening and low pass filtering (for MTZ).**



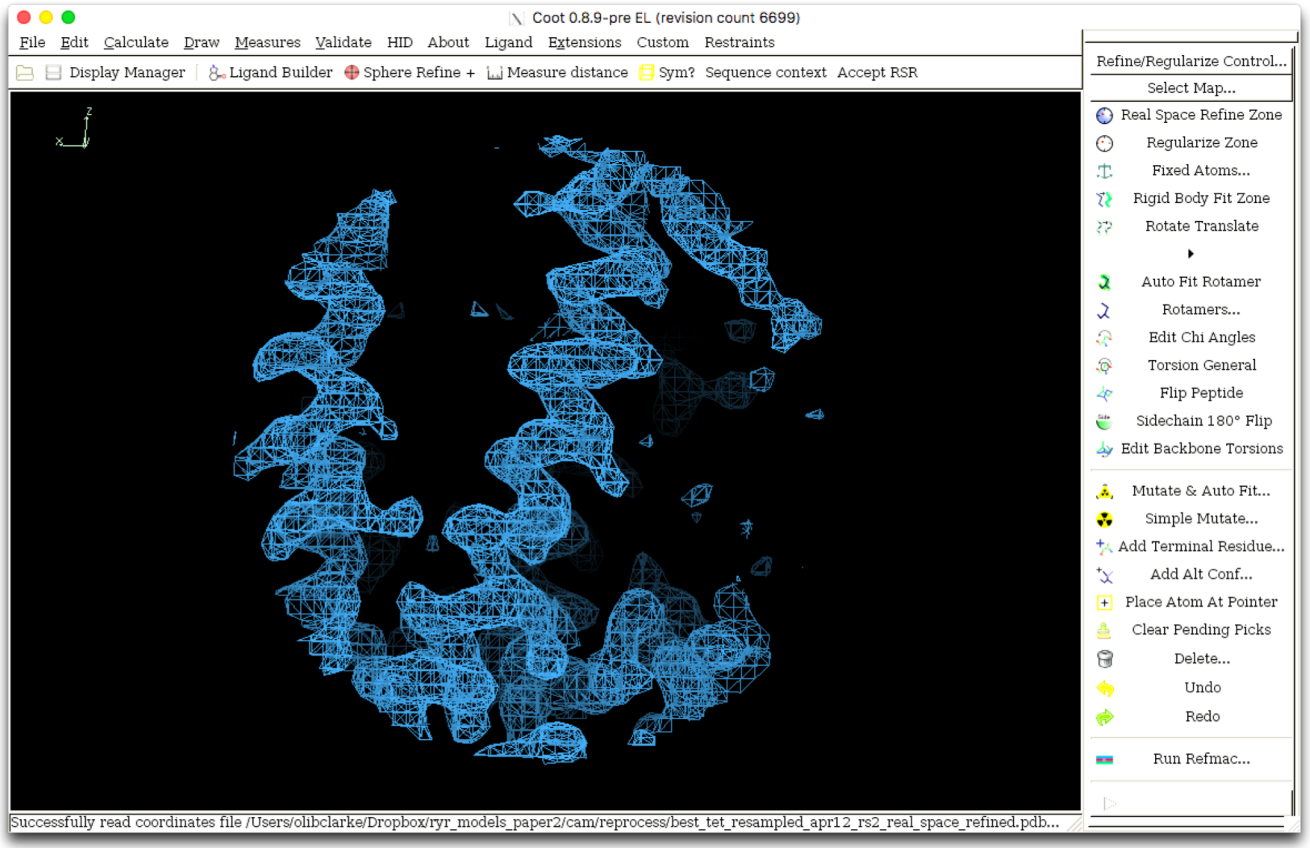
COOT – Crystallographic Object Oriented Toolkit

- Simple, intuitive interface for building and manipulating atomic models in density maps.
- Low computational requirements
- Extensive API – easy to script or modify (using simple Python code)
- **On-the-fly sharpening and low pass filtering (for MTZ).**



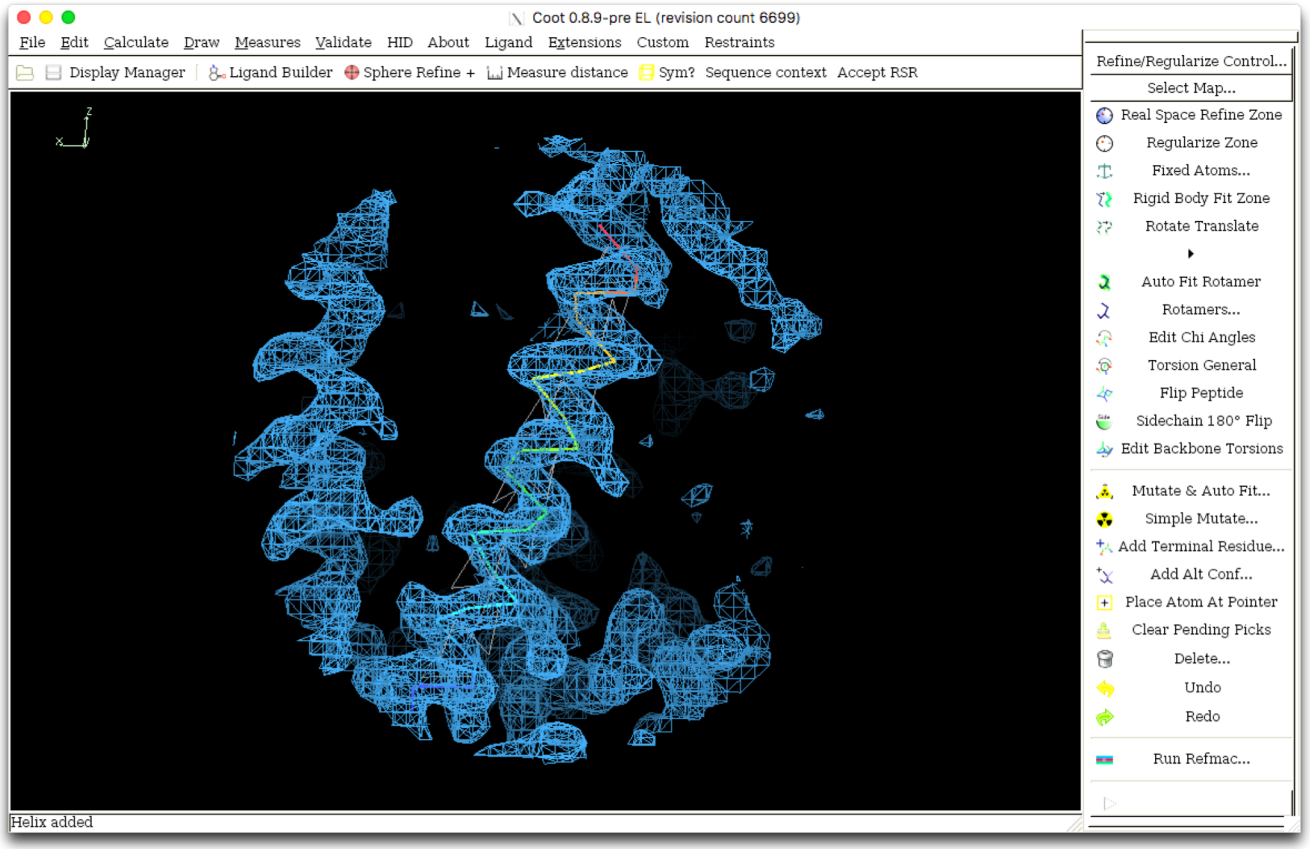
COOT – Crystallographic Object Oriented Toolkit

- Semi-automated helix placement
- Place cursor at the center of the helix and trigger ”Place helix here” (I suggest via a key binding - “h” with *`coot-trimmings`*)
- Coot will attempt to automatically determine the length and direction of the helix.
- Trim/extend, adjust weights, then refine using real-space refine zone. Drag into density to adjust fit.



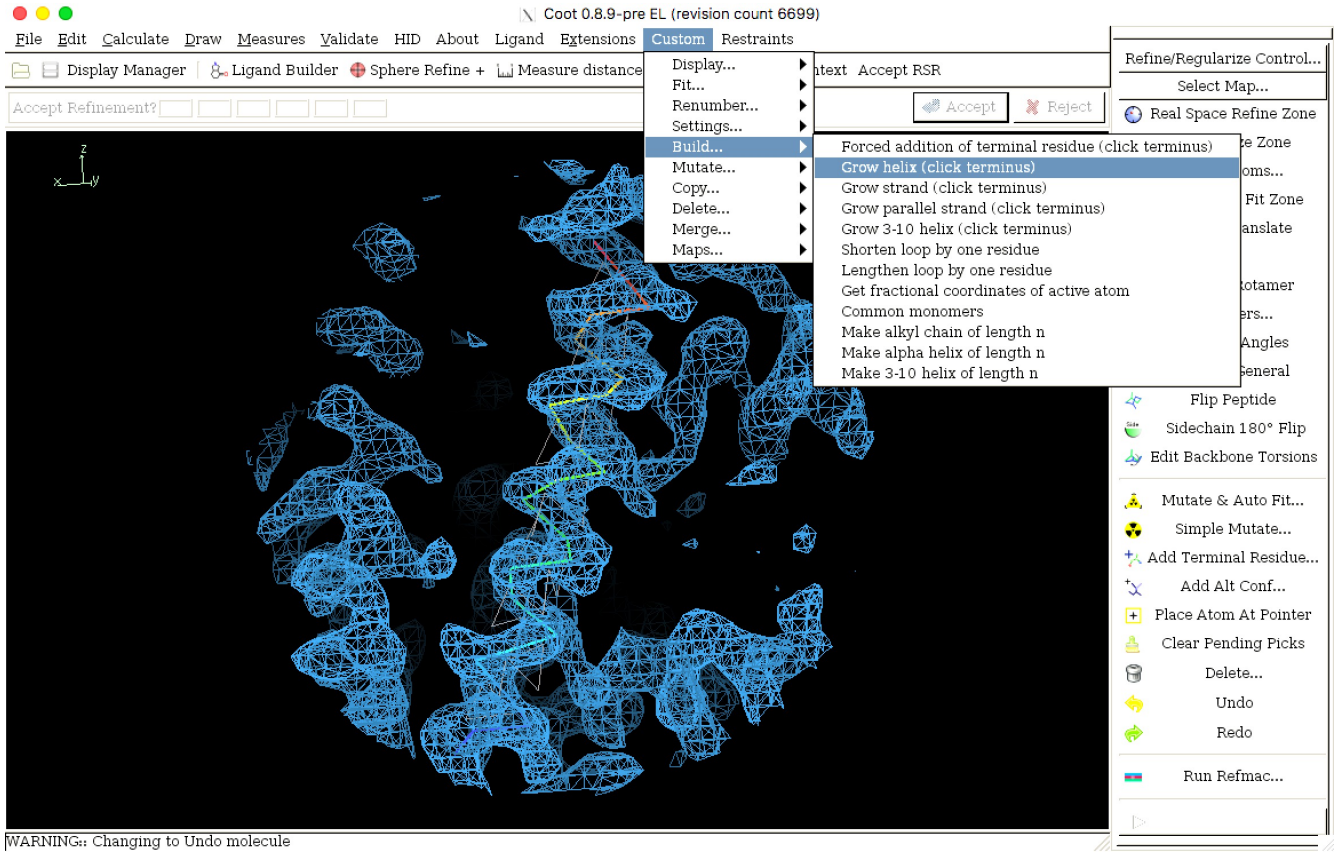
COOT – Crystallographic Object Oriented Toolkit

- Semi-automated helix placement
- **Place cursor at the center of the helix and trigger ”Place helix here” (I suggest via a key binding - “h” with *`coot-trimmings`*)**
- Coot will attempt to automatically determine the length and direction of the helix.
- Trim/extend, adjust weights, then refine using real-space refine zone. Drag into density to adjust fit.



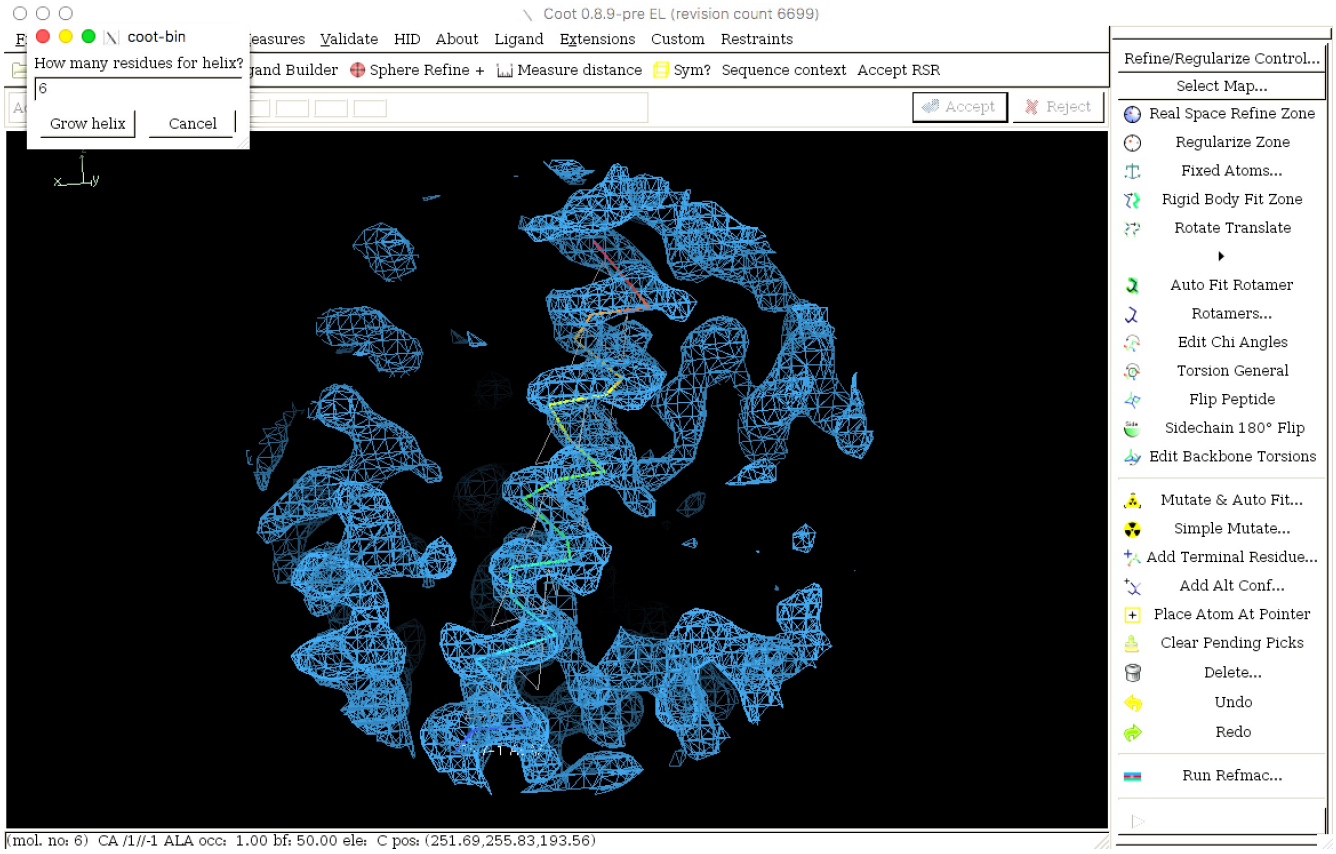
COOT – Crystallographic Object Oriented Toolkit

- Semi-automated helix placement
- Place cursor at the center of the helix and trigger ”Place helix here” (I suggest via a key binding - “h” with *`coot-trimmings`*)
- Coot will attempt to automatically determine the length and direction of the helix.
- **Trim/extend, adjust weights, then refine using real-space refine zone. Drag into density to adjust fit.**



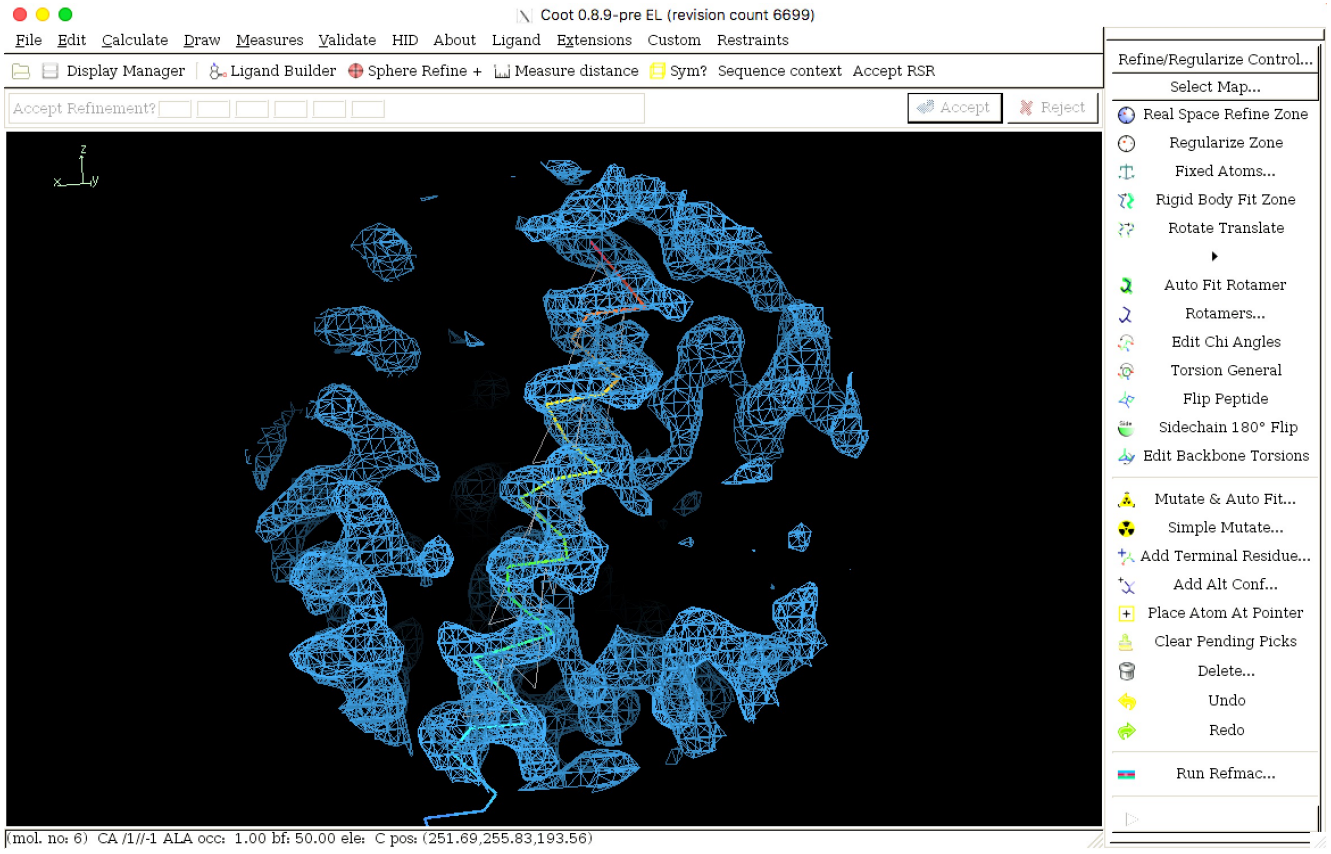
COOT – Crystallographic Object Oriented Toolkit

- Semi-automated helix placement
- Place cursor at the center of the helix and trigger ”Place helix here” (I suggest via a key binding - “h” with *`coot-trimmings`*)
- Coot will attempt to automatically determine the length and direction of the helix.
- **Trim/extend, adjust weights, then refine using real-space refine zone. Drag into density to adjust fit.**



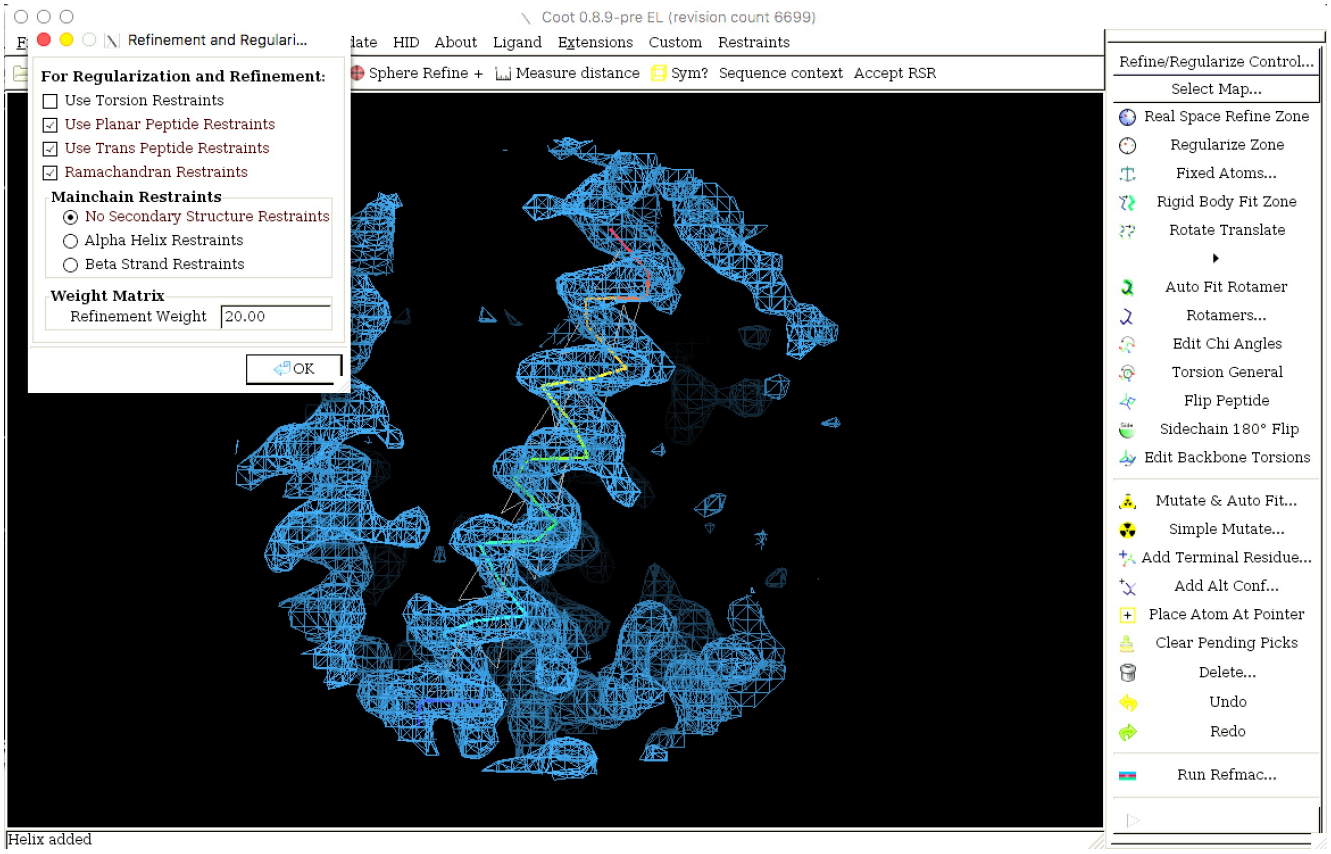
COOT – Crystallographic Object Oriented Toolkit

- Semi-automated helix placement
- Place cursor at the center of the helix and trigger ”Place helix here” (I suggest via a key binding - “h” with *`coot-trimmings`*)
- Coot will attempt to automatically determine the length and direction of the helix.
- **Trim/extend, adjust weights, then refine using real-space refine zone. Drag into density to adjust fit.**



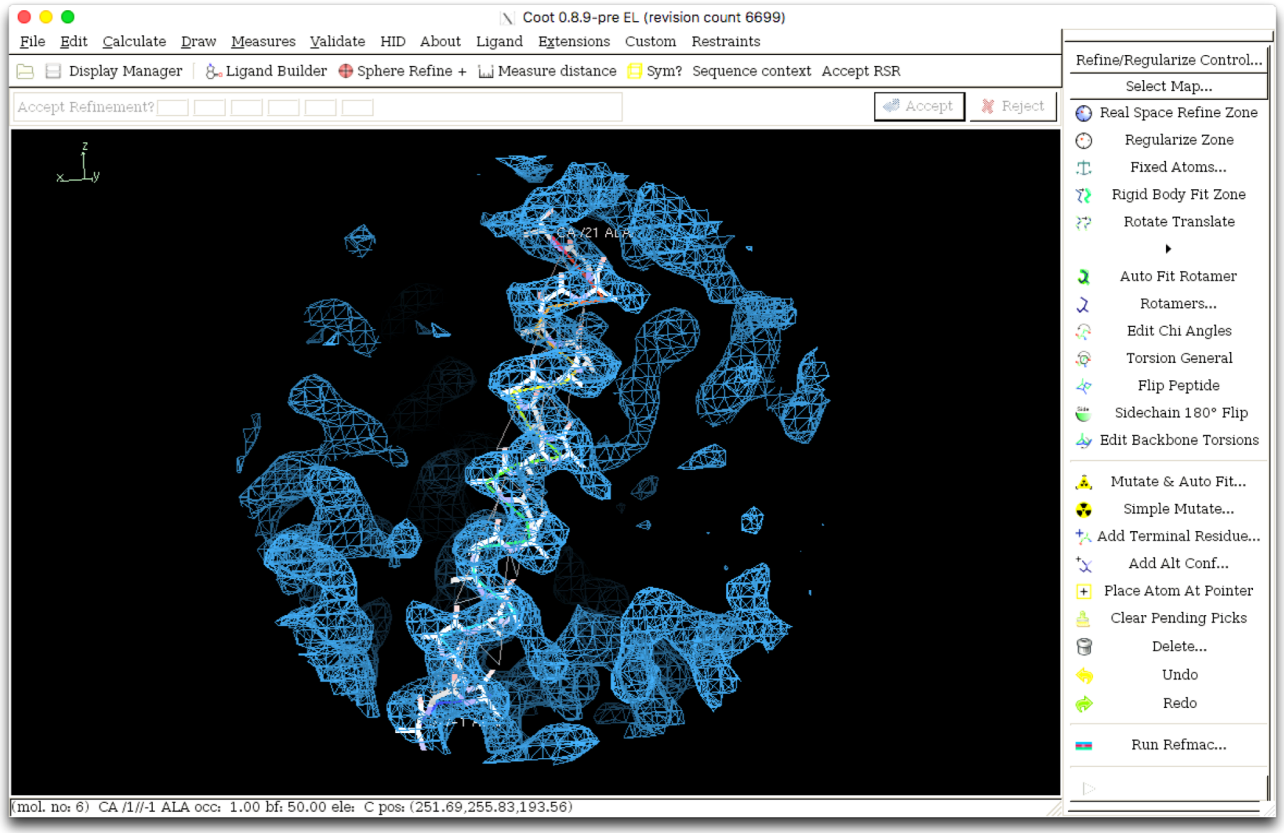
COOT – Crystallographic Object Oriented Toolkit

- Semi-automated helix placement
- Place cursor at the center of the helix and trigger ”Place helix here” (I suggest via a key binding - “h” with *`coot-trimmings`*)
- Coot will attempt to automatically determine the length and direction of the helix.
- **Trim/extend, adjust weights, then refine using real-space refine zone. Drag into density to adjust fit.**



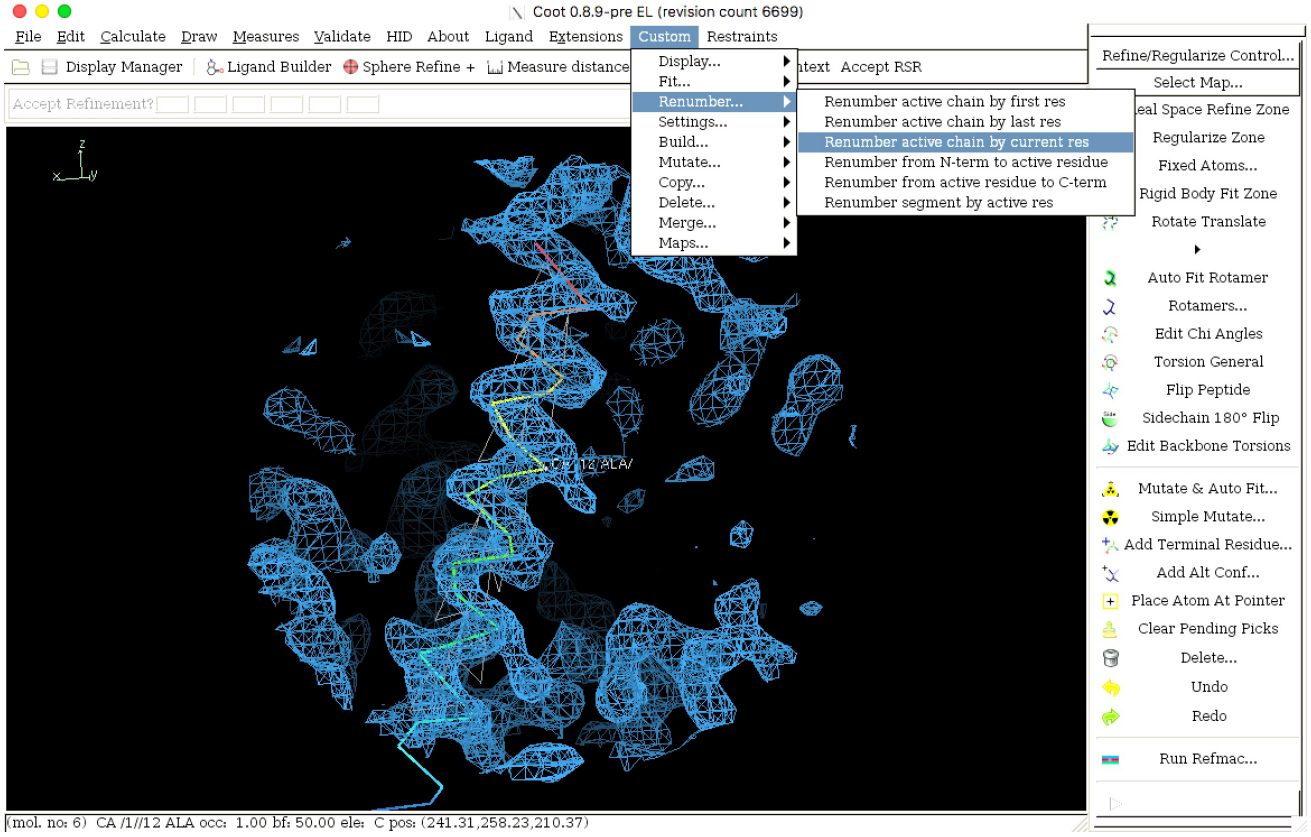
COOT – Crystallographic Object Oriented Toolkit

- Semi-automated helix placement
- Place cursor at the center of the helix and trigger ”Place helix here” (I suggest via a key binding - “h” with *`coot-trimmings`*)
- Coot will attempt to automatically determine the length and direction of the helix.
- **Trim/extend, adjust weights, then refine using real-space refine zone. Drag into density to adjust fit.**



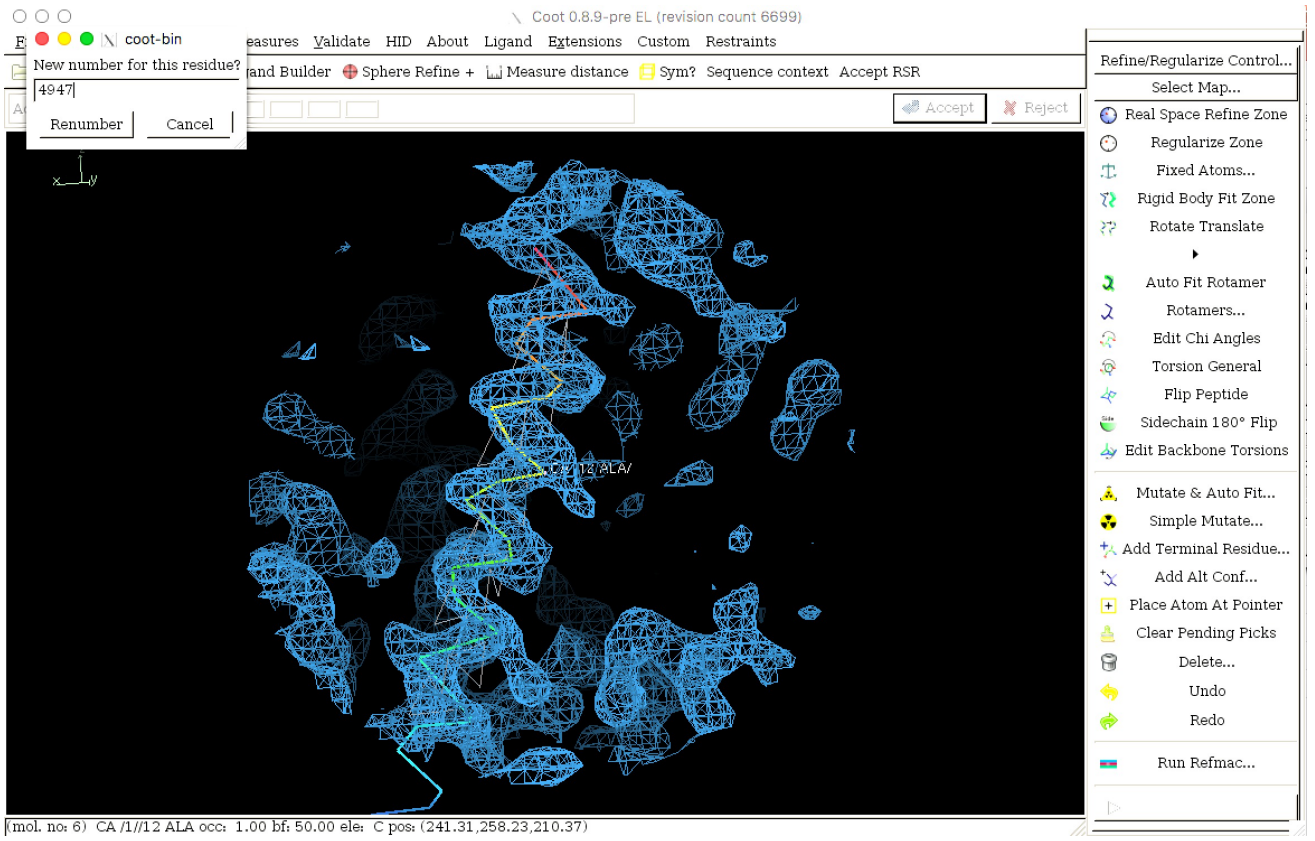
COOT – Crystallographic Object Oriented Toolkit

- Sequence assignment.
- **Adjust numbering to match expected position in sequence.**
- Mutate to match sequence
- Fill sidechains manually.
- Adjust sequence register to optimize local fit to sidechain densities.



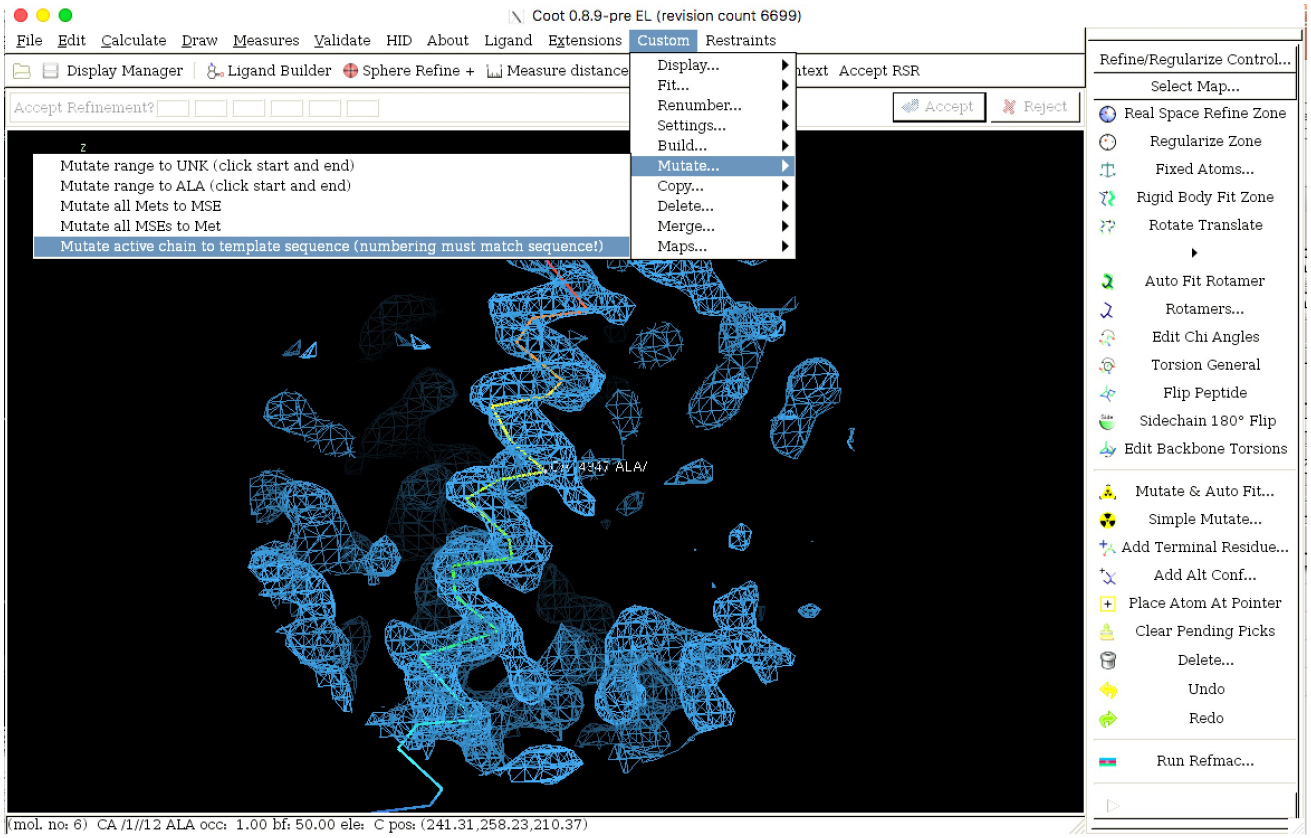
COOT – Crystallographic Object Oriented Toolkit

- Sequence assignment.
- **Adjust numbering to match expected position in sequence.**
- Mutate to match sequence
- Fill sidechains manually.
- Adjust sequence register to optimize local fit to sidechain densities.



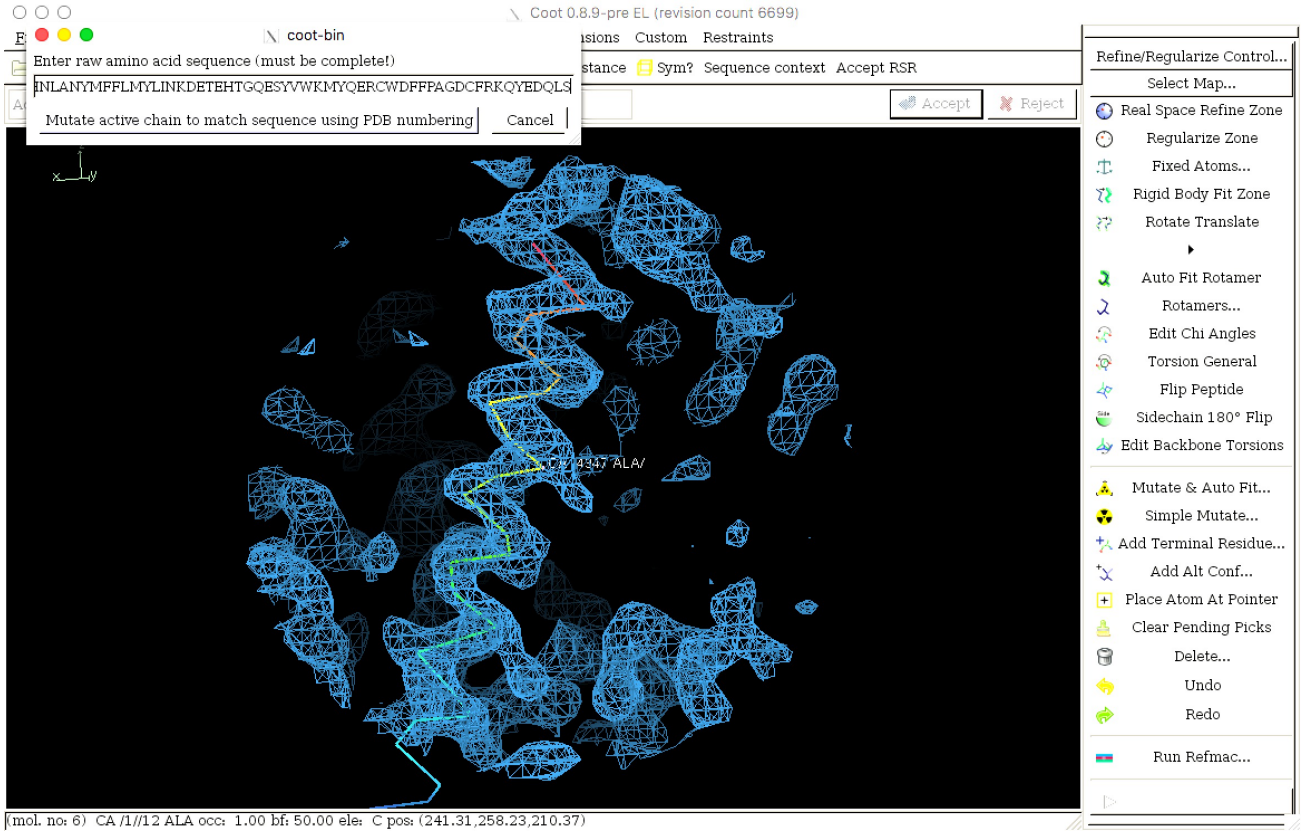
COOT – Crystallographic Object Oriented Toolkit

- Sequence assignment.
- Adjust numbering to match expected position in sequence.
- **Mutate to match sequence**
- Fill sidechains manually.
- Adjust sequence register to optimize local fit to sidechain densities.



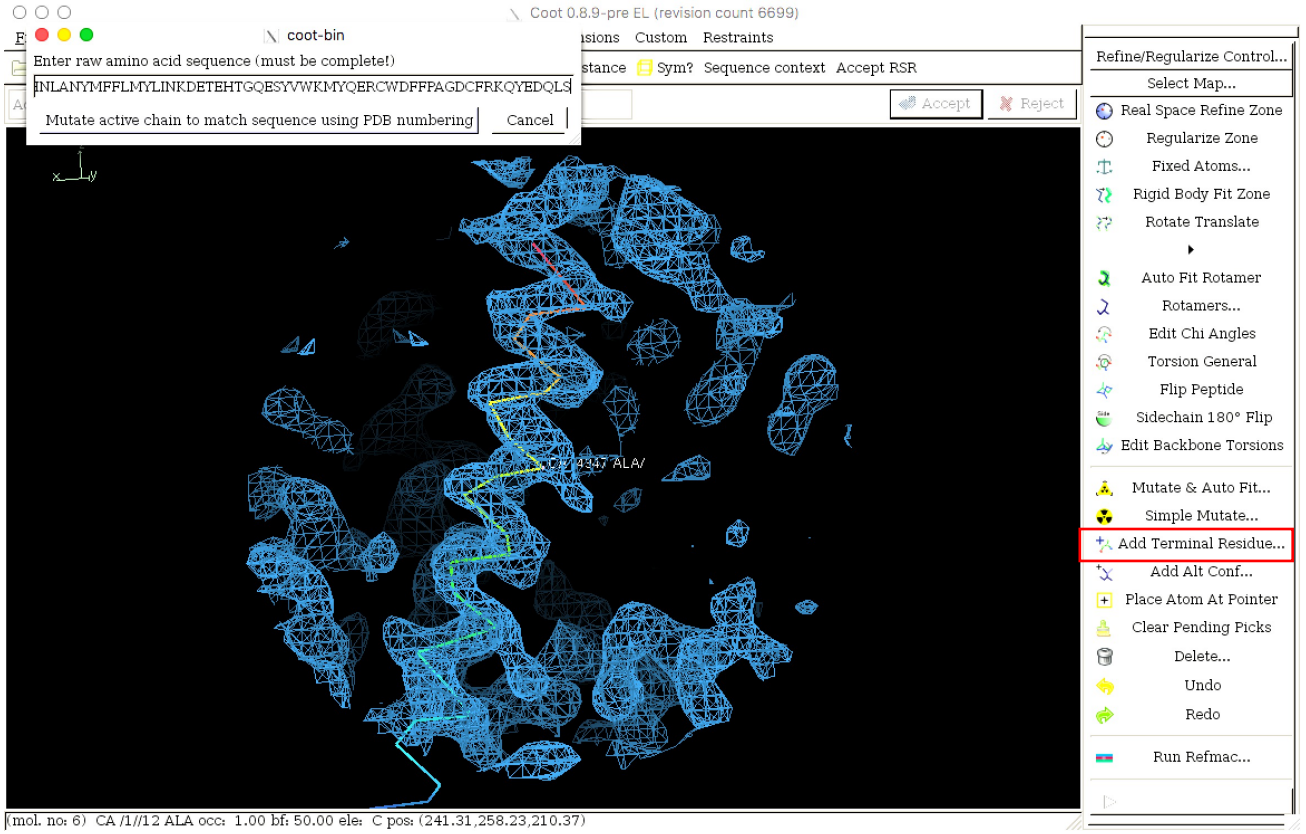
COOT – Crystallographic Object Oriented Toolkit

- Sequence assignment.
- Adjust numbering to match expected position in sequence.
- **Mutate to match sequence**
- Fill sidechains manually.
- Adjust sequence register to optimize local fit to sidechain densities.



COOT – Crystallographic Object Oriented Toolkit

- Sequence assignment.
- Adjust numbering to match expected position in sequence.
- **Mutate to match sequence**
- Fill sidechains manually.
- Adjust sequence register to optimize local fit to sidechain densities.

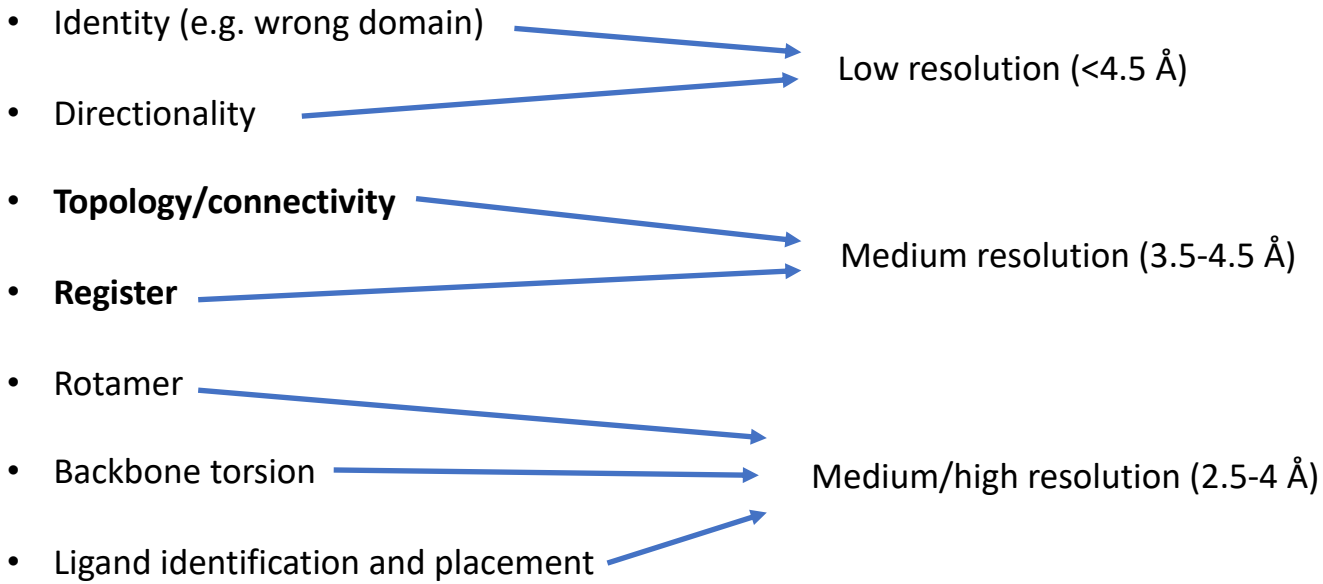


Use 'Add Terminal residue' to extend chain.

Types of errors in macromolecular models

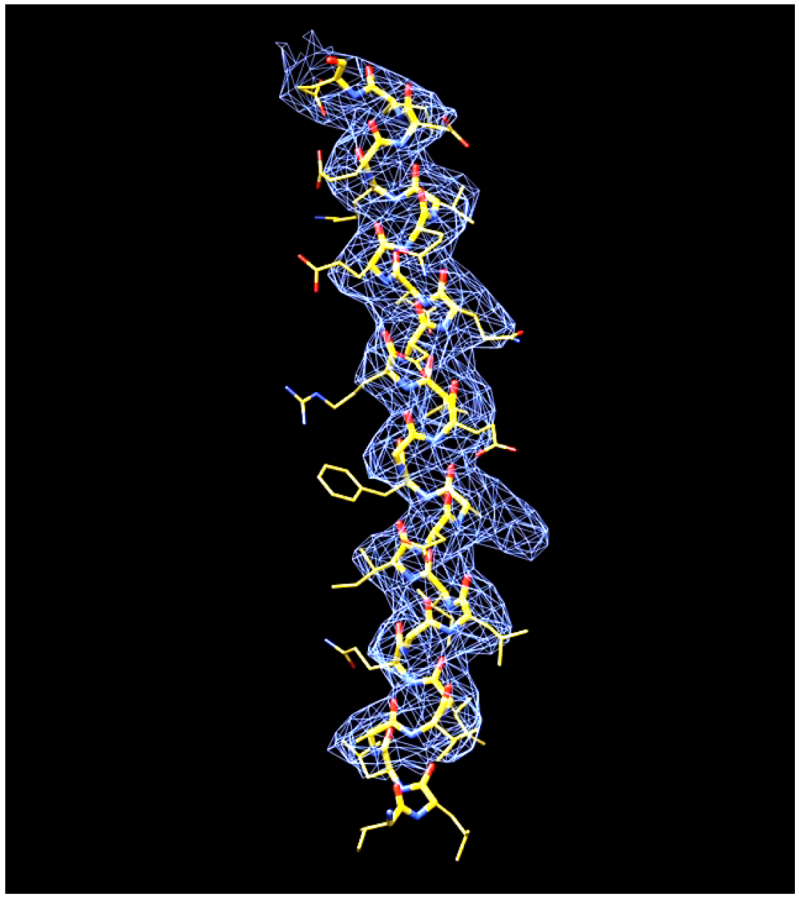
- Identity (e.g. wrong domain)
- Directionality
- Topology/connectivity
- Register
- Rotamer
- Backbone torsion
- Ligand identification and placement

Types of errors in macromolecular models



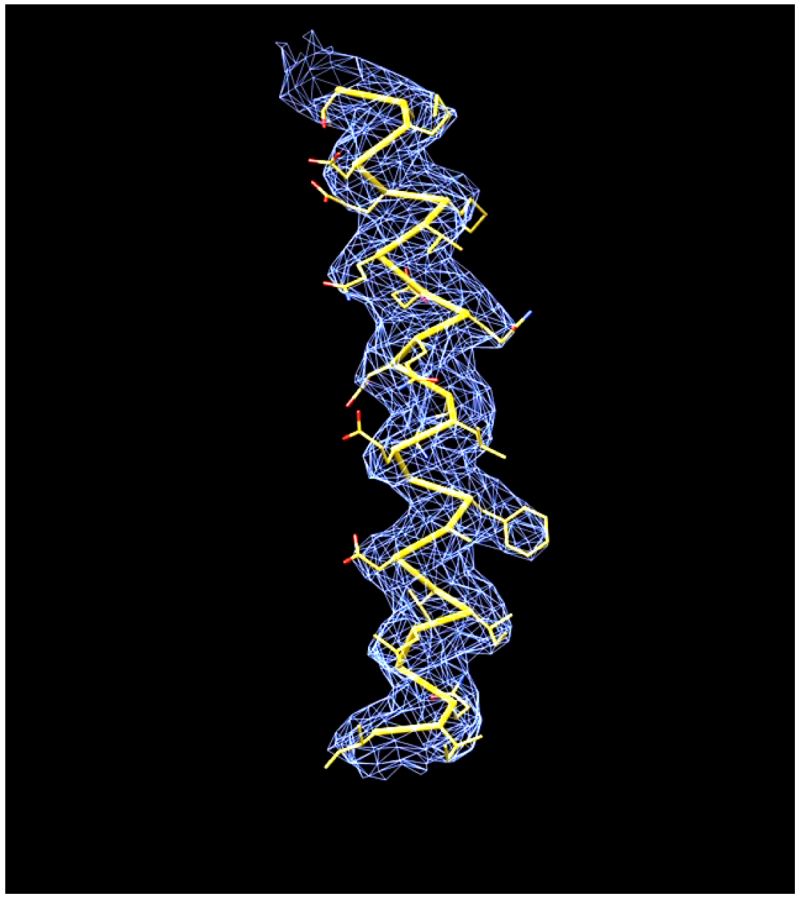
Types of errors in macromolecular models

- Identity (e.g. wrong domain)
- Directionality
- Topology/connectivity
- **Register**
- Rotamer
- Backbone torsion
- Ligand identification and placement



Types of errors in macromolecular models

- Identity (e.g. wrong domain)
- Directionality
- Topology/connectivity
- **Register**
- Rotamer
- Backbone torsion
- Ligand identification and placement



Strategy for identifying and correcting errors.

- Analyse as you go – “sanity checks” on chemistry, nonbonded interactions, surface composition. Use Molprobity for clashes, Chimera or pymol to check e.g. for buried polars, exposed hydrophobics. Monitor agreement with secondary structure, disorder predictions.

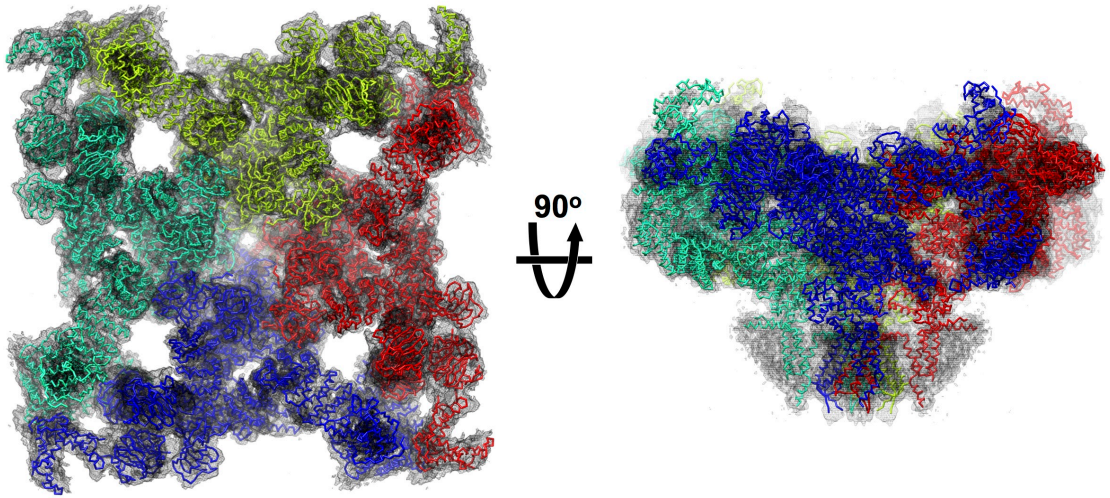
- Use EM-ringer to identify errors in backbone and rotamer geometry.

- Look at everything! Manually check and recheck the fit of every residue in Coot. Tedious but necessary.

- Sometimes, you just can’t tell the right answer. Don’t be afraid to specify sequence ambiguity (use UNKs).

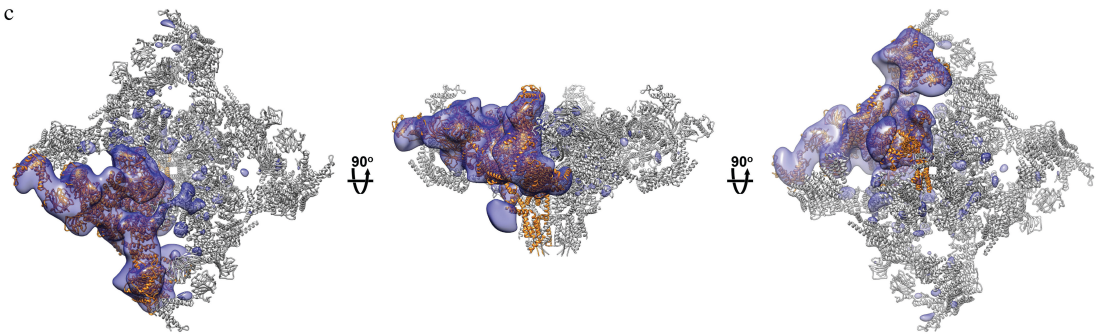
- Half-map FSCs are only really useful to analyse overfitting – they tell you nothing about the local quality or correctness of the model.

Model building in EM – RyR1 as an example



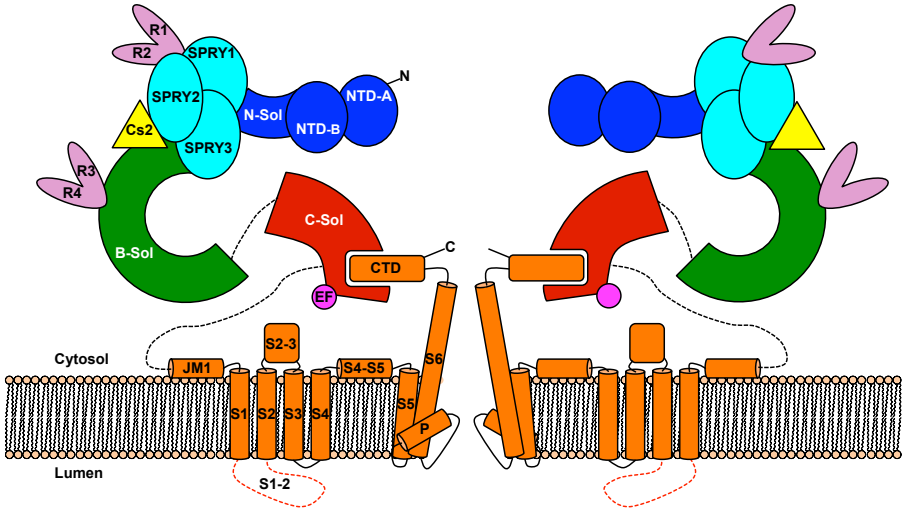
- Starting information:
- Crystal structure of first 500aas
- Homology models for four domains (SPRY1-3, RY12, RY34);
- TM fold apparent from density
- Allowed building of polyalanine trace including known and unknown domains
- Fortuitous 3D class with disordered protomer allowed determination of protomer boundaries

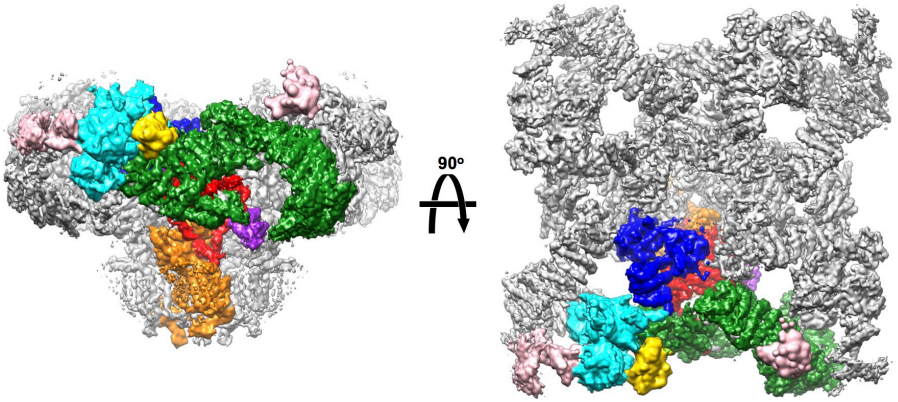
Model building in EM – RyR1 as an example



- Starting information:
- Crystal structure of first 500aas
- Homology models for four domains (SPRY1-3, RY12, RY34);
- TM fold apparent from density
- Allowed building of polyalanine trace including known and unknown domains
- Fortuitous 3D class with disordered protomer allowed determination of protomer boundaries

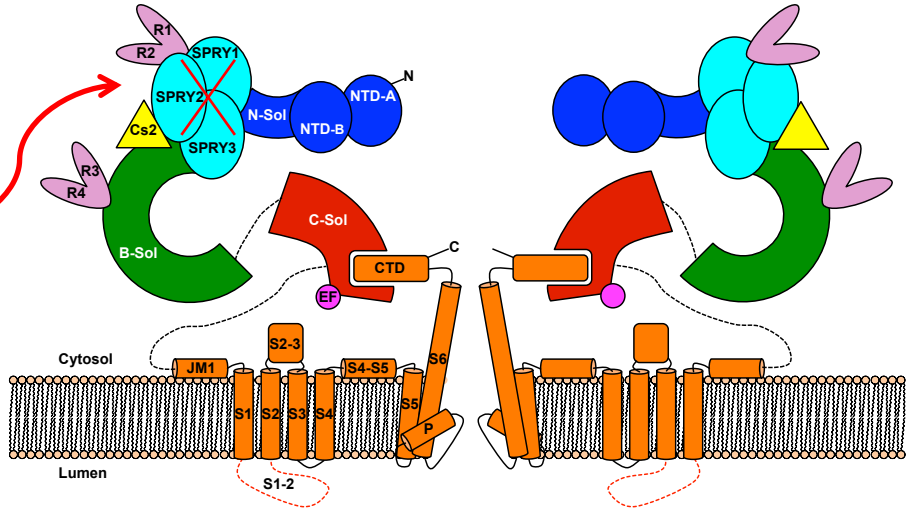
Our initial model allowed location of all previously predicted domains, and several new ones.

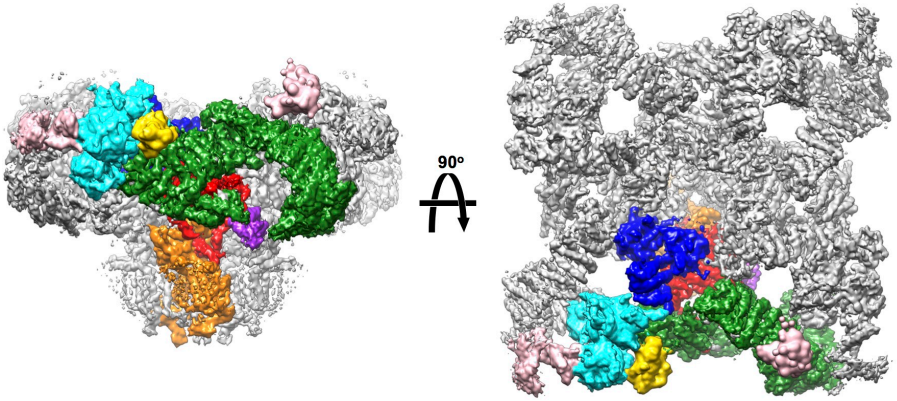




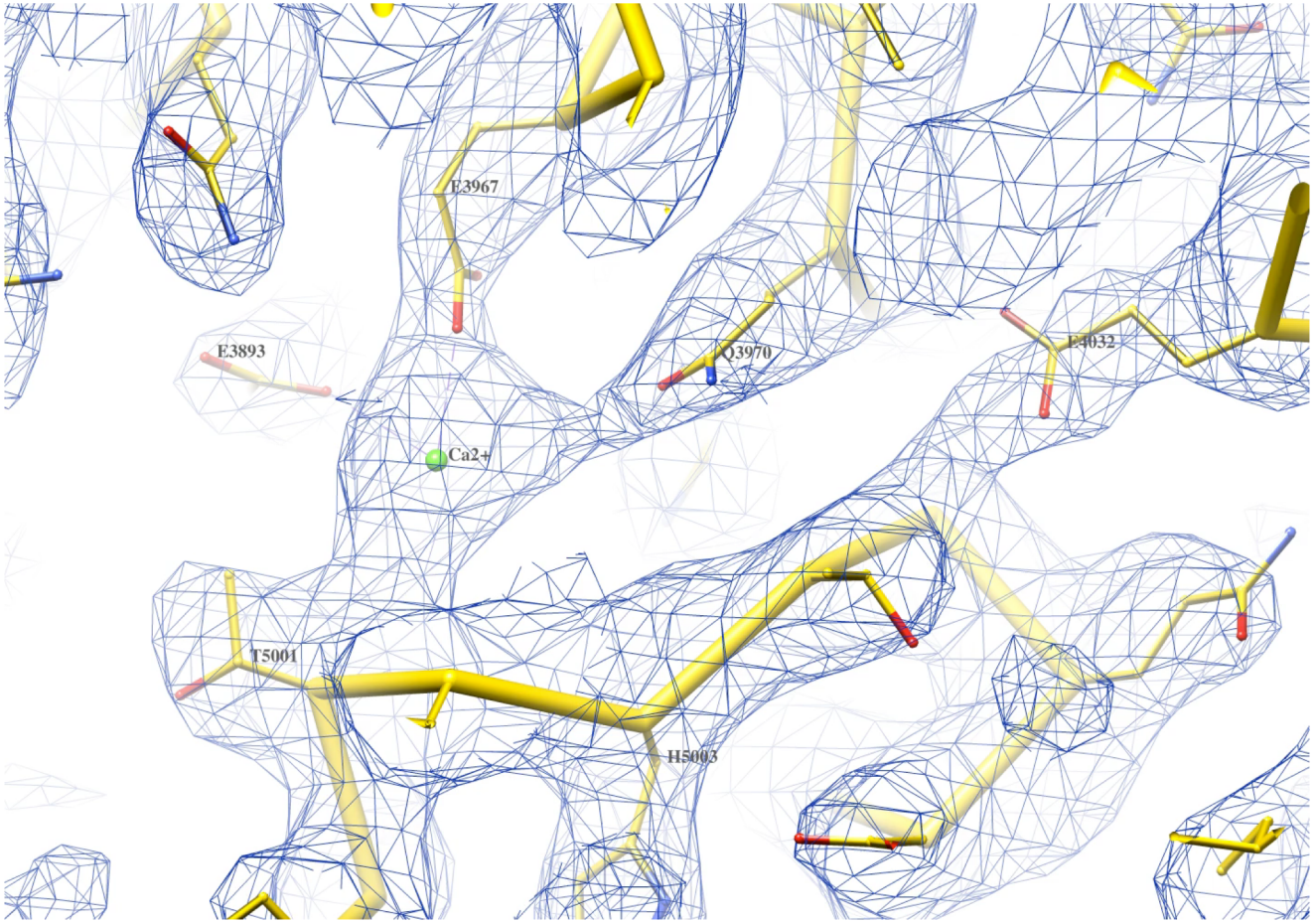
Our initial model allowed location of all previously predicted domains, and several new ones.

We got these wrong. Yan et al. got them right, and we have since updated our model based on their coordinates and our own data.

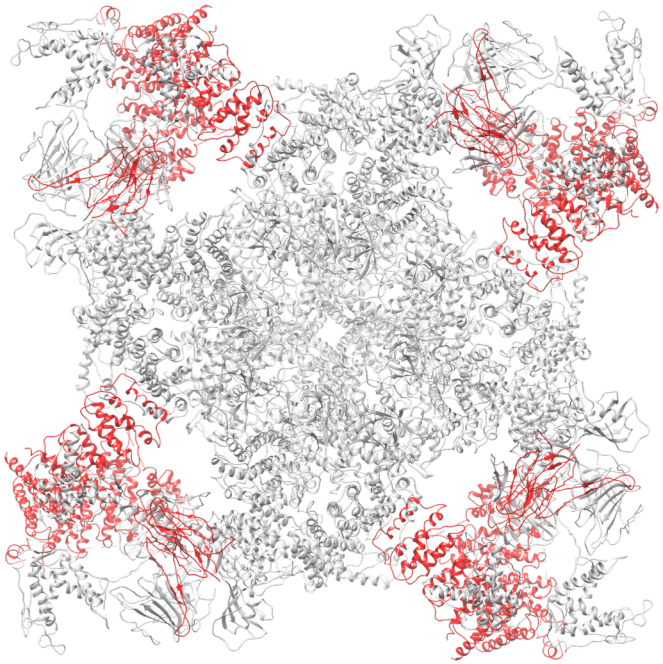




Improved resolution allows building of a near-complete atomic model. Refined using phenix.real_space_refine

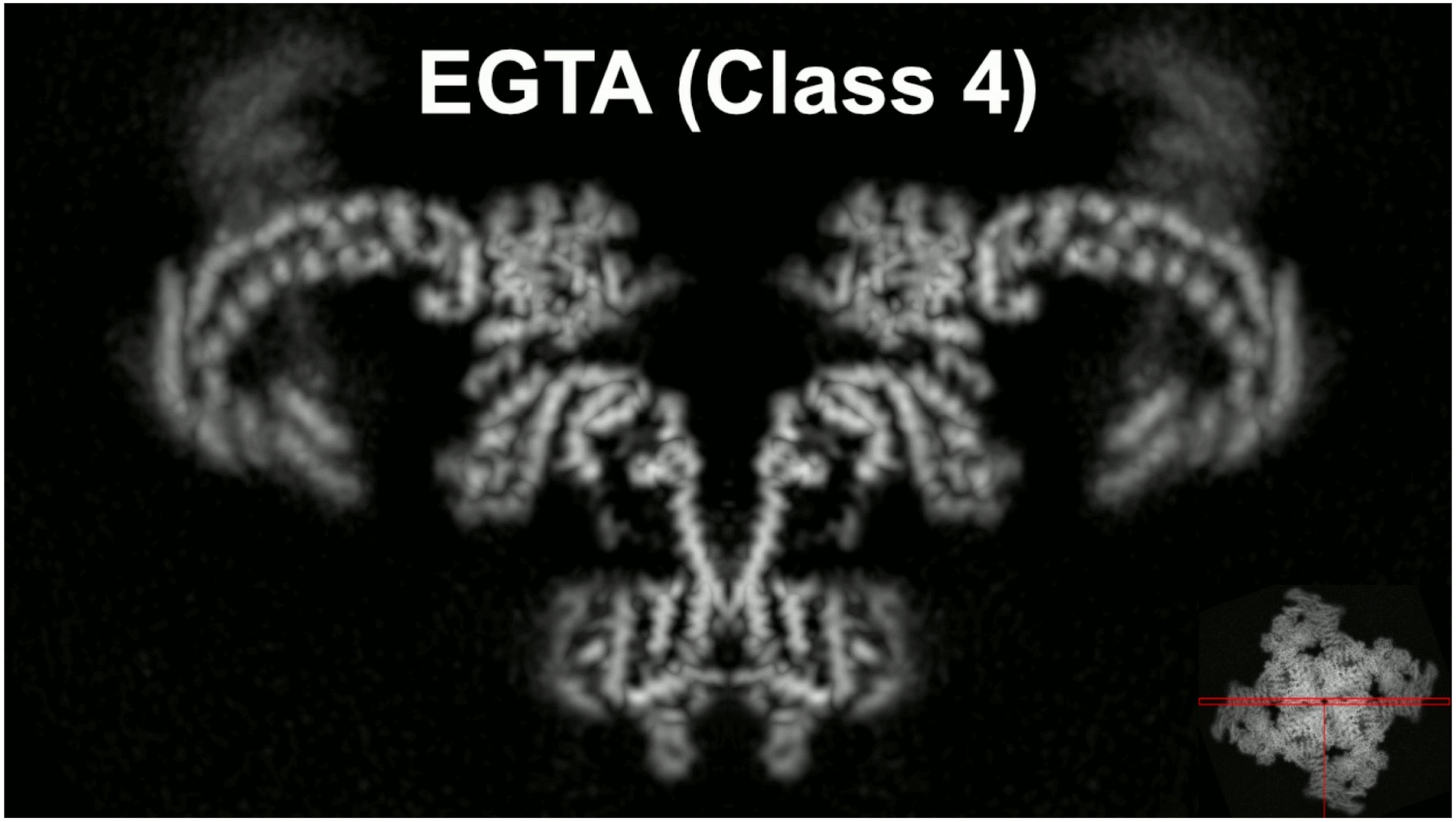


Low local resolution in periphery prevents model completion.

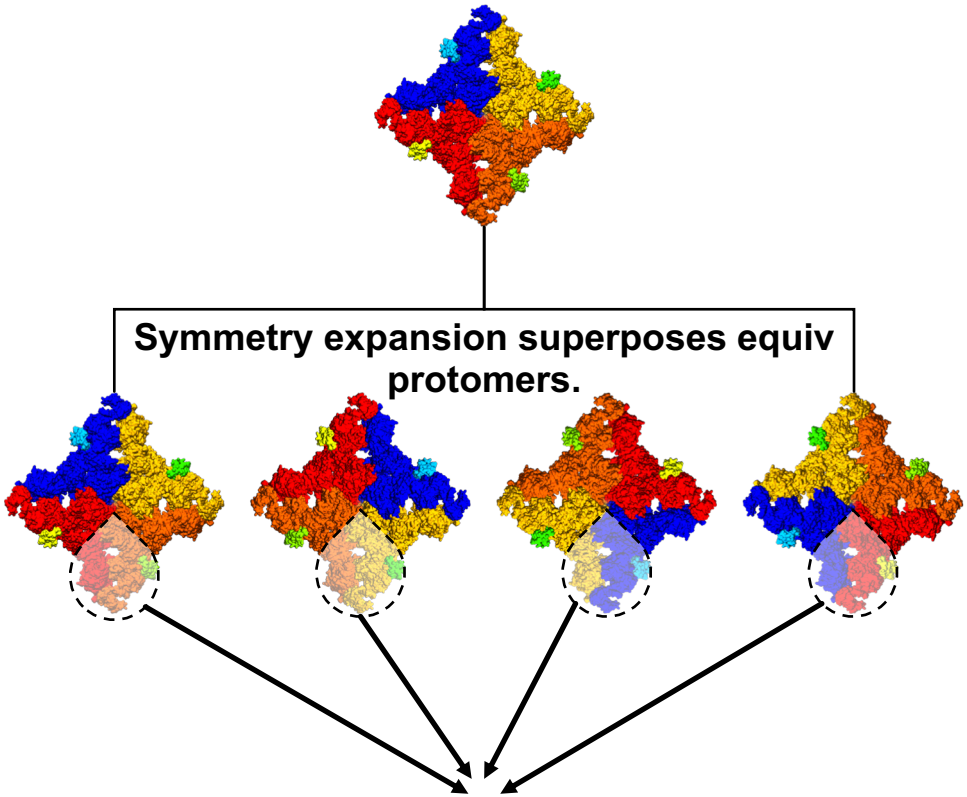


Residues 1250-1650, 2500-3640 still poly-ala – no sequence assignment.

Is lower local resolution in periphery due to unresolved movement of shell?

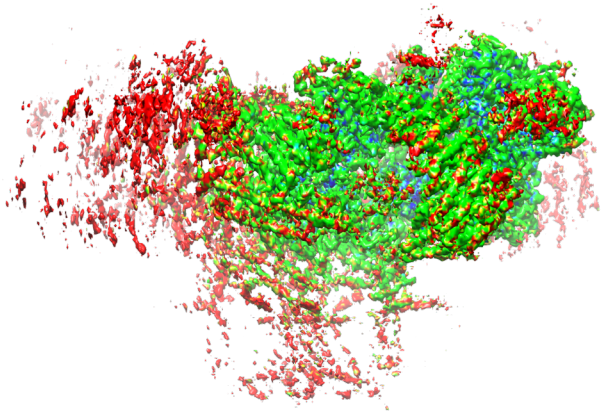


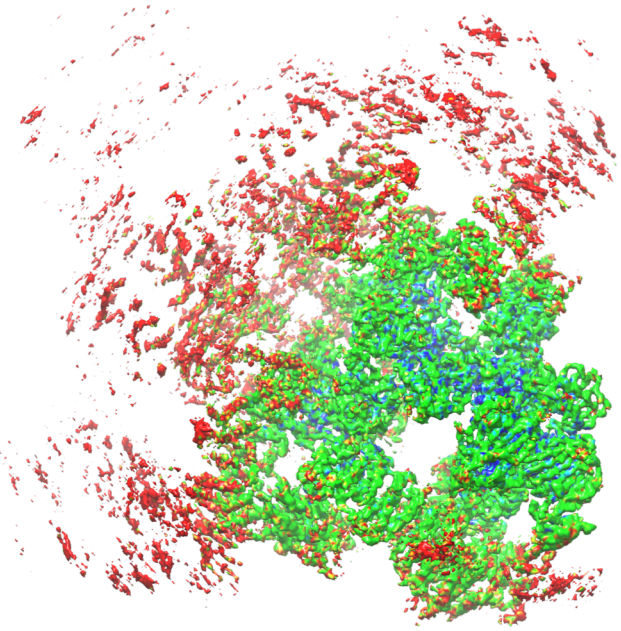
Solution - focused refinement of the shell region of a single protomer in a symmetry-expanded set of particles.



Masked refinement in C1 with local angular searches greatly improves local resolution.

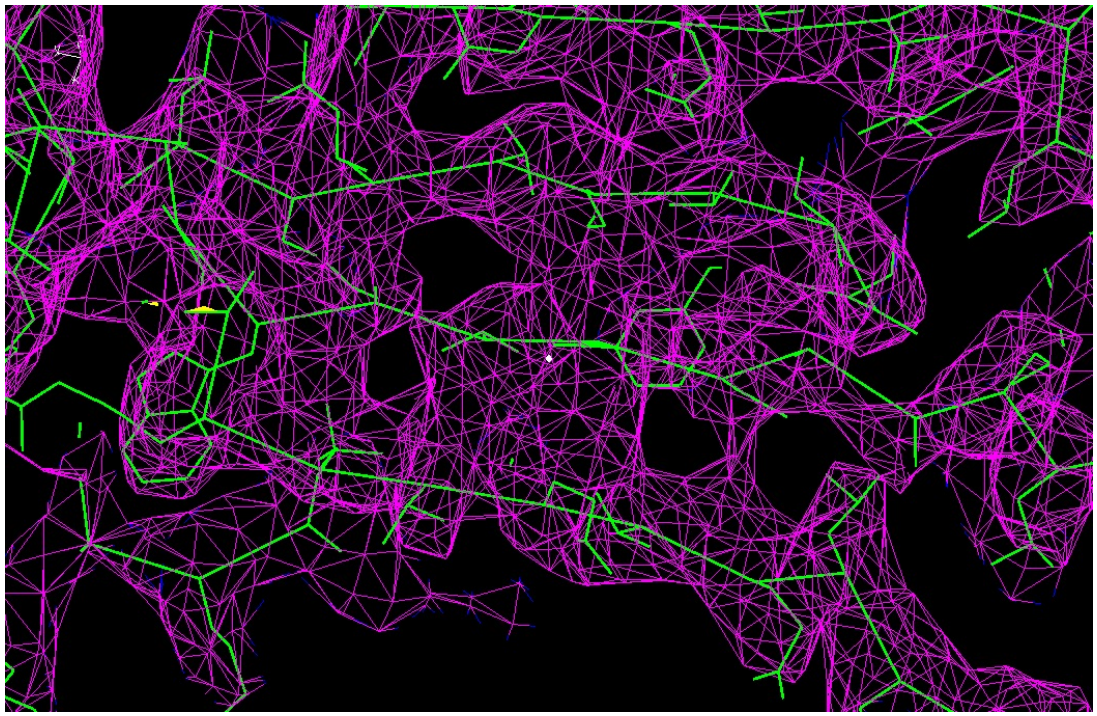
Solution - focused refinement of the shell region of a single protomer in a symmetry-expanded set of particles.





Masked refinement with local angular searches greatly improves local resolution.

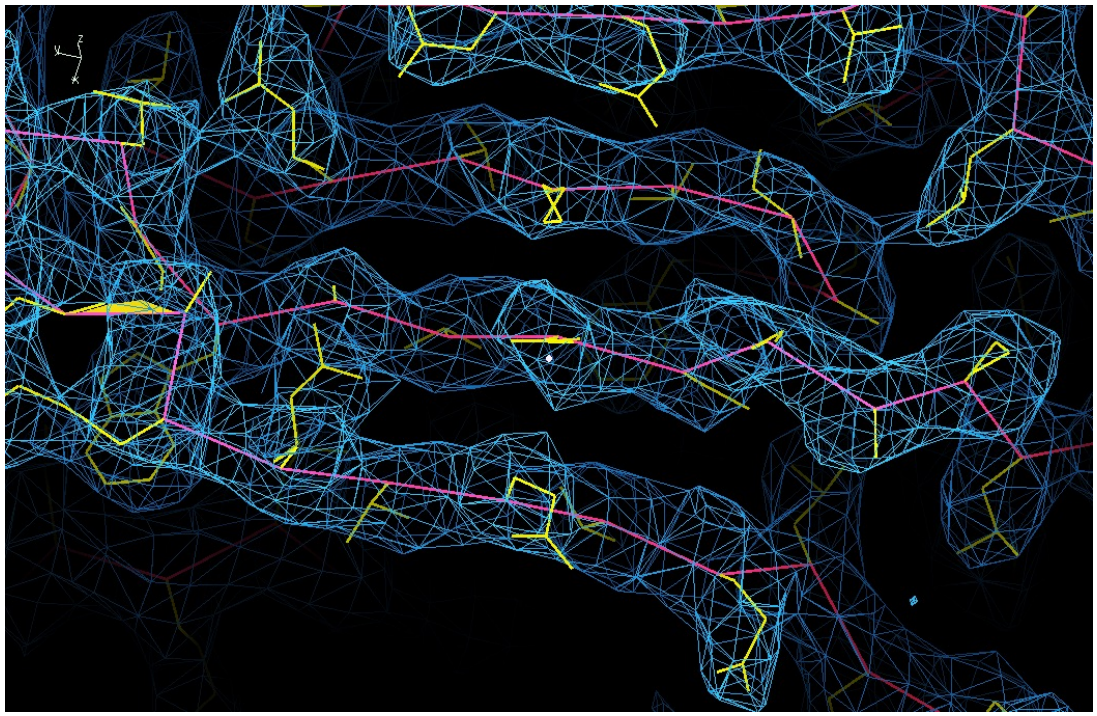
Solution - focused refinement of the shell region of a single protomer in a symmetry-expanded set of particles.



Before

Allows sequence assignment in regions that previously lacked connectivity and/or sidechain definition.

Solution - focused refinement of the shell region of a single protomer in a symmetry-expanded set of particles.



After

Allows sequence assignment in regions that previously lacked connectivity and/or sidechain definition.

Thanks for listening!