EMAN2.12 - Tutorial 2
Dealing with Structural or Conformational Variability in Single Particle Analysis

For the 2015 workshop our intent was to have multiple software package developers all apply their standard techniques to a specimen exhibiting variability. It was a challenge to identify a data set with enough variability in a small number of particles to be suitable for use with laptops at a workshop. Eventually we settled on a ribosome data set Joachim Frank's group made publicly available over a decade ago. While this is not a high resolution data set by any stretch of the imagination, variable data sets rarely are. This data set consists of 10,000 particles, 1/2 of which have a bound EF-G, and 1/2 which don't. Other variabilities also exist within the set, however, the box size is MUCH smaller than it should be for proper CTF processing, and violates EMAN2's requirements. Part of this project is an illustration of how to best handle situations like this.

The issues with this data are:
- box size much too small
- preferred orientation
- defocus groups rather than individual micrographs
- In theory only 2 states, but… (as a test set, that could be viewed as a positive)

EMAN2.1 has at least 5 different programs designed explicitly to explore heterogeneity. While the test data set is small and can be downsampled, most of the 3-D methods for studying heterogeneity would require many hours on a laptop computer, which is more time than we have in the workshop. So, we will begin the tutorial with some 2-D analysis, which would normally be the first step before doing 3-D analysis anyway. The tutorial will explain the various 3-D methods, but you will probably have to complete these after the the end of the workshop.

Step 1 - Canonical procedures
Do the same thing you did in steps 1-12 in the standard single particle tutorial (for homogeneous data). We will not go through the steps in detail here, except where they differ from those previous instructions:

- Unpack the sample data file (ribosome.zip)
- cd ribosome
- run e2projectmanager.py
- set the project parameters (3000 kDa, Cs=2, 200 kV, 2.82 A/pix)
- *CTF → Automated Fitting*
  - As normal, note that SSNR will be depressed by very tight box size, and structure factor may be slightly distorted, but otherwise should be fine
- *CTF→interactive tuning*
  - As normal, many fewer files due to the use of defocus groups. Don't bother looking for bad micrographs.
- *CTF→generate structure factor*
  - You can use *allparticles* for this given the small number of "micrographs"
- *CTF → Automated Fitting*
  - As normal, note that SSNR will be depressed by very tight box size, and structure factor may be slightly distorted, but otherwise should be fine

- *CTF → Generate Output*
  - This one needs to be handled a little differently due to the ridiculously tight boxes. When we do CTF fitting, we can't afford to add any padding to the boxes, since the extra zero regions will seriously screw up the calculations. However, once we have all of the CTF parameters, we can add some padding when we generate output, though we need to combine this with some intelligent masking. We use a soft edged mask to help permit some mis-centering of the particles during later alignment. With a sharp mask there will be a fairly strong bias towards translational alingments of 0,0, even though the regions just inside the mask are pure noise So:
    - Check refinebysnr (why not)
    - Check phaseflip (though we won't really use these)
    - Starting with proctag: small
    - xform.scale:scale=1:clip=168
    - normalize.circlemean:radius=58
    - mask.soft:outer_radius=58:width=6
    - math.fft.resample:n=2
  - The particles we create this way will be phase flipped, nicely normalized, masked and downsampled by 2. If you like you can also generate a set at full scaling (without math.fft.resample and with a different *proctag*). During the workshop, full scale particles will slow processing too much, but if you are following this tutorial when you have more time and a bigger computer you can use the fully sampled data in any of the methods described below.
- *Particle Sets → Build Particle Sets*
  - *allparticles* and *excludebad* checked, though we won't be doing any explicit bad particle exclusion with this example data set.
  - *setname*: **all**
  - Due to the use of defocus groups it is virtually impossible to identify and exclude bad micrographs in this situation, so we simply use all of the particles.
  - ➡If you're particularly clever, you might notice that the sets we get out have 10,001 particles, not 10,000. No, I am not a palindrome fetishist. I accidentally duplicated one particle somewhere when turning the single SPIDER stack into the defocus groups. This shouldn't cause us any particular problems.

<u>Step 2 - Generate 2-D class-averages</u>

This time instead of identifying bad particles, or making class-averages for 3-D reconstruction (though we could also use them for that purpose), we're making them for purposes of looking at structural variability in 2-D. While certainly 3-D is the goal, it can be extremely useful to get a feel for what's going on in 2-D first.

➡ Structural variability can be continuous or discrete. Compositional variability, also known as discrete heterogeneity, is comparatively a relatively easy problem. For example, if you are looking at a ligand binding problem, you need only to decide which of 2 discrete groups each particle belongs in along with determining its orientation. However, for structural variability, also known as continuous heterogeneity, it is quite possible to confuse structural changes with orientation changes, and much like handedness, sometimes it cannot be unambiguously determined from a given data set at a given resolution. Additionally, in such situations there is no "correct" number of classes to subdivide your structure into. If you are looking at a single degree of freedom with 20 Å of motion, then you could argue that you need to subdivide the data into 4 groups if you hope to achieve 5 Å resolution. The practical reality, though, is generally much worse than this.

➡ So, when dealing with unknown sorts of variability, it is critical to use more than one method to examine your data, and insure that all of the different methods are consistent with the interpretation you are trying to make. Despite the fact that many packages (like EMAN and Relion) treat 3-D classification in conceptually similar ways, the details matter, a LOT. If you determine multiple structures from a single data set using one method or software package, and can reproduce those results qualitatively with another method/package, you can be much more confident that you are looking at a trustworthy result. Certainly you can easily find cases where results from EMAN, Relion and SPARX will not agree (usually low resolution results), and in such cases, and unless you are fond of retractions, cherry picking the result you like best, without an awfully convincing rationalization, is probably not a good strategy.

➡ While 2-D results can be difficult to interpret in a 3-D context, they are far less ambiguous, since this analysis does not involve the added complication of needing a self-consistent 3-D structure, and often such analyses can yield surprising amounts of information.

- 2D Analysis → Reference Free Class Averaging
  - *ncls* = **100**
  - *iter* = **8**
  - *nbasisfp* = **16**
  - *naliref* = **16**
  - center: **xform.centerofmass**

There is no symmetry in this structure, so that leads us to want a larger than usual number of classes. We'd like to have the possibility of considering more subtle variations in the structure so we set nbasisfp (number of PCA vectors to generate) to a larger than usual number. Similarly, there are more distinct views of a ribosome than a higher symmetry particle, so we also need more alignment references. We reduce the number of iterations to keep the time reasonable.

This took about 30 minutes to run on my laptop. You can always look at intermediate results if it hasn't finished yet when you need to look at them.

<u>**Step 3 - Reclassify Similar Orientations**</u>

If you have two particles in different conformations how exactly would you define the "same orientation"? Unless you are already splitting the data into 2 structurally homogeneous populations there is no single answer to this question. This means when we classify in 2-D if we want to look at particle variability, we cannot precisely say we are looking for particles in the same orientation. We have to settle for particles in similar orientations.

- Look at your highest numbered allrefs_XX file in a tiled view. Particles in similar orientations should be grouped together. Modify the display so it shows ptcl_repr in the corner, telling you how many particles were used for each average.
- Open the same averages in a single particle view. Use the up/down arrows to scroll to a region in the file with similar looking views, and switch rapidly back and forth among the views you feel are similar. Use this to get an initial feel for the variability present among these

averages, and whether they are similar enough in some aspect to consider together as a group.

- Once you have identified a set of classes to group together:
- 3D Refinement → 2D Heterogeneity Analysis
  - This is the same interface you used previously to identify boxed particles, it simply appears twice in the project manager for these two distinct purposes.
  - Select the same class-average file you were looking at above, and mark the set of class-averages you felt were in a compatible orientation.
  - Use the *Make New Set* button and give the particle a name like **view_1**
  - If you observe other compatible sets of class-averages, press *Clear*, mark the other group, and save this new set with the name **view_2.**

Step 4 - Run a 2-D refinement on view_1

We will now reclassify the particles in view_1 into a few new classes:

- 2D Analysis → Reference Free Class Averaging
  - *ncls* = **12**
  - *iter* = **6**
  - *nbasisfp* = **5**
  - *naliref* = **1**
  - center: **xform.centerofmass**

These particles should all be in pretty much the same orientation, so we don't need multiple alignment references. The number of classes depends somewhat on how many particles you have in view_1. You should target 10-20 particles per class at a minimum. You can reduce the number of classes if necessary. Remember, though, that you inevitably have a few misclassified particles in your view_1 list, so a few of the new class-averages you get will not match the set at all. Once the new class-averages have been generated:

- Look at your highest numbered allrefs_XX file. You should now see more subtle differences among the classes. It is easier to visualize this if you follow this process:
  - Open allrefs_XX in a tiled display window
  - Middle-click to open the control panel, and select *Del* mode.
  - Delete any particles which don't seem to fit with the other averages, or any that are particularly noisy.
  - Hit the Save button (upper right) and save the cleaned up stack as **view_1_avgs.hdf**
  - **e2stacksort.py view_1_avgs.hdf sorted.hdf --simalign rotate_translate_flip --iterative --useali**
  - Open sorted.hdf in a single image viewer and scroll back and forth through the images to see what sort of variations you have in this "view".

  ➡ if you have ImageMagick installed on your computer, you could convert this sequence into a GIF animation with **e2stackanim.py**
- Repeat this process for other orientations with a significant number of particles. Note that unlike 3-D reconstruction, this sort of analysis actually works <u>better</u> with preferred orientation.

Step 5 - Run a normal single particle refinement with all of the data
Before we start trying to split the 3D data into multiple groups it is usually a good idea to do a normal single particle refinement using all of the data to use as a baseline. It also provides necessary information for some of the other methods described in step 7. For this ribosome example, this process should work well.

If you have your own data, and it has very high compositional heterogeneity, this idea may not work as well as we might hope. You may get a map which is visually unappealing and clearly not correct. This fact in and of itself, is valuable, and indeed, we may then jump straight to one of the methods designed to separate our data into groups.

We will not go through the details of how to do this refinement. Based on your experience with the e2refine_easy tutorial you should be able to produce a decent ribosome from this data with little trouble. If you are feeling a bit lazy you can go to the PDB or EMDB and download any 70S ribosome and use it as a starting model rather than generating one yourself. For purposes of the tutorial, the exercise of generating a starting model is only useful if you want the practice.

Step 6 - CHOICES
We probably won't have time to go down any of these paths at the workshop, but when you get back home, you can try some of these other heterogeneity analysis methods.

Method 6.1 - Variance Analysis on the single model refinement
In this paper:
Zhang, W., Kimmel, M., Spahn, C.M., and Penczek, P.A. (2008). Heterogeneity of large macromolecular complexes revealed by 3D cryo-EM variance analysis. Structure. 16:1770-76.

Penczek's group introduced the concept of performing variance analysis to CryoEM maps, to identify which regions of the map were more/less reliable, and/or where there might be motion or other variability in the underlying data. This is not a local resolution estimator like ResMap produces, but rather is a method for essentially putting an error bar on each voxel of your final reconstruction.

EMAN2.1's implementation of this method performs bootstrapping on class averages rather than directly on the 3-D model, but in a later paper, Penczek's group found that the original method needed to take into account the anisotropic particle orientation distribution and came up with a very complicated method to accomplish this. The EMAN implementation accomplishes the same goal through class-based bootstrapping and thus should produce very reliable results.

- This program is called e2refinevariance.py, and has not yet (due to a forgetful faculty member) been added to the projectmanager. So you will need to run it from the command-line.
- You must have a complete refinement from step 5 to run this program (refine_XX folder)
- give it the following options:
  - **--path refine_XX** (an existing complete refine_easy result)
  - **--apix 5.62** (or 2.82 if you used the full data)
  - **--mass 3000**
  - **--input** <particle stack based on refine_XX input>
    - You cannot low-pass filter the results of a variance calculation, so if you want to look for low-resolution variability specifically, you must filter the input data before running the program.

- **--nmodels 100** (the number of bootstrap volumes to generate, to get a useful variance I suggest at least 40-50)
- **--iteration** YY (the iteration number within refine_XX to produce the variance of)
- **--keep3d** (this will keep all 100 bootstrap volumes. A lot of disk space, but if you need to tweak some parameters, can use later with --volfiles)
- **--threads** and **--parallel** as usual

Put this together and we get something like:
e2refinevariance.py --path refine_05 --apix 5.62 --mass 3000 --input sets/all__ctf_flip_small.lst --nmodels 100 --keep3d --iteration 3 --threads 4 --parallel thread:4

The results will appear in a new folder called *refinevar_QQ*

Method 6.2 - Traditional multi-reference refinement

This is, is the "traditional" multi-model refinement concept. Say in a normal 3-D refinement of some particle you would generate M different projections of that structure when trying to determine its orientation. You now also have N different 3-D references, so you make projections of each in the same set of projection directions, giving a total of N*M different projections. For each individual particle you now have to decide not only which of the M orientations its in, but also which of the N references it best matches. This classification leads to having N sets of M classes each, leading to N different reconstructions. Each of these then becomes a reference for the next round of refinement, and things proceed almost exactly as they do in normal single particle analysis.

This general concept is available in virtually any of the available software packages which support heterogeneous particle analysis, often with subtle variations in each.

- ➡ I have heard comments sometimes from people saying “I ran a 3-way alignment in Relion, then I ran one in EMAN2.1, and I liked Relion’s results better”. Is this true? How can you make value judgements like this? I believe the most common cause of this problem is actually due to a mistake in EMAN usage, and one that I may have actually promulgated in the past. Consider:
  - ➡ In the normal single particle analysis tutorial, I describe how the iterative class-averaging process permits EMAN refinements to converge in many fewer cycles than most other software packages. In packages like Relion and Frealign, it is quite common to run for 20-30 cycles before converging. Iterative class averaging means EMAN typically only requires ~4 iterations to achieve the same level of convergence.
  - ➡ Can we make the same argument for multi-reference refinement? NO! While iterative class-averaging remains a useful thing to do, it does not make multi-model refinement converge any faster. In Relion, it runs 25 cycles when splitting a data set among multiple models! In EMAN, if you are seeding the process with a random particle distribution you may need to run quite a few iterations to come to convergence, many more than the typical 4 used for single model refinements (though 25 may be overkill). This, I believe is the greatest cause for this misperception when comparing the packages. Note also that EMAN has a variety of other heterogeneity resolving methods which are not this computationally demanding!

To perform multi model refinement, in the projectmanager:

- *3D Refinement → Multiple Map Refinement*
  - The multimodel refinement needs to be seeded with N different inputs to produce N different outputs. If the inputs are identical, then there is no way to separate them, so there are several mutually exclusive ways to do this:
    1. Provide N different volumes from some other source (like one of the other heterogeneity analysis methods discussed below). To do this, enter a comma-separated list of the filenames in the *models* text box. Do not fill in anything above this in the GUI in this case.
    2. Provide a single reference volume (specified in the *model* text box), then one of:
       - 2.1. *mapfragment* - automatically segment the single map, then generate N different volumes with a random segment excluded from each.
       - 2.2. *randclassify* - in the first round, each particle is randomly assigned to a different class (similar to what Relion does, I believe). This option is probably the least biased
       - 2.3. randphase - The volume is phase-randomized N times beyond some automatic resolution to produce N different seeds
  - *nmodels* is only specified with option 2 above. With option 1, the number of models is the same as the number of provided models
  - *input* - The particles to refine. As with normal single particle refinement, this is one of the sets/*lst files. This MUST be a .lst file to work properly, and the particles must be phase-flipped.
  - *targetres* - Similar to the option in e2refine_easy. However, if you are trying to perform coarse classification, this has little to do with the final resolution you want to achieve. It has more to do with the level of detail at which you expect to see differences. So, for something like the current ribosome example, I suggest a value of **15** or **20** Å.
  - *sym* - **c1** (no symmetry)
  - *mass* - **3000**
  - *iter* - **10** (see the discussion above, you may find you need even larger numbers for subtle variations, but 10 is usually more than enough for big, discrete changes)
  - *apix* - Unlike e2refine_easy, at the moment, this is required. If you use the data downsampled by 2 (you probably should) enter **5.64**
  - fill in *parallel* and *threads* as usual. If you want to run on a cluster, go to the command tab, copy the command out, and run on a cluster as described:
    http://blake.bcm.edu/emanwiki/EMAN2/Parallel/Mpi
  - defaults should be fine for the other options. For a single computer: *Launch*

After this finishes running (will take quite a bit longer than normal refinement):
- Results will be in a folder called *multi_XX*
- Note that e2refinemulti does produce a report/ folder, like e2refine_easy, but at the moment, this functionality has not been completed for e2refinemulti. Don't even bother looking at it for now. At some point we will fix this…
- The final results are in multi_XX/threed_YY_ZZ.hdf
  - Each time you run e2refinemulti in the project XX is increased by 1
  - YY is the iteration number. ie - if you tell it to run 10 iterations, you should be looking at YY=10
  - ZZ is the reference number. There should be N of these matching the number of starting models.
- multi_XX/fsc_mutual_avg_YY.txt contain averaged FSC curves computed among the different maps in each iteration. Unlike all of the other refinements we've done, these FSC curves are

expected to get worse as we refine. As the models gradually diverge from one another, as you expect, agreement becomes worse.

- If the refinement runs to completion, you will also magically see new sets appear, with names like:
  - all__ctf_flip_small-lst_mulXX_itYY_mZ.lst
  - XX is the name of the multi_XX folder
  - YY is the iteration number in that folder
  - Z is the number of the model
  - These sets contain the particles associated with each of the N final maps at the end of the multi-model refinement, but they are only created if e2refinemulti runs to completion

- Once you have completed the multi-model refinement it is CRITICAL that you not just trust these results as they are. There has been no "gold standard" or equivalent methodology to get rid of model bias (this is also true in Relion!). At this point you MUST take these structures and confirm that you are seeing reality not fantasy. This is usually accomplished in a couple of stages:
  - First, you need to run a normal e2refine_easy on the particles extracted for each subpopulation. That is why the new sets/*lst files are produced for you, so you can do this. For this task you have a choice of what to use as an initial model for each of these refinements. For this initial run, it is fine to use the final outputs in multi_XX as initial models. This run will give you some resolution estimates, etc. If these e2refine_easy commands give you better resolution than the refinement you did above in step 5, that is a good sign, as it indicates the particle subpopulation has sufficiently improved homogeneity to produce a higher resolution.
  - Next, we need to insure that these subpopulations are fairly robust. An excellent way to do this is by cross-validation. The idea is to take one of the particle populations we just used, and refine it, not against the map that was used to produce it, but against one of the other maps from multi_XX. ie - in the previous step we refined all__ctf_flip_small-lst_mulXX_itYY_m1.lst against multi_XX/threed_YY_01.hdf, NOW we want to refine the same data using multi_XX/threed_YY_02.hdf (for example) as a starting model. If these new refinements produce maps that look like the particle data, then you have a robust population and you can be moderately more confident that your data separation is valid. However, if the refined map looks more like the threed file you used as a starting model, then your results are likely due to model bias, not a useful classification of your data.
- Note that you can also take the final sets derived from e2refinemult as inputs to another run of e2refinemulti to build up a hierarchy of split data sets.

Method 6.3 - Robust data splitting into 2 populations

Let us say that you are performing a ligand binding experiment or some other experiment where you expect the data to fall into 2 discrete classes. While e2refinemulti can give a very respectable result in these cases, just as there is some orientation uncertainty for each particle, there can also be a substantial classification uncertainty. In 6.2 at the end we tried to make sure that our split particle populations were robust. While this should be true on average, on a per-particle basis there will still be a significant number of incorrectly classified individual particles.

To have completely accurate classification between the models there must be sufficient information in each individual particle image to make the decision between maps. If the ligand in question is small, very often noise and other perturbations will be strong enough to make this per-particle classification at least somewhat inaccurate.

e2classifyligand performs a much more focused process specifically to classify particles. The program takes 2 reference volumes as input, which should at least weakly represent liganded and unliganded populations. The program uses the difference between these references to identify a mask where the most important ligand information exists in 3-D. It then takes an existing single-model refinement (step 5 above), and "carefully" subtracts a projection of the map from each particle (including CTF and other information) then 2-D masks the result, and finally classifies the particle based on this information. More about its usage will be documented here (as soon as I can find or regenerate the material):

http://blake.bcm.edu/emanwiki/EMAN2/Programs/e2classifyligand

Method 6.4 - Split a single model refinement into 2 using 2-D PCA

This is a very new method, which has not yet been published. It is still experimental. That said, it can very quickly split a single 3-D refinement into 2 subvolumes, based on PCA applied to each individual orientation in 2-D. It then uses what I believe is a novel method to group the PCA-split class-averages into 2 different 3-D maps. That is, you can provide this method with the results of a single model refinement (again, step 5), and it will produce 2 maps as output with no other information. I will document how this procedure works in a manuscript soon. For now, I simply encourage you to give it a try and see what you get. Typically I would take the results of this method, and use the 2 resulting maps as inputs to e2refinemulti or e2classifyligand.

- 3D Refinement → *Split map into two subgroups*
  - *path =* **refine**_*XX*
  - *parallel = (standard option)*

- The results of this splitting operation will be put directly in the refine_XX folder:
  - threed_YY_split.hdf (contains the 2 output volumes in an HDF stack)
  - classes_YY_split0.hdf and classes_YY_split1.hdf
  - sets/split_XX_0.lst and sets/split_XX_1.lst

I would suggest running the program, then looking at the resulting threed_YY_split file to see if you observe a biologically interesting change. If you see something that looks interesting, it is important to follow up with either an e2refine_multi, or at least an e2refine_easy using the split particle sets.

Method 6.5 - Breaking the symmetry of a pseudosymmetric structure

This method is a bit of an outlier. It doesn't target heterogeneity per-se. Rather it considers the problem of structures with pseudosymmetries. For example, an icosahedral virus particle with one unique vertex (typically a portal complex) is generally treated with icosahedral symmetry to achieve a high resolution reconstruction. Of course, this is nonsense. It is averaging the unique vertex into all of the others and producing some sort of hybrid average. Nonetheless, one can often achieve very nice high resolution structures of the capsid 'on average' this way. To go from this to an asymmetric structure can be best accomplished by making use of our existing knowledge of the orientation of the particle within the asymmetric unit.

In e2refine_easy, there is a checkbox called "breaksym". If you check this box, the final map it produces will be an asymmetric map, rather than one with the specified symmetry imposed. During refinement it will make use of the specified symmetry, so the initial model must be aligned to the standard orientation of the symmetry axes. Once it determines the orientation of each particle assuming the specified symmetry it then performs an independent search for which asymmetric unit the particle fits into. The final reconstruction will then have broken symmetry, and the particle orientations will cover the full unit hemisphere.

Method 6.6 - Extracting components of individual particles for independent processing
While 'focused classification', ie - using a mask in conjunction with multi-model refinement, is a powerful technique, there are cases with large amounts of flexibility where this method is insufficient. As one very simple example, consider an ostensibly symmetric particle like GroEL with 14 copies of each monomer. Each of these monomers is undergoing some flexible motion independently in solution. In the apical domain this may be as large as ~10 Å. To study dynamics, we would like to look at the 14 monomers individually, rather than the aggregate complex which has too much variability to subclassify.

We have a program based on a method developed in ~2005 in EMAN1 to extract individual monomers or other components of larger particles for independent analysis. The program is called e2extractsubparticles.py.

I'm not quite ready to describe this program in detail here, but if this sounds like something you are interested in trying, contact me (sludtke@bcm.edu), and I will give you the necessary details to try it. Eventually it will become part of this tutorial as it evolves.